**Vastbase G100 V2.2**

**(Build 10)**

**开发者指南**



北京海量数据技术股份有限公司

【版权声明】

©2007-2022 北京海量数据技术股份有限公司 版权所有

本文档著作权归 **北京海量数据技术股份有限公司**（简称“海量数据”）所有，未经海量数据事先书面许可，任何主体不得以任何形式复制、 修改、抄袭、传播全部或部分本文档内容。

北京海量数据技术股份有限公司保留所有的权利。

【服务声明】

本文档意在向客户介绍海量数据全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的产品、服务的种类、服务标准等应由您与海量数据之间的商业合同约定，除非双方另有约定，否则，海量数据对本文档内容不做任何明示或模式的承诺或保证。

**目录**

[1. 字符集 24](#_Toc27793)

[1.1. 字符集列表 24](#_Toc977)

[1.2. 字符集设置 28](#_Toc22103)

[2. Schema 28](#_Toc17121)

[2.1. WDR Snapshot Schema 30](#_Toc2591)

[2.1.1. WDR Snapshot 原信息表 34](#_Toc8773)

[2.1.2. WDR Snapshot 数据表 35](#_Toc4832)

[2.1.3. 查看WDR报告 35](#_Toc19927)

[2.2. DBE\_PERF-Schema 42](#_Toc3785)

[2.2.1. OS 43](#_Toc28125)

[2.2.2. Instance 44](#_Toc17554)

[2.2.3. Memory 45](#_Toc6037)

[2.2.4. File 49](#_Toc16475)

[2.2.5. Transaction 53](#_Toc11016)

[2.2.6. Session/Thread 56](#_Toc7345)

[2.2.7. Query 72](#_Toc20635)

[2.2.8. Lock 109](#_Toc96)

[2.2.9. Wait Events 113](#_Toc7155)

[2.2.10. Configuration 115](#_Toc24644)

[2.2.11. Operator 119](#_Toc18213)

[2.2.12. Workload Manager 125](#_Toc19512)

[2.2.13. Global Plancache 126](#_Toc20559)

[2.2.14. RTO 127](#_Toc2249)

[2.2.15. Cache/IO 128](#_Toc21770)

[2.2.16. Utility 141](#_Toc874)

[2.2.17. Object 152](#_Toc8788)

[2.2.18. Workload 181](#_Toc23848)

[2.3. Information Schema 188](#_Toc25828)

[2.3.1. ADMINISTRABLE\_ROLE\_AUTHORIZATIONS 188](#_Toc26683)

[2.3.2. APPLICABLE\_ROLES 188](#_Toc1286)

[2.3.3. ATTRIBUTES 188](#_Toc5862)

[2.3.4. CHARACTER\_SETS 192](#_Toc21105)

[2.3.5. CHECK\_CONSTRAINT\_ROUTINE\_USAGE 193](#_Toc3711)

[2.3.6. CHECK\_CONSTRAINTS 194](#_Toc1975)

[2.3.7. COLLATIONS 194](#_Toc26523)

[2.3.8. COLLATION\_CHARACTER\_SET\_APPLICABILITY 194](#_Toc10809)

[2.3.9. COLUMN\_DOMAIN\_USAGE 195](#_Toc11439)

[2.3.10. COLUMN\_OPTIONS 195](#_Toc13005)

[2.3.11. COLUNM\_PRIVILEGES 196](#_Toc29350)

[2.3.12. COLUMN\_UDT\_USAGE 196](#_Toc3185)

[2.3.13. COLUMNS 197](#_Toc828)

[2.3.14. CONSTRAINT\_COLUMN\_USAGE 201](#_Toc8645)

[2.3.15. CONSTRAINT\_TABLE\_USAGE 202](#_Toc3871)

[2.3.16. DATA\_TYPE\_PRIVILEGES 202](#_Toc14112)

[2.3.17. DOMAIN\_CONSTRAINTS 203](#_Toc20003)

[2.3.18. DOMAIN\_UDT\_USAGE 203](#_Toc32376)

[2.3.19. DOMAINS 204](#_Toc26125)

[2.3.20. ELEMENT\_TYPES 207](#_Toc31616)

[2.3.21. ENABLED\_ROLES 210](#_Toc29890)

[2.3.22. FOREIGN\_DATA\_WRAPPER\_OPTIONS 210](#_Toc16797)

[2.3.23. FOREIGN\_DATA\_WRAPPERS 211](#_Toc16601)

[2.3.24. FOREIGN\_SERVER\_OPTIONS 211](#_Toc8871)

[2.3.25. FOREIGN\_SERVERS 211](#_Toc27625)

[2.3.26. FOREIGN\_TABLE\_OPTIONS 212](#_Toc23696)

[2.3.27. FOREIGN\_TABLES 212](#_Toc7171)

[2.3.28. KEY\_COLUMN\_USAGE 213](#_Toc1382)

[2.3.29. PARAMETERS 214](#_Toc32309)

[2.3.30. REFERENTIAL\_CONSTRAINTS 216](#_Toc12370)

[2.3.31. ROLE\_COLUMN\_GRANTS 217](#_Toc21012)

[2.3.32. ROLE\_ROUTINE\_GRANTS 217](#_Toc23410)

[2.3.33. ROLE\_TABLE\_GRANTS 218](#_Toc28313)

[2.3.34. ROLE\_UDT\_GRANTS 219](#_Toc30160)

[2.3.35. ROLE\_USAGE\_GRANTS 219](#_Toc29396)

[2.3.36. ROUTINE\_PRIVILEGES 220](#_Toc1473)

[2.3.37. ROUTINES 220](#_Toc10558)

[2.3.38. SCHEMATA 228](#_Toc31592)

[2.3.39. SEQUENCES 228](#_Toc6844)

[2.3.40. SQL\_FEATURES 229](#_Toc22583)

[2.3.41. SQL\_IMPLEMENTATION\_INFO 230](#_Toc30349)

[2.3.42. SQL\_LANGUAGES 230](#_Toc23465)

[2.3.43. SQL\_PACKAGES 231](#_Toc24238)

[2.3.44. SQL\_PARTS 232](#_Toc9858)

[2.3.45. SQL\_SIZING 232](#_Toc25738)

[2.3.46. SQL\_SIZING\_PROFILES 232](#_Toc4442)

[2.3.47. TABLE\_CONSTRAINTS 233](#_Toc8164)

[2.3.48. TABLE\_PRIVILAGES 233](#_Toc6211)

[2.3.49. TABLES 234](#_Toc23139)

[2.3.50. TRIGGERED\_UPDATE\_COLUMNS 235](#_Toc3262)

[2.3.51. TRIGGERS 236](#_Toc23591)

[2.3.52. UDT\_PRIVILEGES 237](#_Toc2158)

[2.3.53. USAGE\_PRIVILEGES 238](#_Toc2632)

[2.3.54. USER\_DEFINED\_TYPES 239](#_Toc11799)

[2.3.55. USER\_MAPPING\_OPTIONS 241](#_Toc20033)

[2.3.56. USER\_MAPPINGS 241](#_Toc29683)

[2.3.57. VIEW\_COLUMN\_USAGE 242](#_Toc13694)

[2.3.58. VIEW\_ROUTINE\_USAGE 242](#_Toc18793)

[2.3.59. VIEW\_TABLE\_USAGE 243](#_Toc29364)

[2.3.60. VIEWS 243](#_Toc15262)

[2.3.61. \_PG\_FOREIGN\_DATA\_WRAPPERS 244](#_Toc30383)

[2.3.62. \_PG\_FOREIGN\_SERVERS 245](#_Toc6016)

[2.3.63. \_PG\_FOREIGN\_TABLE\_COLUMNS 246](#_Toc17580)

[2.3.64. \_PG\_FOREIGN\_TABLES 246](#_Toc15675)

[2.3.65. \_PG\_USER\_MAPPINGS 247](#_Toc1669)

[2.3.66. INFORMATION\_SCHEMA\_CATALOG\_NAME 247](#_Toc14388)

[2.4. DBE\_PLDEBUGGER Schema 248](#_Toc18227)

[2.4.1. DBE\_PLDEBUGGER.turn\_on 252](#_Toc2047)

[2.4.2. DBE\_PLDEBUGGER.turn\_off 253](#_Toc11895)

[2.4.3. DBE\_PLDEBUGGER.local\_debug\_server\_info 253](#_Toc10732)

[2.4.4. DBE\_PLDEBUGGER.attach 254](#_Toc11940)

[2.4.5. DBE\_PLDEBUGGER.info\_locals 254](#_Toc12323)

[2.4.6. DBE\_PLDEBUGGER.next 255](#_Toc24034)

[2.4.7. DBE\_PLDEBUGGER.continue 255](#_Toc23137)

[2.4.8. DBE\_PLDEBUGGER.abort 256](#_Toc15970)

[2.4.9. DBE\_PLDEBUGGER.print\_var 256](#_Toc1490)

[2.4.10. DBE\_PLDEBUGGER.info\_code 256](#_Toc5467)

[2.4.11. DBE\_PLDEBUGGER.step 257](#_Toc13477)

[2.4.12. DBE\_PLDEBUGGER.add\_breakpoint 257](#_Toc11977)

[2.4.13. DBE\_PLDEBUGGER.delete\_breakpoint 258](#_Toc13125)

[2.4.14. DBE\_PLDEBUGGER.info\_breakpoints 258](#_Toc7236)

[2.4.15. DBE\_PLDEBUGGER.backtrace 258](#_Toc20579)

[2.4.16. DBE\_PLDEBUGGER.enable\_breakpoint 259](#_Toc12133)

[2.4.17. DBE\_PLDEBUGGER.disable\_breakpoint 259](#_Toc1581)

[2.4.18. DBE\_PLDEBUGGER.finish 259](#_Toc13469)

[2.4.19. DBE\_PLDEBUGGER.set\_var 259](#_Toc9832)

[2.4.20. 附录 D. SQL 一致性 260](#_Toc22882)

[2.5. DBE\_PLDEVELOPER 261](#_Toc26705)

[2.5.1. DBE\_PLDEVELOPER.gs\_source 261](#_Toc16669)

[2.5.2. DBE\_PLDEVELOPER.gs\_errors 262](#_Toc28258)

[2.6. DB4AI Schema 262](#_Toc9796)

[2.6.1. DB4AI.SNAPSHOT 263](#_Toc19320)

[2.6.2. DB4AI.CREATE\_SNAPSHOT 263](#_Toc4481)

[2.6.3. DB4AI.CREATE\_SNAPSHOT\_INTERNAL 264](#_Toc2536)

[2.6.4. DB4AI.PREPARE\_SNAPSHOT 264](#_Toc16089)

[2.6.5. DB4AI.PREPARE\_SNAPSHOT\_INTERNAL 265](#_Toc3209)

[2.6.6. DB4AI.ARCHIVE\_SNAPSHOT 266](#_Toc18725)

[2.6.7. DB4AI.PUBLISH\_SNAPSHOT 266](#_Toc8379)

[2.6.8. DB4AI.MANAGE\_SNAPSHOT\_INTERNAL 266](#_Toc13931)

[2.6.9. DB4AI.SAMPLE\_SNAPSHOT 267](#_Toc4593)

[2.6.10. DB4AI.PURGE\_SNAPSHOT 267](#_Toc5073)

[2.6.11. DB4AI.PURGE\_SNAPSHOT\_INTERNAL 268](#_Toc21068)

[3. SQL语法参考 268](#_Toc1593)

[3.1. SQL基本要素 268](#_Toc10565)

[3.1.1. VastbaseSQL 268](#_Toc7381)

[3.1.2. 关键字 269](#_Toc8318)

[3.1.3. 数据类型 319](#_Toc23849)

[3.1.4. 常量与宏 395](#_Toc26664)

[3.1.5. 函数和操作符 397](#_Toc31832)

[3.1.6. 表达式 872](#_Toc30636)

[3.1.7. 事务控制 887](#_Toc6071)

[3.1.8. 系统操作 888](#_Toc5232)

[3.2. DML语法一览表 889](#_Toc3821)

[3.3. DDL语法一览表 891](#_Toc30702)

[3.4. DCL语法一览表 898](#_Toc29745)

[3.5. SQL语法 899](#_Toc22404)

[3.5.1. ABORT 899](#_Toc27495)

[3.5.2. ALTER AUDIT POLICY 901](#_Toc25496)

[3.5.3. ALTER AGGREGATE 903](#_Toc22310)

[3.5.4. ALTER DATA SOURCE 904](#_Toc16648)

[3.5.5. ALTER DATABASE 907](#_Toc27375)

[3.5.6. ALTER DEFAULT PRIVILEGES 909](#_Toc22276)

[3.5.7. ALTER DIRECTORY 914](#_Toc13340)

[3.5.8. ALTER EXTENSION 915](#_Toc23741)

[3.5.9. ALTER FOREIGN TABLE 918](#_Toc30547)

[3.5.10. ALTER FUNCTION 920](#_Toc11666)

[3.5.11. ALTER GLOBAL CONFIGURATION 924](#_Toc28289)

[3.5.12. ALTER GROUP 924](#_Toc24482)

[3.5.13. ALTER INDEX 926](#_Toc5025)

[3.5.14. ALTER LANGUAGE 928](#_Toc14316)

[3.5.15. ALTER LARGE OBJECT 929](#_Toc18148)

[3.5.16. ALTER MASKING POLICY 930](#_Toc12384)

[3.5.17. ALTER MATERIALIZED VIEW 933](#_Toc21237)

[3.5.18. ALTER PUBLICATION 934](#_Toc10127)

[3.5.19. ALTER RESOURCE LABEL 936](#_Toc10341)

[3.5.20. ALTER ROLE 937](#_Toc20196)

[3.5.21. ALTER ROW LEVEL SECURITY POLICY 940](#_Toc12687)

[3.5.22. ALTER PROCEDURE 942](#_Toc21383)

[3.5.23. ALTER SCHEMA 946](#_Toc27242)

[3.5.24. ALTER SEQUENCE 948](#_Toc24398)

[3.5.25. ALTER SERVER 950](#_Toc1064)

[3.5.26. ALTER SESSION 952](#_Toc14969)

[3.5.27. ALTER SUBSCRIPTION 954](#_Toc23188)

[3.5.28. ALTER SYNONYM 957](#_Toc23562)

[3.5.29. ALTER SYSTEM KILL SESSION 958](#_Toc29062)

[3.5.30. ALTER SYSTEM SET 959](#_Toc26087)

[3.5.31. ALTER TABLE 961](#_Toc8946)

[3.5.32. ALTER TABLE PARTITION 977](#_Toc13116)

[3.5.33. ALTER TABLE SUBPARTITION 985](#_Toc10831)

[3.5.34. ALTER TABLESPACE 989](#_Toc4784)

[3.5.35. ALTER TEXT SEARCH CONFIGURATION 991](#_Toc24692)

[3.5.36. ALTER TEXT SEARCH DICTIONARY 995](#_Toc21346)

[3.5.37. ALTER TRIGGER 997](#_Toc21275)

[3.5.38. ALTER TYPE 998](#_Toc17110)

[3.5.39. ALTER USER 1001](#_Toc32162)

[3.5.40. ALTER USER MAPPING 1004](#_Toc27445)

[3.5.41. ALTER VIEW 1006](#_Toc21708)

[3.5.42. ANALYZE和ANALYSE 1008](#_Toc7300)

[3.5.43. ANONYMOUS BLOCK 1013](#_Toc9613)

[3.5.44. BEGIN 1014](#_Toc24783)

[3.5.45. CALL 1016](#_Toc1826)

[3.5.46. CHECKPOINT 1019](#_Toc14928)

[3.5.47. CLEAN CONNECTION 1020](#_Toc25350)

[3.5.48. CLOSE 1022](#_Toc17427)

[3.5.49. CLUSTER 1023](#_Toc27018)

[3.5.50. COMMENT 1026](#_Toc20669)

[3.5.51. COMMIT | END 1029](#_Toc16399)

[3.5.52. COMMIT PREPARED 1031](#_Toc16458)

[3.5.53. COPY 1032](#_Toc24119)

[3.5.54. CREATE AGGREGATE 1049](#_Toc19716)

[3.5.55. CREATE AUDIT POLICY 1052](#_Toc31119)

[3.5.56. CREATE CAST 1054](#_Toc20746)

[3.5.57. CREATE CLIENT MASTER KEY 1056](#_Toc7028)

[3.5.58. CREATE COLUMN ENCRYPTION KEY 1058](#_Toc20756)

[3.5.59. CREATE DATA SOURCE 1059](#_Toc6986)

[3.5.60. CREATE DATABASE 1061](#_Toc15964)

[3.5.61. CREATE DATABASE LINK 1072](#_Toc18233)

[3.5.62. CREATE DIRECTORY 1075](#_Toc12503)

[3.5.63. CREATE EXTENSION 1076](#_Toc6159)

[3.5.64. CREATE FOREIGN TABLE 1078](#_Toc9245)

[3.5.65. CREATE FUNCTION 1081](#_Toc28924)

[3.5.66. CREATE GROUP 1089](#_Toc21687)

[3.5.67. CREATE INCREMENTAL MATERIALIZED VIEW 1090](#_Toc12851)

[3.5.68. CREATE INDEX 1092](#_Toc2213)

[3.5.69. CREATE MASKING POLICY 1103](#_Toc24009)

[3.5.70. CREATE MATERIALIZED VIEW 1106](#_Toc9569)

[3.5.71. CREATE MODEL 1107](#_Toc18495)

[3.5.72. CREATE OPERATOR 1109](#_Toc18087)

[3.5.73. CREATE PACKAGE 1112](#_Toc23189)

[3.5.74. CREATE PROCEDURE 1114](#_Toc25800)

[3.5.75. CREATE PUBLICATION 1118](#_Toc23014)

[3.5.76. CREATE RESOURCE LABEL 1119](#_Toc20056)

[3.5.77. CREATE ROLE 1121](#_Toc2256)

[3.5.78. CREATE ROW LEVEL SECURITY POLICY 1127](#_Toc6253)

[3.5.79. CREATE RULE 1132](#_Toc22842)

[3.5.80. CREATE SCHEMA 1134](#_Toc468)

[3.5.81. CREATE SECURITY LABEL 1136](#_Toc22841)

[3.5.82. CREATE SEQUENCE 1137](#_Toc26847)

[3.5.83. CREATE SERVER 1141](#_Toc7199)

[3.5.84. CREATE SUBSCRIPTION 1145](#_Toc12477)

[3.5.85. CREATE SYNONYM 1148](#_Toc10486)

[3.5.86. CREATE TABLE 1152](#_Toc6781)

[3.5.87. CREATE TABLE AS 1180](#_Toc511)

[3.5.88. CREATE TABLE PARTITION 1185](#_Toc30501)

[3.5.89. CREATE TABLE SUBPARTITION 1214](#_Toc13540)

[3.5.90. CREATE TABLESPACE 1235](#_Toc899)

[3.5.91. CREATE TEXT SEARCH CONFIGURATION 1239](#_Toc8639)

[3.5.92. CREATE TEXT SEARCH DICTIONARY 1241](#_Toc942)

[3.5.93. CREATE TRIGGER 1246](#_Toc22316)

[3.5.94. CREATE TYPE 1253](#_Toc9029)

[3.5.95. CREATE USER 1263](#_Toc5106)

[3.5.96. CREATE USER MAPPING 1266](#_Toc2287)

[3.5.97. CREATE VIEW 1268](#_Toc17175)

[3.5.98. CREATE WEAK PASSWORD DICTIONARY 1269](#_Toc16053)

[3.5.99. CURSOR 1271](#_Toc11597)

[3.5.100. DEALLOCATE 1272](#_Toc24090)

[3.5.101. DELETE 1273](#_Toc13091)

[3.5.102. DECLARE 1276](#_Toc9771)

[3.5.103. DO 1278](#_Toc15707)

[3.5.104. DROP AGGREGATE 1279](#_Toc4828)

[3.5.105. DROP AUDIT POLICY 1280](#_Toc15702)

[3.5.106. DROP CAST 1281](#_Toc2150)

[3.5.107. DROP CLIENT MASTER KEY 1282](#_Toc9586)

[3.5.108. DROP COLUMN ENCRYPTION KEY 1283](#_Toc31365)

[3.5.109. DROP DATA SOURCE 1284](#_Toc16850)

[3.5.110. DROP DATABASE 1285](#_Toc30414)

[3.5.111. DROP DATABASE LINK 1286](#_Toc23526)

[3.5.112. DROP DIRECTORY 1287](#_Toc29289)

[3.5.113. DROP EXTENSION 1288](#_Toc10154)

[3.5.114. DROP FOREIGN TABLE 1289](#_Toc32507)

[3.5.115. DROP FUNCTION 1289](#_Toc12156)

[3.5.116. DROP GLOBAL CONFIGURATION 1290](#_Toc26502)

[3.5.117. DROP GROUP 1291](#_Toc23540)

[3.5.118. DROP INDEX 1292](#_Toc32123)

[3.5.119. DROP MASKING POLICY 1293](#_Toc24354)

[3.5.120. DROP MATERIALIZED VIEW 1293](#_Toc24423)

[3.5.121. DROP MODEL 1294](#_Toc18207)

[3.5.122. DROP OPERATOR 1295](#_Toc3536)

[3.5.123. DROP OWNED 1295](#_Toc23005)

[3.5.124. DROP PACKAGE 1296](#_Toc2120)

[3.5.125. DROP PACKAGE BODY 1296](#_Toc11399)

[3.5.126. DROP PROCEDURE 1298](#_Toc15289)

[3.5.127. DROP PUBLICATION 1298](#_Toc15106)

[3.5.128. DROP RESOURCE LABEL 1299](#_Toc5242)

[3.5.129. DROP ROLE 1300](#_Toc10311)

[3.5.130. DROP ROW LEVEL SECURITY POLICY 1300](#_Toc22963)

[3.5.131. DROP RULE 1302](#_Toc22185)

[3.5.132. DROP SCHEMA 1302](#_Toc25865)

[3.5.133. DROP SECURITY LABEL 1303](#_Toc16206)

[3.5.134. DROP SEQUENCE 1304](#_Toc8382)

[3.5.135. DROP SERVER 1305](#_Toc27231)

[3.5.136. DROP SUBSCRIPTION 1306](#_Toc18333)

[3.5.137. DROP SYNONYM 1307](#_Toc13750)

[3.5.138. DROP TABLE 1307](#_Toc32100)

[3.5.139. DROP TABLESPACE 1308](#_Toc17947)

[3.5.140. DROP TEXT SEARCH CONFIGURATION 1309](#_Toc19358)

[3.5.141. DROP TEXT SEARCH DICTIONARY 1310](#_Toc8017)

[3.5.142. DROP TRIGGER 1311](#_Toc11768)

[3.5.143. DROP TYPE 1312](#_Toc372)

[3.5.144. DROP USER 1313](#_Toc29339)

[3.5.145. DROP USER MAPPING 1315](#_Toc24196)

[3.5.146. DROP VIEW 1316](#_Toc5660)

[3.5.147. DROP WEAK PASSWORD DICTIONARY 1316](#_Toc4368)

[3.5.148. EXECUTE 1317](#_Toc16365)

[3.5.149. EXPLAIN 1318](#_Toc1368)

[3.5.150. EXPLAIN PLAN 1326](#_Toc27525)

[3.5.151. FETCH 1328](#_Toc20576)

[3.5.152. GRANT 1334](#_Toc27543)

[3.5.153. MERGE 1350](#_Toc19662)

[3.5.154. INSERT 1354](#_Toc8195)

[3.5.155. LOCK 1360](#_Toc21386)

[3.5.156. MOVE 1365](#_Toc15473)

[3.5.157. PREDICT BY 1367](#_Toc5747)

[3.5.158. PREPARE 1367](#_Toc11451)

[3.5.159. PREPARE TRANSACTION 1368](#_Toc5543)

[3.5.160. PURGE 1370](#_Toc19846)

[3.5.161. REASSIGN OWNED 1372](#_Toc11317)

[3.5.162. REFRESH INCREMENTAL MATERIALIZED VIEW 1373](#_Toc8887)

[3.5.163. MATERIALIZED VIEW 1374](#_Toc22319)

[3.5.164. REINDEX 1375](#_Toc17082)

[3.5.165. RELEASE SAVEPOINT 1380](#_Toc12917)

[3.5.166. RESET 1381](#_Toc9378)

[3.5.167. REVOKE 1382](#_Toc16561)

[3.5.168. ROLLBACK 1388](#_Toc12464)

[3.5.169. ROLLBACK PREPARED 1389](#_Toc93)

[3.5.170. ROLLBACK TO SAVEPOINT 1390](#_Toc11724)

[3.5.171. SAVEPOINT 1391](#_Toc7624)

[3.5.172. SECURITY LABEL 1394](#_Toc15338)

[3.5.173. SELECT 1396](#_Toc21108)

[3.5.174. SELECT INTO 1410](#_Toc19061)

[3.5.175. SET 1413](#_Toc17482)

[3.5.176. SET CONSTRAINTS 1416](#_Toc13106)

[3.5.177. SET ROLE 1417](#_Toc2927)

[3.5.178. SET SESSION AUTHORIZATION 1418](#_Toc9543)

[3.5.179. SET TRANSACTION 1420](#_Toc31682)

[3.5.180. SHOW 1421](#_Toc10553)

[3.5.181. SHUTDOWN 1422](#_Toc30539)

[3.5.182. SNAPSHOT 1423](#_Toc3620)

[3.5.183. START TRANSACTION 1426](#_Toc31009)

[3.5.184. TIMECAPSULE TABLE 1428](#_Toc20863)

[3.5.185. TRUNCATE 1432](#_Toc17560)

[3.5.186. UPDATE 1435](#_Toc24098)

[3.5.187. VACUUM 1438](#_Toc17072)

[3.5.188. VALUES 1442](#_Toc9330)

[3.5.189. WITH FUNCTION 1443](#_Toc10264)

[3.6. 类型转换概述 1446](#_Toc23764)

[3.6.1. 操作符 1447](#_Toc4169)

[3.6.2. UNION,CASE和相关构造 1450](#_Toc3802)

[3.6.3. 函数 1454](#_Toc24200)

[3.6.4. 值存储 1457](#_Toc2764)

[3.7. 全文检索 1458](#_Toc31687)

[3.7.1. 介绍 1458](#_Toc12404)

[3.7.2. 全文检索概述 1458](#_Toc190)

[3.7.3. 表和索引 1464](#_Toc30045)

[3.7.4. 测试和调试文本搜索 1470](#_Toc5563)

[3.7.5. 词典 1474](#_Toc24052)

[3.7.6. 附加功能 1490](#_Toc30630)

[3.7.7. 控制文本搜索 1496](#_Toc12897)

[3.7.8. 解析器 1506](#_Toc22962)

[3.8. 聚合函数嵌套 1512](#_Toc20580)

[3.9. 普通表和分区表在线重建索引 1514](#_Toc3275)

[3.10. SQL并行查询 1517](#_Toc722)

[3.11. 附录 1524](#_Toc14582)

[3.11.1. GIN索引 1524](#_Toc16405)

[3.11.2. 扩展函数 1529](#_Toc6022)

[3.11.3. 扩展语法 1530](#_Toc20090)

[4. 系统表和系统视图 1531](#_Toc9908)

[4.1. 系统表和系统视图概述 1531](#_Toc18420)

[4.2. 系统表 1532](#_Toc17119)

[4.2.1. GS\_ASP 1532](#_Toc24781)

[4.2.2. GS\_AUDITING\_POLICY 1534](#_Toc6885)

[4.2.3. GS\_AUDITING\_POLICY\_ACCESS 1534](#_Toc12582)

[4.2.4. GS\_AUDITING\_POLICY\_FILTERS 1535](#_Toc24631)

[4.2.5. GS\_AUDITING\_POLICY\_PRIVILEGES 1535](#_Toc32566)

[4.2.6. GS\_CLIENT\_GLOBAL\_KEYS 1536](#_Toc22675)

[4.2.7. GS\_CLIENT\_GLOBAL\_KEYS\_ARGS 1536](#_Toc14493)

[4.2.8. GS\_COLUMN\_KEYS 1536](#_Toc26251)

[4.2.9. GS\_COLUMN\_KEYS\_ARGS 1537](#_Toc31568)

[4.2.10. GS\_DB\_PRIVILEGE 1537](#_Toc1855)

[4.2.11. GS\_ENCRYPTED\_COLUMNS 1538](#_Toc21225)

[4.2.12. GS\_ENCRYPTED\_PROC 1538](#_Toc30455)

[4.2.13. GS\_GLOBAL\_CHAIN 1539](#_Toc13657)

[4.2.14. GS\_GLOBAL\_CONFIG 1540](#_Toc31617)

[4.2.15. GS\_MASKING\_POLICY 1540](#_Toc25169)

[4.2.16. GS\_MASKING\_POLICY\_ACTIONS 1541](#_Toc2681)

[4.2.17. GS\_MASKING\_POLICY\_FILTERS 1541](#_Toc8932)

[4.2.18. GS\_MATVIEW 1542](#_Toc29422)

[4.2.19. GS\_MATVIEW\_DEPENDENCY 1542](#_Toc4899)

[4.2.20. GS\_MODEL\_WAREHOUSE 1543](#_Toc11512)

[4.2.21. GS\_OPT\_MODEL 1544](#_Toc11725)

[4.2.22. GS\_PACKAGE 1545](#_Toc18238)

[4.2.23. GS\_POLICY\_LABEL 1545](#_Toc21064)

[4.2.24. GS\_RECYCLEBIN 1546](#_Toc27013)

[4.2.25. GS\_TXN\_SNAPSHOT 1547](#_Toc32626)

[4.2.26. GS\_UID 1547](#_Toc23727)

[4.2.27. GS\_WLM\_EC\_OPERATOR\_INFO 1548](#_Toc4635)

[4.2.28. GS\_WLM\_INSTANCE\_HISTORY 1549](#_Toc19424)

[4.2.29. GS\_WLM\_OPERATOR\_INFO 1550](#_Toc3104)

[4.2.30. GS\_WLM\_PLAN\_ENCODING\_TABLE 1552](#_Toc20388)

[4.2.31. GS\_WLM\_PLAN\_OPERATOR\_INFO 1552](#_Toc20860)

[4.2.32. GS\_WLM\_SESSION\_QUERY\_INFO\_ALL 1553](#_Toc6974)

[4.2.33. GS\_WLM\_USER\_RESOURCE\_HISTORY 1558](#_Toc29388)

[4.2.34. PG\_AGGREGATE 1559](#_Toc9618)

[4.2.35. PG\_AM 1561](#_Toc6431)

[4.2.36. PG\_AMOP 1563](#_Toc12873)

[4.2.37. PG\_AMPROC 1564](#_Toc1284)

[4.2.38. PG\_APP\_WORKLOADGROUP\_MAPPING 1564](#_Toc1308)

[4.2.39. PG\_ATTRDEF 1565](#_Toc25386)

[4.2.40. PG\_ATTRIBUTE 1565](#_Toc6249)

[4.2.41. PG\_AUTH\_HISTORY 1567](#_Toc2853)

[4.2.42. PG\_AUTH\_MEMBERS 1567](#_Toc14717)

[4.2.43. PG\_AUTHID 1568](#_Toc8448)

[4.2.44. PG\_CAST 1570](#_Toc16450)

[4.2.45. PG\_CLASS 1571](#_Toc6349)

[4.2.46. PG\_COLLATION 1575](#_Toc1319)

[4.2.47. PG\_CONSTRAINT 1576](#_Toc21335)

[4.2.48. PG\_CONVERSION 1578](#_Toc3966)

[4.2.49. PG\_DATABASE 1578](#_Toc1427)

[4.2.50. PG\_DB\_ROLE\_SETTING 1580](#_Toc1961)

[4.2.51. PG\_DEFAULT\_ACL 1580](#_Toc26769)

[4.2.52. PG\_DEPEND 1581](#_Toc5068)

[4.2.53. PG\_DESCRIPTION 1582](#_Toc24054)

[4.2.54. PG\_DIRECTORY 1583](#_Toc15062)

[4.2.55. PG\_ENUM 1583](#_Toc21833)

[4.2.56. PG\_EXTENSION 1584](#_Toc2344)

[4.2.57. PG\_EXTENSION\_DATA\_SOURCE 1584](#_Toc24724)

[4.2.58. PG\_FOREIGN\_DATA\_WRAPPER 1585](#_Toc19236)

[4.2.59. PG\_FOREIGN\_SERVER 1586](#_Toc6956)

[4.2.60. PG\_FOREIGN\_TABLE 1587](#_Toc5470)

[4.2.61. PG\_HASHBUCKET 1587](#_Toc23453)

[4.2.62. PG\_INDEX 1588](#_Toc20306)

[4.2.63. PG\_INHERITS 1590](#_Toc12810)

[4.2.64. PG\_JOB 1590](#_Toc23753)

[4.2.65. PG\_JOB\_PROC 1592](#_Toc32390)

[4.2.66. PG\_LANGUAGE 1593](#_Toc9038)

[4.2.67. PG\_LARGEOBJECT 1594](#_Toc1836)

[4.2.68. PG\_LARGEOBJECT\_METADATA 1595](#_Toc21601)

[4.2.69. PG\_NAMESPACE 1595](#_Toc29698)

[4.2.70. PG\_OBJECT 1596](#_Toc27221)

[4.2.71. PG\_OPCLASS 1597](#_Toc1769)

[4.2.72. PG\_OPERATOR 1597](#_Toc29941)

[4.2.73. PG\_OPFAMILY 1599](#_Toc22574)

[4.2.74. PG\_PARTITION 1599](#_Toc31836)

[4.2.75. PG\_PLTEMPLATE 1601](#_Toc25716)

[4.2.76. PG\_PROC 1602](#_Toc13236)

[4.2.77. PG\_PUBLICATION 1605](#_Toc24331)

[4.2.78. PG\_PUBLICATION\_REL 1606](#_Toc17071)

[4.2.79. PG\_RANGE 1606](#_Toc5112)

[4.2.80. PG\_REPLICATION\_ORIGIN 1607](#_Toc25586)

[4.2.81. PG\_RESOURCE\_POOL 1607](#_Toc16178)

[4.2.82. PG\_REWRITE 1609](#_Toc18392)

[4.2.83. PG\_RLSPOLICY 1609](#_Toc10983)

[4.2.84. PG\_SECLABEL 1610](#_Toc26153)

[4.2.85. PG\_SHDEPEND 1610](#_Toc13507)

[4.2.86. PG\_SHDESCRIPTION 1612](#_Toc32450)

[4.2.87. PG\_SHSECLABEL 1612](#_Toc6799)

[4.2.88. PG\_STATISTIC 1613](#_Toc18494)

[4.2.89. PG\_STATISTIC\_EXT 1614](#_Toc12447)

[4.2.90. PG\_SUBSCRIPTION 1616](#_Toc1310)

[4.2.91. PG\_SYNONYM 1616](#_Toc30958)

[4.2.92. PG\_TABLESPACE 1617](#_Toc15110)

[4.2.93. PG\_TRIGGER 1617](#_Toc2007)

[4.2.94. PG\_TS\_CONFIG 1618](#_Toc25862)

[4.2.95. PG\_TS\_CONFIG\_MAP 1619](#_Toc19399)

[4.2.96. PG\_TS\_DICT 1619](#_Toc265)

[4.2.97. PG\_TS\_PARSER 1620](#_Toc14180)

[4.2.98. PG\_TS\_TEMPLATE 1620](#_Toc6624)

[4.2.99. PG\_TYPE 1621](#_Toc2549)

[4.2.100. PG\_USER\_MAPPING 1624](#_Toc14982)

[4.2.101. PG\_USER\_STATUS 1625](#_Toc4120)

[4.2.102. PG\_WORKLOAD\_GROUP 1625](#_Toc23686)

[4.2.103. PGXC\_CLASS 1626](#_Toc18176)

[4.2.104. PGXC\_GROUP 1626](#_Toc12983)

[4.2.105. PGXC\_NODE 1627](#_Toc24276)

[4.2.106. PGXC\_SLICE 1629](#_Toc31396)

[4.2.107. PLAN\_TABLE\_DATA 1629](#_Toc15598)

[4.2.108. STATEMENT\_HISTORY 1631](#_Toc4693)

[4.3. 系统视图 1634](#_Toc25852)

[4.3.1. GS\_AUDITING 1634](#_Toc15713)

[4.3.2. GS\_AUDITING\_ACCESS 1635](#_Toc10331)

[4.3.3. GS\_AUDITING\_PRIVILEGE 1636](#_Toc7007)

[4.3.4. GS\_DB\_PRIVILEGES 1636](#_Toc13608)

[4.3.5. GS\_FILE\_STAT 1637](#_Toc9025)

[4.3.6. GS\_GSC\_MEMORY\_DETAIL 1637](#_Toc10956)

[4.3.7. GS\_INSTANCE\_TIME 1638](#_Toc21016)

[4.3.8. GS\_LABELS 1639](#_Toc18460)

[4.3.9. GS\_LSC\_MEMORY\_DETAIL 1639](#_Toc29001)

[4.3.10. GS\_MASKING 1640](#_Toc10049)

[4.3.11. GS\_MATVIEWS 1640](#_Toc14159)

[4.3.12. GS\_OS\_RUN\_INFO 1641](#_Toc24059)

[4.3.13. GS\_REDO\_STAT 1641](#_Toc24421)

[4.3.14. GS\_SESSION\_CPU\_STATISTICS 1642](#_Toc19721)

[4.3.15. GS\_SESSION\_MEMORY 1642](#_Toc30690)

[4.3.16. GS\_SESSION\_MEMORY\_CONTEXT 1643](#_Toc32695)

[4.3.17. GS\_SESSION\_MEMORY\_DETAIL 1644](#_Toc25608)

[4.3.18. GS\_SESSION\_MEMORY\_STATISTICS 1645](#_Toc17626)

[4.3.19. GS\_SESSION\_STAT 1646](#_Toc13701)

[4.3.20. GS\_SESSION\_TIME 1646](#_Toc13516)

[4.3.21. GS\_SQL\_COUNT 1646](#_Toc25362)

[4.3.22. GS\_STAT\_SESSION\_CU 1649](#_Toc13345)

[4.3.23. GS\_THREAD\_MEMORY\_CONTEXT 1649](#_Toc19648)

[4.3.24. GS\_TOTAL\_MEMORY\_DETAIL 1650](#_Toc16686)

[4.3.25. GS\_WLM\_CGROUP\_INFO 1651](#_Toc17398)

[4.3.26. GS\_WLM\_EC\_OPERATOR\_STATISTICS 1652](#_Toc2711)

[4.3.27. GS\_WLM\_OPERATOR\_HISTORY 1653](#_Toc15415)

[4.3.28. GS\_WLM\_OPERATOR\_STATISTICS 1654](#_Toc32639)

[4.3.29. GS\_WLM\_PLAN\_OPERATOR\_HISTORY 1656](#_Toc13084)

[4.3.30. GS\_WLM\_REBUILD\_USER\_RESOURCE\_POOL 1656](#_Toc19989)

[4.3.31. GS\_WLM\_RESOURCE\_POOL 1656](#_Toc4398)

[4.3.32. GS\_WLM\_SESSION\_INFO 1657](#_Toc11142)

[4.3.33. GS\_WLM\_SESSION\_HISTORY 1657](#_Toc4740)

[4.3.34. GS\_WLM\_SESSION\_INFO\_ALL 1662](#_Toc22697)

[4.3.35. GS\_WLM\_SESSION\_STATISTICS 1667](#_Toc32076)

[4.3.36. GS\_WLM\_USER\_INFO 1670](#_Toc12509)

[4.3.37. MPP\_TABLES 1670](#_Toc23981)

[4.3.38. PG\_AVAILABLE\_EXTENSION\_VERSIONS 1671](#_Toc15593)

[4.3.39. PG\_AVAILABLE\_EXTENSIONS 1671](#_Toc31135)

[4.3.40. PG\_COMM\_DELAY 1672](#_Toc20969)

[4.3.41. PG\_COMM\_RECV\_STREAM 1673](#_Toc1205)

[4.3.42. PG\_COMM\_SEND\_STREAM 1674](#_Toc17543)

[4.3.43. PG\_COMM\_STATUS 1675](#_Toc6881)

[4.3.44. PG\_CONTROL\_GROUP\_CONFIG 1676](#_Toc15119)

[4.3.45. PG\_CURSORS 1676](#_Toc705)

[4.3.46. PG\_EXT\_STATS 1677](#_Toc15654)

[4.3.47. PG\_GET\_INVALID\_BACKENDS 1679](#_Toc31482)

[4.3.48. PG\_GET\_SENDERS\_CATCHUP\_TIME 1680](#_Toc22053)

[4.3.49. PG\_GROUP 1680](#_Toc13215)

[4.3.50. PG\_GTT\_ATTACHED\_PIDS 1680](#_Toc1900)

[4.3.51. PG\_GTT\_RELSTATS 1681](#_Toc23493)

[4.3.52. PG\_GTT\_STATS 1681](#_Toc29223)

[4.3.53. PG\_INDEXES 1682](#_Toc11434)

[4.3.54. PG\_LOCKS 1683](#_Toc7257)

[4.3.55. PG\_NODE\_ENV 1685](#_Toc26391)

[4.3.56. PG\_OS\_THREADS 1685](#_Toc4908)

[4.3.57. PG\_PREPARED\_STATEMENTS 1686](#_Toc7406)

[4.3.58. PG\_PREPARED\_XACTS 1686](#_Toc31959)

[4.3.59. PG\_PUBLICATION\_TABLES 1687](#_Toc12286)

[4.3.60. PG\_REPLICATION\_ORIGIN\_STATUS 1687](#_Toc32408)

[4.3.61. PG\_REPLICATION\_SLOTS 1688](#_Toc30861)

[4.3.62. PG\_RLSPOLICIES 1688](#_Toc16274)

[4.3.63. PG\_ROLES 1689](#_Toc22854)

[4.3.64. PG\_RULES 1691](#_Toc22418)

[4.3.65. PG\_RUNNING\_XACTS 1691](#_Toc29538)

[4.3.66. PG\_SECLABELS 1692](#_Toc12553)

[4.3.67. PG\_SESSION\_IOSTAT 1693](#_Toc1346)

[4.3.68. PG\_SESSION\_WLMSTAT 1694](#_Toc24505)

[4.3.69. PG\_SETTINGS 1695](#_Toc3811)

[4.3.70. PG\_SHADOW 1696](#_Toc14822)

[4.3.71. PG\_STAT\_ACTIVITY 1698](#_Toc29385)

[4.3.72. PG\_STAT\_ACTIVITY\_NG 1700](#_Toc12306)

[4.3.73. PG\_STAT\_ALL\_INDEXES 1702](#_Toc16204)

[4.3.74. PG\_STAT\_ALL\_TABLES 1703](#_Toc32650)

[4.3.75. PG\_STAT\_BAD\_BLOCK 1705](#_Toc27518)

[4.3.76. PG\_STAT\_BGWRITER 1706](#_Toc32325)

[4.3.77. PG\_STAT\_DATABASE 1707](#_Toc27642)

[4.3.78. PG\_STAT\_DATABASE\_CONFLICTS 1708](#_Toc29421)

[4.3.79. PG\_STAT\_REPLICATION 1709](#_Toc26452)

[4.3.80. PG\_STAT\_SUBSCRIPTION 1710](#_Toc16906)

[4.3.81. PG\_STAT\_SYS\_INDEXES 1711](#_Toc20244)

[4.3.82. PG\_STAT\_SYS\_TABLES 1712](#_Toc16805)

[4.3.83. PG\_STAT\_USER\_FUNCTIONS 1713](#_Toc2386)

[4.3.84. PG\_STAT\_USER\_INDEXES 1714](#_Toc20190)

[4.3.85. PG\_STAT\_USER\_TABLES 1714](#_Toc30833)

[4.3.86. PG\_STAT\_XACT\_ALL\_TABLES 1716](#_Toc23876)

[4.3.87. PG\_STAT\_XACT\_SYS\_TABLES 1716](#_Toc22932)

[4.3.88. PG\_STAT\_XACT\_USER\_FUNCTIONS 1717](#_Toc13818)

[4.3.89. PG\_STAT\_XACT\_USER\_TABLES 1718](#_Toc30646)

[4.3.90. PG\_STATIO\_ALL\_INDEXES 1718](#_Toc3303)

[4.3.91. PG\_STATIO\_ALL\_SEQUENCES 1719](#_Toc24723)

[4.3.92. PG\_STATIO\_ALL\_TABLES 1719](#_Toc1157)

[4.3.93. PG\_STATIO\_SYS\_INDEXES 1720](#_Toc29119)

[4.3.94. PG\_STATIO\_SYS\_SEQUENCES 1721](#_Toc15628)

[4.3.95. PG\_STATIO\_SYS\_TABLES 1721](#_Toc1958)

[4.3.96. PG\_STATIO\_USER\_INDEXES 1722](#_Toc12809)

[4.3.97. PG\_STATIO\_USER\_SEQUENCES 1722](#_Toc9537)

[4.3.98. PG\_STATIO\_USER\_TABLES 1723](#_Toc420)

[4.3.99. PG\_STATS 1724](#_Toc25288)

[4.3.100. PG\_TABLES 1726](#_Toc21284)

[4.3.101. PG\_TDE\_INFO 1727](#_Toc8877)

[4.3.102. PG\_THREAD\_WAIT\_STATUS 1727](#_Toc32362)

[4.3.103. PG\_TIMEZONE\_ABBREVS 1743](#_Toc18331)

[4.3.104. PG\_TIMEZONE\_NAMES 1744](#_Toc25728)

[4.3.105. PG\_TOTAL\_MEMORY\_DETAIL 1744](#_Toc15191)

[4.3.106. PG\_TOTAL\_USER\_RESOURCE\_INFO 1745](#_Toc10610)

[4.3.107. PG\_TOTAL\_USER\_RESOURCE\_INFO\_OID 1746](#_Toc21933)

[4.3.108. PG\_USER 1748](#_Toc16900)

[4.3.109. \_PG\_USER\_MAPPINGS 1749](#_Toc27607)

[4.3.110. PG\_VARIABLE\_INFO 1750](#_Toc7173)

[4.3.111. PG\_VIEWS 1751](#_Toc21498)

[4.3.112. PG\_WLM\_STATISTICS 1751](#_Toc7756)

[4.3.113. PGXC\_PREPARED\_XACTS 1752](#_Toc17839)

[4.3.114. PLAN\_TABLE 1752](#_Toc26028)

[4.3.115. STATEMENT\_HISTORY 1754](#_Toc1971)

[4.3.116. SUMMARY\_STATIO\_ALL\_INDEXES 1757](#_Toc3671)

[5. 物化视图 1757](#_Toc7215)

[5.1. 全量物化视图 1758](#_Toc5515)

[5.1.1. 概述 1758](#_Toc26686)

[5.1.2. 使用 1758](#_Toc32128)

[5.1.3. 支持和约束 1759](#_Toc30870)

[5.2. 增量物化视图 1760](#_Toc28144)

[5.2.1. 概述 1760](#_Toc21630)

[5.2.2. 使用 1760](#_Toc7600)

[5.2.3. 支持和约束 1762](#_Toc6522)

[6. PL/SQL 1762](#_Toc30252)

[6.1. 存储过程 1762](#_Toc11604)

[6.1.1. 动态语句 1762](#_Toc10193)

[6.1.2. 基本语句 1771](#_Toc362)

[6.1.3. 控制语句 1775](#_Toc24226)

[6.1.4. 锁操作 1792](#_Toc31078)

[6.1.5. 游标 1792](#_Toc134)

[6.1.6. record 1798](#_Toc22737)

[6.1.7. Retry管理 1802](#_Toc27421)

[6.1.8. 事务管理 1803](#_Toc30239)

[6.1.9. 数据类型 1814](#_Toc5401)

[6.1.10. 数组 1815](#_Toc10013)

[6.1.11. 调试 1816](#_Toc22375)

[6.1.12. 声明语法 1819](#_Toc14839)

[6.1.13. PL/pgSQL语言函数 1821](#_Toc10017)

[7. 并发控制 1822](#_Toc27159)

[7.1. 事务管理 1822](#_Toc19444)

[7.1.1. 管理事务 1822](#_Toc27093)

[7.1.2. 自治事务 1824](#_Toc15032)

[8. 插件管理 1830](#_Toc10666)

[8.1. PostGIS插件 1830](#_Toc21616)

[8.2. pg\_zhtrgm插件 1833](#_Toc22524)

[8.3. PGroonga插件 1837](#_Toc16812)

[8.4. uuid-ossp插件 1841](#_Toc26581)

[8.5. wal2json插件 1845](#_Toc19230)

[9. 外部数据封装器 1850](#_Toc24999)

[9.1. Oracle\_FDW 1850](#_Toc22486)

[9.2. DBLINK 1852](#_Toc5914)

[9.3. JDBC\_FDW 1855](#_Toc17992)

[9.4. MySQL\_FDW 1860](#_Toc14975)

[9.5. PostgreSQL\_FDW 1863](#_Toc16006)

[10. AI特性 1868](#_Toc7834)

[10.1. AI4DB数据库自治运维 1868](#_Toc20314)

[10.1.1. DBMIND模式说明 1869](#_Toc18900)

[10.1.2. DBMIND支持的组件 1876](#_Toc29989)

[10.1.3. DBMIND的AI子功能 1884](#_Toc8234)

[10.2. DB4AI: 数据库驱动AI 1913](#_Toc24740)

[10.2.1. 原生DB4AI引擎 1913](#_Toc16711)

[10.2.2. 全流程AI 1927](#_Toc17688)

[10.3. AI in DB: 数据库内AI功能 1938](#_Toc25768)

[10.3.1. 智能Explain: SQL语句查询时间预测 1938](#_Toc28764)

[11. GUC参数说明 1945](#_Toc3014)

[11.1. GUC使用说明 1945](#_Toc17640)

[11.2. 文件位置 1946](#_Toc25232)

[11.3. 连接和认证 1948](#_Toc11490)

[11.3.1. 连接设置 1948](#_Toc10430)

[11.3.2. 安全和认证（postgresql.conf） 1954](#_Toc5792)

[11.3.3. 通信库参数 1966](#_Toc4810)

[11.4. 资源消耗 1969](#_Toc20040)

[11.4.1. 内存 1969](#_Toc18315)

[11.4.2. 磁盘空间 1980](#_Toc25975)

[11.4.3. 内核资源使用 1981](#_Toc1990)

[11.4.4. 基于开销的清理延迟 1982](#_Toc8466)

[11.4.5. 后端写进程 1985](#_Toc14132)

[11.4.6. 异步IO 1989](#_Toc26287)

[11.5. 预写式日志 1994](#_Toc31435)

[11.5.1. 设置 1994](#_Toc11834)

[11.5.2. 检查点 2004](#_Toc32341)

[11.5.3. 日志回放 2008](#_Toc25021)

[11.5.4. 归档 2011](#_Toc17292)

[11.6. 双机复制 2014](#_Toc9336)

[11.6.1. 发送端服务器 2014](#_Toc12124)

[11.6.2. 主服务器 2022](#_Toc938)

[11.6.3. 备服务器 2031](#_Toc28157)

[11.7. 内存表 2036](#_Toc26304)

[11.8. 查询规划 2038](#_Toc19704)

[11.8.1. 优化器方法配置 2038](#_Toc441)

[11.8.2. 优化器开销常量 2050](#_Toc25188)

[11.8.3. 基因查询优化器 2053](#_Toc25611)

[11.8.4. 其他优化器选项 2056](#_Toc21199)

[11.9. 错误报告和日志 2079](#_Toc20298)

[11.9.1. 记录日志的位置 2079](#_Toc21427)

[11.9.2. 记录日志的时间 2085](#_Toc7258)

[11.9.3. 记录日志的内容 2089](#_Toc13936)

[11.9.4. 使用CSV格式写日志 2101](#_Toc27390)

[11.10. 告警检测 2103](#_Toc26645)

[11.11. 运行时统计 2106](#_Toc30456)

[11.11.1. 查询和索引统计收集器 2106](#_Toc29832)

[11.11.2. 性能统计 2110](#_Toc1820)

[11.12. 负载管理 2111](#_Toc20423)

[11.13. 自动清理 2123](#_Toc19968)

[11.14. 客户端连接缺省设置 2129](#_Toc32344)

[11.14.1. 语句行为 2129](#_Toc4404)

[11.14.2. 区域和格式化 2137](#_Toc15913)

[11.14.3. 其他缺省 2143](#_Toc21280)

[11.15. 锁管理 2145](#_Toc7014)

[11.16. 版本和平台兼容性 2151](#_Toc23810)

[11.16.1. 历史版本兼容性 2151](#_Toc30997)

[11.16.2. 平台和客户端兼容性 2155](#_Toc12700)

[11.17. 容错性 2164](#_Toc4979)

[11.18. 连接池参数 2168](#_Toc30356)

[11.19. 事务 2169](#_Toc29295)

[11.20. 双数据库实例复制参数 2173](#_Toc4502)

[11.21. 开发人员选项 2173](#_Toc13486)

[11.22. 审计 2184](#_Toc24495)

[11.22.1. 审计开关 2184](#_Toc8189)

[11.22.2. 用户和权限审计 2188](#_Toc24716)

[11.22.3. 操作审计 2190](#_Toc2599)

[11.23. 升级参数 2198](#_Toc15472)

[11.24. 其它选项 2199](#_Toc1401)

[11.25. 等待事件 2206](#_Toc802)

[11.26. Query 2207](#_Toc22150)

[11.27. 系统性能快照 2212](#_Toc13103)

[11.28. 安全配置 2215](#_Toc9445)

[11.29. 全局临时表 2218](#_Toc21538)

[11.30. HyperLogLog 2218](#_Toc9981)

[11.31. 用户自定义函数 2222](#_Toc936)

[11.32. 定时任务 2223](#_Toc18719)

[11.33. 线程池 2225](#_Toc5313)

[11.34. 备份恢复 2226](#_Toc4731)

[11.35. Undo 2227](#_Toc6493)

[11.36. DCF参数设置 2228](#_Toc24245)

[11.37. 闪回相关参数 2239](#_Toc12836)

[11.38. 回滚相关参数 2241](#_Toc31011)

[11.39. 预留参数 2241](#_Toc7478)

[11.40. AI特性 2241](#_Toc20218)

[11.41. Global SysCache参数 2243](#_Toc1058)

[11.42. date类型控制参数 2244](#_Toc19051)

[12. 配置运行参数 2246](#_Toc26956)

[12.1. 重设参数 2246](#_Toc1258)

[12.2. 查看参数当前取值 2253](#_Toc2723)

[13. 接口参考 2254](#_Toc26346)

[13.1. 应用程序接口 2254](#_Toc25642)

[14. 错误日志信息参考 2255](#_Toc26333)

[14.1. 内核错误信息 2255](#_Toc7518)

# 字符集

## 字符集列表

下表显示了可以在Vastbase中支持使用的字符集：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **描述** | **语言** | **服务端？** | **字节/字符** | **别名** |
| BIG5 | Big Five | 繁体中文 | 否 | 1-2 | WIN950, Windows950 |
| EUC\_CN | 扩展UNIX编码-中国 | 简体中文 | 是 | 1-3 |  |
| EUC\_JP | 扩展UNIX编码-日本 | 日文 | 是 | 1-3 |  |
| EUC\_JIS\_2004 | 扩展UNIX编码-日本, JIS X 0213 | 日文 | 是 | 1-3 |  |
| EUC\_KR | 扩展UNIX编码-韩国 | 韩文 | 是 | 1-3 |  |
| EUC\_TW | 扩展UNIX编码-台湾 | 繁体中文，台湾话 | 是 | 1-3 |  |
| GB18030 | 国家标准 | 中文 | 是 | 1-4 |  |
| GBK | 扩展国家标准 | 简体中文 | 是 | 1-2 | WIN936, Windows936 |
| ISO\_8859\_5 | ISO 8859-5, ECMA 113 | 拉丁语/西里尔语 | 是 | 1 |  |
| ISO\_8859\_6 | ISO 8859-6, ECMA 114 | 拉丁语/阿拉伯语 | 是 | 1 |  |
| ISO\_8859\_7 | ISO 8859-7, ECMA 118 | 拉丁语/希腊语 | 是 | 1 |  |
| ISO\_8859\_8 | ISO 8859-8, ECMA 121 | 拉丁语/希伯来语 | 是 | 1 |  |
| JOHAB | JOHAB | 韩语 | 否 | 1-3 |  |
| KOI8R | KOI8-R | 西里尔语（俄语） | 是 | 1 | KOI8 |
| KOI8U | KOI8-U | 西里尔语（乌克兰语） | 是 | 1 |  |
| LATIN1 | ISO 8859-1, ECMA 94 | 西欧 | 是 | 1 | ISO88591 |
| LATIN2 | ISO 8859-2, ECMA 94 | 中欧 | 是 | 1 | ISO88592 |
| LATIN3 | ISO 8859-3, ECMA 94 | 南欧 | 是 | 1 | ISO88593 |
| LATIN4 | ISO 8859-4, ECMA 94 | 北欧 | 是 | 1 | ISO88594 |
| LATIN5 | ISO 8859-9, ECMA 128 | 土耳其语 | 是 | 1 | ISO88599 |
| LATIN6 | ISO 8859-10, ECMA 144 | 日耳曼语 | 是 | 1 | ISO885910 |
| LATIN7 | ISO 8859-13 | 波罗的海 | 是 | 1 | ISO885913 |
| LATIN8 | ISO 8859-14 | 凯尔特语 | 是 | 1 | ISO885914 |
| LATIN9 | ISO 8859-15 | 带欧罗巴和口音的LATIN1 | 是 | 1 | ISO885915 |
| LATIN10 | ISO 8859-16, ASRO SR 14111 | 罗马尼亚语 | 是 | 1 | ISO885916 |
| MULE\_INTERNAL | Mule内部编码 | 多语种编辑器 | 是 | 1-4 |  |
| SJIS | Shift JIS | 日语 | 否 | 1-2 | Mskanji, ShiftJIS, WIN932, Windows932 |
| SHIFT\_JIS\_2004 | Shift JIS, JIS X 0213 | 日语 | 否 | 1-2 |  |
| SQL\_ASCII | 未指定（见文本） | 任意 | 是 | 1 |  |
| UHC | 统一韩语编码 | 韩语 | 否 | 1-2 | WIN949, Windows949 |
| UTF8 | Unicode, 8-bit | 所有 | 是 | 1-4 | Unicode |
| WIN866 | Windows CP866 | 西里尔语 | 是 | 1 | ALT |
| WIN874 | Windows CP874 | 泰语 | 是 | 1 |  |
| WIN1250250 | Windows CP1250 | 中欧 | 是 | 1 |  |
| WIN1251 | Windows CP1251 | 西里尔语 | 是 | 1 | WIN |
| WIN1252 | Windows CP1252 | 西欧 | 是 | 1 |  |
| WIN1253 | Windows CP1253 | 希腊语 | 是 | 1 |  |
| WIN1254 | Windows CP1254 | 土耳其语 | 是 | 1 |  |
| WIN1255 | Windows CP1255 | 希伯来语 | 是 | 1 |  |
| WIN1256 | Windows CP1256 | 阿拉伯语 | 是 | 1 |  |
| WIN1257 | Windows CP1257 | 波罗的海 | 是 | 1 |  |
| WIN1258 | Windows CP1258 | 越南语 | 是 | 1 | ABC, TCVN, TCVN5712, VSCII |

需要注意并非所有的客户端API都支持上面列出的字符集。

SQL\_ASCII设置与其他设置表现得相当不同。如果服务器字符集是SQL\_ASCII，服务器把字节值0-127根据ASCII标准解释，而字节值128-255则当作无法解析的字符。如果设置为SQL\_ASCII，就不会有编码转换。因此，这个设置基本不是用来声明所使用的指定编码，因为这个声明会忽略编码。在大多数情况下，如果你使用了任何非ASCII数据，那么使用SQL\_ASCII设置都是不明智的，因为Vastbase将无法帮助你转换或者校验非ASCII字符。image1

## 字符集设置

用户可以在数据库创建时指定非缺省编码，但是指定的编码必须与所选的区域相兼容：

createdb -E GB18030 -T template0 --lc-collate=zh\_CN.gb18030 --lc-ctype=zh\_CN.gb18030 testdb

将创建一个使用GB18030字符集以及zh\_CN区域的 名字叫testdb的数据库。另外一种实现方法是使用SQL命令：

CREATE DATABASE testdb WITH ENCODING 'GB18030' LC\_COLLATE='zh\_CN.gb18030' LC\_CTYPE='zh\_CN.gb18030' TEMPLATE=template0;

注意上述命令声明拷贝template0数据库。当拷贝任何其他数据库时， 来自源数据库的编码和区域设置不能被改变，因为可能导致数据损坏。

数据库的编码是存储在pg\_database系统表中的。 你可以用vsql的-l选项或\l命令列出这些编码。

# Schema

Schema又称作模式。通过管理Schema，允许多个用户使用同一数据库而不相互干扰，可以将数据库对象组织成易于管理的逻辑组，同时便于将第三方应用添加到相应的Schema下而不引起冲突。

每个数据库包含一个或多个Schema。数据库中的每个Schema包含表和其他类型的对象。数据库创建初始，默认具有一个名为public的Schema，且所有用户都拥有此Schema的usage权限，只有系统管理员和初始化用户可以在public Schema下创建函数、存储过程和同义词对象，其他用户即使赋予create权限后也不可以创建上述三种对象。可以通过Schema分组数据库对象。Schema类似于操作系统目录，但Schema不能嵌套。默认只有初始化用户可以在pg\_catalog模式下创建对象。

相同的数据库对象名称可以应用在同一数据库的不同Schema中，而没有冲突。例如，a\_schema和b\_schema都可以包含名为mytable的表。具有所需权限的用户可以访问数据库的多个Schema中的对象。

CREATE USER创建用户的同时，系统会在执行该命令的数据库中，为该用户创建一个同名的SCHEMA。

数据库对象是创建在数据库搜索路径中的第一个Schema内的。有关默认情况下的第一个Schema情况及如何变更Schema顺序等更多信息，请参见"搜索路径"。

**创建、修改和删除Schema**

* 要创建Schema，请使用CREATE SCHEMA（参考：SQL语法参考->SQL语法->CREATE SCHEMA章节）。默认初始用户和系统管理员可以创建Schema，其他用户需要具备数据库的CREATE权限才可以在该数据库中创建Schema，赋权方式请参考：SQL语法参考->SQL语法->GRANT章节中将数据库的访问权限赋予指定的用户或角色中的语法。
* 要更改Schema名称或者所有者，请使用ALTER SCHEMA（参考：SQL语法参考->SQL语法->ALTER SCHEMA章节）。Schema所有者可以更改Schema。
* 要删除Schema及其对象，请使用DROP SCHEMA（参考：SQL语法参考->SQL语法->DROP SCHEMA章节）。Schema所有者可以删除Schema。
* 要在Schema内创建表，请以schema\_name.table\_name格式创建表。不指定schema\_name时，对象默认创建到"搜索路径"中的第一个Schema内。
* 要查看Schema所有者，请对系统表PG\_NAMESPACE和PG\_USER执行如下关联查询。语句中的schema\_name请替换为实际要查找的Schema名称。

vastbase=# SELECT s.nspname,u.usename AS nspowner FROM pg\_namespace s, pg\_user u WHERE nspname='schema\_name' AND s.nspowner = u.usesysid;

* 要查看所有Schema的列表，请查询PG\_NAMESPACE系统表。

vastbase=# SELECT \* FROM pg\_namespace;

* 要查看属于某Schema下的表列表，请查询系统视图PG\_TABLES。例如，以下查询会返回Schema PG\_CATALOG中的表列表。

vastbase=# SELECT distinct(tablename),schemaname from pg\_tables where schemaname = 'pg\_catalog';

**搜索路径**

搜索路径定义在search\_path参数中参考：GUC参数说明->客户端连接缺省设置->语句行为章节，参数取值形式为采用逗号分隔的Schema名称列表。如果创建对象时未指定目标Schema，则该对象会被添加到搜索路径中列出的第一个Schema中。当不同Schema中存在同名的对象时，查询对象未指定Schema的情况下，将从搜索路径中包含该对象的第一个Schema中返回对象。

* 要查看当前搜索路径，请使用SHOW（参考：SQL语法参考->SQL语法->ALTER SCHEMA章节）。

vastbase=# SHOW SEARCH\_PATH;

search\_path

----------------

"$user",public

(1 row)

search\_path参数的默认值为：“$user”,public。$user表示与当前会话用户名同名的Schema名，如果这样的模式不存在，$user将被忽略。所以默认情况下，用户连接数据库后，如果数据库下存在同名Schema，则对象会添加到同名Schema下，否则对象被添加到Public Schema下。

* 要更改当前会话的默认Schema，请使用SET命令。
* 执行如下命令将搜索路径设置为myschema、public，首先搜索myschema。

vastbase=# SET SEARCH\_PATH TO myschema, public;

## WDR Snapshot Schema

WDR Snasphot在启动后（打开参数enable\_wdr\_snapshot 参考：GUC参数说明->系统性能快照），会在用户表空间"pg\_default"，数据库"postgres"下新建schema "snapshot"，用于持久化WDR快照数据。默认初始化用户或monadmin用户可以访问Snapshot Schema。

根据参数wdr\_snapshot\_retention\_days（ 参考：GUC参数说明->系统性能快照）来自动管理快照的生命周期。

WDR Snapshot 生成性能报告

**功能描述**

基于WDR Snapshot数据表汇总、统计，生成性能报告，默认初始化用户或监控管理员用户可以生成报告。

**操作步骤**

1、执行如下命令新建报告文件。

touch /home/vastbase/wdrTestNode.html

2、连接数据库。

3、执行如下命令查询已经生成的快照，以获取快照的snapshot\_id。

select \* from snapshot.snapshot;

4、执行如下命令手动创建快照。数据库中只有一个快照或者需要查看在当前时间段数据库的监控数据，可以选择手动执行快照操作，该命令需要用户具有sysadmin权限。（可选）

select create\_wdr\_snapshot();

执行“cm\_ctl query -Cdvi”，回显中“Central Coordinator State”下显示的信息即为CCN信息。

5、执行如下命令，在本地生成HTML格式的WDR报告。

（1）执行如下命令，设置报告格式。\a: 不显示表行列符号， \t: 不显示列名 ，\o: 指定输出文件。

\a

\t

\o /home/vastbase/wdrTestNode.html

（2）执行如下命令，生成HTML格式的WDR报告。

select generate\_wdr\_report(begin\_snap\_id Oid, end\_snap\_id Oid, int report\_type, int report\_scope, int node\_name );

示例一：生成集群级别的报告：

select generate\_wdr\_report(1, 2, 'all', 'cluster',null);

示例二：生成某个节点的报告：

select generate\_wdr\_report(1, 2, 'all', 'node', pgxc\_node\_str()::cstring);

（3）执行如下命令关闭输出选项及格式化输出命令。

\o \a \t

6、在/home/vastbase/下根据需要查看WDR报告。

generate\_wdr\_report函数参数说明如下表所示

| **参数** | **说明** | **取值范围** |
| --- | --- | --- |
| begin\_snap\_id | 查询时间段开始的snapshot的id（表snapshot.snapshot中的snapshot\_id）。 | - |
| end\_snap\_id | 查询时间段结束snapshot的id。默认end\_snap\_id大于begin\_snap\_id（表snapshot.snapshot中的snapshot\_id）。 | - |
| report\_type | 指定生成report的类型。例如，summary/detail/all。 | summary：汇总数据。detail：明细数据。all：包含summary和detail。 |
| report\_scope | 指定生成report的范围，可以为cluster或者node。 | cluster：数据库级别的信息。node：节点级别的信息。 |
| node\_name | 在report\_scope指定为node时，需要把该参数指定为对应节点的名称。（节点名称可以执行select \* from pg\_node\_env;查询）。在report\_scope为cluster时，该值可以省略或者指定为空或NULL。 | node：Vastbase中的节点名称。cluster：省略/空/NULL。 |

**注意事项**

* 需进入系统库查看快照信息。
* WDR Snapshot启动（即参数enable\_wdr\_snapshot为on时），且快照数量大于等于2。

**示例**

1、修改参数。

alter system set enable\_wdr\_snapshot=on;

2、创建报告文件。

touch /home/vastbase/wdrTestNode.html

3、登录数据库后切换至系统库。

\c postgres

4、查询已经生成的快照。

select \* from snapshot.snapshot;

结果显示如下：

snapshot\_id | start\_ts | end\_ts

-------------+-------------------------------+-------------------------------

432 | 2022-07-05 10:34:25.477173+08 | 2022-07-05 10:34:40.441504+08

435 | 2022-07-05 13:20:23.230771+08 | 2022-07-05 13:20:38.65221+08

5、生成格式化性能报告wdrTestNode.html。

\a \t \o /home/om/wdrTestNode.html

6、查询node节点名称

select \* from pg\_node\_env;

结果如下所示，node\_name 即为节点名称。

node\_name | host | process | port | installpath | datapath | log\_directory

-----------+-----------+---------+------+-------------------------------+------------------------------+---------------

node1 | localhost | 148191 | 5432 | /home/vastbase/local/vastbase | /home/vastbase/data/vastbase | pg\_log

(1 row)

7、向性能报告wdrTestNode.html中写入数据。

select generate\_wdr\_report(432, 435, 'all', 'node', 'node1');

8、关闭性能报告wdrTestNode.html。

postgres=# \o

9、生成格式化性能报告wdrTestCluster.html。

postgres=# \o /home/om/wdrTestCluster.html

10、向格式化性能报告wdrTestCluster.html中写入数据。

select generate\_wdr\_report(432, 435, 'all', 'cluster');

11、关闭性能报告wdrTestCluster.html。

postgres=# \o \a \t

### WDR Snapshot 原信息表

fig: **须知：** 用户应该禁止对Snapshot schema下的表进行增删改等操作，人为对这些表的修改或破坏可能会导致WDR各种异常情况甚至WDR不可用。

#### SNAPSHOT

SNAPSHOT表记录当前系统中存储的WDR快照数据的索引信息、开始时间和结束时间。只能在系统库中查询到结果，在用户库中无法查询。

**表 1** SNAPSHOT表属性

| **名称** | **类型** | **描述** | **示例** |
| --- | --- | --- | --- |
| snapshot\_id | bigint | WDR快照序号。 | 1 |
| start\_ts | timestamp | WDR快照的开始时间。 | 2019-12-28 17:11:27.423742+08 |
| end\_ts | timestamp | WDR快照的结束时间。 | 2019-12-28 17:11:43.67726+08 |

#### SNAPSHOT.TABLES\_SNAP\_TIMESTAMP

TABLES\_SNAP\_TIMESTAMP表记录所有存储的WDR snapshot中数据库、表对象、以及数据采集的开始和结束时间。

**表 1** TABLES\_SNAP\_TIMESTAMP表属性

| **名称** | **类型** | **描述** | **示例** |
| --- | --- | --- | --- |
| snapshot\_id | bigint | WDR快照序号。 | 1 |
| db\_name | text | WDR snapshot对应的database。 | tpcc1000 |
| tablename | text | WDR snapshot对应的table。 | snap\_xc\_statio\_all\_indexes |
| start\_ts | timestamp | WDR快照的开始时间。 | 2019-12-28 17:11:27.425849+08 |
| end\_ts | timestamp | WDR快照的结束时间。 | 2019-12-28 17:11:27.707398+08 |

### WDR Snapshot 数据表

WDR Snapshot数据表命名原则：snap\_{源数据表}。

WDR Snapshot数据表来源为DBE\_PERF Schema（参考：系统表和系统视图->schema->DBE\_PERF Schema章节）下的视图。

### 查看WDR报告

#### Database Stat

Database Stat列名称及描述如下表所示。

**表 1** Database Stat报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Name | 数据库名称。 |
| Backends | 连接到该数据库的后端数。 |
| Xact Commit | 此数据库中已经提交的事务数。 |
| Xact Rollback | 此数据库中已经回滚的事务数。 |
| Blks Read | 在这个数据库中读取的磁盘块的数量。 |
| Blks Hit | 高速缓存中已经发现的磁盘块的次数。 |
| Tuple Returned | 顺序扫描的行数。 |
| Tuple Fetched | 随机扫描的行数。 |
| Tuple Inserted | 通过数据库查询插入的行数。 |
| Tuple Updated | 通过数据库查询更新的行数。 |
| Tup Deleted | 通过数据库查询删除的行数。 |
| Conflicts | 由于数据库恢复冲突取消的查询数量。 |
| Temp Files | 通过数据库查询创建的临时文件数量。 |
| Temp Bytes | 通过数据库查询写入临时文件的数据总量。 |
| Deadlocks | 在该数据库中检索的死锁数。 |
| Blk Read Time | 通过数据库后端读取数据文件块花费的时间，以毫秒计算。 |
| Blk Write Time | 通过数据库后端写入数据文件块花费的时间，以毫秒计算。 |
| Stats Reset | 重置当前状态统计的时间。 |

#### Load Profile

Load Profile指标名称及描述如下表所示。

**表 1** Load Profile报表主要内容

| **指标名称** | **描述** |
| --- | --- |
| DB Time(us) | 作业运行的elapse time总和。 |
| CPU Time(us) | 作业运行的CPU时间总和。 |
| Redo size(blocks) | 产生的WAL的大小（块数）。 |
| Logical read (blocks) | 表或者索引文件的逻辑读（块数）。 |
| Physical read (blocks) | 表或者索引的物理读（块数）。 |
| Physical write (blocks) | 表或者索引的物理写（块数）。 |
| Read IO requests | 表或者索引的读次数。 |
| Write IO requests | 表或者索引的写次数。 |
| Read IO (MB) | 表或者索引的读大小（MB）。 |
| Write IO (MB) | 表或者索引的写大小（MB）。 |
| Logons | 登录次数。 |
| Executes (SQL) | SQL执行次数。 |
| Rollbacks | 回滚事务数。 |
| Transactions | 事务数。 |
| SQL response time P95(us) | 95%的SQL的响应时间。 |
| SQL response time P80(us) | 80%的SQL的响应时间。 |

#### Instance Efficiency Percentages

Instance Efficiency Percentages指标名称及描述如下表所示。

**表 1** Instance Efficiency Percentages报表主要内容

| **指标名称** | **描述** |
| --- | --- |
| Buffer Hit % | Buffer Pool命中率。 |
| Effective CPU % | CPU time占DB time的比例。 |
| WalWrite NoWait % | 访问WAL Buffer的event次数占总wait event的比例。 |
| Soft Parse % | 软解析的次数占总的解析次数的比例。 |
| Non-Parse CPU % | 非parse的时间占执行总时间的比例。 |

#### IO Profile

IO Profile指标名称及描述如下表所示。

**表 1** IO Profile指标表主要内容

| **指标名称** | **描述** |
| --- | --- |
| Database requests | Database IO次数。 |
| Database (MB) | Database IO数据量。 |
| Database (blocks) | Database IO数据块。 |
| Redo requests | Redo IO次数。 |
| Redo (MB) | Redo IO量。 |

#### SQL Statistics

SQL Statistics列名称及描述如下表所示。

**表 1** SQL Statistics报表主要内容

| **列名称** | **描述** |
| --- | --- |
| Unique SQL Id | 归一化的SQL ID。 |
| Node Name | 节点名称。 |
| User Name | 用户名称。 |
| Tuples Read | 访问的元组数量。 |
| Calls | 调用次数。 |
| Min Elapse Time(us) | 最小执行时间（us）。 |
| Max Elapse Time(us) | 最大执行时间（us）。 |
| Total Elapse Time(us) | 总执行时间（us）。 |
| Avg Elapse Time(us) | 平均执行时间（us）。 |
| Returned Rows | SELECT返回行数。 |
| Tuples Affected | Insert/Update/Delete行数。 |
| Logical Read | Buffer逻辑读次数。 |
| Physical Read | Buffer物理读次数。 |
| CPU Time(us) | CPU时间（us）。 |
| Data IO Time(us) | IO上的时间花费（us）。 |
| Sort Count | 排序执行的次数。 |
| Sort Time(us) | 排序执行的时间（us）。 |
| Sort Mem Used(KB) | 排序过程中使用的work memory大小（KB）。 |
| Sort Spill Count | 排序过程中，若发生落盘，写文件的次数。 |
| Sort Spill Size(KB) | 排序过程中，若发生落盘，使用的文件大小（KB）。 |
| Hash Count | hash执行的次数。 |
| Hash Time(us) | hash执行的时间（us）。 |
| Hash Mem Used(KB) | hash过程中使用的work memory大小（KB）。 |
| Hash Spill Count | hash过程中，若发生落盘，写文件的次数。 |
| Hash Spill Size(KB) | hash过程中，若发生落盘，使用的文件大小（KB）。 |
| SQL Text | 归一化SQL字符串。 |

#### Cache IO Stats

Cache IO Stats包含User table和User index两张表，列名称及描述如下所示。

* **User table IO activity ordered by heap blks hit ratio**

**表 1** User table IO activity ordered by heap blks hit ratio报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Name | Database名称。 |
| Schema Name | Schema名称。 |
| Table Name | Table名称。 |
| %Heap Blks Hit Ratio | 此表的Buffer Pool命中率。 |
| Heap Blks Read | 该表中读取的磁盘块数。 |
| Heap Blks Hit | 此表缓存命中数。 |
| Idx Blks Read | 表中所有索引读取的磁盘块数。 |
| Idx Blks Hit | 表中所有索引命中缓存数。 |
| Toast Blks Read | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| Toast Blks Hit | 此表的TOAST表命中缓冲区数（如果存在）。 |
| Tidx Blks Read | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| Tidx Blks Hit | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

* **User index IO activity ordered by idx blks hit ratio**

**表 2** User index IO activity ordered by idx blks hit ratio报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Name | Database名称。 |
| Schema Name | Schema名称。 |
| Table Name | Table名称。 |
| Index Name | Index名称。 |
| %Idx Blks Hit Ratio | Index的命中率。 |
| Idx Blks Read | 所有索引读取的磁盘块数。 |
| Idx Blks Hit | 所有索引命中缓存数。 |

#### Object stats

Object stats包含User Tables stats、User index stats和Bad lock stats三张表，列名称及描述如下所示。

* **User Tables stats**

**表 1** User Tables stats报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Name | Database名称。 |
| Schema | Schema名称。 |
| Relname | Relation名称。 |
| Seq Scan | 此表发起的顺序扫描数。 |
| Seq Tup Read | 顺序扫描抓取的活跃行数。 |
| Index Scan | 此表发起的索引扫描数。 |
| Index Tup Fetch | 索引扫描抓取的活跃行数。 |
| Tuple Insert | 插入行数。 |
| Tuple Update | 更新行数。 |
| Tuple Delete | 删除行数。 |
| Tuple Hot Update | HOT更新行数（即没有更新所需的单独索引）。 |
| Live Tuple | 估计活跃行数。 |
| Dead Tuple | 估计死行数。 |
| Last Vacuum | 最后一次此表是手动清理的（不计算VACUUM FULL）时间。 |
| Last Autovacuum | 上次被autovacuum守护进程清理的时间。 |
| Last Analyze | 上次手动分析这个表的时间。 |
| Last Autoanalyze | 上次被autovacuum守护进程分析的时间。 |
| Vacuum Count | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| Autovacuum Count | 这个表被autovacuum清理的次数。 |
| Analyze Count | 这个表被手动分析的次数。 |
| Autoanalyze Count | 这个表被autovacuum守护进程分析的次数。 |

* **User index stats**

**表 2** User index stats报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Name | Database名称。 |
| Schema | Schema名称。 |
| Relname | Relation名称。 |
| Index Relname | Index名称。 |
| Index Scan | 索引上开始的索引扫描数。 |
| Index Tuple Read | 通过索引上扫描返回的索引项数。 |
| Index Tuple Fetch | 通过使用索引的简单索引扫描抓取的表行数。 |

* **Bad lock stats**

**表 3** Bad lock stats报表主要内容

| **列名称** | **描述** |
| --- | --- |
| DB Id | 数据库的OID。 |
| Tablespace Id | 表空间的OID。 |
| Relfilenode | 文件对象ID。 |
| Fork Number | 文件类型。 |
| Error Count | 失败计数。 |
| First Time | 第一次发生时间。 |
| Last Time | 最近一次发生时间。 |

#### SQL Detail

SQL Detail列名称及描述如下表所示。

**表 1** SQL Detail报表主要内容

| **列名称** | **描述** |
| --- | --- |
| Unique SQL Id | 归一化SQL ID。 |
| User Name | 用户名称。 |
| Node Name | 节点名称。Node模式下不显示该字段。 |
| SQL Text | 归一化SQL文本。 |

## DBE\_PERF-Schema

DBE\_PERF Schema内视图主要用来诊断性能问题，也是WDR Snapshot的数据来源。数据库安装后，默认只有初始用户具有模式dbe\_perf的权限。若是由旧版本升级而来，为保持权限的前向兼容，模式dbe\_perf的权限与旧版本保持一致。从OS、Instance、Memory等多个维度划分组织视图，并且符合如下命名规范：

* GLOBAL\_开头的视图，代表从数据库节点请求数据，并将数据追加对外返回，不会处理数据。
* SUMMARY\_开头的视图，代表是将Vastbase内的数据概述，多数情况下是返回数据库节点（有时只有数据库主节点的）的数据，会对数据进行加工和汇聚。
* 非这两者开头的视图，一般代表本地视图，不会向其它数据库节点请求数据。

### OS

#### OS\_RUNTIME

显示当前操作系统运行的状态信息。

**表 1** OS\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| id | integer | 编号。 |
| name | text | 操作系统运行状态名称。 |
| value | numeric | 操作系统运行状态值。 |
| comments | text | 操作系统运行状态注释。 |
| cumulative | boolean | 操作系统运行状态的值是否为累加值。 |

#### GLOBAL\_OS\_RUNTIME

提供Vastbase中所有正常节点下的操作系统运行状态信息。

**表 1** GLOBAL\_OS\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| id | integer | 编号。 |
| name | text | 操作系统运行状态名称。 |
| value | numeric | 操作系统运行状态值。 |
| comments | text | 操作系统运行状态注释。 |
| cumulative | boolean | 操作系统运行状态的值是否为累加值。 |

#### OS\_THREADS

提供当前节点下所有线程的状态信息。

**表 1** OS\_THREADS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| pid | bigint | 数据库进程中正在运行的线程号。 |
| lwpid | integer | 与pid对应的轻量级线程号。 |
| thread\_name | text | 与pid对应的线程名称。 |
| creation\_time | timestamp with time zone | 与pid对应的线程创建的时间。 |

#### GLOBAL\_OS\_THREADS

提供Vastbase中所有正常节点下的线程状态信息。

**表 1** GLOBAL\_OS\_THREADS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| pid | bigint | 当前节点进程中正在运行的线程号。 |
| lwpid | integer | 与pid对应的轻量级线程号。 |
| thread\_name | text | 与pid对应的线程名称。 |
| creation\_time | timestamp with time zone | 与pid对应的线程创建的时间。 |

### Instance

#### INSTANCE\_TIME

提供当前节点下的各种时间消耗信息，主要分为以下类型：

* DB\_TIME：作业在多核下的有效时间花销。
* CPU\_TIME：CPU的时间花销。
* EXECUTION\_TIME：执行器内的时间花销。
* PARSE\_TIME：SQL解析的时间花销。
* PLAN\_TIME：生成Plan的时间花销。
* REWRITE\_TIME：SQL重写的时间花销。
* PL\_EXECUTION\_TIME：plpgsql（存储过程）执行的时间花销。
* PL\_COMPILATION\_TIME：plpgsql（存储过程）编译的时间花销。
* NET\_SEND\_TIME：网络上的时间花销。
* DATA\_IO\_TIME：IO上的时间花销。

**表 1** INSTANCE\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 类型名称。 |
| value | bigint | 时间值（单位：微秒）。 |

#### GLOBAL\_INSTANCE\_TIME

提供Vastbase中所有正常节点下的各种时间消耗信息（时间类型见instance\_time视图）。

**表 1** GLOBAL\_INSTANCE\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程的名称。 |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 类型名称。 |
| value | bigint | 时间值（单位：微秒）。 |

### Memory

#### MEMORY\_NODE\_DETAIL

显示某个数据库节点内存使用情况。

**表 1** MEMORY\_NODE\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 数据库进程名称。 |
| memorytype | text | 内存使用的名称。  - max\_process\_memory：Vastbase实例所占用的内存大小。  - process\_used\_memory：进程所使用的内存大小。  - max\_dynamic\_memory：最大动态内存。  - dynamic\_used\_memory：已使用的动态内存。  - dynamic\_peak\_memory：内存的动态峰值。  - dynamic\_used\_shrctx：最大动态共享内存上下文。  - dynamic\_peak\_shrctx：共享内存上下文的动态峰值。  - max\_shared\_memory：最大共享内存。  - shared\_used\_memory：已使用的共享内存。  - max\_cstore\_memory：列存所允许使用的最大内存。  - cstore\_used\_memory：列存已使用的内存大小。  - max\_sctpcomm\_memory：sctp通信所允许使用的最大内存。  - sctpcomm\_used\_memory：sctp通信已使用的内存大小。  - sctpcomm\_peak\_memory：sctp通信的内存峰值。  - other\_used\_memory：其他已使用的内存大小。  - gpu\_max\_dynamic\_memory：GPU最大动态内存。  - gpu\_dynamic\_used\_memory：GPU已使用的动态内存。  - gpu\_dynamic\_peak\_memory：GPU内存的动态峰值。  - pooler\_conn\_memory：链接池申请内存计数。  - pooler\_freeconn\_memory：链接池空闲连接的内存计数。  - storage\_compress\_memory：存储模块压缩使用的内存大小。  - udf\_reserved\_memory：UDF预留的内存大小。  - min\_dynamic\_memory：最小动态内存。  - max\_backend\_memory：后端进程可用最大内存。  - backend\_used\_memory：后端进程使用的内存大小。 |
| memorymbytes | integer | 内存使用的大小，单位为MB。 |

#### GLOBAL\_MEMORY\_NODE\_DETAIL

显示当前Vastbase中所有正常节点下的内存使用情况。

**表 1** GLOBAL\_MEMORY\_NODE\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 数据库进程名称。 |
| memorytype | text | 内存使用的名称。  - max\_process\_memory：Vastbase实例所占用的内存大小。  - process\_used\_memory：进程所使用的内存大小。  - max\_dynamic\_memory：最大动态内存。  - dynamic\_used\_memory：已使用的动态内存。  - dynamic\_peak\_memory：内存的动态峰值。  - dynamic\_used\_shrctx：最大动态共享内存上下文。  - dynamic\_peak\_shrctx：共享内存上下文的动态峰值。  - max\_shared\_memory：最大共享内存。  - shared\_used\_memory：已使用的共享内存。  - max\_cstore\_memory：列存所允许使用的最大内存。  - cstore\_used\_memory：列存已使用的内存大小。  - max\_sctpcomm\_memory：sctp通信所允许使用的最大内存。  - sctpcomm\_used\_memory：sctp通信已使用的内存大小。  - sctpcomm\_peak\_memory：sctp通信的内存峰值。  - other\_used\_memory：其他已使用的内存大小。  - gpu\_max\_dynamic\_memory：GPU最大动态内存。  - gpu\_dynamic\_used\_memory：GPU已使用的动态内存。  - gpu\_dynamic\_peak\_memory：GPU内存的动态峰值。  - pooler\_conn\_memory：链接池申请内存计数。  - pooler\_freeconn\_memory：链接池空闲连接的内存计数。  - storage\_compress\_memory：存储模块压缩使用的内存大小。  - udf\_reserved\_memory：UDF预留的内存大小。  - min\_dynamic\_memory：最小动态内存。  - max\_backend\_memory：后端进程可用最大内存。  - backend\_used\_memory：后端进程使用的内存大小。 |
| memorymbytes | integer | 内存使用的大小，单位为MB。 |

#### GS\_SHARED\_MEMORY\_DETAIL

查询当前节点所有已产生的共享内存上下文的使用信息。

**表 1** GS\_SHARED\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| contextname | text | 内存上下文的名称。 |
| level | smallint | 内存上下文的级别。 |
| parent | text | 上级内存上下文。 |
| totalsize | bigint | 共享内存总大小（单位：字节）。 |
| freesize | bigint | 共享内存剩余大小（单位：字节）。 |
| usedsize | bigint | 共享内存使用大小（单位：字节）。 |

#### GLOBAL\_SHARED\_MEMORY\_DETAIL

查询Vastbase中所有正常节点下的共享内存上下文的使用信息。

**表 1** GLOBAL\_SHARED\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| contextname | text | 内存上下文的名称。 |
| level | smallint | 内存上下文的级别。 |
| parent | text | 上级内存上下文。 |
| totalsize | bigint | 共享内存总大小（单位：字节）。 |
| freesize | bigint | 共享内存剩余大小（单位：字节）。 |
| usedsize | bigint | 共享内存使用大小（单位：字节）。 |

### File

#### FILE\_IOSTAT

通过对数据文件IO的统计，反映数据的IO性能，用以发现IO操作异常等性能问题。

**表 1** FILE\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| filenum | oid | 文件标识。 |
| dbid | oid | 数据库标识。 |
| spcid | oid | 表空间标识。 |
| phyrds | bigint | 读物理文件的数目。 |
| phywrts | bigint | 写物理文件的数目。 |
| phyblkrd | bigint | 读物理文件块的数目。 |
| phyblkwrt | bigint | 写物理文件块的数目。 |
| readtim | bigint | 读文件的总时长（单位：微秒）。 |
| writetim | bigint | 写文件的总时长（单位：微秒）。 |
| avgiotim | bigint | 读写文件的平均时长（单位：微秒）。 |
| lstiotim | bigint | 最后一次读文件时长（单位：微秒）。 |
| miniotim | bigint | 读写文件的最小时长（单位：微秒）。 |
| maxiowtm | bigint | 读写文件的最大时长（单位：微秒）。 |

#### SUMMARY\_FILE\_IOSTAT

通过Vastbase内对数据文件汇聚IO的统计，反映数据的IO性能，用以发现IO操作异常等性能问题。

**表 1** SUMMARY\_FILE\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| filenum | oid | 文件标识。 |
| dbid | oid | 数据库标识。 |
| spcid | oid | 表空间标识。 |
| phyrds | numeric | 读物理文件的数目。 |
| phywrts | numeric | 写物理文件的数目。 |
| phyblkrd | numeric | 读物理文件块的数目。 |
| phyblkwrt | numeric | 写物理文件块的数目。 |
| readtim | numeric | 读文件的总时长（单位：微秒）。 |
| writetim | numeric | 写文件的总时长（单位：微秒）。 |
| avgiotim | bigint | 读写文件的平均时长（单位：微秒）。 |
| lstiotim | bigint | 最后一次读文件时长（单位：微秒）。 |
| miniotim | bigint | 读写文件的最小时长（单位：微秒）。 |
| maxiowtm | bigint | 读写文件的最大时长（单位：微秒）。 |

#### GLOBAL\_FILE\_IOSTAT

得到所有节点上的数据文件IO统计信息。

**表 1** GLOBAL\_FILE\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| filenum | oid | 文件标识。 |
| dbid | oid | 数据库标识。 |
| spcid | oid | 表空间标识。 |
| phyrds | bigint | 读物理文件的数目。 |
| phywrts | bigint | 写物理文件的数目。 |
| phyblkrd | bigint | 读物理文件块的数目。 |
| phyblkwrt | bigint | 写物理文件块的数目。 |
| readtim | bigint | 读文件的总时长（单位：微秒）。 |
| writetim | bigint | 写文件的总时长（单位：微秒）。 |
| avgiotim | bigint | 读写文件的平均时长（单位：微秒）。 |
| lstiotim | bigint | 最后一次读文件时长（单位：微秒）。 |
| miniotim | bigint | 读写文件的最小时长（单位：微秒）。 |
| maxiowtm | bigint | 读写文件的最大时长（单位：微秒）。 |

#### FILE\_REDO\_IOSTAT

本节点Redo（WAL）相关的统计信息。

**表 1** FILE\_REDO\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| phywrts | bigint | 向wal buffer中写的次数。 |
| phyblkwrt | bigint | 向wal buffer中写的block的块数。 |
| writetim | bigint | 向xlog文件中写操作的时间（单位：微秒）。 |
| avgiotim | bigint | 平均写xlog的时间（writetim/phywrts）（单位：微秒）。 |
| lstiotim | bigint | 最后一次写xlog的时间（单位：微秒）。 |
| miniotim | bigint | 最小的写xlog时间（单位：微秒）。 |
| maxiowtm | bigint | 最大的写xlog时间（单位：微秒）。 |

#### SUMMARY\_FILE\_REDO\_IOSTAT

Vastbase内汇聚所有的Redo（WAL）相关的统计信息。

**表 1** SUMMARY\_FILE\_REDO\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| phywrts | numeric | 向wal buffer中写的次数。 |
| phyblkwrt | numeric | 向wal buffer中写的block的块数。 |
| writetim | numeric | 向xlog文件中写操作的时间（单位：微秒）。 |
| avgiotim | bigint | 平均写xlog的时间（writetim/phywrts）（单位：微秒）。 |
| lstiotim | bigint | 最后一次写xlog的时间（单位：微秒）。 |
| miniotim | bigint | 最小的写xlog时间（单位：微秒）。 |
| maxiowtm | bigint | 最大的写xlog时间（单位：微秒）。 |

#### GLOBAL\_FILE\_REDO\_IOSTAT

得到Vastbase内各节点的Redo（WAL）相关统计信息。

**表 1** GLOBALXC\_FILE\_REDO\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| phywrts | bigint | 向wal buffer中写的次数。 |
| phyblkwrt | bigint | 向wal buffer中写的block的块数。 |
| writetim | bigint | 向xlog文件中写操作的时间（单位：微秒）。 |
| avgiotim | bigint | 平均写xlog的时间（writetim/phywrts）（单位：微秒）。 |
| lstiotim | bigint | 最后一次写xlog的时间（单位：微秒）。 |
| miniotim | bigint | 最小的写xlog时间（单位：微秒）。 |
| maxiowtm | bigint | 最大的写xlog时间（单位：微秒）。 |

#### LOCAL\_REL\_IOSTAT

获取当前节点中数据文件IO状态的累计值，显示为所有数据文件IO状态的总和。

**表 1** LOCAL\_REL\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| phyrds | bigint | 读物理文件的数目。 |
| phywrts | bigint | 写物理文件的数目。 |
| phyblkrd | bigint | 读物理文件的块的数目。 |
| phyblkwrt | bigint | 写物理文件的块的数目。 |

#### GLOBAL\_REL\_IOSTAT

获取所有节点上的数据文件IO统计信息。

**表 1** GLOBAL\_REL\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| phyrds | bigint | 读物理文件的数目。 |
| phywrts | bigint | 写物理文件的数目。 |
| phyblkrd | bigint | 读物理文件块的数目。 |
| phyblkwrt | bigint | 写物理文件块的数目。 |

#### SUMMARY\_REL\_IOSTAT

获取所有节点上的数据文件IO统计信息。

**表 1** SUMMARY\_REL\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| phyrds | numeric | 读物理文件的数目。 |
| phywrts | numeric | 写物理文件的数目。 |
| phyblkrd | numeric | 读物理文件的块的数目。 |
| phyblkwrt | numeric | 写物理文件的块的数目。 |

### Transaction

#### TRANSACTIONS\_PREPARED\_XACTS

显示当前准备好进行两阶段提交的事务的信息。

**表 1** TRANSACTIONS\_PREPARED\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| transaction | xid | 预备事务的数字事务标识。 |
| gid | text | 赋予该事务的全局事务标识。 |
| prepared | timestamp with time zone | 事务准备好提交的时间。 |
| owner | name | 执行该事务的用户的名称。 |
| database | name | 执行该事务所在的数据库名。 |

#### SUMMARY\_TRANSACTIONS\_PREPARED\_XACTS

显示Vastbase中数据库主节点当前准备好进行两阶段提交的事务的信息。

**表 1** SUMMARY\_TRANSACTIONS\_PREPARED\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| transaction | xid | 预备事务的数字事务标识。 |
| gid | text | 赋予该事务的全局事务标识。 |
| prepared | timestamp with time zone | 事务准备好提交的时间。 |
| owner | name | 执行该事务的用户的名称。 |
| database | name | 执行该事务所在的数据库名。 |

#### GLOBAL\_TRANSACTIONS\_PREPARED\_XACTS

显示各节点当前准备好进行两阶段提交的事务的信息。

**表 1** GLOBAL\_TRANSACTIONS\_PREPARED\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| transaction | xid | 预备事务的数字事务标识。 |
| gid | text | 赋予该事务的全局事务标识。 |
| prepared | timestamp with time zone | 事务准备好提交的时间。 |
| owner | name | 执行该事务的用户的名称。 |
| database | name | 执行该事务所在的数据库名。 |

#### TRANSACTIONS\_RUNNING\_XACTS

显示当前节点运行事务的信息。

**表 1** TRANSACTIONS\_RUNNING\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| handle | integer | 事务对应的事务管理器中的槽位句柄，该值恒为-1。 |
| gxid | xid | 事务id号。 |
| state | tinyint | 事务状态（3：prepared或者0：starting）。 |
| node | text | 节点名称。 |
| xmin | xid | 节点上当前数据涉及的最小事务号xmin。 |
| vacuum | boolean | 标志当前事务是否是lazy vacuum事务。 |
| timeline | bigint | 标志数据库重启次数。 |
| prepare\_xid | xid | 处于prepared状态的事务的id号，若不在prepared状态，值为0。 |
| pid | bigint | 事务对应的线程id。 |
| next\_xid | xid | 其余节点发送给当前节点的事务id，该值恒为0。 |

#### SUMMARY\_TRANSACTIONS\_RUNNING\_XACTS

显示集群中各个节点运行事务的信息，字段内容和transactions\_running\_xacts一致。

**表 1** SUMMARY\_TRANSACTIONS\_RUNNING\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| handle | integer | 事务对应的事务管理器中的槽位句柄，该值恒为-1。 |
| gxid | xid | 事务id号。 |
| state | tinyint | 事务状态（3：prepared或者0：starting）。 |
| node | text | 节点名称。 |
| xmin | xid | 节点上当前数据涉及的最小事务号xmin。 |
| vacuum | boolean | 标志当前事务是否是lazy vacuum事务。 |
| timeline | bigint | 标志数据库重启次数。 |
| prepare\_xid | xid | 处于prepared状态的事务的id号，若不在prepared状态，值为0。 |
| pid | bigint | 事务对应的线程id。 |
| next\_xid | xid | 其余节点发送给当前节点的事务id，该值恒为0。 |

#### GLOBAL\_TRANSACTIONS\_RUNNING\_XACTS

显示集群中各个节点运行事务的信息。

**表 1** GLOBAL\_TRANSACTIONS\_RUNNING\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| handle | integer | 事务对应的事务管理器中的槽位句柄，该值恒为-1 |
| gxid | xid | 事务id号。 |
| state | tinyint | 事务状态（3：prepared或者0：starting）。 |
| node | text | 节点名称。 |
| xmin | xid | 节点上当前数据涉及的最小事务号xmin。 |
| vacuum | boolean | 标志当前事务是否是lazy vacuum事务。 |
| timeline | bigint | 标志数据库重启次数。 |
| prepare\_xid | xid | 处于prepared状态的事务的id号，若不在prepared状态，值为0。 |
| pid | bigint | 事务对应的线程id。 |
| next\_xid | xid | 其余节点发送给当前节点的事务id，该值恒为0。 |

### Session/Thread

#### GLOBAL\_SESSION\_STAT\_ACTIVITY

显示Vastbase内各节点上正在运行的线程相关的信息。

**表 1** GLOBAL\_SESSION\_STAT\_ACTIVITY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| coorname | text | 数据库进程名称。 |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| datname | text | 用户会话在后台连接到的数据库名称。 |
| pid | bigint | 后台线程ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| usename | text | 登录该后台的用户名。 |
| application\_name | text | 连接到该后台的应用名。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| backend\_start | timestampwith time zone | 该过程开始的时间，即当客户端连接服务器时间。 |
| xact\_start | timestampwith time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestampwith time zone | 开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。 |
| state\_change | timestampwith time zone | 上次状态改变的时间。 |
| waiting | boolean | 如果后台当前正等待锁则为true。 |
| enqueue | text | 该字段不支持。 |
| state | text | 该后台当前总体状态。可能值是：   * active：后台正在执行一个查询。 * idle：后台正在等待一个新的客户端命令。 * idle in transaction：后台在事务中，但是目前无法执行查询。 * idle in transaction (aborted)：这个状态除说明事务中有某个语句导致了错误外，类似于idle in transaction * fastpath function call：后台正在执行一个fast-path函数。 * disabled：如果后台禁用track\_activities，则报告这个状态。 * 说明：普通用户只能查看到自己帐户所对应的会话状态。即其他帐户的state信息为空。例如以judy用户连接数据库后，在pg\_stat\_activity中查看到的普通用户joe及初始用户vastbase的state信息为空。   vastbase=# SELECT datname, usename, usesysid,state,pid FROM pg\_stat\_activity;  datname | usename | usesysid | state | pid  ———-+———+———-+——–+—————–postgres | vastbase | 10 | |139968752121616  postgres | vastbase | 10 | |139968903116560  db\_tpcds | judy | 16398 | active |139968391403280  postgres | vastbase | 10 | |139968643069712  postgres | vastbase | 10 | |139968680818448  postgres | joe | 16390 | |139968563377936  (6 rows) |
| resource\_pool | name | 用户使用的资源池。 |
| query\_id | bigint | 查询语句的ID。 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| unique\_sql\_id | bigint | 语句的unique sql id。 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

#### SESSION\_STAT\_ACTIVITY

显示当前节点上正在运行的线程相关的信息。

**表 1** SESSION\_STAT\_ACTIVITY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| datname | name | 用户会话在后台连接到的数据库名称。 |
| pid | bigint | 后台线程ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| usename | name | 登录该后台的用户名。 |
| application\_name | text | 连接到该后台的应用名。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| backend\_start | timestampwith time zone | 该过程开始的时间，即当客户端连接服务器时间。 |
| xact\_start | timestampwith time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestampwith time zone | 开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。 |
| state\_change | timestampwith time zone | 上次状态改变的时间。 |
| waiting | boolean | 如果后台当前正等待锁则为true。 |
| enqueue | text | 该字段不支持。 |
| state | text | 该后台当前总体状态。可能值是：   * active：后台正在执行一个查询。 * idle：后台正在等待一个新的客户端命令。 * idle in transaction：后台在事务中，但是目前无法执行查询。 * idle in transaction (aborted)：这个状态除说明事务中有某个语句导致了错误外，类似于idle in transaction * fastpath function call：后台正在执行一个fast-path函数。 * disabled：如果后台禁用track\_activities，则报告这个状态。   说明：disabled：如果后台禁用track\_activities，则报告这个状态。  vastbase=# SELECT datname, usename, usesysid,state,pid FROM pg\_stat\_activity;  datname | usename | usesysid | state | pid  ———-+———+———-+——–+—————–postgres | vastbase | 10 | |139968752121616  postgres | vastbase | 10 | |139968903116560  db\_tpcds | judy | 16398 | active |139968391403280  postgres | vastbase | 10 | |139968643069712  postgres | vastbase | 10 | |139968680818448  postgres | joe | 16390 | |139968563377936  (6 rows) |
| resource\_pool | name | 用户使用的资源池。 |
| query\_id | bigint | 查询语句的ID。 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| unique\_sql\_id | bigint | 语句的unique sql id。 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

#### GLOBAL\_THREADPOOL\_STATUS

GLOBAL\_THREADPOOL\_STATUS视图显示在所有节点上的线程池中工作线程及会话的状态信息。具体的字段详见[表1](LOCAL_THREADPOOL_STATUS.md#zh-cn_topic_0237122641_zh-cn_topic_0059778133_tc25f02433de2419f8da4d0a8c2c8e562)。（章节LOCAL\_THREADPOOL\_STATUS）

#### LOCAL\_THREADPOOL\_STATUS

LOCAL\_THREADPOOL\_STATUS视图显示线程池下工作线程及会话的状态信息。该视图仅在线程池开启（enable\_thread\_pool = on）时生效。

**表 1** LOCAL\_THREADPOOL\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| group\_id | integer | 线程池组ID。 |
| bind\_numa\_id | integer | 该线程池组绑定的NUMA ID。 |
| bind\_cpu\_number | integer | 该线程池组绑定的CPU信息。如果未绑定CPU，该值为NULL。 |
| listener | integer | 该线程池组的Listener线程数量。 |
| worker\_info | text | 线程池中线程相关信息，包括以下信息：   * default：该线程池组中的初始线程数量。 * new：该线程池组中新增线程的数量。 * expect：该线程池组中预期线程的数量。 * actual：该线程池组中实际线程的数量。 * idle：该线程池组中空闲线程的数量。 * pending：该线程池组中等待线程的数量。 |
| session\_info | text | 线程池中会话相关信息，包括以下信息：   * total：该线程池组中所有的会话数量。 * waiting：该线程池组中等待调度的会话数量。 * running：该线程池中正在执行的会话数量。 * idle：该线程池组中空闲的会话数量。 |
| stream\_info | text | stream池相关信息，包含以下信息：   * total：该stream池组中所有的线程数量。 * running：该stream池中正在执行的线程数量。 * idle：该stream池组中空闲的线程数量。 |

#### SESSION\_STAT

当前节点以会话线程或AutoVacuum线程为单位，统计会话状态信息。

**表 1** SESSION\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| statid | integer | 统计编号。 |
| statname | text | 统计会话名称。 |
| statunit | text | 统计会话单位。 |
| value | bigint | 统计会话值。 |

#### GLOBAL\_SESSION\_STAT

各节点上以会话线程或AutoVacuum线程为单位，统计会话状态信息。

**表 1** GLOBAL\_SESSION\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| sessid | text | 线程启动时间+线程标识。 |
| statid | integer | 统计编号。 |
| statname | text | 统计会话名称。 |
| statunit | text | 统计会话单位。 |
| value | bigint | 统计会话值。 |

#### SESSION\_TIME

用于统计当前节点会话线程的运行时间信息，及各执行阶段所消耗时间。

**表 1** SESSION\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 会话类型名称。 |
| value | bigint | 会话值。 |

#### GLOBAL\_SESSION\_TIME

用于统计各节点会话线程的运行时间信息，及各执行阶段所消耗时间。

**表 1** GLOBAL\_SESSION\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| sessid | text | 线程启动时间+线程标识。 |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 会话类型名称。 |
| value | bigint | 会话值。 |

#### SESSION\_MEMORY

统计Session级别的内存使用情况，包含执行作业在数据节点上Vastbase线程和Stream线程分配的所有内存，单位为MB。

**表 1** SESSION\_MEMORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| init\_mem | integer | 当前正在执行作业进入执行器前已分配的内存。 |
| used\_mem | integer | 当前正在执行作业已分配的内存。 |
| peak\_mem | integer | 当前正在执行作业已分配的内存峰值。 |

#### GLOBAL\_SESSION\_MEMORY

统计各节点的Session级别的内存使用情况，包含执行作业在数据节点上Vastbase线程和Stream线程分配的所有内存，单位为MB。

**表 1** GLOBAL\_SESSION\_MEMORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| sessid | text | 线程启动时间+线程标识。 |
| init\_mem | integer | 当前正在执行作业进入执行器前已分配的内存。 |
| used\_mem | integer | 当前正在执行作业已分配的内存。 |
| peak\_mem | integer | 当前正在执行作业已分配的内存峰值。 |

#### SESSION\_MEMORY\_DETAIL

统计线程的内存使用情况，以MemoryContext节点来统计。

**表 1** SESSION\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| sesstype | text | 线程名称。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 内存上下文的重要级别。 |
| parent | text | 父级内存上下文名称。 |
| totalsize | bigint | 总申请内存大小（单位：字节）。 |
| freesize | bigint | 空闲内存大小（单位：字节）。 |
| usedsize | bigint | 使用内存大小（单位：字节）。 |

#### GLOBAL\_SESSION\_MEMORY\_DETAIL

统计各节点的线程的内存使用情况，以MemoryContext节点来统计。

**表 1** GLOBAL\_SESSION\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| sessid | text | 线程启动时间+线程标识。 |
| sesstype | text | 线程名称。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 内存上下文的重要级别。 |
| parent | text | 父级内存上下文名称。 |
| totalsize | bigint | 总申请内存大小（单位：字节）。 |
| freesize | bigint | 空闲内存大小（单位：字节）。 |
| usedsize | bigint | 使用内存大小（单位：字节）。 |

#### THREAD\_WAIT\_STATUS

通过该视图可以检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况，具体事件信息请参考：系统表和系统视图->系统视图->PG\_THREAD\_WAIT\_STATUS章节中表2 等待状态列表、表3 轻量级锁等待事件列表、表4 IO等待事件列表和表5 事务锁等待事件列表。

**表 1** THREAD\_WAIT\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| db\_name | text | 数据库名称。 |
| thread\_name | text | 线程名称。 |
| query\_id | bigint | 查询ID，对应debug\_query\_id。 |
| tid | bigint | 当前线程的线程号。 |
| sessionid | bigint | session的ID。 |
| lwtid | integer | 当前线程的轻量级线程号。 |
| psessionid | bigint | streaming线程的父线程。 |
| tlevel | integer | streaming线程的层级。 |
| smpid | integer | 并行线程的ID。 |
| wait\_status | text | 当前线程的等待状态。 |
| wait\_event | text | 如果wait\_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则为空。 |

#### GLOBAL\_THREAD\_WAIT\_STATUS

通过该视图可以检测所有节点上工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。具体事件信息请参考：系统表和系统视图->系统视图->PG\_THREAD\_WAIT\_STATUS章节中表2 等待状态列表、表3 轻量级锁等待事件列表、表4 IO等待事件列表和表5 事务锁等待事件列表。

通过GLOBAL\_THREAD\_WAIT\_STATUS视图，可以查看Vastbase全局各个节点上所有SQL语句产生的线程之间的调用层次关系，以及各个线程的阻塞等待状态，从而更容易定位hang以及类似现象的原因。

GLOBAL\_THREAD\_WAIT\_STATUS视图和THREAD\_WAIT\_STATUS视图列定义完全相同，这是由于GLOBAL\_THREAD\_WAIT\_STATUS视图本质是到Vastbase中各个节点上查询THREAD\_WAIT\_STATUS视图汇总的结果。

**表 1** GLOBAL\_THREAD\_WAIT\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| db\_name | text | 数据库名称。 |
| thread\_name | text | 线程名称。 |
| query\_id | bigint | 查询ID，对应debug\_query\_id。 |
| tid | bigint | 当前线程的线程号。 |
| sessionid | bigint | session的ID。 |
| lwtid | integer | 当前线程的轻量级线程号。 |
| psessionid | bigint | streaming线程的父线程。 |
| tlevel | integer | streaming线程的层级。 |
| smpid | integer | 并行线程的ID。 |
| wait\_status | text | 当前线程的等待状态。 |
| wait\_event | text | 如果wait\_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则是空。 |

#### SESSION\_CPU\_RUNTIME

SESSION\_CPU\_RUNTIME视图显示和当前用户执行复杂作业（正在运行）时的负载管理CPU使用的信息。

**表 1** SESSION\_CPU\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| usename | name | 登录到该后端的用户名。 |
| pid | bigint | 后端线程ID。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位为ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位为ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位为ms。 |
| query | text | 正在执行的语句。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |

#### SESSION\_MEMORY\_RUNTIME

SESSION\_MEMORY\_RUNTIME视图显示和当前用户执行复杂作业正在运行时的负载管理内存使用的信息。

**表 1** SESSION\_MEMORY\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| usename | name | 登录到该后端的用户名。 |
| pid | bigint | 后端线程ID。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位为ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位为ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位为ms。 |
| query | text | 正在执行的语句。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |

#### STATEMENT\_IOSTAT\_COMPLEX\_RUNTIME

STATEMENT\_IOSTAT\_COMPLEX\_RUNTIME视图显示当前用户执行作业正在运行时的IO负载管理相关信息。以下涉及到iops，对于行存，均以万次/s为单位，对于列存，均以次/s为单位。

**表 1** STATEMENT\_IOSTAT\_COMPLEX\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| query\_id | bigint | 作业id。 |
| mincurriops | integer | 该作业当前io在各数据库节点中的最小值。 |
| maxcurriops | integer | 该作业当前io在各数据库节点中的最大值。 |
| minpeakiops | integer | 在作业运行时，作业io峰值中，各数据库节点的最小值。 |
| maxpeakiops | integer | 在作业运行时，作业io峰值中，各数据库节点的最大值。 |
| io\_limits | integer | 该作业所设GUC参数io\_limits。 |
| io\_priority | text | 该作业所设GUC参数io\_priority。 |
| query | text | 作业。 |
| node\_group | text | 作业所属用户对应的逻辑Vastbase。 |
| curr\_io\_limits | integer | 使用io\_priority管控io时的实时io\_limits值。 |

#### LOCAL\_ACTIVE\_SESSION

LOCAL\_ACTIVE\_SESSION视图显示本节点上的ACTIVE SESSION PROFILE内存中的样本。

**表 1** LOCAL\_ACTIVE\_SESSION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sampleid | bigint | 采样ID。 |
| sample\_time | timestamp with time zone | 采样的时间。 |
| need\_flush\_sample | boolean | 该样本是否需要刷新的磁盘。 |
| databaseid | oid | 数据库ID。 |
| thread\_id | bigint | 线程的ID。 |
| sessionid | bigint | 会话的ID。 |
| start\_time | timestamp with time zone | 会话的启动时间。 |
| event | text | 具体的事件名称。 |
| lwtid | integer | 当前线程的轻量级线程号。 |
| psessionid | bigint | streaming线程的父线程。 |
| tlevel | integer | streaming线程的层级。与执行计划的层级（ID）相对应。 |
| smpid | integer | smp执行模式下并行线程的并行编号。 |
| userid | oid | session用户的ID。 |
| application\_name | text | 应用的名称。 |
| client\_addr | inet | client端的地址。 |
| client\_hostname | text | client端的名称。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号。 |
| query\_id | bigint | debug query id。 |
| unique\_query\_id | bigint | unique query id。 |
| user\_id | oid | unique query的key中的user\_id。 |
| cn\_id | integer | cn id，在DN上表示下发该unique sql的节点id，unique query的key中的cn\_id。 |
| unique\_query | text | 规范化后的UniqueSQL文本串。 |
| locktag | text | 会话等待锁信息，可通过locktag\_decode解析。 |
| lockmode | text | 会话等待锁模式。 |
| block\_sessionid | bigint | 如果会话正在等待锁，阻塞该会话获取锁的会话标识。 |
| final\_block\_sessionid | bigint | 表示源头阻塞会话ID。 |
| wait\_status | text | 描述event列的更多详细信息。 |
| global\_sessionid | text | 全局会话ID。 |

### Query

#### STATEMENT\_COMPLEX\_HISTORY

STATEMENT\_COMPLEX\_HISTORY视图显示在数据库主节点上执行作业结束后的负载管理记录。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| dbname | text | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 数据库进程名称 |
| username | text | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| block\_time | bigint | 语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| finish\_time | timestamp with time zone | 语句执行的结束时间。 |
| duration | bigint | 语句实际执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句预估执行时间，单位ms。 |
| status | text | 语句执行结束状态：正常为finished，异常为aborted。 |
| abort\_info | text | 语句执行结束状态为aborted时显示异常信息。 |
| resource\_pool | text | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句预估使用内存。 |
| min\_peak\_memory | integer | 语句在数据库节点上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库节点上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句数据库节点间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库节点上的下盘信息：None：数据库节点均未下盘。All：数据库节点均下盘。[a:b]：数量为b个数据库节点中有a个数据库节点下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库节点上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库节点上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库节点上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库节点的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库节点间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库节点上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库节点上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库节点上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库节点间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Spill file size large than 256MBBroadcast size large than 100MBEarly spillSpill times is greater than 3Spill on memory adaptiveHash table conflict |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑vastbase。 |
| cpu\_top1\_node\_name | text | cpu使用率第1的节点名称。 |
| cpu\_top2\_node\_name | text | cpu使用率第2的节点名称。 |
| cpu\_top3\_node\_name | text | cpu使用率第3的节点名称。 |
| cpu\_top4\_node\_name | text | cpu使用率第4的节点名称。 |
| cpu\_top5\_node\_name | text | cpu使用率第5的节点名称。 |
| mem\_top1\_node\_name | text | 内存使用量第1的节点名称。 |
| mem\_top2\_node\_name | text | 内存使用量第2的节点名称。 |
| mem\_top3\_node\_name | text | 内存使用量第3的节点名称。 |
| mem\_top4\_node\_name | text | 内存使用量第4的节点名称。 |
| mem\_top5\_node\_name | text | 内存使用量第5的节点名称。 |
| cpu\_top1\_value | bigint | cpu使用率。 |
| cpu\_top2\_value | bigint | cpu使用率。 |
| cpu\_top3\_value | bigint | cpu使用率。 |
| cpu\_top4\_value | bigint | cpu使用率。 |
| cpu\_top5\_value | bigint | cpu使用率。 |
| mem\_top1\_value | bigint | 内存使用量。 |
| mem\_top2\_value | bigint | 内存使用量。 |
| mem\_top3\_value | bigint | 内存使用量。 |
| mem\_top4\_value | bigint | 内存使用量。 |
| mem\_top5\_value | bigint | 内存使用量。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |

#### SUMMARY\_STATEMENT

获得各数据库主节点的执行语句（归一化SQL）的全量信息（包含数据库节点）。

**表 1** SUMMARY\_STATEMENT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| node\_id | integer | 节点的ID。 |
| user\_name | name | 用户名称。 |
| user\_id | oid | 用户OID。 |
| unique\_sql\_id | bigint | 归一化的SQL ID。 |
| query | text | 归一化的SQL。备注：长度受track\_activity\_query\_size控制。 |
| n\_calls | bigint | 调用次数。 |
| min\_elapse\_time | bigint | SQL在内核内的最小运行时间（单位：微秒）。 |
| max\_elapse\_time | bigint | SQL在内核内的最大运行时间（单位：微秒）。 |
| total\_elapse\_time | bigint | SQL在内核内的总运行时间（单位：微秒）。 |
| n\_returned\_rows | bigint | SELECT返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行。 |
| n\_tuples\_returned | bigint | 顺序扫描行。 |
| n\_tuples\_inserted | bigint | 插入行。 |
| n\_tuples\_updated | bigint | 更新行。 |
| n\_tuples\_deleted | bigint | 删除行。 |
| n\_blocks\_fetched | bigint | buffer的块访问次数。 |
| n\_blocks\_hit | bigint | buffer的块命中次数。 |
| n\_soft\_parse | bigint | 软解析次数。 |
| n\_hard\_parse | bigint | 硬解析次数。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 |
| net\_send\_info | text | 通过物理连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_recv\_info | text | 通过物理连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_send\_info | text | 通过逻辑连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_recv\_info | text | 通过逻辑连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| last\_updated | timestamp with time zone | 最后一次更新该语句的时间。 |
| sort\_count | bigint | 排序执行的次数。 |
| sort\_time | bigint | 排序执行的时间（单位：微秒）。 |
| sort\_mem\_used | bigint | 排序过程中使用的work memory大小（单位：KB）。 |
| sort\_spill\_count | bigint | 排序过程中，若发生落盘，写文件的次数。 |
| sort\_spill\_size | bigint | 排序过程中，若发生落盘，使用的文件大小（单位：KB）。 |
| hash\_count | bigint | hash执行的次数。 |
| hash\_time | bigint | hash执行的时间（单位：微秒）。 |
| hash\_mem\_used | bigint | hash过程中使用的work memory大小（单位：KB）。 |
| hash\_spill\_count | bigint | hash过程中，若发生落盘，写文件的次数。 |
| hash\_spill\_size | bigint | hash过程中，若发生落盘，使用的文件大小（单位：KB）。 |

#### STATEMENT

获得当前节点的执行语句（归一化SQL）的信息。查询视图必须具有sysadmin权限或者monitor admin权限。数据库主节点上可以看到此数据库主节点接收到的归一化的SQL的全量统计信息（包含数据库节点）；数据库节点上仅可看到归一化的SQL的此节点执行的统计信息。

**表 1** STATEMENT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| node\_id | integer | 节点的ID。 |
| user\_name | name | 用户名称。 |
| user\_id | oid | 用户OID。 |
| unique\_sql\_id | bigint | 归一化的SQL ID。 |
| query | text | 归一化的SQL。备注：长度受track\_activity\_query\_size控制。 |
| n\_calls | bigint | 调用次数。 |
| min\_elapse\_time | bigint | SQL在内核内的最小运行时间（单位：微秒）。 |
| max\_elapse\_time | bigint | SQL在内核内的最大运行时间（单位：微秒）。 |
| total\_elapse\_time | bigint | SQL在内核内的总运行时间（单位：微秒）。 |
| n\_returned\_rows | bigint | SELECT返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行。 |
| n\_tuples\_returned | bigint | 顺序扫描行。 |
| n\_tuples\_inserted | bigint | 插入行。 |
| n\_tuples\_updated | bigint | 更新行。 |
| n\_tuples\_deleted | bigint | 删除行。 |
| n\_blocks\_fetched | bigint | buffer的块访问次数。 |
| n\_blocks\_hit | bigint | buffer的块命中次数。 |
| n\_soft\_parse | bigint | 软解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 |
| n\_hard\_parse | bigint | 硬解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 |
| net\_send\_info | text | 通过物理连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_recv\_info | text | 通过物理连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_send\_info | text | 通过逻辑连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_recv\_info | text | 通过逻辑连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| sort\_count | bigint | 排序执行的次数。 |
| sort\_time | bigint | 排序执行的时间（单位：微秒）。 |
| sort\_mem\_used | bigint | 排序过程中使用的work memory大小（单位：KB）。 |
| sort\_spill\_count | bigint | 排序过程中，若发生落盘，写文件的次数。 |
| sort\_spill\_size | bigint | 排序过程中，若发生落盘，使用的文件大小（单位：KB）。 |
| hash\_count | bigint | hash执行的次数。 |
| hash\_time | bigint | hash执行的时间（单位：微秒）。 |
| hash\_mem\_used | bigint | hash过程中使用的work memory大小（单位：KB）。 |
| hash\_spill\_count | bigint | hash过程中，若发生落盘，写文件的次数。 |
| hash\_spill\_size | bigint | hash过程中，若发生落盘，使用的文件大小（单位：KB）。 |
| last\_updated | timestamp with time zone | 最后一次更新该语句的时间。 |

#### STATEMENT\_COUNT

显示数据库当前节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）和（DDL、DML、DCL）统计信息。

fig: **说明：**

普通用户查询STATEMENT\_COUNT视图仅能看到该用户当前节点的统计信息；管理员权限用户查询STATEMENT\_COUNT视图则能看到所有用户当前节点的统计信息。当Vastbase或该节点重启时，计数将清零，并重新开始计数。计数以节点收到的查询数为准，Vastbase内部进行的查询。例如，数据库主节点收到一条查询，若下发多条查询数据库节点，那将在数据库节点上进行相应次数的计数。

**表 1** STATEMENT\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| user\_name | text | 用户名。 |
| select\_count | bigint | select语句统计结果。 |
| update\_count | bigint | update语句统计结果。 |
| insert\_count | bigint | insert语句统计结果。 |
| delete\_count | bigint | delete语句统计结果。 |
| mergeinto\_count | bigint | merge into语句统计结果。 |
| ddl\_count | bigint | DDL语句的数量。 |
| dml\_count | bigint | DML语句的数量。 |
| dcl\_count | bigint | DCL语句的数量。 |
| total\_select\_elapse | bigint | 总select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费(单位：微秒)。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| total\_update\_elapse | bigint | 总update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费(单位：微秒)。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | bigint | 总insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | bigint | 总delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |

#### GLOBAL\_STATEMENT\_COUNT

显示数据库各节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）和（DDL、DML、DCL）统计信息。

**表 1** GLOBAL\_STATEMENT\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| user\_name | text | 用户名。 |
| select\_count | bigint | select语句统计结果。 |
| update\_count | bigint | update语句统计结果。 |
| insert\_count | bigint | insert语句统计结果。 |
| delete\_count | bigint | delete语句统计结果。 |
| mergeinto\_count | bigint | merge into语句统计结果。 |
| ddl\_count | bigint | DDL语句的数量。 |
| dml\_count | bigint | DML语句的数量。 |
| dcl\_count | bigint | DCL语句的数量。 |
| total\_select\_elapse | bigint | 总select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费（单位：微秒）。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| total\_update\_elapse | bigint | 总update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费（单位：微秒）。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | bigint | 总insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | bigint | 总delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |

#### SUMMARY\_STATEMENT\_COUNT

显示数据库汇聚各节点（数据库节点）当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）和（DDL、DML、DCL）统计信息。

**表 1** SUMMARY\_STATEMENT\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| user\_name | text | 用户名。 |
| select\_count | numeric | select语句统计结果。 |
| update\_count | numeric | update语句统计结果。 |
| insert\_count | numeric | insert语句统计结果。 |
| delete\_count | numeric | delete语句统计结果。 |
| mergeinto\_count | numeric | merge into语句统计结果。 |
| ddl\_count | numeric | DDL语句的数量。 |
| dml\_count | numeric | DML语句的数量。 |
| dcl\_count | numeric | DCL语句的数量。 |
| total\_select\_elapse | numeric | 总select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费（单位：微秒）。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| total\_update\_elapse | numeric | 总update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费（单位：微秒）。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | numeric | 总insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | numeric | 总delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |

#### GLOBAL\_STATEMENT\_COMPLEX\_HISTORY

显示各个节点执行作业结束后的负载管理记录。

**表 1** GLOBAL\_STATEMENT\_COMPLEX\_HISTORY的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| dbname | text | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 数据库进程名称。 |
| username | text | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| block\_time | bigint | 语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| finish\_time | timestamp with time zone | 语句执行的结束时间。 |
| duration | bigint | 语句实际执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句预估执行时间，单位ms。 |
| status | text | 语句执行结束状态：正常为finished，异常为aborted。 |
| abort\_info | text | 语句执行结束状态为aborted时显示异常信息。 |
| resource\_pool | text | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句预估使用内存。 |
| min\_peak\_memory | integer | 语句在数据库节点上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库节点上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句数据库节点间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库节点上的下盘信息：   * None：数据库节点均未下盘。 * All：数据库节点均下盘。 * [a:b]：数量为b个数据库节点中有a个数据库节点下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库节点上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库节点上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库节点上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库节点的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库节点间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库节点上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库节点上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库节点上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库节点间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Spill file size large than 256MB。 * Broadcast size large than 100MB。 * Early spill。 * Spill times is greater than 3。 * Spill on memory adaptive。 * Hash table conflict。 |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |
| cpu\_top1\_node\_name | text | cpu使用率第1的节点名称。 |
| cpu\_top2\_node\_name | text | cpu使用率第2的节点名称。 |
| cpu\_top3\_node\_name | text | cpu使用率第3的节点名称。 |
| cpu\_top4\_node\_name | text | cpu使用率第4的节点名称。 |
| cpu\_top5\_node\_name | text | cpu使用率第5的节点名称。 |
| mem\_top1\_node\_name | text | 内存使用量第1的节点名称。 |
| mem\_top2\_node\_name | text | 内存使用量第2的节点名称。 |
| mem\_top3\_node\_name | text | 内存使用量第3的节点名称。 |
| mem\_top4\_node\_name | text | 内存使用量第4的节点名称。 |
| mem\_top5\_node\_name | text | 内存使用量第5的节点名称。 |
| cpu\_top1\_value | bigint | cpu使用率。 |
| cpu\_top2\_value | bigint | cpu使用率。 |
| cpu\_top3\_value | bigint | cpu使用率。 |
| cpu\_top4\_value | bigint | cpu使用率。 |
| cpu\_top5\_value | bigint | cpu使用率。 |
| mem\_top1\_value | bigint | 内存使用量。 |
| mem\_top2\_value | bigint | 内存使用量。 |
| mem\_top3\_value | bigint | 内存使用量。 |
| mem\_top4\_value | bigint | 内存使用量。 |
| mem\_top5\_value | bigint | 内存使用量。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |

#### GLOBAL\_STATEMENT\_COMPLEX\_HISTORY\_TABLE

显示各个节点执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。具体的字段请参考：系统表和系统视图->schema->DBE\_PERF Schema->Query->GLOBAL\_STATEMENT\_COMPLEX\_HISTORY中的字段。

#### STATEMENT\_COMPLEX\_HISTORY\_TABLE

STATEMENT\_COMPLEX\_HISTORY\_TABLE系统表显示数据库主节点执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。具体的字段请参考GS\_WLM\_SESSION\_HISTORY的表1。

#### GLOBAL\_STATEMENT\_COMPLEX\_RUNTIME

显示当前用户在各个节点上正在执行的作业的负载管理记录。

**表 1** GLOBAL\_STATEMENT\_COMPLEX\_RUNTIME的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据OID。 |
| dbname | name | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 数据库进程名称 |
| username | name | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| pid | bigint | 后端线程ID。 |
| block\_time | bigint | 语句执行前的阻塞时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| duration | bigint | 语句已经执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句执行预估总时间，单位ms。 |
| estimate\_left\_time | bigint | 语句执行预估剩余时间，单位ms。 |
| enqueue | text | 工作负载管理资源状态。 |
| resource\_pool | name | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句预估使用内存，单位MB。 |
| min\_peak\_memory | integer | 语句在数据库节点上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库节点上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句在数据库节点间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库节点上的下盘信息：   * None：数据库节点均未下盘。 * All：数据库节点均下盘。 * [a:b]：数量为b个数据库节点中有a个数据库节点下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库节点上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库节点上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库节点上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库节点的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库节点间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库节点上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库节点上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库节点上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库节点间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Spill file size large than 256MB。 * Broadcast size large than 100MB。 * Early spill。 * Spill times is greater than 3。 * Spill on memory adaptive。 * Hash table conflict。 |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 正在执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |

#### STATEMENT\_RESPONSETIME\_PERCENTILE

获取VastbaseSQL响应时间P80，P95分布信息。

**表 1** STATEMENT\_RESPONSETIME\_PERCENTILE的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| p80 | bigint | Vastbase80%的SQL的响应时间（单位：微秒）。 |
| p95 | bigint | Vastbase95%的SQL的响应时间（单位：微秒）。 |

#### STATEMENT\_COMPLEX\_RUNTIME

STATEMENT\_COMPLEX\_RUNTIME视图显示当前用户在数据库主节点上正在执行的作业的负载管理记录。

**表 1** STATEMENT\_COMPLEX\_RUNTIME的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据OID。 |
| dbname | name | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 数据库进程名称。 |
| username | name | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| pid | bigint | 后端线程ID。 |
| block\_time | bigint | 语句执行前的阻塞时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| duration | bigint | 语句已经执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句执行预估总时间，单位ms。 |
| estimate\_left\_time | bigint | 语句执行预估剩余时间，单位ms。 |
| enqueue | text | 工作负载管理资源状态。 |
| resource\_pool | name | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句预估使用内存，单位MB。 |
| min\_peak\_memory | integer | 语句在数据库节点上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库节点上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句在数据库节点间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库节点上的下盘信息：   * None：数据库节点均未下盘。 * All：数据库节点均下盘。 * [a:b]：数量为b个数据库节点中有a个数据库节点下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库节点上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库节点上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库节点上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库节点的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库节点间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库节点上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库节点上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库节点上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库节点间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Spill file size large than 256MB。 * Broadcast size large than 100MB。 * Early spill。 * Spill times is greater than 3。 * Spill on memory adaptive。 * Hash table conflict。 |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 正在执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |

#### STATEMENT\_WLMSTAT\_COMPLEX\_RUNTIME

STATEMENT\_WLMSTAT\_COMPLEX\_RUNTIME视图显示和当前用户执行作业正在运行时的负载管理相关信息。

**表 1** STATEMENT\_WLMSTAT\_COMPLEX\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| datname | name | 连接后端的数据库名称。 |
| threadid | bigint | 后端线程ID。 |
| processid | integer | 后端线程的pid。 |
| usesysid | oid | 登录后端的用户OID。 |
| appname | text | 连接到后端的应用名。 |
| usename | name | 登录到该后端的用户名。 |
| priority | bigint | 语句所在Cgroups的优先级。 |
| attribute | text | 语句的属性：  Ordinary：语句发送到数据库后被解析前的默认属性。  Simple：简单语句。  Complicated：复杂语句。  Internal：数据库内部语句。 |
| block\_time | bigint | 语句当前为止的pending的时间，单位s。 |
| elapsed\_time | bigint | 语句当前为止的实际执行时间，单位s。 |
| total\_cpu\_time | bigint | 语句在上一时间周期内的数据库节点上CPU使用的总时间，单位s。 |
| cpu\_skew\_percent | integer | 语句在上一时间周期内的数据库节点上CPU使用的倾斜率。 |
| statement\_mem | integer | 语句执行使用的statement\_mem，预留字段。 |
| active\_points | integer | 语句占用的资源池并发点数。 |
| dop\_value | integer | 语句的从资源池中获取的dop值。 |
| control\_group | text | 该字段不支持。 |
| status | text | 该字段不支持。 |
| enqueue | text | 语句当前的排队情况，包括：  Global：在全局队列中排队。  Respool：在资源池队列中排队。  CentralQueue：在中心协调节点（CCN）中排队。  Transaction：语句处于一个事务块中。  StoredProc：句处于一个存储过程中。  None：未在排队。  Forced None：事务块语句或存储过程语句由于超出设定的等待时间而强制执行。 |
| resource\_pool | name | 语句当前所在的资源池。 |
| query | text | 该后端的最新查询。如果state状态是active（活的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| is\_plana | boolean | 逻辑Vastbase模式下，语句当前是否占用其他逻辑Vastbase的资源执行。该值默认为f（否）。 |
| node\_group | text | 语句所属用户对应的逻辑Vastbase。 |

#### GS\_SLOW\_QUERY\_INFO（废弃）

GS\_SLOW\_QUERY\_INFO视图显示当前节点上已经转储的慢查询信息。此数据是从内核中转储到系统表中的数据。当设置GUC参数enable\_resource\_record为on时，系统会定时（周期为3分钟）将内核中query信息导入GS\_WLM\_SESSION\_QUERY\_INFO\_ALL系统表，开启此功能会占用系统存储空间并对性能有一定影响。用户通过查询GS\_SLOW\_QUERY\_INFO视图，可以查看已经转储的慢查询信息，本版本中已废弃。

表 1 GS\_SLOW\_QUERY\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| dbname | text | 数据库名称。 |
| schemaname | text | schema名称。 |
| nodename | text | 节点名称。 |
| username | text | 用户名。 |
| queryid | bigint | 归一化ID。 |
| query | text | query语句。 |
| start\_time | timestamp with time zone | 开始执行时间。 |
| finish\_time | timestamp with time zone | 结束执行时间。 |
| duration | bigint | 执行持续时间（毫秒）。 |
| query\_plan | text | 计划信息。 |
| n\_returned\_rows | bigint | Select返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行数。 |
| n\_tuples\_returned | bigint | 顺序扫描行数。 |
| n\_tuples\_inserted | bigint | 插入行数。 |
| n\_tuples\_updated | bigint | 更新行数。 |
| n\_tuples\_deleted | bigint | 删除行数。 |
| n\_blocks\_fetched | bigint | Cache加载次数。 |
| n\_blocks\_hit | bigint | Cache命中数。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| net\_send\_time | bigint | 网络上的时间花费（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费(单位：微秒)。 |

#### GS\_SLOW\_QUERY\_HISTORY（废弃）

GS\_SLOW\_QUERY\_HISTORY显示当前节点上未转储的慢查询信息。具体字段信息请参考GS\_SLOW\_QUERY\_INFO。该视图只有system admin和monitor admin用户有权限查询，本版本中已废弃。

#### GLOBAL\_SLOW\_QUERY\_HISTORY（废弃）

GS\_SLOW\_QUERY\_HISTORY显示所有节点上未转储的慢查询信息，本版本中已废弃。具体字段信息请参考GS\_SLOW\_QUERY\_INFO。

#### GLOBAL\_SLOW\_QUERY\_INFO（废弃）

GS\_SLOW\_QUERY\_HISTORY显示所有节点上已经转储的慢查询信息，本版本中已废弃。具体字段信息请参考GS\_SLOW\_QUERY\_INFO。

### Lock

#### GLOBAL\_LOCKS

GLOBAL\_LOCKS视图用于查看各节点各打开事务所持有的锁信息。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| locktype | text | 被锁定对象的类型：relation、extend、page、tuple、transactionid、virtualxid、object、userlock、advisory。 |
| database | oid | 被锁定对象所在数据库的OID：  如果被锁定的对象是共享对象，则OID为0。  如果是一个事务ID，则为NULL。 |
| relation | oid | 关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。 |
| page | integer | 关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。 |
| tuple | smallint | 页面里边的行编号，如果对象不是行，则为NULL。 |
| virtualxid | text | 事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。 |
| transactionid | xid | 事务的ID，如果对象不是一个事务ID，则为NULL。 |
| classid | oid | 包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。 |
| objid | oid | 对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。 |
| objsubid | smallint | 对于表的一个字段，这是字段编号；对于其他对象类型，这个字段是零；如果这个对象不是普通数据库对象，则为NULL。 |
| virtualtransaction | text | 持有此锁或者在等待此锁的事务的虚拟ID。 |
| pid | bigint | 持有或者等待这个锁的服务器线程的逻辑ID。如果锁是被一个预备事务持有的，则为NULL。 |
| mode | text | 这个线程持有的或者是期望的锁模式。 |
| granted | boolean | 如果锁是持有锁，则为TRUE。  如果锁是等待锁，则为FALSE。 |
| fastpath | boolean | 如果通过fast-path获得锁，则为TRUE；如果通过主要的锁表获得，则为FALSE。 |

#### LOCKS

LOCKS视图用于查看各打开事务所持有的锁信息。

表 1 LOCKS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| locktype | text | 被锁定对象的类型：relation、extend、page、tuple、transactionid、virtualxid、object、userlock、advisory。 |
| database | oid | 被锁定对象所在数据库的OID：   * 如果被锁定的对象是共享对象，则OID为0。 * 如果是一个事务ID，则为NULL。 |
| relation | oid | 关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。 |
| page | integer | 关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。 |
| tuple | smallint | 页面里边的行编号，如果对象不是行，则为NULL。 |
| bucket | integer | 哈希桶号。 |
| virtualxid | text | 事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。 |
| transactionid | xid | 事务的ID，如果对象不是一个事务ID，则为NULL。 |
| classid | oid | 包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。 |
| objid | oid | 对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。 |
| objsubid | smallint | 对于表的一个字段，这是字段编号；对于其他对象类型，这个字段是0；如果这个对象不是普通数据库对象，则为NULL。 |
| virtualtransaction | text | 持有此锁或者在等待此锁的事务的虚拟ID。 |
| pid | bigint | 持有或者等待这个锁的服务器线程的逻辑ID。如果锁是被一个预备事务持有的，则为NULL。 |
| sessionid | bigint | 持有或者等待这个锁的会话ID。如果锁是被一个预备事务持有的，则为NULL。 |
| mode | text | 这个线程持有的或者是期望的锁模式。 |
| granted | boolean | * 如果锁是持有锁，则为TRUE。 * 如果锁是等待锁，则为FALSE。 |
| fastpath | boolean | 如果通过fast-path获得锁，则为TRUE；如果通过主要的锁表获得，则为FALSE。 |
| locktag | text | 会话等待锁信息，可通过locktag\_decode()函数解析。 |
| global\_sessionid | text | 全局会话ID。 |

### Wait Events

#### WAIT\_EVENTS

WAIT\_EVENTS显示当前节点的event的等待相关的统计信息。具体事件信息见参考：系统表和系统视图->系统视图->PG\_THREAD\_WAIT\_STATUS章节等待状态列表、轻量级锁等待事件列表、IO等待事件列表和事务锁等待事件列表。关于每种事务锁对业务的影响程度，请参考LOCK语法小节的详细描述。

表 1 WAIT\_EVENTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 数据库进程名称。 |
| type | text | event类型。 |
| event | text | event名称。 |
| wait | bigint | 等待次数。 |
| failed\_wait | bigint | 失败的等待次数。 |
| total\_wait\_time | bigint | 总等待时间（单位：微秒）。 |
| avg\_wait\_time | bigint | 平均等待时间（单位：微秒）。 |
| max\_wait\_time | bigint | 最大等待时间（单位：微秒）。 |
| min\_wait\_time | bigint | 最小等待时间（单位：微秒）。 |
| last\_updated | timestamp with time zone | 最后一次更新该事件的时间。 |

#### GLOBAL\_WAIT\_EVENTS

GLOBAL\_WAIT\_EVENTS视图显示各节点的event的等待相关的统计信息。

**表 1** GLOBAL\_WAIT\_EVENTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 数据库进程名称。 |
| type | text | event类型。 |
| event | text | event名称。 |
| wait | bigint | 等待次数。 |
| failed\_wait | bigint | 失败的等待次数。 |
| total\_wait\_time | bigint | 总等待时间（单位：微秒）。 |
| avg\_wait\_time | bigint | 平均等待时间（单位：微秒）。 |
| max\_wait\_time | bigint | 最大等待时间（单位：微秒）。 |
| min\_wait\_time | bigint | 最小等待时间（单位：微秒）。 |
| last\_updated | timestamp with time zone | 最后一次更新该事件的时间。 |

### Configuration

#### CONFIG\_SETTINGS

CONFIG\_SETTINGS视图显示数据库运行时参数的相关信息。

**表 1** CONFIG\_SETTINGS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | text | 参数名称。 |
| setting | text | 参数当前值。 |
| unit | text | 参数的隐式结构。 |
| category | text | 参数的逻辑组。 |
| short\_desc | text | 参数的简单描述。 |
| extra\_desc | text | 参数的详细描述。 |
| context | text | 设置参数值的上下文，包括internal、postmaster、sighup、backend、superuser、user。 |
| vartype | text | 参数类型，包括bool、enum、integer、real、string。 |
| source | text | 参数的赋值方式。 |
| min\_val | text | 参数最大值。如果参数类型不是数值型，那么该字段值为null。 |
| max\_val | text | 参数最小值。如果参数类型不是数值型，那么该字段值为null。 |
| enumvals | text[] | enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。 |
| boot\_val | text | 数据库启动时参数默认值。 |
| reset\_val | text | 数据库重置时参数默认值。 |
| sourcefile | text | 设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。 |
| sourceline | integer | 设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。 |

#### GLOBAL\_CONFIG\_SETTINGS

GLOBAL\_CONFIG\_SETTINGS显示各节点数据库运行时参数的相关信息。

表 1 GLOBAL\_CONFIG\_SETTINGS的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| name | text | 参数名称。 |
| setting | text | 参数当前值。 |
| unit | text | 参数的隐式结构。 |
| category | text | 参数的逻辑组。 |
| short\_desc | text | 参数的简单描述。 |
| extra\_desc | text | 参数的详细描述。 |
| context | text | 设置参数值的上下文，包括internal、postmaster、sighup、backend、superuser、user。 |
| vartype | text | 参数类型，包括bool、enum、integer、real、string。 |
| source | text | 参数的赋值方式。 |
| min\_val | text | 参数最小值。如果参数类型不是数值型，那么该字段值为null。 |
| max\_val | text | 参数最大值。如果参数类型不是数值型，那么该字段值为null。 |
| enumvals | text[] | enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。 |
| boot\_val | text | 数据库启动时参数默认值。 |
| reset\_val | text | 数据库重置时参数默认值。 |
| sourcefile | text | 设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。 |
| sourceline | integer | 设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。 |

### Operator

#### OPERATOR\_HISTORY\_TABLE

OPERATOR\_HISTORY\_TABLE系统表显示执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表中的数据。

**表 1** OPERATOR\_HISTORY\_TABLE的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间（ms）。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库节点上的最小内存峰值（MB）。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值（MB）。 |
| average\_peak\_memory | integer | 当前算子在数据库节点上的平均内存峰值（MB）。 |
| memory\_skew\_percent | integer | 当前算子在数据库节点间的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量（MB），默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量（MB），默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量（MB），默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库节点上的最小执行时间（ms）。 |
| max\_cpu\_time | bigint | 该算子在数据库节点上的最大执行时间（ms）。 |
| total\_cpu\_time | bigint | 该算子在数据库节点上的总执行时间（ms）。 |
| cpu\_skew\_percent | integer | 数据库节点间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Sort/SetOp/HashAgg/HashJoin spill。Spill file size large than 256MB。Broadcast size large than 100MB。Early spill。Spill times is greater than 3。Spill on memory adaptive。Hash table conflict。 |

#### OPERATOR\_HISTORY

OPERATOR\_HISTORY视图显示的是当前用户数据库主节点上执行作业结束后的算子的相关记录。记录的数据同GS\_WLM\_OPERATOR\_INFO表（参考：系统表和系统视图->系统表->GS\_WLM\_OPERATOR\_INFO章节）。

#### OPERATOR\_RUNTIME

OPERATOR\_RUNTIME视图显示当前用户正在执行的作业的算子相关信息。

**表 1** OPERATOR\_RUNTIME的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间（ms）。 |
| status | text | 当前算子的执行状态，包括finished和running。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库节点上的最小内存峰值（MB）。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值（MB）。 |
| average\_peak\_memory | integer | 当前算子在数据库节点上的平均内存峰值（MB）。 |
| memory\_skew\_percent | integer | 当前算子在数据库节点的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量（MB），默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量（MB），默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量（MB），默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库节点上的最小执行时间（ms）。 |
| max\_cpu\_time | bigint | 该算子在数据库节点上的最大执行时间(ms)。 |
| total\_cpu\_time | bigint | 该算子在数据库节点上的总执行时间(ms)。 |
| cpu\_skew\_percent | integer | 数据库节点间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Sort/SetOp/HashAgg/HashJoin spill。Spill file size large than 256MB。Broadcast size large than 100MB。Early spill。Spill times is greater than 3。Spill on memory adaptive。Hash table conflict。 |

#### GLOBAL\_OPERATOR\_HISTORY

GLOBAL\_OPERATOR\_HISTORY系统视图显示的是当前用户在数据库主节点上执行作业结束后的算子的相关记录。

**表 1** GLOBAL\_OPERATOR\_HISTORY的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间(ms)。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库节点上的最小内存峰值(MB)。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值(MB)。 |
| average\_peak\_memory | integer | 当前算子在数据库节点上的平均内存峰值(MB)。 |
| memory\_skew\_percent | integer | 当前算子在数据库节点间的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量(MB)，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量(MB)，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量(MB)，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库节点上的最小执行时间(ms)。 |
| max\_cpu\_time | bigint | 该算子在数据库节点上的最大执行时间(ms)。 |
| total\_cpu\_time | bigint | 该算子在数据库节点上的总执行时间(ms)。 |
| cpu\_skew\_percent | integer | 数据库节点间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息： 1、Sort/SetOp/HashAgg/HashJoin spill。 2、Spill file size large than 256MB。 3、Broadcast size large than 100MB。 4、Early spill。 5、Spill times is greater than 3。 6、Spill on memory adaptive。 7、Hash table conflict。 |

#### GLOBAL\_OPERATOR\_HISTORY\_TABLE

GLOBAL\_OPERATOR\_HISTORY\_TABLE视图显示数据库主节点执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表GS\_WLM\_OPERATOR\_INFO中的数据。该视图是查询数据库主节点系统表GS\_WLM\_OPERATOR\_INFO的汇聚视图。表字段同GLOBAL\_OPERATOR\_HISTORY表1（参考：系统表和系统视图->schema->DBE\_PERF Schema->OperatorGLOBAL\_OPERATOR\_HISTORY章节）。

#### GLOBAL\_OPERATOR\_RUNTIME

GLOBAL\_OPERATOR\_RUNTIME视图显示当前用户在数据库主节点上正在执行的作业的算子相关信息。

**表 1** GLOBAL\_OPERATOR\_RUNTIME的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间(ms)。 |
| status | text | 当前算子的执行状态，包括finished和running。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库节点上的最小内存峰值(MB)。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值(MB)。 |
| average\_peak\_memory | integer | 当前算子在数据库节点上的平均内存峰值(MB)。 |
| memory\_skew\_percent | integer | 当前算子在数据库节点的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最小数据量(MB)，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量(MB)，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点上下盘的平均数据量(MB)，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库节点上的最小执行时间(ms)。 |
| max\_cpu\_time | bigint | 该算子在数据库节点上的最大执行时间(ms)。 |
| total\_cpu\_time | bigint | 该算子在数据库节点上的总执行时间(ms)。 |
| cpu\_skew\_percent | integer | 数据库节点间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Sort/SetOp/HashAgg/HashJoin spill。Spill file size large than 256MB。Broadcast size large than 100MB。Early spill。Spill times is greater than 3。Spill on memory adaptive。Hash table conflict。 |

### Workload Manager

#### WLM\_USER\_RESOURCE\_CONFIG

WLM\_USER\_RESOURCE\_CONFIG视图显示用户的资源配置信息。

**表 1** WLM\_USER\_RESOURCE\_CONFIG字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| userid | oid | 用户oid。 |
| username | name | 用户名称。 |
| sysadmin | boolean | 是否是sysadmin。 |
| rpoid | oid | 资源池的oid。 |
| respool | name | 资源池的名称。 |
| parentid | oid | 父用户的oid。 |
| totalspace | bigint | 占用总空间大小。 |
| spacelimit | bigint | 空间大上限。 |
| childcount | integer | 子用户数量。 |
| childlist | text | 子用户的列表。 |

#### WLM\_USER\_RESOURCE\_RUNTIME

WLM\_USER\_RESOURCE\_RUNTIME视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在GUC参数“use\_workload\_manager”为“on”时才有效。

**表 1** WLM\_USER\_RESOURCE\_RUNTIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| username | name | 用户名。 |
| used\_memory | integer | 正在使用的内存大小，单位MB。 |
| total\_memory | integer | 可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 |
| used\_cpu | integer | 正在使用的CPU核数。 |
| total\_cpu | integer | 在该机器节点上，用户关联控制组的CPU核数总和。 |
| used\_space | bigint | 已使用的存储空间大小，单位KB。 |
| total\_space | bigint | 可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。 |
| used\_temp\_space | bigint | 已使用的临时空间大小（预留字段，暂未使用），单位KB。 |
| total\_temp\_space | bigint | 可使用的临时空间大小（预留字段，暂未使用），单位KB，值为-1表示未限制最大临时存储空间。 |
| used\_spill\_space | bigint | 已使用的下盘空间大小（预留字段，暂未使用），单位KB。 |
| total\_spill\_space | bigint | 可使用的下盘空间大小（预留字段，暂未使用），单位KB，值为-1表示未限制最大下盘空间。 |

### Global Plancache

GPC相关视图在enable\_global\_plancache打开且线程池打开的状态下才有效。

#### GLOBAL\_PLANCACHE\_STATUS

GLOBAL\_PLANCACHE\_STATUS视图显示GPC全局计划缓存状态信息。

**表 1** GLOBAL\_PLANCACHE\_STATUS字段

#### GLOBAL\_PLANCACHE\_CLEAN

GLOBAL\_PLANCACHE\_CLEAN视图用于清理所有节点上无人使用的全局计划缓存。返回值为Boolean类型。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 所属节点名称。 |
| query | text | 查询语句text。 |
| refcount | integer | 被引用次数。 |
| valid | bool | 是否合法。 |
| databaseid | oid | 所属数据库id。 |
| schema\_name | text | 所属schema。 |
| params\_num | integer | 参数数量。 |
| func\_id | oid | 该plancache所在存储过程oid，如果不属于存储过程则为0。 |

### RTO

#### global\_rto\_status

global\_rto\_status视图显示关于主机和备机的日志流控信息（本节点除外、备DN上不可使用）。

**表 1** remote\_rto\_status参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点的名称，包含主机和备机。 |
| rto\_info | text | 流控的信息，包含了备机当前的日志流控时间（单位：秒），备机通过GUC参数设置的预期流控时间（单位：秒），为了达到这个预期主机所需要的睡眠时间（单位：微秒）。 |

### Cache/IO

#### STATIO\_USER\_TABLES

STATIO\_USER\_TABLES视图显示命名空间中所有用户关系表的I/O状态信息。

**表 1** STATIO\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

#### SUMMARY\_STATIO\_USER\_TABLES

SUMMARY\_STATIO\_USER\_TABLES视图显示Vastbase内汇聚的命名空间中所有用户关系表的I/O状态信息。

**表 1** SUMMARY\_STATIO\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | numeric | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | numeric | 此表缓存命中数。 |
| idx\_blks\_read | numeric | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 表中所有索引命中缓存数。 |
| toast\_blks\_read | numeric | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | numeric | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | numeric | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | numeric | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### GLOBAL\_STATIO\_USER\_TABLES

GLOBAL\_STATIO\_USER\_TABLES视图显示各节点的命名空间中所有用户关系表的I/O状态信息。

**表 1** GLOBAL\_STATIO\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 节点名称。 |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 此表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### STATIO\_USER\_INDEXES

STATIO\_USER\_INDEXES视图显示当前节点命名空间中所有用户关系表索引的I/O状态信息。

**表 1** STATIO\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

#### SUMMARY\_STATIO\_USER\_INDEXES

SUMMARY\_STATIO\_USER\_INDEXES视图显示Vastbase内汇聚的命名空间中所有用户关系表索引的I/O状态信息。

**表 1** SUMMARY\_STATIO\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

#### GLOBAL\_STATIO\_USER\_INDEXES

GLOBAL\_STATIO\_USER\_INDEXES视图显示各节点的命名空间中所有用户关系表索引的I/O状态信息。

**表 1** GLOBAL\_STATIO\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

#### STATIO\_USER\_SEQUENCES

STATIO\_USER\_SEQUENCE视图显示当前节点的命名空间中所有用户关系表类型为序列的I/O状态信息。

**表 1** STATIO\_USER\_SEQUENCE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### SUMMARY\_STATIO\_USER\_SEQUENCES

SUMMARY\_STATIO\_USER\_SEQUENCES视图显示Vastbase内汇聚的命名空间中所有用户关系表类型为序列的I/O状态信息。

**表 1** SUMMARY\_STATIO\_USER\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | numeric | 从序列中读取的磁盘块数。 |
| blks\_hit | numeric | 序列中缓存命中数。 |

#### GLOBAL\_STATIO\_USER\_SEQUENCES

GLOBAL\_STATIO\_USER\_SEQUENCES视图显示各节点的命名空间中所有用户关系表类型为序列的I/O状态信息。

**表 1** GLOBAL\_STATIO\_USER\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### STATIO\_SYS\_TABLES

STATIO\_SYS\_TABLES视图显示命名空间中所有系统表的I/O状态信息。

**表 1** STATIO\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

#### SUMMARY\_STATIO\_SYS\_TABLES

SUMMARY\_STATIO\_SYS\_TABLES视图显示Vastbase内汇聚的命名空间中所有系统表的I/O状态信息。

**表 1** SUMMARY\_STATIO\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | numeric | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | numeric | 此表缓存命中数。 |
| idx\_blks\_read | numeric | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 表中所有索引命中缓存数。 |
| toast\_blks\_read | numeric | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | numeric | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | numeric | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | numeric | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### GLOBAL\_STATIO\_SYS\_TABLES

GLOBAL\_STATIO\_SYS\_TABLES视图显示各节点的命名空间中所有系统表的I/O状态信息。

**表 1** GLOBAL\_STATIO\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 此表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### STATIO\_SYS\_INDEXES

STATIO\_SYS\_INDEXES显示命名空间中所有系统表索引的I/O状态信息。

**表 1** STATIO\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

#### SUMMARY\_STATIO\_SYS\_INDEXES

SUMMARY\_STATIO\_SYS\_INDEXES视图显示Vastbase内汇聚的命名空间中所有系统表索引的I/O状态信息。

**表 1** SUMMARY\_STATIO\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

#### GLOBAL\_STATIO\_SYS\_INDEXES

GLOBAL\_STATIO\_SYS\_INDEXES视图显示各节点的命名空间中所有系统表索引的I/O状态信息。

**表 1** GLOBAL\_STATIO\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

#### STATIO\_SYS\_SEQUENCES

STATIO\_SYS\_SEQUENCES显示命名空间中所有系统表为序列的I/O状态信息。

**表 1** STATIO\_SYS\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### SUMMARY\_STATIO\_SYS\_SEQUENCES

SUMMARY\_STATIO\_SYS\_SEQUENCES视图显示Vastbase内汇聚的命名空间中所有系统表为序列的I/O状态信息。

**表 1** SUMMARY\_STATIO\_SYS\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | numeric | 从序列中读取的磁盘块数。 |
| blks\_hit | numeric | 序列中缓存命中数。 |

#### GLOBAL\_STATIO\_SYS\_SEQUENCES

GLOBAL\_STATIO\_SYS\_SEQUENCES视图显示各节点的命名空间中所有系统表为序列的I/O状态信息。

**表 1** GLOBAL\_STATIO\_SYS\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### STATIO\_ALL\_TABLES

STATIO\_ALL\_TABLES视图将包含数据库中每个表（包括TOAST表）的一行，显示出特定表I/O的统计。

**表 1** STATIO\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

#### SUMMARY\_STATIO\_ALL\_TABLES

SUMMARY\_STATIO\_ALL\_TABLES视图将包含Vastbase内汇聚的数据库中每个表（包括TOAST表）的I/O的统计。

**表 1** SUMMARY\_STATIO\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | numeric | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | numeric | 此表缓存命中数。 |
| idx\_blks\_read | numeric | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 表中所有索引命中缓存数。 |
| toast\_blks\_read | numeric | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | numeric | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | numeric | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | numeric | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### GLOBAL\_STATIO\_ALL\_TABLES

GLOBAL\_STATIO\_ALL\_TABLES视图将包含各节点的数据库中每个表（包括TOAST表）的I/O的统计。

**表 1** GLOBAL\_STATIO\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 此表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 此表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 此表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 此表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 此表的TOAST表索引命中缓冲区数（如果存在）。 |

#### STATIO\_ALL\_INDEXES

STATIO\_ALL\_INDEXES视图包含数据库中的每个索引行，显示特定索引的I/O的统计。

**表 1** STATIO\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

#### GLOBAL\_STATIO\_ALL\_INDEXES

GLOBAL\_STATIO\_ALL\_INDEXES视图包含各节点的数据库中的每个索引行，显示特定索引的I/O的统计。

**表 1** GLOBAL\_STATIO\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

#### STATIO\_ALL\_SEQUENCES

STATIO\_ALL\_SEQUENCES视图包含数据库中每个序列的每一行，显示特定序列关于I/O的统计。

**表 1** STATIO\_ALL\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### SUMMARY\_STATIO\_ALL\_SEQUENCES

SUMMARY\_STATIO\_ALL\_SEQUENCES视图包含Vastbase内汇聚的数据库中每个序列的每一行,显示特定序列关于I/O的统计。

**表 1** SUMMARY\_STATIO\_ALL\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | numeric | 从序列中读取的磁盘块数。 |
| blks\_hit | numeric | 序列中缓存命中数。 |

#### GLOBAL\_STATIO\_ALL\_SEQUENCES

GLOBAL\_STATIO\_ALL\_SEQUENCES包含各节点的数据库中每个序列的每一行，显示特定序列关于I/O的统计。

**表 1** GLOBAL\_STATIO\_ALL\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

#### GLOBAL\_STAT\_DB\_CU

GLOBAL\_STAT\_DB\_CU视图用于查询Vastbase，每个数据库的CU命中情况。可以通过pg\_stat\_reset()进行清零。

**表 1** GLOBAL\_STAT\_DB\_CU字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name1 | text | 数据库进程名称。 |
| db\_name | text | 数据库名。 |
| mem\_hit | bigint | 内存命中次数。 |
| hdd\_sync\_read | bigint | 硬盘同步读次数。 |
| hdd\_asyn\_read | bigint | 硬盘异步读次数。 |

#### GLOBAL\_STAT\_SESSION\_CU

GLOBAL\_STAT\_SESSION\_CU用于查询Vastbase各个节点，当前运行session的CU命中情况。session退出相应的统计数据会清零。Vastbase重启后，统计数据也会清零。

**表 1** GLOBAL\_STAT\_SESSION\_CU字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| mem\_hit | integer | 内存命中次数。 |
| hdd\_sync\_read | integer | 硬盘同步读次数。 |
| hdd\_asyn\_read | integer | 硬盘异步读次数。 |

### Utility

#### REPLICATION\_STAT

REPLICATION\_STAT用于描述日志同步状态信息，如发起端发送日志位置、收端接收日志位置等。

**表 1** REPLICATION\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pid | bigint | 线程的PID。 |
| usesysid | oid | 用户系统ID。 |
| usename | name | 用户名。 |
| application\_name | text | 程序名称。 |
| client\_addr | inet | 客户端地址。 |
| client\_hostname | text | 客户端名。 |
| client\_port | integer | 客户端端口。 |
| backend\_start | timestamp with time zone | 程序启动时间。 |
| state | text | 日志复制的状态： 追赶状态。 一致的流状态。 |
| sender\_sent\_location | text | 发送端发送日志位置。 |
| receiver\_write\_location | text | 接收端write日志位置。 |
| receiver\_flush\_location | text | 接收端flush日志位置。 |
| receiver\_replay\_location | text | 接收端replay日志位置。 |
| sync\_priority | integer | 同步复制的优先级（0表示异步）。 |
| sync\_state | text | 同步状态：异步复制、同步复制、潜在同步者。 |

#### GLOBAL\_REPLICATION\_STAT

GLOBAL\_REPLICATION\_STAT视图用于获得各节点描述日志同步状态信息，如发起端发送日志位置、收端接收日志位置等。

**表 1** GLOBAL\_REPLICATION\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| pid | bigint | 线程的PID。 |
| usesysid | oid | 用户系统ID。 |
| usename | name | 用户名。 |
| application\_name | text | 程序名称。 |
| client\_addr | inet | 客户端地址。 |
| client\_hostname | text | 客户端名。 |
| client\_port | integer | 客户端端口。 |
| backend\_start | timestamp with time zone | 程序启动时间。 |
| state | text | 日志复制的状态：追赶状态、一致的流状态。 |
| sender\_sent\_location | text | 发送端发送日志位置。 |
| receiver\_write\_location | text | 接收端write日志位置。 |
| receiver\_flush\_location | text | 接收端flush日志位置。 |
| receiver\_replay\_location | text | 接收端replay日志位置。 |
| sync\_priority | integer | 同步复制的优先级（0表示异步）。 |
| sync\_state | text | 同步状态：异步复制、同步复制、潜在同步者。 |

#### REPLICATION\_SLOTS

REPLICATION\_SLOTS视图用于查看复制节点的信息。

**表 1** REPLICATION\_SLOTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| slot\_name | text | 复制节点的名称。 |
| plugin | text | 插件名称。 |
| slot\_type | text | 复制节点的类型。 |
| datoid | oid | 复制节点的数据库OID。 |
| database | name | 复制节点的数据库名称。 |
| active | boolean | 复制节点是否为激活状态。 |
| xmin | xid | 复制节点事务标识。 |
| catalog\_xmin | xid | 逻辑复制槽对应的最早解码事务标识。 |
| restart\_lsn | text | 复制节点的Xlog文件信息。 |
| dummy\_standby | boolean | 复制节点是否为假备。 |

#### GLOBAL\_REPLICATION\_SLOTS

GLOBAL\_REPLICATION\_SLOTS视图用于查看Vastbase各节点的复制节点的信息。

**表 1** GLOBAL\_REPLICATION\_SLOTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| slot\_name | text | 复制节点的名称。 |
| plugin | text | 插件名称。 |
| slot\_type | text | 复制节点的类型。 |
| datoid | oid | 复制节点的数据库OID。 |
| database | name | 复制节点的数据库名称。 |
| active | boolean | 复制节点是否为激活状态。 |
| x\_min | xid | 复制节点事务标识。 |
| catalog\_xmin | xid | 逻辑复制槽对应的最早解码事务标识. |
| restart\_lsn | text | 复制节点的Xlog文件信息。 |
| dummy\_standby | boolean | 复制节点是否为假备。 |

#### BGWRITER\_STAT

BGWRITER\_STAT视图显示关于后端写进程活动的统计信息。

**表 1** BGWRITER\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| checkpoints\_timed | bigint | 执行的定期检查点数。 |
| checkpoints\_req | bigint | 执行的需求检查点数。 |
| checkpoint\_write\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被写入到磁盘，以毫秒为单位。 |
| checkpoint\_sync\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被同步到磁盘，以毫秒为单位。 |
| buffers\_checkpoint | bigint | 检查点写缓冲区数量。 |
| buffers\_clean | bigint | 后端写进程写缓冲区数量。 |
| maxwritten\_clean | bigint | 后端写进程停止清理扫描时间数，因为它写了太多缓冲区。 |
| buffers\_backend | bigint | 通过后端直接写缓冲区数。 |
| buffers\_backend\_fsync | bigint | 后端不得不执行自己的fsync调用的时间数（通常后端写进程处理这些即使后端确实自己写）。 |
| buffers\_alloc | bigint | 分配的缓冲区数量。 |
| stats\_reset | timestamp with time zone | 这些统计被重置的时间。 |

#### GLOBAL\_BGWRITER\_STAT

GLOBAL\_BGWRITER\_STAT视图显示各节点关于后端写进程活动的统计信息。

**表 1** GLOBAL\_BGWRITER\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| checkpoints\_timed | bigint | 执行的定期检查点数。 |
| checkpoints\_req | bigint | 执行的需求检查点数。 |
| checkpoint\_write\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被写入到磁盘，以毫秒为单位。 |
| checkpoint\_sync\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被同步到磁盘，以毫秒为单位。 |
| buffers\_checkpoint | bigint | 检查点写缓冲区数量。 |
| buffers\_clean | bigint | 后端写进程写缓冲区数量。 |
| maxwritten\_clean | bigint | 后端写进程停止清理扫描时间数，因为它写了太多缓冲区。 |
| buffers\_backend | bigint | 通过后端直接写缓冲区数。 |
| buffers\_backend\_fsync | bigint | 后端不得不执行自己的fsync调用的时间数（通常后端写进程处理这些即使后端确实自己写）。 |
| buffers\_alloc | bigint | 分配的缓冲区数量。 |
| stats\_reset | timestamp with time zone | 这些统计被重置的时间。 |

#### GLOBAL\_CKPT\_STATUS

GLOBAL\_CKPT\_STATUS视图用于显示Vastbase所有实例的检查点信息和各类日志刷页情况。

**表 1** GLOBAL\_CKPT\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| ckpt\_redo\_point | test | 当前实例的检查点。 |
| ckpt\_clog\_flush\_num | bigint | 从启动到当前时间clog刷盘页面数。 |
| ckpt\_csnlog\_flush\_num | bigint | 从启动到当前时间csnlog刷盘页面数。 |
| ckpt\_multixact\_flush\_num | bigint | 从启动到当前时间multixact刷盘页面数。 |
| ckpt\_predicate\_flush\_num | bigint | 从启动到当前时间predicate刷盘页面数。 |
| ckpt\_twophase\_flush\_num | bigint | 从启动到当前时间twophase刷盘页面数。 |

#### GLOBAL\_DOUBLE\_WRITE\_STATUS

GLOBAL\_DOUBLE\_WRITE\_STATUS视图显示Vastbase所有实例的双写文件的情况。它是由每个节点的local\_double\_write\_stat视图组成，属性完全一致。

**表 1** GLOBAL\_DOUBLE\_WRITE\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| curr\_dwn | bigint | 当前双写文件的序列号。 |
| curr\_start\_page | bigint | 当前双写文件恢复起始页面。 |
| file\_trunc\_num | bigint | 当前双写文件复用的次数。 |
| file\_reset\_num | bigint | 当前双写文件写满后发生重置的次数。 |
| total\_writes | bigint | 当前双写文件总的I/O次数。 |
| low\_threshold\_writes | bigint | 低效率写双写文件的I/O次数（一次I/O刷页数量少于16页面）。 |
| high\_threshold\_writes | bigint | 高效率写双写文件的I/O次数（一次I/O刷页数量多于一批，421个页面）。 |
| total\_pages | bigint | 当前刷页到双写文件区的总的页面个数。 |
| low\_threshold\_pages | bigint | 低效率刷页的页面个数。 |
| high\_threshold\_pages | bigint | 高效率刷页的页面个数。 |
| file\_id | bigint | 当前双写文件的id号。 |

#### GLOBAL\_PAGEWRITER\_STATUS

GLOBAL\_PAGEWRITER\_STATUS视图显示Vastbase实例的刷页信息和检查点信息。

**表 1** GLOBAL\_PAGEWRITER\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| pgwr\_actual\_flush\_total\_num | bigint | 从启动到当前时间总计刷脏页数量。 |
| pgwr\_last\_flush\_num | integer | 上一批刷脏页数量。 |
| remain\_dirty\_page\_num | bigint | 当前预计还剩余多少脏页。 |
| queue\_head\_page\_rec\_lsn | text | 当前实例的脏页队列第一个脏页的recovery\_lsn。 |
| queue\_rec\_lsn | text | 当前实例的脏页队列的recovery\_lsn。 |
| current\_xlog\_insert\_lsn | text | 当前实例XLog写入的位置。 |
| ckpt\_redo\_point | text | 当前实例的检查点。 |

#### GLOBAL\_RECORD\_RESET\_TIME

GLOBAL\_RECORD\_RESET\_TIME用于重置（重启、主备倒换、数据库删除）汇聚Vastbase统计信息时间。

**表 1** GLOBAL\_RECORD\_RESET\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| reset\_time | timestamp with time zone | 重置时间点。 |

#### GLOBAL\_REDO\_STATUS

GLOBAL\_REDO\_STATUS视图显示Vastbase实例的日志回放情况。

**表 1** GLOBAL\_REDO\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| redo\_start\_ptr | bigint | 当前实例日志回放的起始点。 |
| redo\_start\_time | bigint | 当前实例日志回放的起始UTC时间。 |
| redo\_done\_time | bigint | 当前实例日志回放的结束UTC时间。 |
| curr\_time | bigint | 当前实例的当前UTC时间。 |
| min\_recovery\_point | bigint | 当前实例日志的最小一致性点位置。 |
| read\_ptr | bigint | 当前实例日志的读取位置。 |
| last\_replayed\_read\_ptr | bigint | 当前实例的日志回放位置。 |
| recovery\_done\_ptr | bigint | 当前实例启动完成时的回放位置。 |
| read\_xlog\_io\_counter | bigint | 当前实例读取回放日志的io次数计数。 |
| read\_xlog\_io\_total\_dur | bigint | 当前实例读取回放日志的io总时延。 |
| read\_data\_io\_counter | bigint | 当前实例回放过程中读取数据页面的io次数计数。 |
| read\_data\_io\_total\_dur | bigint | 当前实例回放过程中读取数据页面的io总时延。 |
| write\_data\_io\_counter | bigint | 当前实例回放过程中写数据页面的io次数计数。 |
| write\_data\_io\_total\_dur | bigint | 当前实例回放过程中写数据页面的io总时延。 |
| process\_pending\_counter | bigint | 当前实例回放过程中日志分发线程的同步次数计数。 |
| process\_pending\_total\_dur | bigint | 当前实例回放过程中日志分发线程的同步总时延。 |
| apply\_counter | bigint | 当前实例回放过程中回放线程的同步次数计数。 |
| apply\_total\_dur | bigint | 当前实例回放过程中回放线程的同步总时延。 |
| speed | bigint | 当前实例日志回放速率。 |
| local\_max\_ptr | bigint | 当前实例启动成功后本地收到的回放日志的最大值。 |
| primary\_flush\_ptr | bigint | 主机落盘日志的位置。 |
| worker\_info | text | 当前实例回放线程信息，若没有开并行回放则该值为空。 |

#### GLOBAL\_RECOVERY\_STATUS

GLOBAL\_RECOVERY\_STATUS视图显示关于主机和备机的日志流控信息。

**表 1** GLOBAL\_RECOVERY\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 主机进程名称，包含主机和备机。 |
| standby\_node\_name | text | 备机进程名称。 |
| source\_ip | text | 主机的IP地址。 |
| source\_port | integer | 主机的端口号。 |
| dest\_ip | text | 备机的IP地址。 |
| dest\_port | integer | 备机的端口号。 |
| current\_rto | bigint | 备机当前的日志流控时间，单位秒。 |
| target\_rto | bigint | 备机通过GUC参数设置的预期流控时间，单位秒。 |
| current\_sleep\_time | bigint | 为了达到这个预期主机所需要的睡眠时间，单位微妙。 |

#### CLASS\_VITAL\_INFO

CLASS\_VITAL\_INFO视图用于做WDR时校验相同的表或者索引的oid是否一致。

**表 1** CLASS\_VITAL\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的oid。 |
| schemaname | name | schema名称。 |
| relname | name | 表名。 |
| relkind | “char” | 表示对象类型，取值范围如下： r：表示普通表。 t：表示toast表。 i：表示索引。 |

#### USER\_LOGIN

USER\_LOGIN用来记录用户登录和退出次数的相关信息。

**表 1** USER\_LOGIN字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| user\_name | text | 用户名称。 |
| user\_id | integer | 用户oid（同pg\_authid中的oid字段）。 |
| login\_counter | bigint | 登录次数。 |
| logout\_counter | bigint | 退出次数。 |

#### SUMMARY\_USER\_LOGIN

SUMMARY\_USER\_LOGIN用来记录数据库主节点上用户登录和退出次数的相关信息。

**表 1** SUMMARY\_USER\_LOGIN字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| user\_name | text | 用户名称。 |
| user\_id | integer | 用户oid（同pg\_authid中的oid字段）。 |
| login\_counter | bigint | 登录次数。 |
| logout\_counter | bigint | 退出次数。 |

#### GLOBAL\_SINGLE\_FLUSH\_DW\_STATUS

GLOBAL\_SINGLE\_FLUSH\_DW\_STATUS视图显示数据库所有实例单页面淘汰双写文件信息。展示内容中，/前是第一个版本双写文件刷页情况，/后是第二个版本双写文件刷页情况。

**表 1** GLOBAL\_SINGLE\_FLUSH\_DW\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 实例名称。 |
| bgwr\_actual\_flush\_total\_num | bigint | 从启动到当前时间bgwriter线程总计刷脏页数量。 |
| bgwr\_last\_flush\_num | integer | bgwriter线程上一批刷脏页数量。 |
| candidate\_slots | integer | 当前候选buffer链中页面个数。 |
| get\_buffer\_from\_list | bigint | buffer淘汰从候选buffer链中获取页面的次数。 |
| get\_buffer\_clock\_sweep | bigint | buffer淘汰从原淘汰方案中获取页面的次数。 |

#### GLOBAL\_CANDIDATE\_STATUS

GLOBAL\_CANDIDATE\_STATUS视图显示整个数据库所有实例候选buffer个数，buffer淘汰信息。

**表 1** GLOBAL\_GET\_BGWRITER\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| candidate\_slots | integer | 当前Normal Buffer Pool候选buffer链中页面个数。 |
| get\_buf\_from\_list | bigint | Normal Buffer Pool，buffer淘汰从候选buffer链中获取页面的次数。 |
| get\_buf\_clock\_sweep | bigint | Normal Buffer Pool，buffer淘汰从原淘汰方案中获取页面的次数。 |
| seg\_candidate\_slots | integer | 当前Segment Buffer Pool候选buffer链中页面个数。 |
| seg\_get\_buf\_from\_list | bigint | Segment Buffer Pool，buffer淘汰从候选buffer链中获取页面的次数。 |
| seg\_get\_buf\_clock\_sweep | bigint | Segment Buffer Pool，buffer淘汰从原淘汰方案中获取页面的次数。 |

### Object

#### STAT\_USER\_TABLES

显示当前节点所有命名空间中用户自定义普通表的状态信息。

**表 1** STAT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（即没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次该表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析该表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | bigint | 该表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 该表被autovacuum清理的次数。 |
| analyze\_count | bigint | 该表被手动分析的次数。 |
| autoanalyze\_count | bigint | 该表被autovacuum守护进程分析的次数。 |

#### SUMMARY\_STAT\_USER\_TABLES

Vastbase内汇聚所有命名空间中用户自定义普通表的状态信息。

**表 1** SUMMARY\_STAT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（即没有更新所需的单独索引）。 |
| n\_live\_tup | numeric | 估计活跃行数。 |
| n\_dead\_tup | numeric | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | numeric | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | numeric | 这个表被autovacuum清理的次数。 |
| analyze\_count | numeric | 这个表被手动分析的次数。 |
| autoanalyze\_count | numeric | 这个表被autovacuum守护进程分析的次数。 |

#### GLOBAL\_STAT\_USER\_TABLES

得到各节点所有命名空间中用户自定义普通表的状态信息。

**表 1** GLOBAL\_STAT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（即没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被手动分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |

#### STAT\_USER\_INDEXES

显示数据库中用户自定义普通表的索引状态信息。

**表 1** STAT\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 此索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引的模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### SUMMARY\_STAT\_USER\_INDEXES

Vastbase内汇聚所有数据库中用户自定义普通表的索引状态信息。

**表 1** SUMMARY\_STAT\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | numeric | 索引上开始的索引扫描数。 |
| idx\_tup\_read | numeric | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | numeric | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### GLOBAL\_STAT\_USER\_INDEXES

得到各节点数据库中用户自定义普通表的索引状态信息。

**表 1** GLOBAL\_STAT\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### STAT\_SYS\_TABLES

显示单节点内pg\_catalog、information\_schema以及pg\_toast模式下的所有系统表的统计信息。

**表 1** STAT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次该表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析该表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 该表被autovacuum清理的次数。 |
| analyze\_count | bigint | 该表被手动分析的次数。 |
| autoanalyze\_count | bigint | 该表被autovacuum守护进程分析的次数。 |

#### SUMMARY\_STAT\_SYS\_TABLES

Vastbase内汇聚pg\_catalog、information\_schema以及pg\_toast模式下的所有系统表的统计信息。

**表 1** SUMMARY\_STAT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | numeric | 估计活跃行数。 |
| n\_dead\_tup | numeric | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | numeric | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | numeric | 这个表被autovacuum清理的次数。 |
| analyze\_count | numeric | 这个表被手动分析的次数。 |
| autoanalyze\_count | numeric | 这个表被autovacuum守护进程分析的次数。 |

#### GLOBAL\_STAT\_SYS\_TABLES

得到各节点pg\_catalog、information\_schema以及pg\_toast模式下的所有系统表的统计信息。

**表 1** GLOBAL\_STAT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 节点名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被手动分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |

#### STAT\_SYS\_INDEXES

显示pg\_catalog、information\_schema以及pg\_toast模式中所有系统表的索引状态信息。

**表 1** STAT\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 此索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引的模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### SUMMARY\_STAT\_SYS\_INDEXES

Vastbase内汇聚pg\_catalog、information\_schema以及pg\_toast模式中所有系统表的索引状态信息。

**表 1** SUMMARY\_STAT\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | numeric | 索引上开始的索引扫描数。 |
| idx\_tup\_read | numeric | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | numeric | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### GLOBAL\_STAT\_SYS\_INDEXES

得到各节点pg\_catalog、information\_schema以及pg\_toast模式中所有系统表的索引状态信息。

**表 1** GLOBAL\_STAT\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### STAT\_ALL\_TABLES

本节点内数据库中每个表（包括TOAST表）的一行的统计信息。

**表 1** STAT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次该表是手动清理的（不计算VACUUM FULL）的时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析该表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析时间。 |
| vacuum\_count | bigint | 该表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 该表被autovacuum清理的次数。 |
| analyze\_count | bigint | 该表被手动分析的次数。 |
| autoanalyze\_count | bigint | 该表被autovacuum守护进程分析的次数。 |

#### SUMMARY\_STAT\_ALL\_TABLES

Vastbase内汇聚数据库中每个表的一行（包括TOAST表）的统计信息。

**表 1** SUMMARY\_STAT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | numeric | 估计活跃行数。 |
| n\_dead\_tup | numeric | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）的时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析时间。 |
| vacuum\_count | numeric | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | numeric | 这个表被autovacuum清理的次数。 |
| analyze\_count | numeric | 这个表被手动分析的次数。 |
| autoanalyze\_count | numeric | 这个表被autovacuum守护进程分析的次数。 |

#### GLOBAL\_STAT\_ALL\_TABLES

得到各节点数据中每个表的一行（包括TOAST表）的统计信息。

**表 1** GLOBAL\_STAT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次此表是手动清理的（不计算VACUUM FULL）的时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析时间。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被手动分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |

#### STAT\_ALL\_INDEXES

将包含本节点数据库中的每个索引行，显示访问特定索引的统计。

**表 1** STAT\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### SUMMARY\_STAT\_ALL\_INDEXES

将包含Vastbase内汇聚数据库中的每个索引行，显示访问特定索引的统计。

**表 1** SUMMARY\_STAT\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | numeric | 索引上开始的索引扫描数。 |
| idx\_tup\_read | numeric | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | numeric | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### GLOBAL\_STAT\_ALL\_INDEXES

将包含各节点数据库中的每个索引行，显示访问特定索引的统计。

**表 1** GLOBAL\_STAT\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引中模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

#### STAT\_DATABASE

视图将包含本节点中每个数据库的统计信息。

**表 1** STAT\_DATABASE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 数据库的OID。 |
| datname | name | 此数据库的名称。 |
| numbackends | integer | 当前连接到该数据库的后端数。这是在返回一个反映目前状态值的视图中唯一的列；自上次重置所有其他列返回累积值。 |
| xact\_commit | bigint | 此数据库中已经提交的事务数。 |
| xact\_rollback | bigint | 此数据库中已经回滚的事务数。 |
| blks\_read | bigint | 在这个数据库中读取的磁盘块的数量。 |
| blks\_hit | bigint | 高速缓存中已经发现的磁盘块的次数，这样读取是不必要的（这只包括PostgreSQL缓冲区高速缓存，没有操作系统的文件系统缓存）。 |
| tup\_returned | bigint | 通过数据库查询返回的行数。 |
| tup\_fetched | bigint | 通过数据库查询抓取的行数。 |
| tup\_inserted | bigint | 通过数据库查询插入的行数。 |
| tup\_updated | bigint | 通过数据库查询更新的行数。 |
| tup\_deleted | bigint | 通过数据库查询删除的行数。 |
| conflicts | bigint | 由于数据库恢复冲突取消的查询数量（只在备用服务器发生的冲突）。请参见STAT\_DATABASE\_CONFLICTS章节获取更多信息。 |
| temp\_files | bigint | 通过数据库查询创建的临时文件数量。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不管log\_temp\_files设置。 |
| temp\_bytes | bigint | 通过数据库查询写入临时文件的数据总量。计算所有临时文件，不论为什么创建临时文件，而且不管log\_temp\_files设置。 |
| deadlocks | bigint | 在该数据库中检索的死锁数。 |
| blk\_read\_time | double precision | 通过数据库后端读取数据文件块花费的时间，以毫秒计算。 |
| blk\_write\_time | double precision | 通过数据库后端写入数据文件块花费的时间，以毫秒计算。 |
| stats\_reset | timestamp with time zone | 重置当前状态统计的时间。 |

#### SUMMARY\_STAT\_DATABASE

视图将包含数据库内汇聚的每个数据库的每一行，显示数据库统计。

**表 1** SUMMARY\_STAT\_DATABASE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datname | name | 这个数据库的名称。 |
| numbackends | bigint | 当前连接到该数据库的后端数。这是在返回一个反映目前状态值的视图中唯一的列；自上次重置所有其他列返回累积值。 |
| xact\_commit | numeric | 此数据库中已经提交的事务数。 |
| xact\_rollback | numeric | 此数据库中已经回滚的事务数。 |
| blks\_read | numeric | 在这个数据库中读取的磁盘块的数量。 |
| blks\_hit | numeric | 高速缓存中已经发现的磁盘块的次数，这样读取是不必要的（这只包括openGauss缓冲区高速缓存，没有操作系统的文件系统缓存）。 |
| tup\_returned | numeric | 通过数据库查询返回的行数。 |
| tup\_fetched | numeric | 通过数据库查询抓取的行数。 |
| tup\_inserted | bigint | 通过数据库查询插入的行数。 |
| tup\_updated | bigint | 通过数据库查询更新的行数。 |
| tup\_deleted | bigint | 通过数据库查询删除的行数。 |
| conflicts | bigint | 由于数据库恢复冲突取消的查询数量（只在备用服务器发生的冲突）。请参见STAT\_DATABASE\_CONFLICTS章节获取更多信息。 |
| temp\_files | numeric | 通过数据库查询创建的临时文件数量。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不管log\_temp\_files设置。 |
| temp\_bytes | numeric | 通过数据库查询写入临时文件的数据总量。计算所有临时文件，不论为什么创建临时文件，而且不管log\_temp\_files设置。 |
| deadlocks | bigint | 在该数据库中检索的死锁数。 |
| blk\_read\_time | double precision | 通过数据库后端读取数据文件块花费的时间，以毫秒计算。 |
| blk\_write\_time | double precision | 通过数据库后端写入数据文件块花费的时间，以毫秒计算。 |
| stats\_reset | timestamp with time zone | 重置当前状态统计的时间。 |

#### GLOBAL\_STAT\_DATABASE

视图将包含Vastbase中各节点的每个数据库的每一行，显示数据库统计。

**表 1** GLOBAL\_STAT\_DATABASE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| datid | oid | 数据库的OID。 |
| datname | name | 这个数据库的名称。 |
| numbackends | integer | 当前连接到该数据库的后端数。这是在返回一个反映目前状态值的视图中唯一的列；自上次重置所有其他列返回累积值。 |
| xact\_commit | bigint | 此数据库中已经提交的事务数。 |
| xact\_rollback | bigint | 此数据库中已经回滚的事务数。 |
| blks\_read | bigint | 在这个数据库中读取的磁盘块的数量。 |
| blks\_hit | bigint | 高速缓存中已经发现的磁盘块的次数，这样读取是不必要的（这只包括数据库内核缓冲区高速缓存，没有操作系统的文件系统缓存）。 |
| tup\_returned | bigint | 通过数据库查询返回的行数。 |
| tup\_fetched | bigint | 通过数据库查询抓取的行数。 |
| tup\_inserted | bigint | 通过数据库查询插入的行数。 |
| tup\_updated | bigint | 通过数据库查询更新的行数。 |
| tup\_deleted | bigint | 通过数据库查询删除的行数。 |
| conflicts | bigint | 由于数据库恢复冲突取消的查询数量（只在备用服务器发生的冲突）。请参见STAT\_DATABASE\_CONFLICTS获取更多信息。 |
| temp\_files | bigint | 通过数据库查询创建的临时文件数量。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不管log\_temp\_files设置。 |
| temp\_bytes | bigint | 通过数据库查询写入临时文件的数据总量。计算所有临时文件，不论为什么创建临时文件，而且不管log\_temp\_files设置。 |
| deadlocks | bigint | 在该数据库中检索的死锁数。 |
| blk\_read\_time | double precision | 通过数据库后端读取数据文件块花费的时间，以毫秒计算。 |
| blk\_write\_time | double precision | 通过数据库后端写入数据文件块花费的时间，以毫秒计算。 |
| stats\_reset | timestamp with time zone | 重置当前状态统计的时间。 |

#### STAT\_DATABASE\_CONFLICTS

显示当前节点数据库冲突状态的统计信息。

**表 1** STAT\_DATABASE\_CONFLICTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 数据库标识。 |
| datname | name | 数据库名称。 |
| confl\_tablespace | bigint | 冲突的表空间的数目。 |
| confl\_lock | bigint | 冲突的锁数目。 |
| confl\_snapshot | bigint | 冲突的快照数目。 |
| confl\_bufferpin | bigint | 冲突的缓冲区数目。 |
| confl\_deadlock | bigint | 冲突的死锁数目。 |

#### SUMMARY\_STAT\_DATABASE\_CONFLICTS

显示Vastbase内汇聚的数据库冲突状态的统计信息。

**表 1** SUMMARY\_STAT\_DATABASE\_CONFLICTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datname | name | 数据库名称。 |
| confl\_tablespace | bigint | 冲突的表空间的数目。 |
| confl\_lock | bigint | 冲突的锁数目。 |
| confl\_snapshot | bigint | 冲突的快照数目。 |
| confl\_bufferpin | bigint | 冲突的缓冲区数目。 |
| confl\_deadlock | bigint | 冲突的死锁数目。 |

#### GLOBAL\_STAT\_DATABASE\_CONFLICTS

显示每个节点的数据库冲突状态的统计信息。

**表 1** GLOBAL\_STAT\_DATABASE\_CONFLICTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| datid | oid | 数据库标识。 |
| datname | name | 数据库名称。 |
| confl\_tablespace | bigint | 冲突的表空间的数目。 |
| confl\_lock | bigint | 冲突的锁数目。 |
| confl\_snapshot | bigint | 冲突的快照数目。 |
| confl\_bufferpin | bigint | 冲突的缓冲区数目。 |
| confl\_deadlock | bigint | 冲突的死锁数目。 |

#### STAT\_XACT\_ALL\_TABLES

显示命名空间中所有普通表和toast表的事务状态信息。

**表 1** STAT\_XACT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### SUMMARY\_STAT\_XACT\_ALL\_TABLES

显示Vastbase内汇聚的命名空间中所有普通表和toast表的事务状态信息。

**表 1** SUMMARY\_STAT\_XACT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（比如没有更新所需的单独索引）。 |

#### GLOBAL\_STAT\_XACT\_ALL\_TABLES

显示各节点的命名空间中所有普通表和toast表的事务状态信息。

**表 1** GLOBAL\_STAT\_XACT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### STAT\_XACT\_SYS\_TABLES

显示当前节点命名空间中系统表的事务状态信息。

**表 1** STAT\_XACT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### SUMMARY\_STAT\_XACT\_SYS\_TABLES

显示Vastbase内汇聚的命名空间中系统表的事务状态信息。

**表 1** SUMMARY\_STAT\_XACT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（比如没有更新所需的单独索引）。 |

#### GLOBAL\_STAT\_XACT\_SYS\_TABLES

显示各节点命名空间中系统表的事务状态信息。

**表 1** GLOBAL\_STAT\_XACT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 节点名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### STAT\_XACT\_USER\_TABLES

显示当前节点命名空间中用户表的事务状态信息。

**表 1** STAT\_XACT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### SUMMARY\_STAT\_XACT\_USER\_TABLES

显示数据库内汇聚的命名空间中用户表的事务状态信息。

**表 1** SUMMARY\_STAT\_XACT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | numeric | 此表发起的顺序扫描数。 |
| seq\_tup\_read | numeric | 顺序扫描抓取的活跃行数。 |
| idx\_scan | numeric | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | numeric | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | numeric | 插入行数。 |
| n\_tup\_upd | numeric | 更新行数。 |
| n\_tup\_del | numeric | 删除行数。 |
| n\_tup\_hot\_upd | numeric | HOT更新行数（比如没有更新所需的单独索引）。 |

#### GLOBAL\_STAT\_XACT\_USER\_TABLES

显示各节点命名空间中用户表的事务状态信息。

**表 1** GLOBAL\_STAT\_XACT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| relid | oid | 表的OID。 |
| schemaname | name | 此表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 此表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 此表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

#### STAT\_XACT\_USER\_FUNCTIONS

视图包含当前节点本事务内函数执行的统计信息。

**表 1** STAT\_XACT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcid | oid | 函数标识。 |
| schemaname | name | 模式的名称。 |
| funcname | name | 函数名称。 |
| calls | bigint | 函数被调用的次数。 |
| total\_time | double precision | 函数的总执行时长。 |
| self\_time | double precision | 当前线程调用函数的总的时长。 |

#### SUMMARY\_STAT\_XACT\_USER\_FUNCTIONS

视图包含Vastbase内汇聚的本事务内函数执行的统计信息。

**表 1** SUMMARY\_STAT\_XACT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 模式的名称。 |
| funcname | name | 函数名称。 |
| calls | numeric | 函数被调用的次数。 |
| total\_time | double precision | 函数的总执行时长。 |
| self\_time | double precision | 当前线程调用函数的总的时长。 |

#### GLOBAL\_STAT\_XACT\_USER\_FUNCTIONS

视图包含各节点本事务内函数执行的统计信息。

**表 1** GLOBAL\_STAT\_XACT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 节点名称。 |
| funcid | oid | 函数标识。 |
| schemaname | name | 模式的名称。 |
| funcname | name | 函数名称。 |
| calls | bigint | 函数被调用的次数。 |
| total\_time | double precision | 此函数及其调用的所有其他函数所花费的总时间。 |
| self\_time | double precision | 在此函数本身中花费的总时间（不包括它调用的其他函数）。 |

#### STAT\_BAD\_BLOCK

获得当前节点表、索引等文件的读取失败信息。

**表 1** STAT\_BAD\_BLOCK字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 数据库进程名称。 |
| databaseid | integer | database的oid。 |
| tablespaceid | integer | tablespace的oid。 |
| relfilenode | integer | relation的file node。 |
| bucketid | smallint | 一致性hash bucket ID。 |
| forknum | integer | fork编号。 |
| error\_count | integer | error的数量。 |
| first\_time | timestamp with time zone | 坏块第一次出现的时间。 |
| last\_time | timestamp with time zone | 坏块最后出现的时间。 |

#### SUMMARY\_STAT\_BAD\_BLOCK

获得Vastbase内汇聚的表、索引等文件的读取失败信息。

**表 1** SUMMARY\_STAT\_BAD\_BLOCK字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| databaseid | integer | database的oid。 |
| tablespaceid | integer | tablespace的oid。 |
| relfilenode | integer | relation的file node。 |
| forknum | bigint | fork编号。 |
| error\_count | bigint | error的数量。 |
| first\_time | timestamp with time zone | 坏块第一次出现的时间。 |
| last\_time | timestamp with time zone | 坏块最后出现的时间。 |

#### GLOBAL\_STAT\_BAD\_BLOCK

获得各节点的表、索引等文件的读取失败信息。

**表 1** GLOBAL\_STAT\_BAD\_BLOCK字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 数据库进程名称。 |
| databaseid | integer | database的oid。 |
| tablespaceid | integer | tablespace的oid。 |
| relfilenode | integer | relation的file node。 |
| forknum | integer | fork编号。 |
| error\_count | integer | error的数量。 |
| first\_time | timestamp with time zone | 坏块第一次出现的时间。 |
| last\_time | timestamp with time zone | 坏块最后出现的时间。 |

#### STAT\_USER\_FUNCTIONS

STAT\_USER\_FUNCTIONS视图显示命名空间中用户自定义函数（函数语言为非内部语言）的状态信息。

**表 1** STAT\_USER\_FUNCTIONS字段

| 名称 | 类型 | 描述 |
| --- | --- | --- |
| funcid | oid | 函数标识。 |
| schemaname | name | schema的名称。 |
| funcname | name | 用户自定义函数的名称。 |
| calls | bigint | 函数被调用的次数。 |
| total\_time | double precision | 调用此函数花费的总时间，包含调用其它函数的时间（单位：毫秒）。 |
| self\_time | double precision | 调用此函数自己花费的时间，不包含调用其它函数的时间（单位：毫秒）。 |

#### SUMMARY\_STAT\_USER\_FUNCTIONS

SUMMARY\_STAT\_USER\_FUNCTIONS用来统计所数据库节点用户自定义函数的相关统计信息。

**表 1** SUMMARY\_STAT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | schema的名称。 |
| funcname | name | 用户function的名称。 |
| calls | numeric | 总调用次数。 |
| total\_time | double precision | 调用此function的总时间花费，包含调用其它function的时间（单位：毫秒）。 |
| self\_time | double precision | 调用此function自己时间的花费，不包含调用其它function的时间（单位：毫秒）。 |

#### GLOBAL\_STAT\_USER\_FUNCTIONS

提供Vastbase中各个节点的用户所创建的函数的状态的统计信息。

**表 1** GLOBAL\_STAT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| funcid | oid | 函数的id。 |
| schemaname | name | 此函数所在模式的名称。 |
| funcname | name | 函数名称。 |
| calls | bigint | 该函数被调用的次数。 |
| total\_time | double precision | 此函数及其调用的所有其他函数所花费的总时间（以毫秒为单位）。 |
| self\_time | double precision | 在此函数本身中花费的总时间（不包括它调用的其他函数），以毫秒为单位。 |

### Workload

#### WORKLOAD\_SQL\_COUNT

显示当前节点workload上的SQL数量分布。普通用户只可以看到自己在workload上的SQL分布；初始用户可以看到总的workload的负载情况。

**表 1** WORKLOAD\_SQL\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| workload | name | 负载名称。 |
| select\_count | bigint | select数量。 |
| update\_count | bigint | update数量。 |
| insert\_count | bigint | insert数量。 |
| delete\_count | bigint | delete数量。 |
| ddl\_count | bigint | ddl数量。 |
| dml\_count | bigint | dml数量。 |
| dcl\_count | bigint | dcl数量。 |

#### SUMMARY\_WORKLOAD\_SQL\_COUNT

显示Vastbase内各数据库主节点的workload上的SQL数量分布。

**表 1** SUMMARY\_WORKLOAD\_SQL\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| workload | name | 负载名称。 |
| select\_count | bigint | select数量。 |
| update\_count | bigint | update数量。 |
| insert\_count | bigint | insert数量。 |
| delete\_count | bigint | delete数量。 |
| ddl\_count | bigint | ddl数量。 |
| dml\_count | bigint | dml数量。 |
| dcl\_count | bigint | dcl数量。 |

#### WORKLOAD\_TRANSACTION

当前节点上负载的事务信息。

**表 1** WORKLOAD\_TRANSACTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| workload | name | 负载的名称。 |
| commit\_counter | bigint | 用户事务commit数量。 |
| rollback\_counter | bigint | 用户事务rollback数量。 |
| resp\_min | bigint | 用户事务最小响应时间（单位：微秒）。 |
| resp\_max | bigint | 用户事务最大响应时间（单位：微秒）。 |
| resp\_avg | bigint | 用户事务平均响应时间（单位：微秒）。 |
| resp\_total | bigint | 用户事务总响应时间（单位：微秒）。 |
| bg\_commit\_counter | bigint | 后台事务commit数量。 |
| bg\_rollback\_counter | bigint | 后台事务rollback数量。 |
| bg\_resp\_min | bigint | 后台事务最小响应时间（单位：微秒）。 |
| bg\_resp\_max | bigint | 后台事务最大响应时间（单位：微秒）。 |
| bg\_resp\_avg | bigint | 后台事务平均响应时间（单位：微秒）。 |
| bg\_resp\_total | bigint | 后台事务总响应时间（单位：微秒）。 |

#### SUMMARY\_WORKLOAD\_TRANSACTION

显示Vastbase内汇聚的负载事务信息。

**表 1** SUMMARY\_WORKLOAD\_TRANSACTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| workload | name | 负载的名称。 |
| commit\_counter | numeric | 用户事务commit数量。 |
| rollback\_counter | numeric | 用户事务rollback数量。 |
| resp\_min | bigint | 用户事务最小响应时间（单位：微秒）。 |
| resp\_max | bigint | 用户事务最大响应时间（单位：微秒）。 |
| resp\_avg | bigint | 用户事务平均响应时间（单位：微秒）。 |
| resp\_total | numeric | 用户事务总响应时间（单位：微秒）。 |
| bg\_commit\_counter | numeric | 后台事务commit数量。 |
| bg\_rollback\_counter | numeric | 后台事务rollback数量。 |
| bg\_resp\_min | bigint | 后台事务最小响应时间（单位：微秒）。 |
| bg\_resp\_max | bigint | 后台事务最大响应时间（单位：微秒）。 |
| bg\_resp\_avg | bigint | 后台事务平均响应时间（单位：微秒）。 |
| bg\_resp\_total | numeric | 后台事务总响应时间（单位：微秒）。 |

#### GLOBAL\_WORKLOAD\_TRANSACTION

显示各节点上的workload的负载信息。

**表 1** GLOBAL\_WORKLOAD\_TRANSACTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| workload | name | 负载的名称。 |
| commit\_counter | bigint | 用户事务commit数量。 |
| rollback\_counter | bigint | 用户事务rollback数量。 |
| resp\_min | bigint | 用户事务最小响应时间（单位：微秒）。 |
| resp\_max | bigint | 用户事务最大响应时间（单位：微秒）。 |
| resp\_avg | bigint | 用户事务平均响应时间（单位：微秒）。 |
| resp\_total | bigint | 用户事务总响应时间（单位：微秒）。 |
| bg\_commit\_counter | bigint | 后台事务commit数量。 |
| bg\_rollback\_counter | bigint | 后台事务rollback数量。 |
| bg\_resp\_min | bigint | 后台事务最小响应时间（单位：微秒）。 |
| bg\_resp\_max | bigint | 后台事务最大响应时间（单位：微秒）。 |
| bg\_resp\_avg | bigint | 后台事务平均响应时间（单位：微秒）。 |
| bg\_resp\_total | bigint | 后台事务总响应时间（单位：微秒）。 |

#### WORKLOAD\_SQL\_ELAPSE\_TIME

WORKLOAD\_SQL\_ELAPSE\_TIME用来统计workload（业务负载）上的SUID信息。

**表 1** WORKLOAD\_SQL\_ELAPSE\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| workload | name | workload（业务负载）名称。 |
| total\_select\_elapse | bigint | 总select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费（单位：微秒）。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| total\_update\_elapse | bigint | 总update的时间花费（单位：微秒）。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | bigint | 总insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | bigint | 总delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |

#### SUMMARY\_WORKLOAD\_SQL\_ELAPSE\_TIME

SUMMARY\_WORKLOAD\_SQL\_ELAPSE\_TIME用来统计数据库主节点上workload（业务）负载的SUID信息。

**表 1** SUMMARY\_WORKLOAD\_SQL\_ELAPSE\_TIM字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 数据库进程名称。 |
| workload | name | workload（业务负载）名称。 |
| total\_select\_elapse | bigint | 总select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费（单位：微秒）。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| total\_update\_elapse | bigint | 总update的时间花费（单位：微秒）。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | bigint | 总insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | bigint | 总delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |

#### USER\_TRANSACTION

USER\_TRANSACTION用来统计用户执行的事务信息。monadmin用户能看到所有用户执行事务的信息，普通用户只能查询到自己执行的事务信息。

**表 1** USER\_TRANSACTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| username | name | 用户的名称。 |
| commit\_counter | bigint | 用户事务commit数量。 |
| rollback\_counter | bigint | 用户事务rollback数量。 |
| resp\_min | bigint | 用户事务最小响应时间（单位：微秒）。 |
| resp\_max | bigint | 用户事务最大响应时间（单位：微秒）。 |
| resp\_avg | bigint | 用户事务平均响应时间（单位：微秒）。 |
| resp\_total | bigint | 用户事务总响应时间（单位：微秒）。 |
| bg\_commit\_counter | bigint | 后台事务commit数量。 |
| bg\_rollback\_counter | bigint | 后台事务rollback数量。 |
| bg\_resp\_min | bigint | 后台事务最小响应时间（单位：微秒）。 |
| bg\_resp\_max | bigint | 后台事务最大响应时间（单位：微秒）。 |
| bg\_resp\_avg | bigint | 后台事务平均响应时间（单位：微秒）。 |
| bg\_resp\_total | bigint | 后台事务总响应时间（单位：微秒）。 |

#### GLOBAL\_USER\_TRANSACTION

GLOBAL\_USER\_TRANSACTION用来统计全局用户执行的事务信息。

**表 1** GLOBAL\_USER\_TRANSACTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | name | 节点名称。 |
| username | name | 用户的名称。 |
| commit\_counter | bigint | 用户事务commit数量。 |
| rollback\_counter | bigint | 用户事务rollback数量。 |
| resp\_min | bigint | 用户事务最小响应时间（单位：微秒）。 |
| resp\_max | bigint | 用户事务最大响应时间（单位：微秒）。 |
| resp\_avg | bigint | 用户事务平均响应时间（单位：微秒）。 |
| resp\_total | bigint | 用户事务总响应时间（单位：微秒）。 |
| bg\_commit\_counter | bigint | 后台事务commit数量。 |
| bg\_rollback\_counter | bigint | 后台事务rollback数量。 |
| bg\_resp\_min | bigint | 后台事务最小响应时间（单位：微秒）。 |
| bg\_resp\_max | bigint | 后台事务最大响应时间（单位：微秒）。 |
| bg\_resp\_avg | bigint | 后台事务平均响应时间（单位：微秒）。 |
| bg\_resp\_total | bigint | 后台事务总响应时间（单位：微秒）。 |

## Information Schema

信息模式本身是一个名为information\_schema的模式。这个模式自动存在于所有数据库中。信息模式由一组视图构成，它们包含定义在当前数据库中对象的信息。这个模式的拥有者是初始数据库用户，并且该用户自然地拥有这个模式上的所有特权，包括删除它的能力。

### ADMINISTRABLE\_ROLE\_AUTHORIZATIONS

administrable\_role\_authorizations标识了当前用户拥有 admin 选项的所有角色。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| 受赠人 | sql\_identifier | 授予此角色成员资格的角色的名称（可以是当前用户，或者在嵌套角色成员资格的情况下是不同的角色） |
| 角色名称 | sql\_identifier | 角色名称 |
| is\_grantable | yes or no | 总是yes |

### APPLICABLE\_ROLES

标识当前用户可以使用其权限的所有角色。当前用户本身也是一个适用的角色。适用角色的集合通常用于权限检查。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| 被授予用户 | sql\_identifier | 授予此角色成员资格的角色的名称（可以是当前用户，或者在嵌套角色成员资格的情况下是不同的角色） |
| 角色名称 | sql\_identifier | 角色名称 |
| is\_grantable | yes or no | YES，授予者对角色有管理员选项。NO，没有。 |

### ATTRIBUTES

包含有关数据库中定义的复合数据类型的属性的信息。（该视图不提供有关表列的信息，这些信息有时在 PostgreSQL 上下文中称为属性。）仅显示当前用户可以访问的那些属性（通过成为该类型的所有者或对该类型具有某些特权） 。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| outt\_catalog | sql\_identifier | 包含数据类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 包含数据类型的模式名称。 |
| udt\_name | sql\_identifier | 数据类型的名称。 |
| attribute\_name | sql\_identifier | 属性名称。 |
| ordinal\_position | cardinal\_number | 数据类型中属性的序号位置（计数从 1 开始）。 |
| attribute\_default | character\_data | 属性的默认表达式。 |
| is\_nullable | yes\_or\_no | 如果属性可能为空，则为YES，如果已知它不可为空，则为NO 。 |
| data\_type | character\_data | 属性的数据类型，如果它是内置类型，或者ARRAY如果它是某个数组（在这种情况下，请参见视图element\_types），否则为 USER-DEFINED （在这种情况下，类型在attribute\_udt\_name中标识 并关联列）。 |
| character\_maximum\_length | cardinal\_number | 如果data\_type标识字符或位串类型，则声明最大长度；对于所有其他数据类型，或者如果没有声明最大长度，则为 null。 |
| character\_octet\_length | cardinal\_number | 如果data\_type标识字符类型，则以八位字节（字节）为单位的数据的最大可能长度；所有其他数据类型为 null。最大八位字节长度取决于声明的字符最大长度（见上文）和服务器编码。 |
| character\_set\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能 |
| character\_set\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能 |
| character\_set\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能 |
| collation\_catalog | sql\_identifier | 包含属性排序规则的数据库的名称（始终为当前数据库），如果默认或属性的数据类型不可排序则为 null |
| collation\_schema | sql\_identifier | 包含属性排序规则的模式的名称，如果默认或属性的数据类型不可排序则为 null |
| collation\_name | sql\_identifier | 属性排序规则的名称，默认为 null 或者属性的数据类型不可排序 |
| numeric\_precision | cardinal\_number | 如果data\_type标识数字类型，则此列包含此属性的类型的（声明的或隐含的）精度。精度表示有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| numeric\_precision\_radix | cardinal\_number | 如果data\_type标识数字类型，则此列指示numeric\_precision和numeric\_scale列中的值以哪个基表示。该值为 2 或 10。对于所有其他数据类型，此列为空。 |
| numeric\_scale | cardinal\_number | 如果data\_type标识一个精确的数字类型，则此列包含此属性的类型的（声明的或隐含的）比例。刻度表示小数点右侧的有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| datetime\_precision | cardinal\_number | 如果data\_type标识日期、时间、时间戳或间隔类型，则此列包含此属性类型的（声明的或隐含的）小数秒精度，即秒值中小数点后保留的小数位数. 对于所有其他数据类型，此列为空。 |
| interval\_type | character\_data | 如果data\_type标识一个区间类型，则此列包含该属性的区间包含哪些字段的规范，例如YEAR TO MONTH、DAY TO SECOND等。如果未指定字段限制（即，区间接受所有字段），对于所有其他数据类型，此字段为空。 |
| interval\_precision | cardinal\_number | 适用于PostgreSQL中不可用的功能（请参阅datetime\_precision了解间隔类型属性的小数秒精度）。 |
| attribute\_udt\_catalog | sql\_identifier | 定义属性数据类型的数据库的名称（始终为当前数据库）。 |
| attribute\_udt\_schema | sql\_identifier | 定义属性数据类型的模式的名称。 |
| attribute\_udt\_name | sql\_identifier | 属性数据类型的名称。 |
| scope\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为 null，因为数组在PostgreSQL中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 列的数据类型描述符的标识符，在与表有关的数据类型描述符中唯一。这主要用于与此类标识符的其他实例连接（标识符的具体格式没有定义，也不保证在以后的版本中保持不变）。 |
| is\_derived\_reference\_attribute | yes\_or\_no | 适用于PostgreSQL中不可用的功能 |

### CHARACTER\_SETS

character\_sets标识当前数据库中可用的字符集。由于 Vastbase 不支持一个数据库中的多个字符集，因此该视图仅显示一个，即数据库编码。

* 角色曲目：字符的抽象集合，例如 UNICODE、UCS或LATIN1。不作为 SQL 对象公开，但在此视图中可见。
* 字符编码形式：一些字符库的编码。大多数较旧的字符集仅使用一种编码形式，因此它们没有单独的名称（例如，LATIN1是适用于 LATIN1集的编码形式）。但例如 Unicode 有UTF8、UTF16等编码形式（PostgreSQL 并不都支持）。编码表单不作为 SQL 对象公开，但在此视图中可见。
* 字符集：一个命名的 SQL 对象，用于标识字符库、字符编码和默认排序规则。预定义的字符集通常与编码形式具有相同的名称，但用户可以定义其他名称。例如，字符集UTF8通常会识别字符集UCS、编码形式 UTF8和一些默认排序规则。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| character\_set\_catalog | sql\_identifier | 字符集当前未实现为模式对象，因此该列为空。 |
| character\_set\_schema | sql\_identifier | 字符集当前未实现为模式对象，因此该列为空。 |
| character\_set\_name | sql\_identifier | 字符集名称，目前实现为显示数据库编码的名称。 |
| character\_repertoire | sql\_identifier | 字符集，如果编码为UTF8则显示UCS，否则仅显示编码名称。 |
| form\_of\_use | sql\_identifier | 字符编码形式，与数据库编码相同。 |
| default\_collate\_catalog | sql\_identifier | 包含默认排序规则的数据库的名称（始终是当前数据库，如果标识了任何排序规则）。 |
| default\_collate\_schema | sql\_identifier | 包含默认排序规则的架构名称。 |
| default\_collate\_name | sql\_identifier | 默认排序规则的名称。默认排序规则被标识为与 当前数据库的COLLATE和CTYPE设置匹配的排序规则。如果没有这样的排序规则，则此列以及关联的架构和目录列为空。 |

### CHECK\_CONSTRAINT\_ROUTINE\_USAGE

标识检查约束使用的例程（函数和过程）。仅显示当前启用的角色拥有的那些例程。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库） |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称 |
| constraint\_name | sql\_identifier | 约束的名称 |
| specific\_catalog | sql\_identifier | 包含函数的数据库的名称（始终为当前数据库） |
| specific\_schema | sql\_identifier | 包含函数的模式的名称 |
| specific\_name | sql\_identifier | 函数的“特定名称”。 |

### CHECK\_CONSTRAINTS

包含所有检查约束，定义在表或域上，由当前启用的角色拥有（表或域的所有者是约束的所有者）。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |
| check\_clause | character\_data | 检查约束的检查表达式。 |

### COLLATIONS

视图collations包含当前数据库中可用的排序规则。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| collation\_catalog | sql\_identifier | 包含排序规则的数据库的名称（始终为当前数据库）。 |
| collation\_schema | sql\_identifier | 包含排序规则的架构名称。 |
| collation\_name | sql\_identifier | 默认排序规则的名称。 |
| pad\_attribute | character\_data | 始终没有 PAD （ PostgreSQL 不支持替代的PAD SPACE ）。 |

### COLLATION\_CHARACTER\_SET\_APPLICABILITY

标识可用排序规则适用于哪个字符集。在Vastbase中，每个数据库只有一个字符集，所以这个视图没有提供太多有用的信息。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| collation\_catalog | sql\_identifier | 包含排序规则的数据库的名称（始终为当前数据库）。 |
| collation\_schema | sql\_identifier | 包含排序规则的架构名称。 |
| collation\_name | sql\_identifier | 默认排序规则的名称。 |
| character\_set\_catalog | sql\_identifier | 字符集当前未实现为模式对象，因此该列为 null。 |
| character\_set\_schema | sql\_identifier | 字符集当前未实现为模式对象，因此该列为 null。 |
| character\_set\_name | sql\_identifier | 字符集的名称。 |

### COLUMN\_DOMAIN\_USAGE

标识所有列（表或视图的），这些列使用当前数据库中定义的并由当前启用的角色拥有的某些域。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| domain\_catalog | sql\_identifier | 包含域的数据库的名称（始终是当前数据库）。 |
| domain\_schema | sql\_identifier | 包含域的架构名称。 |
| domain\_name | sql\_identifier | 域名。 |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| column\_name | sql\_identifier | 列名。 |

### COLUMN\_OPTIONS

包含为当前数据库中的外部表列定义的所有选项。仅显示当前用户有权访问的那些外部表列（通过成为所有者或具有某些特权）。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含外部表的数据库的名称（始终是当前数据库）。 |
| table\_schema | sql\_identifier | 包含外部表的模式的名称。 |
| table\_name | sql\_identifier | 外部表的名称。 |
| column\_name | sql\_identifier | 列名。 |
| option\_name | sql\_identifier | 选项名称。 |
| option\_value | 字符数据 | 期权价值。 |

### COLUNM\_PRIVILEGES

标识在列上授予当前启用的角色或当前启用的角色的所有权限。列、授予者和被授予者的每个组合都有一行。如果已在整个表上授予权限，它将在此视图中显示为对每一列的授予，但仅适用于列粒度可能的权限类型：SELECT、INSERT、 UPDATE、REFERENCES。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 授予权限的角色的名称。 |
| grantee | sql\_identifier | 授予权限的角色的名称。 |
| table\_catalog | sql\_identifier | 包含该列的表的数据库名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含包含该列的表的架构的名称。 |
| table\_name | sql\_identifier | 包含该列的表的名称。 |
| column\_name | sql\_identifier | 列名。 |
| privilege\_type | character\_data | 权限类型：SELECT、INSERT、 UPDATE或REFERENCES。 |
| is\_grantable | yes\_or\_no | 如果特权是可授予的，则为YES ，否则为NO。 |

### COLUMN\_UDT\_USAGE

标识使用当前启用的角色拥有的数据类型的所有列。请注意，在Vastbase中，内置数据类型的行为类似于用户定义的类型，因此它们也包含在此处。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| outt\_catalog | sql\_identifier | 定义列数据类型（域的基础类型，如果适用）的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 列数据类型（域的基础类型，如果适用）在其中定义的模式的名称。 |
| udt\_name | sql\_identifier | 列数据类型的名称（域的基础类型，如果适用）。 |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| column\_name | sql\_identifier | 列名。 |

### COLUMNS

包含有关数据库中所有表列（或视图列）的信息。不包括系统列（oid等）。仅显示当前用户有权访问的那些列（通过成为所有者或具有某些特权）。

由于数据类型可以在 SQL 中以多种方式定义，并且Vastbase包含定义数据类型的其他方法，因此它们在信息模式中的表示可能有些困难。列 data\_type应该标识列的底层内置类型。在Vastbase中，这意味着类型是在系统目录模式pg\_catalog中定义的。如果应用程序可以专门处理众所周知的内置类型（例如，以不同的方式格式化数字类型或使用精度列中的数据），则此列可能很有用。udt\_name、udt\_schema和 udt\_catalog列始终标识列的基础数据类型，即使列基于域。（由于Vastbase 将内置类型视为用户定义类型，因此此处也出现内置类型。这是 SQL 标准的扩展。）如果应用程序希望根据类型以不同方式处理数据，则应使用这些列，因为在这种情况下，列是否真的基于域并不重要。如果列基于域，则域的标识存储在domain\_name、domain\_schema和domain\_catalog列中。如果需要将列与其关联的数据类型配对并将域视为单独的类型，可以编写coalesce(domain\_name, udt\_name)等。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| column\_name | sql\_identifier | 列名。 |
| ordinal\_position | cardinal\_number | 表中列的序号位置（计数从 1 开始）。 |
| column\_default | character\_data | 列的默认表达式。 |
| is\_nullable | yes\_or\_no | 如果该列可能为空，则为YES ，如果已知它不可为空，则为NO 。非空约束是已知列不可为空的一种方式，但也可以有其他方式。 |
| data\_type | character\_data | 列的数据类型，如果它是内置类型，或者 ARRAY如果它是某个数组（在这种情况下，请参阅视图element\_types），否则为 USER-DEFINED（在这种情况下，类型在udt\_name中标识并关联列）。如果该列基于域，则该列引用该域的基础类型（并且该域在domain\_name和相关列中标识）。 |
| character\_maximum\_length | cardinal\_number | 如果data\_type标识字符或位串类型，则声明最大长度；对于所有其他数据类型，或者如果没有声明最大长度，则为 null。 |
| character\_octet\_length | cardinal\_number | 如果data\_type标识字符类型，则以八位字节（字节）为单位的数据的最大可能长度；所有其他数据类型为 null。最大八位字节长度取决于声明的字符最大长度（见上文）和服务器编码。 |
| numeric\_precision | cardinal\_number | 如果data\_type标识数字类型，则此列包含此列类型的（声明的或隐含的）精度。精度表示有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| numeric\_precision\_radix | cardinal\_number | 如果data\_type标识数字类型，则此列指示numeric\_precision和numeric\_scale列中的值以哪个基表示。该值为 2 或 10。对于所有其他数据类型，此列为空。 |
| numeric\_scale | cardinal\_number | 如果data\_type标识了一个精确的数字类型，则此列包含此列的类型的（声明的或隐含的）比例。刻度表示小数点右侧的有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| datetime\_precision | cardinal\_number | 如果data\_type标识日期、时间、时间戳或间隔类型，则此列包含此列类型的（声明的或隐含的）小数秒精度，即秒值中小数点后保留的小数位数. 对于所有其他数据类型，此列为空。 |
| interval\_type | character\_data | 如果data\_type标识一个区间类型，则此列包含该列的区间包含哪些字段的规范，例如 YEAR TO MONTH、DAY TO SECOND等。如果未指定字段限制（即，区间接受所有字段），对于所有其他数据类型，此字段为空。 |
| interval\_precision | cardinal\_number | 适用于Vastbase中不可用的功能（请参阅datetime\_precision了解间隔类型列的小数秒精度）。 |
| character\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_catalog | sql\_identifier | 包含列排序规则的数据库的名称（始终为当前数据库），默认为 null 或列的数据类型不可排序。 |
| collation\_schema | sql\_identifier | 包含列排序规则的架构名称，默认为 null 或列的数据类型不可排序。 |
| collation\_name | sql\_identifier | 列的排序规则名称，默认为 null 或列的数据类型不可排序。 |
| domain\_catalog | sql\_identifier | 如果列具有域类型，则为定义域的数据库的名称（始终为当前数据库），否则为 null。 |
| domain\_schema | sql\_identifier | 如果该列具有域类型，则为定义域的模式的名称，否则为 null。 |
| domain\_name | sql\_identifier | 如果该列具有域类型，则为域的名称，否则为 null。 |
| outt\_catalog | sql\_identifier | 定义列数据类型（域的基础类型，如果适用）的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 列数据类型（域的基础类型，如果适用）在其中定义的模式的名称。 |
| udt\_name | sql\_identifier | 列数据类型的名称（域的基础类型，如果适用）。 |
| scope\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为 null，因为数组在Vastbase中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 列的数据类型描述符的标识符，在与表有关的数据类型描述符中唯一。这主要用于与此类标识符的其他实例连接。（标识符的具体格式没有定义，也不保证在以后的版本中保持不变）。 |
| is\_self\_reference | yes\_or\_no | Vastbase中不可用的功能。 |
| is\_identity | yes\_or\_no | Vastbase中不可用的功能。 |
| identity\_generation | character\_data | Vastbase中不可用的功能。 |
| identity\_start | character\_data | Vastbase中不可用的功能。 |
| identity\_increment | character\_data | Vastbase中不可用的功能。 |
| identity\_maximum | character\_data | Vastbase中不可用的功能。 |
| identity\_minimum | character\_data | Vastbase中不可用的功能。 |
| identity\_cycle | yes\_or\_no | Vastbase中不可用的功能。 |
| is\_generated | character\_data | Vastbase中不可用的功能。 |
| generation\_expression | character\_data | Vastbase中不可用的功能。 |
| is\_updateable | yes\_or\_no | YES如果列是可更新的，NO如果不是（基表中的列总是可更新的，视图中的列不一定）。 |

### CONSTRAINT\_COLUMN\_USAGE

标识当前数据库中由某个约束使用的所有列。仅显示当前启用的角色拥有的表中包含的那些列。对于检查约束，此视图标识检查表达式中使用的列。对于外键约束，此视图标识外键引用的列。对于唯一或主键约束，此视图标识受约束的列。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含某个约束使用的列的表的数据库的名称（始终是当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称，该表包含某些约束使用的列。 |
| table\_name | sql\_identifier | 包含某个约束使用的列的表的名称。 |
| column\_name | sql\_identifier | 某些约束使用的列的名称。 |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |

### CONSTRAINT\_TABLE\_USAGE

标识当前数据库中由某个约束使用并由当前启用的角色拥有的所有表。（这与视图table\_constraints不同，后者标识所有表约束以及定义它们的表。）对于外键约束，此视图标识外键引用的表。对于唯一键或主键约束，此视图仅标识约束所属的表。此视图中不包括检查约束和非空约束。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含某些约束使用的表的数据库的名称（始终是当前数据库）。 |
| table\_schema | sql\_identifier | 包含由某些约束使用的表的模式的名称。 |
| table\_name | sql\_identifier | 某些约束使用的表的名称。 |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |

### DATA\_TYPE\_PRIVILEGES

标识当前用户可以访问的所有数据类型描述符，方法是作为所描述对象的所有者或对其具有某些特权。每当在表列、域或函数的定义中使用数据类型（作为参数或返回类型）时，都会生成数据类型描述符，并存储有关该数据类型在该实例中如何使用的一些信息（例如，声明的最大长度，如果适用）。每个数据类型描述符都被分配了一个任意标识符，该标识符在为一个对象（表、域、函数）分配的数据类型描述符标识符中是唯一的。此视图可能对应用程序没有用处，但它用于定义信息模式中的一些其他视图。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| object\_catalog | sql\_identifier | 包含所描述对象的数据库的名称（始终为当前数据库）。 |
| object\_schema | sql\_identifier | 包含所描述对象的模式的名称。 |
| object\_name | sql\_identifier | 描述对象的名称。 |
| object\_type | character\_data | 所描述对象的类型：TABLE（数据类型描述符属于该表的列）、DOMAIN（数据类型描述符属于该域）、ROUTINE（数据类型描述符属于参数或返回数据）之一该函数的类型）。 |
| dtd\_identifier | sql\_identifier | 数据类型描述符的标识符，在同一对象的数据类型描述符中是唯一的。 |

### DOMAIN\_CONSTRAINTS

包含属于当前数据库中定义的域的所有约束。仅显示当前用户有权访问的那些域（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |
| domain\_catalog | sql\_identifier | 包含域的数据库的名称（始终为当前数据库）。 |
| domain\_schema | sql\_identifier | 包含域的架构的名称。 |
| domain\_name | sql\_identifier | 域名。 |
| is\_deferrable | yes\_or\_no | 如果约束是可延迟的，则为YES ，否则为NO。 |
| initially\_deferred | yes\_or\_no | 如果约束是可延迟的并且最初是延迟的，则为YES ，否则为NO。 |

### DOMAIN\_UDT\_USAGE

标识基于当前启用角色拥有的数据类型的所有域。请注意，在Vastbase中，内置数据类型的行为类似于用户定义的类型，因此它们也包含在此处。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| outt\_catalog | sql\_identifier | 定义域数据类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 定义域数据类型的模式的名称。 |
| udt\_name | sql\_identifier | 域数据类型的名称。 |
| domain\_catalog | sql\_identifier | 包含域的数据库的名称（始终为当前数据库）。 |
| domain\_schema | sql\_identifier | 包含域的架构的名称。 |
| domain\_name | sql\_identifier | 域名。 |

### DOMAINS

包含当前数据库中定义的所有域。仅显示当前用户有权访问的那些域（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| domain\_catalog | sql\_identifier | 包含域的数据库的名称（始终为当前数据库）。 |
| domain\_schema | sql\_identifier | 包含域的架构的名称。 |
| domain\_name | sql\_identifier | 域名。 |
| data\_type | character\_data | 域的数据类型，如果它是内置类型，或者 ARRAY，如果它是某个数组（在这种情况下，请参见视图element\_types），否则为 USER-DEFINED（在这种情况下，类型在udt\_name中标识并关联列）。 |
| character\_maximum\_length | cardinal\_number | 如果域有字符或位串类型，则声明最大长度；对于所有其他数据类型，或者如果没有声明最大长度，则为 null。 |
| character\_octet\_length | cardinal\_number | 如果域具有字符类型，则以八位字节（字节）为单位的数据的最大可能长度；所有其他数据类型为 null。最大八位字节长度取决于声明的字符最大长度（见上文）和服务器编码。 |
| character\_set\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| collation\_catalog | sql\_identifier | 包含域排序规则的数据库的名称（始终为当前数据库），默认为 null 或域的数据类型不可排序。 |
| collation\_schema | sql\_identifier | 包含域排序规则的架构名称，默认为 null 或域的数据类型不可排序。 |
| collation\_name | sql\_identifier | 域的排序规则的名称，默认为空或域的数据类型不可排序。 |
| numeric\_precision | cardinal\_number | 如果域具有数字类型，则此列包含此域的类型的（声明的或隐含的）精度。精度表示有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如 numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| numeric\_precision\_radix | cardinal\_number | 如果域具有数字类型，则此列指示numeric\_precision和numeric\_scale列中的值在哪个基 中表示。该值为 2 或 10。对于所有其他数据类型，此列为空。 |
| numeric\_scale | cardinal\_number | 如果域具有精确的数字类型，则此列包含此域的类型的（声明的或隐含的）比例。刻度表示小数点右侧的有效位数。它可以用十进制（以 10 为底）或二进制（以 2 为底）表示，如numeric\_precision\_radix列中指定的那样。对于所有其他数据类型，此列为空。 |
| datetime\_precision | cardinal\_number | 如果data\_type标识日期、时间、时间戳或间隔类型，则此列包含此域的类型的（声明的或隐含的）小数秒精度，即秒值中小数点后保留的小数位数. 对于所有其他数据类型，此列为空。 |
| interval\_type | character\_data | 如果data\_type标识一个区间类型，则此列包含该区间包含的字段的规范，例如 YEAR TO MONTH、DAY TO SECOND等。如果未指定字段限制（即，区间接受所有字段），对于所有其他数据类型，此字段为空。 |
| interval\_precision | cardinal\_number | 适用于PostgreSQL中不可用的功能（请参阅datetime\_precision了解间隔类型域的小数秒精度）。 |
| domain\_default | character\_data | 域的默认表达式。 |
| outt\_catalog | sql\_identifier | 定义域数据类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 定义域数据类型的模式的名称。 |
| udt\_name | sql\_identifier | 域数据类型的名称。 |
| scope\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为 null，因为数组在PostgreSQL中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 域的数据类型描述符的标识符，在与域有关的数据类型描述符中唯一（这是微不足道的，因为域仅包含一个数据类型描述符）。这主要用于与此类标识符的其他实例连接（标识符的具体格式没有定义，也不保证在以后的版本中保持不变）。 |

### ELEMENT\_TYPES

包含数组元素的数据类型描述符。当表列、复合类型属性、域、函数参数或函数返回值被定义为数组类型时，相应的information schema视图仅在data\_type列中包含ARRAY。要获取有关数组元素类型的信息，您可以将相应的视图与此视图连接起来。例如，要显示具有数据类型和数组元素类型的表的列，如果适用，您可以执行以下操作：

SELECT c.column\_name, c.data\_type, e.data\_type AS element\_type  
FROM information\_schema.columns c LEFT JOIN information\_schema.element\_types e  
 ON ((c.table\_catalog, c.table\_schema, c.table\_name, 'TABLE', c.dtd\_identifier)  
 = (e.object\_catalog, e.object\_schema, e.object\_name, e.object\_type, e.collection\_type\_identifier))  
WHERE c.table\_schema = '...' AND c.table\_name = '...'  
ORDER BY c.ordinal\_position;

此视图仅包括当前用户通过成为所有者或具有某些特权而可以访问的对象。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| object\_catalog | sql\_identifier | 包含使用所描述数组的对象的数据库的名称（始终为当前数据库）。 |
| object\_schema | sql\_identifier | 包含使用正在描述的数组的对象的模式的名称。 |
| object\_name | sql\_identifier | 使用正在描述的数组的对象的名称。 |
| object\_type | character\_data | 使用所描述数组的对象的类型：TABLE（该数组由该表的列使用）、USER-DEFINED TYPE（该数组由该复合类型的属性使用）、DOMAIN（该数组由该域使用）， ROUTINE（该数组由参数或该函数的返回数据类型使用）。 |
| collection\_type\_identifier | sql\_identifier | 正在描述的数组的数据类型描述符的标识符。使用它来连接其他信息模式视图的 dtd\_identifier列。 |
| data\_type | character\_data | 数组元素的数据类型，如果是内置类型，则为USER-DEFINED（在这种情况下，类型在udt\_name和相关列中标识）。 |
| character\_maximum\_length | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| character\_octet\_length | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| character\_set\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| collation\_catalog | sql\_identifier | 包含元素类型排序规则的数据库名称（始终为当前数据库），如果默认为 null 或元素的数据类型不可排序。 |
| collation\_schema | sql\_identifier | 包含元素类型排序规则的模式名称，默认为 null 或元素的数据类型不可排序。 |
| collation\_name | sql\_identifier | 元素类型的排序规则名称，默认为 null 或者元素的数据类型不可排序。 |
| numeric\_precision | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| numeric\_precision\_radix | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| numeric\_scale | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| datetime\_precision | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| interval\_type | character\_data | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| interval\_precision | cardinal\_number | 始终为 null，因为此信息不适用于PostgreSQL中的数组元素数据类型。 |
| domain\_default | character\_data | 尚未实现。 |
| outt\_catalog | sql\_identifier | 定义元素数据类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 元素的数据类型在其中定义的模式的名称。 |
| udt\_name | sql\_identifier | 元素的数据类型名称。 |
| scope\_catalog | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于PostgreSQL中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为 null，因为数组在PostgreSQL中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 元素的数据类型描述符的标识符。这目前没有用。 |

### ENABLED\_ROLES

标识当前"启用的角色"。启用的角色被递归地定义为当前用户以及所有已授予具有自动继承属性的启用角色的角色。这些都是当前用户直接或间接拥有的所有角色，自动继承其成员资格。

对于权限检查，应用"适用角色"的集合，它可以比启用的角色集合更广泛。所以一般来说，最好使用applicable\_roles。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| role\_name | sql\_identifier | 角色名称 |

### FOREIGN\_DATA\_WRAPPER\_OPTIONS

包含为当前数据库中的外部数据包装器定义的所有选项。仅显示当前用户有权访问的那些外部数据包装器（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_data\_wrapper\_catalog | sql\_identifier | 定义外部数据包装器的数据库的名称（始终为当前数据库）。 |
| foreign\_data\_wrapper\_name | sql\_identifier | 外部数据包装器的名称。 |
| option\_name | sql\_identifier | 选项名称。 |
| option\_value | character\_data | 期权价值。 |

### FOREIGN\_DATA\_WRAPPERS

包含当前数据库中定义的所有外部数据包装器。仅显示当前用户有权访问的那些外部数据包装器（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_data\_wrapper\_catalog | sql\_identifier | 包含外部数据包装器的数据库的名称（始终为当前数据库）。 |
| foreign\_data\_wrapper\_name | sql\_identifier | 外部数据包装器的名称。 |
| authorization\_identifier | sql\_identifier | 国外服务器所有者的名字。 |
| library\_name | character\_data | 实现此外部数据包装器的库的文件名。 |
| foreign\_data\_wrapper\_language | character\_data | 用于实现此外部数据包装器的语言。 |

### FOREIGN\_SERVER\_OPTIONS

包含为当前数据库中的外部服务器定义的所有选项。仅显示当前用户有权访问的那些外部服务器（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_server\_catalog | sql\_identifier | 定义外部服务器的数据库的名称（始终为当前数据库）。 |
| foreign\_server\_name | sql\_identifier | 国外服务器名称。 |
| option\_name | sql\_identifier | 选项名称。 |
| option\_value | character\_data | 选项值。 |

### FOREIGN\_SERVERS

包含当前数据库中定义的所有外部服务器。仅显示当前用户有权访问的那些外部服务器（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_server\_catalog | sql\_identifier | 定义外部服务器的数据库的名称（始终为当前数据库）。 |
| foreign\_server\_name | sql\_identifier | 国外服务器名称。 |
| foreign\_data\_wrapper\_catalog | sql\_identifier | 包含外部服务器使用的外部数据包装器的数据库的名称（始终是当前数据库）。 |
| foreign\_data\_wrapper\_name | sql\_identifier | 外部服务器使用的外部数据包装器的名称。 |
| foreign\_server\_type | 字符数据 | 外部服务器类型信息，如果在创建时指定。 |
| foreign\_server\_version | 字符数据 | 外部服务器版本信息，如果在创建时指定。 |
| authorization\_identifier | sql\_identifier | 国外服务器所有者的名字。 |

### FOREIGN\_TABLE\_OPTIONS

包含为当前数据库中的外部表定义的所有选项。仅显示当前用户有权访问的那些外部表（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_table\_catalog | sql\_identifier | 包含外部表的数据库的名称（始终是当前数据库）。 |
| foreign\_table\_schema | sql\_identifier | 包含外部表的模式的名称。 |
| foreign\_table\_name | sql\_identifier | 外部表的名称。 |
| option\_name | sql\_identifier | 选项名称。 |
| option\_value | character\_data | 选项值。 |

### FOREIGN\_TABLES

包含当前数据库中定义的所有外部表。仅显示当前用户有权访问的那些外部表（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| foreign\_table\_catalog | sql\_identifier | 定义外部表的数据库的名称（始终为当前数据库）。 |
| foreign\_table\_schema | sql\_identifier | 包含外部表的模式的名称。 |
| foreign\_table\_name | sql\_identifier | 外部表的名称。 |
| foreign\_server\_catalog | sql\_identifier | 定义外部服务器的数据库的名称（始终为当前数据库）。 |
| foreign\_server\_name | sql\_identifier | 国外服务器名称。 |

### KEY\_COLUMN\_USAGE

标识当前数据库中受某些唯一、主键或外键约束的所有列。此视图中不包括检查约束。仅显示当前用户通过成为所有者或具有某些特权而可以访问的那些列。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |
| table\_catalog | sql\_identifier | 包含受此约束限制的列的表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含受此约束限制的列的表的架构的名称。 |
| table\_name | sql\_identifier | 包含受此约束限制的列的表的名称。 |
| column\_name | sql\_identifier | 受此约束限制的列的名称。 |
| ordinal\_position | cardinal\_number | 约束键中列的序号位置（计数从 1 开始）。 |
| position\_in\_unique\_constraint | cardinal\_number | 对于外键约束，引用列在其唯一约束中的序号位置（计数从 1 开始）；否则为空。 |

### PARAMETERS

包含有关当前数据库中所有函数的参数（参数）的信息。仅显示当前用户有权访问的那些功能（通过成为所有者或具有某些特权）。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| specific\_catalog | sql\_identifier | 包含函数的数据库的名称（始终为当前数据库）。 |
| specific\_schema | sql\_identifier | 包含函数的模式的名称。 |
| specific\_name | sql\_identifier | 函数的“特定名称”。有关详细信息，请参阅：schema->information-Schema->ROUTINES章节获取更多信息。 |
| ordinal\_position | cardinal\_number | 参数在函数参数列表中的序号位置（计数从 1 开始）。 |
| parameter\_mode | character\_data | IN为输入参数， OUT为输出参数， INOUT为输入/输出参数。 |
| is\_result | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| as\_locator | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| parameter\_name | sql\_identifier | 参数的名称，如果参数没有名称，则返回 null。 |
| data\_type | character\_data | 参数的数据类型，如果它是内置类型，或者ARRAY，如果它是某个数组（在这种情况下，请参见视图element\_types），否则为 USER-DEFINED（在这种情况下，类型在udt\_name中标识并关联列）。 |
| character\_maximum\_length | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| character\_octet\_length | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| character\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_catalog | sql\_identifier | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| collation\_schema | sql\_identifier | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| collation\_name | sql\_identifier | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| numeric\_precision | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| numeric\_precision\_radix | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| numeric\_scale | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| datetime\_precision | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| interval\_type | character\_data | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| interval\_precision | cardinal\_number | 始终为 null，因为此信息不适用于Vastbase中的参数数据类型。 |
| outt\_catalog | sql\_identifier | 定义参数的数据类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 参数的数据类型在其中定义的模式的名称。 |
| udt\_name | sql\_identifier | 参数的数据类型名称。 |
| scope\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为 null，因为数组在Vastbase中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 参数的数据类型描述符的标识符，在与函数有关的数据类型描述符中唯一。这主要用于与此类标识符的其他实例连接。（标识符的具体格式没有定义，也不保证在以后的版本中保持不变）。 |

### REFERENTIAL\_CONSTRAINTS

包含当前数据库中的所有引用（外键）约束。仅显示当前用户对引用表具有写访问权限的那些约束（通过成为所有者或具有除SELECT之外的某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |
| unique\_constraint\_catalog | sql\_identifier | 包含外键约束引用的唯一或主键约束的数据库的名称（始终为当前数据库）。 |
| unique\_constraint\_schema | sql\_identifier | 包含外键约束引用的唯一键或主键约束的模式的名称。 |
| unique\_constraint\_name | sql\_identifier | 外键约束引用的唯一键或主键约束的名称。 |
| match\_option | character\_data | 外键约束的匹配选项： FULL、PARTIAL或NONE。 |
| update\_rule | character\_data | 外键约束的更新规则：CASCADE、SET NULL、SET DEFAULT、 RESTRICT或NO ACTION。 |
| delete\_rule | character\_data | 外键约束的删除规则：CASCADE、SET NULL、SET DEFAULT、 RESTRICT或NO ACTION。 |

### ROLE\_COLUMN\_GRANTS

标识那些在字段上赋予或被赋予为当前角色的所有权限。 更多信息可以在column\_privileges中找到。这个视图和column\_privileges 唯一有效的不同就是这个视图忽略了某些字段，这些字段是通过授予PUBLIC 使得当前用户可以访问的字段。

| **名字** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 被赋予这个权限的用户名称 |
| grantee | sql\_identifier | 被赋予这个权限的角色名称 |
| table\_catalog | sql\_identifier | 包含此字段的表所在的数据库的名字（总是当前数据库） |
| table\_schema | sql\_identifier | 包含该字段的表所在模式的名称 |
| table\_name | sql\_identifier | 包含该字段的表名称 |
| column\_name | sql\_identifier | 该字段的名称 |
| privilege\_type | character\_data | 权限的类型：SELECT,INSERT, UPDATE或者REFERENCES |
| is\_grantable | yes\_or\_no | 如果权限是可以赋予的，则为YES，否则，为NO |

### ROLE\_ROUTINE\_GRANTS

标出所有在函数上赋予或被赋予当前角色的权限。 更多的信息可以在routine\_privileges里找到。在该视图与routine\_privileges 之间实际仅有的差异是该视图忽略那些通过赋权给PUBLIC使当前用户可以访问的函数。

| **名字** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 被赋予该权限的角色名称 |
| grantee | sql\_identifier | 被赋予此权限的角色的名称 |
| specific\_catalog | sql\_identifier | 包含此函数的数据库名称（总是当前数据库） |
| specific\_schema | sql\_identifier | 包含此函数的模式名称 |
| specific\_name | sql\_identifier | 函数的"specific name"（具体的名称）。 参阅：schema->information-Schema->ROUTINES章节获取更多信息。 |
| routine\_catalog | sql\_identifier | 包含此函数的数据库名称（总是当前数据库） |
| routine\_schema | sql\_identifier | 包含此函数的模式名称 |
| routine\_name | sql\_identifier | 函数的名称（可能重复，因为有重载） |
| privilege\_type | character\_data | 总是EXECUTE（函数的唯一的权限类型） |
| is\_grantable | yes\_or\_no | 如果权限可以赋予，那么是YES，否则为NO |

### ROLE\_TABLE\_GRANTS

标识在表或者视图上赋予或被赋予当前角色的全部权限。 更多信息可以在table\_privileges找到。在该视图与table\_privileges 之间实际仅有的差异是该视图忽略那些通过赋权给PUBLIC使当前用户可以访问的表。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 赋予权限的角色名 |
| grantee | sql\_identifier | 被赋予权限的角色名 |
| table\_catalog | sql\_identifier | 包含此表的数据库名（总是当前数据库） |
| table\_schema | sql\_identifier | 包含此表的模式名 |
| table\_name | sql\_identifier | 表名 |
| privilege\_type | character\_data | 权限类型：SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, 或 TRIGGER |
| is\_grantable | yes\_or\_no | 如果权限可以赋予则为YES，否则为NO |
| with\_hierarchy | yes\_or\_no | 在SQL标准里，WITH HIERARCHY OPTION 是一个分开的（子）权限，允许在表继承层次结构上进行创建操作。 在PostgreSQL中，这包含在了SELECT权限中， 所以如果这个权限为SELECT则这个字段显示YES， 否则为NO。 |

### ROLE\_UDT\_GRANTS

用于标出赋予或被赋予当前角色的用户定义类型上的 USAGE权限。更多的信息可以在udt\_privileges里找到。 在该视图与udt\_privileges 之间实际仅有的差异是该视图忽略那些通过赋权给PUBLIC使当前用户可以访问的对象。 因为数据类型在PostgreSQL中并没有真实的权力，但是只有一个隐式的赋权给PUBLIC， 所以这个视图是空的。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 赋予该权限的角色的名字 |
| grantee | sql\_identifier | 被赋予该权限的角色的名字 |
| udt\_catalog | sql\_identifier | 包含该类型的数据库的名字（总是当前数据库） |
| udt\_schema | sql\_identifier | 包含该类型的模式的名字 |
| udt\_name | sql\_identifier | 类型名 |
| privilege\_type | character\_data | 总是TYPE USAGE |
| is\_grantable | yes\_or\_no | 如果权限是可赋予的，那么就是YES，否则为NO |

### ROLE\_USAGE\_GRANTS

用于标出当前角色赋予或被赋予的各种对象的 USAGE权限。更多的信息可以在usage\_privileges里找到。 在该视图与usage\_privileges 之间实际仅有的差异是该视图忽略那些通过赋权给PUBLIC使当前用户可以访问的对象。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 赋予该权限的角色名称。 |
| grantee | sql\_identifier | 被赋予该权限的角色的名称。 |
| object\_catalog | sql\_identifier | 包含该对象的数据库的名字（总是当前数据库）。 |
| object\_schema | sql\_identifier | 如果适用，是包含该对象的模式的名字，否则是一个空字符串。 |
| object\_name | sql\_identifier | 对象的名字。 |
| object\_type | character\_data | COLLATION 或 DOMAIN 或 FOREIGN DATA WRAPPER 或 FOREIGN SERVER 或 SEQUENCE。 |
| privilege\_type | character\_data | Always USAGE。 |
| is\_grantable | yes\_or\_no | 如果权限可以赋予，则为YES，否则为NO。 |

### ROUTINE\_PRIVILEGES

标识在函数上所有赋予当前用户或者由当前用户赋予的权限。 每个函数，授权人，和权限接受人的组合都有一行。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 赋予权限的角色名 |
| grantee | sql\_identifier | 被授予权限的角色名 |
| specific\_catalog | sql\_identifier | 包含该函数的数据库名称（总是当前数据库） |
| specific\_schema | sql\_identifier | 包含该函数的模式名 |
| specific\_name | sql\_identifier | 函数的"specific name"（具体的名字）。 参阅：schema->information-Schema->ROUTINES章节获取更多信息。 |
| routine\_catalog | sql\_identifier | 包含该函数的数据库名称（总是当前数据库） |
| routine\_schema | sql\_identifier | 包含该函数的模式名称 |
| routine\_name | sql\_identifier | 函数名（可能会因重载而重复） |
| privilege\_type | character\_data | 总是EXECUTE (用于函数的唯一的权限类型) |
| is\_grantable | yes\_or\_no | 如果权限是可赋予的，则为YES，否则为NO |

### ROUTINES

包含当前数据库中的所有函数。仅显示当前用户有权访问的哪些功能（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| specific\_catalog | sql\_identifier | 包含函数的数据库的名称（始终为当前数据库）。 |
| specific\_schema | sql\_identifier | 包含函数的模式的名称。 |
| specific\_name | sql\_identifier | 函数的“特定名称”。这是一个在模式中唯一标识函数的名称，即使函数的真实名称被重载。特定名称的格式没有定义，它应该仅用于将其与特定例程名称的其他实例进行比较。 |
| routine\_catalog | sql\_identifier | 包含函数的数据库的名称（始终为当前数据库）。 |
| routine\_schema | sql\_identifier | 包含函数的模式的名称。 |
| routine\_name | sql\_identifier | 函数名（重载时可能重复）。 |
| routine\_type | character\_data | AlwaysFUNCTION（将来可能会有其他类型的例程）。 |
| module\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| module\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| module\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| udt\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| udt\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| udt\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| data\_type | character\_data | 返回函数的数据类型，如果它是内置类型，或者ARRAY如果它是某个数组（在这种情况下，请参阅视图ELEMENT\_TYPES），否则为USER-DEFINED（在这种情况下，类型在type\_udt\_name和相关列）。 |
| character\_maximum\_length | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| character\_octet\_length | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| character\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_catalog | sql\_identifier | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| collation\_schema | sql\_identifier | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| collation\_name | sql\_identifier | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| numeric\_precision | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| numeric\_precision\_radix | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| numeric\_scale | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| datetime\_precision | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| interval\_type | character\_data | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| interval\_precision | cardinal\_number | 始终为null，因为此信息不适用于Vastbase中的返回数据类型。 |
| type\_udt\_catalog | sql\_identifier | 定义函数返回数据类型的数据库名称（始终为当前数据库）。 |
| type\_udt\_schema | sql\_identifier | 定义函数返回数据类型的模式名称。 |
| type\_udt\_name | sql\_identifier | 函数返回数据类型的名称。 |
| scope\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| scope\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| maximum\_cardinality | cardinal\_number | 始终为null，因为数组在Vastbase中始终具有无限的最大基数。 |
| dtd\_identifier | sql\_identifier | 此函数的返回数据类型的数据类型描述符的标识符，在与该函数有关的数据类型描述符中是唯一的。这主要用于与此类标识符的其他实例连接。（标识符的具体格式没有定义，也不保证在以后的版本中保持不变）。 |
| routine\_body | character\_data | 如果该函数是SQL函数，则为SQL，否则为EXTERNAL。 |
| routine\_definition | character\_data | 函数的源文本（如果函数不属于当前启用的角色，则为null）。（根据SQL标准，此列仅在routine\_body为SQL时适用，但在Vastbase中，它将包含创建函数时指定的任何源文本）。 |
| external\_name | character\_data | 如果此函数是C函数，则为函数的外部名称（链接符号）；否则为空。（这与routine\_definition中显示的值相同）。 |
| external\_language | character\_data | 编写函数的语言。 |
| parameter\_style | character\_data | AlwaysGENERAL（SQL标准定义了其他参数样式，这些样式在Vastbase中不可用）。 |
| is\_deterministic | yes\_or\_no | 如果函数被声明为不可变（在SQL标准中称为确定性），则为YES，否则为NO。（您不能通过信息模式查询Vastbase中可用的其他波动级别）。 |
| sql\_data\_access | character\_data | 总是MODIFIES，这意味着该函数可能会修改SQL数据。此信息对Vastbase没有用处。 |
| is\_null\_call | yes\_or\_no | 如果函数在其任何参数为null时自动返回null，则为YES，否则为NO。 |
| sql\_path | character\_data | 适用于Vastbase中不可用的功能。 |
| schema\_level\_routine | yes\_or\_no | 总是YES（相反的是用户定义类型的方法，这是Vastbase中不可用的功能）。 |
| max\_dynamic\_result\_sets | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| is\_user\_defined\_cast | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| is\_implicitly\_invocable | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| security\_type | character\_data | 如果该函数以当前用户的权限运行，则为INVOKER，如果该函数以定义它的用户的权限运行，则为DEFINER。 |
| to\_sql\_specific\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| to\_sql\_specific\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| to\_sql\_specific\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| as\_locator | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| created | time\_stamp | 适用于Vastbase中不可用的功能。 |
| last\_altered | time\_stamp | 适用于Vastbase中不可用的功能。 |
| new\_savepoint\_level | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| is\_udt\_dependent | yes\_or\_no | 目前总是NO。替代YES适用于Vastbase中不可用的功能。 |
| result\_cast\_from\_data\_type | character\_data | 适用于Vastbase中不可用的功能。 |
| result\_cast\_as\_locator | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| result\_cast\_char\_max\_length | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_char\_octet\_length | character\_data | 适用于Vastbase中不可用的功能。 |
| result\_cast\_char\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_char\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_char\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_collation\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_collation\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_collation\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_numeric\_precision | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_numeric\_precision\_radix | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_numeric\_scale | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_datetime\_precision | character\_data | 适用于Vastbase中不可用的功能。 |
| result\_cast\_interval\_type | character\_data | 适用于Vastbase中不可用的功能。 |
| result\_cast\_interval\_precision | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_type\_udt\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_type\_udt\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_type\_udt\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_scope\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_scope\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_scope\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| result\_cast\_maximum\_cardinality | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| result\_cast\_dtd\_identifier | sql\_identifier | 适用于Vastbase中不可用的功能。 |

### SCHEMATA

包含当前数据库中由当前启用的角色拥有的所有模式。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| catalog\_name | sql\_identifier | 包含架构的数据库的名称（始终为当前数据库）。 |
| schema\_name | sql\_identifier | 架构的名称。 |
| schema\_owner | sql\_identifier | 架构所有者的姓名。 |
| default\_character\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| default\_character\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| default\_character\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| sql\_path | character\_data | 适用于Vastbase中不可用的功能。 |

### SEQUENCES

包含当前数据库中定义的所有序列。仅显示当前用户有权访问的那些序列（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| sequence\_catalog | sql\_identifier | 包含序列的数据库的名称（始终为当前数据库）。 |
| sequence\_schema | sql\_identifier | 包含序列的模式的名称。 |
| sequence\_name | sql\_identifier | 序列名称。 |
| data\_type | character\_data | 序列的数据类型。在Vastbase中，这目前总是bigint。 |
| numeric\_precision | cardinal\_number | 此列包含序列数据类型的（声明的或隐含的）精度（见上文）。精度表示有效位数。它可以用十进制（以10为底）或二进制（以2为底）表示，如numeric\_precision\_radix列中指定的那样。 |
| numeric\_precision\_radix | cardinal\_number | 此列指示numeric\_precision和numeric\_scale列中的值以哪个基数表示。该值为2或10。 |
| numeric\_scale | cardinal\_number | 此列包含序列数据类型的（声明的或隐含的）比例（见上文）。刻度表示小数点右侧的有效位数。它可以用十进制（以10为底）或二进制（以2为底）表示，如numeric\_precision\_radix列中指定的那样。 |
| start\_value | character\_data | 序列的起始值。 |
| minimum\_value | character\_data | 序列的最小值。 |
| maximum\_value | character\_data | 序列的最大值。 |
| increment | character\_data | 序列的增量。 |
| cycle\_option | yes\_or\_no | 如果序列循环，则为YES，否则为NO。 |

### SQL\_FEATURES

含有关 SQL 标准中定义的正式特性受Vastbase支持的信息。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| feature\_id | character\_data | 特征的标识符字符串。 |
| feature\_name | character\_data | 功能的描述性名称。 |
| sub\_feature\_id | character\_data | 子特征的标识符字符串，如果不是子特征，则为零长度字符串。 |
| sub\_feature\_name | character\_data | 子功能的描述性名称，如果不是子功能，则为零长度字符串。 |
| is\_supported | yes\_or\_no | 如果当前版本的Vastbase完全支持该功能，则为YES，否则为NO。 |
| is\_verified\_by | character\_data | 始终为 null，因为Vastbase开发组不执行功能一致性的正式测试。 |
| comments | character\_data | 可能是关于功能支持状态的评论。 |

### SQL\_IMPLEMENTATION\_INFO

包含有关由 SQL 标准实现定义的各个方面的信息。此信息主要用于 ODBC 接口的上下文中；其他界面的用户可能会发现这些信息没什么用。因此，这里不描述各个实现信息项；您可以在 ODBC 接口的描述中找到它们。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| implementation\_info\_id | character\_data | 实现信息项的标识符字符串。 |
| implementation\_info\_name | character\_data | 实施信息项的描述性名称。 |
| integer\_value | cardinal\_number | 实现信息项的值，如果该值包含在character\_value列中，则为 null。 |
| character\_value | character\_data | 实现信息项的值，如果该值包含在integer\_value列中，则为 null。 |
| comments | character\_data | 可能是与实施信息项有关的评论。 |

### SQL\_LANGUAGES

标识Vastbase支持的每种SQL语言绑定的一行。Vastbase支持C语言中的直接SQL和嵌入式SQL用户都可以从这张表中学到的全部内容。

此表已从SQL:2008中的SQL标准中删除，因此没有引用SQL:2003之后的标准的条目。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| sql\_language\_source | character\_data | 语言定义的来源名称；总是ISO9075，即SQL标准。 |
| sql\_language\_year | character\_data | sql\_language\_source中引用的标准获得批准的年份。 |
| sql\_language\_conformance | character\_data | 语言绑定的标准一致性级别。对于ISO9075:2003，这始终是CORE。 |
| sql\_language\_integrity | character\_data | 始终为空（此值与SQL标准的早期版本相关）。 |
| sql\_language\_implementation | character\_data | 始终为空。 |
| sql\_language\_binding\_style | character\_data | 语言绑定样式，DIRECT或EMBEDDED。 |
| sql\_language\_programming\_language | character\_data | 编程语言，如果绑定样式是EMBEDDED，否则为null。Vastbase只支持C语言。 |

### SQL\_PACKAGES

包含有关 SQL 标准中定义的哪些功能包受Vastbase支持的信息。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| feature\_id | character\_data | 包的标识符字符串。 |
| feature\_name | character\_data | 包的描述性名称。 |
| is\_supported | yes\_or\_no | 如果当前版本的Vastbase完全支持包，则为YES，否则为NO。 |
| is\_verified\_by | character\_data | 始终为 null，因为Vastbase开发组不执行功能一致性的正式测试。 |
| comments | character\_data | 可能是关于包的支持状态的评论。 |

### SQL\_PARTS

包含有关 SQL 标准的几个部分中的哪些部分受Vastbase支持的信息。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| feature\_id | character\_data | 包含零件编号的标识符字符串。 |
| feature\_name | character\_data | 零件的描述性名称。 |
| is\_supported | yes\_or\_no | 如果当前版本的Vastbase完全支持该部分，则为YES，否则为NO。 |
| is\_verified\_by | character\_data | 始终为 null，因为Vastbase开发组不执行功能一致性的正式测试。 |
| comments | character\_data | 可能是关于部件支持状态的评论。 |

### SQL\_SIZING

包含有关Vastbase中各种大小限制和最大值的信息。此信息主要用于ODBC接口的上下文中；其他界面的用户可能会发现这些信息没什么用。出于这个原因，这里没有描述各个尺寸的项目；用户可以在ODBC接口的描述中找到它们。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| sizing\_id | cardinal\_number | 尺码项目的标识符。 |
| sizing\_name | character\_data | 尺码项目的描述性名称。 |
| supported\_value | cardinal\_number | 尺码项目的值，如果尺寸不受限制或无法确定，则为0，如果不支持尺码项目适用的功能，则为null。 |
| comments | character\_data | 可能是与尺码项目有关的评论。 |

### SQL\_SIZING\_PROFILES

包含有关SQL 标准的各种配置文件所需的sql\_sizing值的信息。Vastbase不跟踪任何 SQL 配置文件，因此该表为空。

| **姓名** | **数据类型** | **描述** |
| --- | --- | --- |
| sizing\_id | cardinal\_number | 尺码项目的标识符。 |
| sizing\_name | character\_data | 尺码项目的描述性名称。 |
| profile\_id | character\_data | 配置文件的标识符字符串。 |
| required\_value | cardinal\_number | SQL 配置文件对尺码项目所需的值，如果配置文件对尺码项目没有限制，则为 0，如果配置文件不需要尺码项目适用的任何功能，则为 null。 |
| comments | character\_data | 可能是与配置文件中的尺码项目有关的评论。 |

### TABLE\_CONSTRAINTS

包含属于当前用户拥有或具有除SELECT之外的某些特权的表的所有约束 。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| constraint\_catalog | sql\_identifier | 包含约束的数据库的名称（始终为当前数据库）。 |
| constraint\_schema | sql\_identifier | 包含约束的模式的名称。 |
| constraint\_name | sql\_identifier | 约束的名称。 |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| constraint\_type | character\_data | 约束类型：CHECK、FOREIGN KEY、PRIMARY KEY或 UNIQUE。 |
| is\_deferrable | yes\_or\_no | 如果约束是可延迟的，则为YES ，否则为NO。 |
| initially\_deferred | yes\_or\_no | 如果约束是可延迟的并且最初是延迟的，则为YES ，否则为NO。 |

### TABLE\_PRIVILAGES

标识表或视图上授予当前启用的角色或当前启用的角色的所有特权。表、授予者和被授予者的每个组合都有一行。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 授予权限的角色的名称。 |
| grantee | sql\_identifier | 授予权限的角色的名称。 |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| privilege\_type | character\_data | 权限类型：SELECT、INSERT、 UPDATE、DELETE、TRUNCATE、 REFERENCES或TRIGGER。 |
| is\_grantable | yes\_or\_no | 如果特权是可授予的，则为YES ，否则为NO。 |
| with\_hierarchy | yes\_or\_no | 在 SQL 标准中，WITH HIERARCHY OPTION是一个单独的（子）权限，允许对表继承层次结构进行某些操作。在 Vastbase 中，这包含在 SELECT权限中，因此如果权限为 SELECT ，则此列显示YES，否则显示NO。 |

### TABLES

包含当前数据库中定义的所有表和视图。仅显示当前用户有权访问的那些表和视图（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的模式的名称。 |
| table\_name | sql\_identifier | 表名。 |
| table\_type | character\_data | 表的类型：BASE TABLE用于持久基表（普通表类型），VIEW用于视图， FOREIGN TABLE用于外部表，或LOCAL TEMPORARY用于临时表。 |
| self\_referencing\_column\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| reference\_generation | character\_data | 适用于Vastbase中不可用的功能。 |
| user\_defined\_type\_catalog | sql\_identifier | 如果表是类型表，则为包含基础数据类型的数据库的名称（始终为当前数据库），否则为 null。 |
| user\_defined\_type\_schema | sql\_identifier | 如果表是类型表，则为包含基础数据类型的模式的名称，否则为 null。 |
| user\_defined\_type\_name | sql\_identifier | 如果表是类型表，则为基础数据类型的名称，否则为 null。 |
| is\_insertable\_into | yes\_or\_no | YES，如果表是可插入的，NO ，如果不是（基表总是可插入的，视图不一定）。 |
| is\_typed | yes\_or\_no | 如果表是类型表，则为YES ，否则为NO。 |
| commit\_action | character\_data | 尚未实现。 |

### TRIGGERED\_UPDATE\_COLUMNS

对于当前数据库中指定列列表的触发器（如UPDATE OF column1, column2），视图trigger\_update\_columns标识这些列。此视图中不包含未指定列列表的触发器。仅显示当前用户拥有或具有除 SELECT on 之外的某些特权的那些列。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| trigger\_catalog | sql\_identifier | 包含触发器的数据库的名称（始终为当前数据库）。 |
| trigger\_schema | sql\_identifier | 包含触发器的架构名称。 |
| trigger\_name | sql\_identifier | 触发器名称。 |
| event\_object\_catalog | sql\_identifier | 包含定义触发器的表的数据库的名称（始终为当前数据库）。 |
| event\_object\_schema | sql\_identifier | 包含定义触发器的表的模式的名称。 |
| event\_object\_table | sql\_identifier | 定义触发器的表的名称。 |
| event\_object\_column | sql\_identifier | 定义触发器的列的名称。 |

### TRIGGERS

视图触发器包含当前数据库中针对当前用户拥有或具有除SELECT之外的某些特权的表和视图定义的所有触发器。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| trigger\_catalog | sql\_identifier | 包含触发器的数据库的名称（始终为当前数据库）。 |
| trigger\_schema | sql\_identifier | 包含触发器的架构名称。 |
| trigger\_name | sql\_identifier | 触发器名称。 |
| event\_manipulation | character\_data | 触发触发器的事件（INSERT、UPDATE或DELETE）。 |
| event\_object\_catalog | sql\_identifier | 包含定义触发器的表的数据库的名称（始终为当前数据库）。 |
| event\_object\_schema | sql\_identifier | 包含定义触发器的表的模式的名称。 |
| event\_object\_table | sql\_identifier | 定义触发器的表的名称。 |
| action\_order | cardinal\_number | 尚未实现。 |
| action\_condition | character\_data | 触发器的WHEN条件，如果没有则为 null（如果表不属于当前启用的角色，则为 null）。 |
| action\_statement | character\_data | 由触发器执行的语句（目前总是EXECUTE PROCEDURE function (...)）。 |
| action\_orientation | character\_data | 标识触发器是为每个已处理的行触发一次还是为每个语句（ROW或STATEMENT）触发一次。 |
| action\_timing | character\_data | 触发器触发的时间（BEFORE、AFTER或 INSTEAD OF）。 |
| action\_reference\_old\_table | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| action\_reference\_new\_table | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| action\_reference\_old\_row | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| action\_reference\_new\_row | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| created | time\_stamp | 适用于Vastbase中不可用的功能。 |

Vastbase中的触发器与影响信息模式中表示的 SQL 标准有两个不兼容之处。首先，触发器名称对于Vastbase中的每个表都是本地的，而不是独立的模式对象。因此，可以在一个模式中定义重复的触发器名称，只要它们属于不同的表。（trigger\_catalog和trigger\_schema实际上是与定义触发器的表有关的值。）其次，触发器可以定义为在Vastbase中触发多个事件（例如，ON INSERT OR UPDATE)，而 SQL 标准只允许一个。如果触发器被定义为触发多个事件，则它在信息模式中表示为多行，每种类型的事件一个。

由于这两个问题，视图触发器的主键实际上是(trigger\_catalog, trigger\_schema, event\_object\_table, trigger\_name, event\_manipulation) 而不是SQL 标准指定的(trigger\_catalog, trigger\_schema, trigger\_name) 。尽管如此，如果您以符合 SQL 标准的方式定义触发器（触发器名称在模式中是唯一的，并且每个触发器只有一个事件类型），这不会影响您。

### UDT\_PRIVILEGES

标识 在用户定义类型上授予当前启用的角色或当前启用的角色的USAGE权限。类型、授予者和被授予者的每种组合都有一行。此视图仅显示复合类型请参阅USER\_DEFINED\_TYPES了解原）； 有关域权限，请参见USAGE\_PRIVILEGES。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 授予权限的角色的名称。 |
| grantee | sql\_identifier | 授予权限的角色的名称。 |
| udt\_catalog | sql\_identifier | 包含该类型的数据库的名称（始终为当前数据库）。 |
| udt\_schema | sql\_identifier | 包含类型的模式的名称。 |
| udt\_name | sql\_identifier | 类型名称。 |
| privilege\_type | character\_data | 始终键入用法。 |
| is\_grantable | yes\_or\_no | 如果特权是可授予的，则为YES ，否则为NO。 |

### USAGE\_PRIVILEGES

标识了在各种对象上授予当前启用的角色或当前启用的角色的USAGE特权。在Vastbase中，这目前适用于排序规则、域、外部数据包装器、外部服务器和序列。对象、授予者和被授予者的每个组合都有一行。

由于排序规则在Vastbase中没有真正的权限，因此该视图显示了所有者为所有排序规则授予PUBLIC的隐式不可授予的USAGE权限。然而，其他对象类型显示了真正的特权。

在Vastbase中，除了USAGE权限之外，序列还支持SELECT和UPDATE权限。这些是非标准的，因此在信息模式中不可见。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| grantor | sql\_identifier | 授予权限的角色的名称。 |
| grantee | sql\_identifier | 授予权限的角色的名称。 |
| object\_catalog | sql\_identifier | 包含对象的数据库的名称（始终为当前数据库）。 |
| object\_schema | sql\_identifier | 包含对象的模式的名称（如果适用），否则为空字符串。 |
| object\_name | sql\_identifier | 对象名称。 |
| object\_type | character\_data | COLLATION或DOMAIN或FOREIGNDATAWRAPPER或FOREIGNSERVER或SEQUENCE。 |
| privilege\_type | character\_data | 始终使用。 |
| is\_grantable | yes\_or\_no | 如果特权是可授予的，则为YES，否则为NO。 |

### USER\_DEFINED\_TYPES

包含当前数据库中定义的所有复合类型。仅显示当前用户有权访问的那些类型（通过成为所有者或具有某些特权）。

SQL知道两种用户定义类型：结构化类型（在Vastbase中也称为复合类型）和不同类型（在Vastbase中未实现）。为了面向未来，请使用user\_defined\_type\_category列来区分这些。其他用户定义的类型，例如基本类型和枚举，它们是Vastbase扩展，这里没有显示。对于域，请参阅：schema->information-Schema->DOMAINs章节。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| user\_defined\_type\_catalog | sql\_identifier | 包含该类型的数据库的名称（始终为当前数据库）。 |
| user\_defined\_type\_schema | sql\_identifier | 包含类型的架构的名称。 |
| user\_defined\_type\_name | sql\_identifier | 类型名称。 |
| user\_defined\_type\_category | character\_data | 目前始终是结构化的。 |
| is\_instantiable | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| is\_final | yes\_or\_no | 适用于Vastbase中不可用的功能。 |
| ordering\_form | character\_data | 适用于Vastbase中不可用的功能。 |
| ordering\_category | character\_data | 适用于Vastbase中不可用的功能。 |
| ordering\_routine\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| ordering\_routine\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| ordering\_routine\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| reference\_type | character\_data | 适用于Vastbase中不可用的功能。 |
| data\_type | character\_data | 适用于Vastbase中不可用的功能。 |
| character\_maximum\_length | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| character\_octet\_length | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| character\_set\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| character\_set\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_catalog | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_schema | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| collation\_name | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| numeric\_precision | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| numeric\_precision\_radix | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| numeric\_scale | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| datetime\_precision | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| interval\_type | character\_data | 适用于Vastbase中不可用的功能。 |
| interval\_precision | cardinal\_number | 适用于Vastbase中不可用的功能。 |
| source\_dtd\_identifier | sql\_identifier | 适用于Vastbase中不可用的功能。 |
| ref\_dtd\_identifier | sql\_identifier | 适用于Vastbase中不可用的功能。 |

### USER\_MAPPING\_OPTIONS

包含为当前数据库中的用户映射定义的所有选项。仅显示当前用户可以访问相应外部服务器的那些用户映射（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| authorization\_identifier | sql\_identifier | 被映射的用户的名称，如果映射是公共的，则为PUBLIC。 |
| foreign\_server\_catalog | sql\_identifier | 此映射使用的外部服务器在其中定义的数据库的名称（始终为当前数据库）。 |
| foreign\_server\_name | sql\_identifier | 此映射使用的外部服务器的名称。 |
| option\_name | sql\_identifier | 选项名称。 |
| option\_value | character\_data | 期权的价值。此列将显示为空，除非当前用户是被映射的用户，或者映射是针对PUBLIC并且当前用户是服务器所有者，或者当前用户是超级用户。目的是保护作为用户映射选项存储的密码信息。 |

### USER\_MAPPINGS

包含当前数据库中定义的所有用户映射。仅显示当前用户可以访问相应外部服务器的那些用户映射（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| authorization\_identifier | sql\_identifier | 被映射的用户的名称，如果映射是公共的，则为PUBLIC。 |
| foreign\_server\_catalog | sql\_identifier | 此映射使用的外部服务器在其中定义的数据库的名称（始终为当前数据库）。 |
| foreign\_server\_name | sql\_identifier | 此映射使用的外部服务器的名称。 |

### VIEW\_COLUMN\_USAGE

标识在视图的查询表达式（定义视图的SELECT语句）中使用的所有列 。仅当包含该列的表由当前启用的角色拥有时，才会包含该列。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| view\_catalog | sql\_identifier | 包含视图的数据库的名称（始终为当前数据库）。 |
| view\_schema | sql\_identifier | 包含视图的架构名称。 |
| view\_name | sql\_identifier | 视图名称。 |
| table\_catalog | sql\_identifier | 包含视图使用的列的表的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含表的架构的名称，该表包含视图使用的列。 |
| table\_name | sql\_identifier | 包含视图使用的列的表的名称。 |
| column\_name | sql\_identifier | 视图使用的列的名称。 |

### VIEW\_ROUTINE\_USAGE

标识在视图的查询表达式（定义视图的SELECT语句）中使用的所有例程（函数和过程）。仅当该例程由当前启用的角色拥有时，才会包含该例程。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含视图的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含视图的架构名称。 |
| table\_name | sql\_identifier | 视图名称。 |
| specific\_catalog | sql\_identifier | 包含函数的数据库的名称（始终为当前数据库）。 |
| specific\_schema | sql\_identifier | 包含函数的模式的名称。 |
| specific\_name | sql\_identifier | 函数的“特定名称”。有关详细信息，请参阅：schema->information-Schema->ROUTINES章节获取更多信息。 |

### VIEW\_TABLE\_USAGE

标识在视图的查询表达式（定义视图的SELECT语句）中使用的所有表。仅当该表由当前启用的角色拥有时，才会包含该表。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| view\_catalog | sql\_identifier | 包含视图的数据库的名称（始终为当前数据库）。 |
| view\_schema | sql\_identifier | 包含视图的架构名称。 |
| view\_name | sql\_identifier | 视图名称。 |
| table\_catalog | sql\_identifier | 包含视图使用的表的数据库的名称（始终是当前数据库）。 |
| table\_schema | sql\_identifier | 包含视图使用的表的模式的名称。 |
| table\_name | sql\_identifier | 视图使用的表的名称。 |

### VIEWS

包含当前数据库中定义的所有视图。仅显示当前用户有权访问的那些视图（通过成为所有者或具有某些特权）。

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| table\_catalog | sql\_identifier | 包含视图的数据库的名称（始终为当前数据库）。 |
| table\_schema | sql\_identifier | 包含视图的架构名称。 |
| table\_name | sql\_identifier | 视图名称。 |
| view\_definition | character\_data | 定义视图的查询表达式（如果视图不属于当前启用的角色，则为 null）。 |
| check\_option | character\_data | 适用于Vastbase中不可用的功能。 |
| is\_updatable | yes\_or\_no | YES如果视图是可更新的（允许UPDATE和DELETE），NO如果不是。 |
| is\_insertable\_into | yes\_or\_no | YES如果视图可插入（允许INSERT）， NO如果不是。 |
| is\_trigger\_updatable | yes\_or\_no | 如果视图上定义了 INSTEAD OF UPDATE触发器，则为YES ，否则为NO。 |
| is\_trigger\_deletable | yes\_or\_no | 如果视图上定义了 INSTEAD OF DELETE触发器，则为YES ，否则为NO。 |
| is\_trigger\_insertable\_into | yes\_or\_no | 如果视图上定义了 INSTEAD OF INSERT触发器，则为YES ，否则为NO。 |

### \_PG\_FOREIGN\_DATA\_WRAPPERS

显示外部数据封装器的信息。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_FOREIGN\_DATA\_WRAPPERS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 外部数据封装器的oid。 |
| fdwowner | oid | 外部数据封装器的所有者的oid。 |
| fdwoptions | text[] | 外部数据封装器指定选项，使用“keyword=value”格式的字符串。 |
| foreign\_data\_wrapper\_catalog | information\_schema.sql\_identifier | 外部封装器所在的数据库名称（永远为当前数据库）。 |
| foreign\_data\_wrapper\_name | information\_schema.sql\_identifier | 外部数据封装器名称。 |
| authorization\_identifier | information\_schema.sql\_identifier | 外部数据封装器所有者的角色名称。 |
| foreign\_data\_wrapper\_language | information\_schema.character\_data | 外部数据封装器的实现语言。 |

### \_PG\_FOREIGN\_SERVERS

显示外部服务器的信息。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_FOREIGN\_SERVERS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 外部服务器的oid。 |
| srvoptions | text[] | 外部服务器指定选项，使用“keyword=value”格式的字符串。 |
| foreign\_server\_catalog | information\_schema.sql\_identifier | 外部服务器所在database名称（永远为当前数据库）。 |
| foreign\_server\_name | information\_schema.sql\_identifier | 外部服务器名称。 |
| foreign\_data\_wrapper\_catalog | information\_schema.sql\_identifier | 外部数据封装器所在database名称（永远为当前数据库）。 |
| foreign\_data\_wrapper\_name | information\_schema.sql\_identifier | 外部数据封装器名称。 |
| foreign\_server\_type | information\_schema.character\_data | 外部服务器的类型。 |
| foreign\_server\_version | information\_schema.character\_data | 外部服务器的版本。 |
| authorization\_identifier | information\_schema.sql\_identifier | 外部服务器的所有者的角色名称。 |

### \_PG\_FOREIGN\_TABLE\_COLUMNS

显示外部表的列信息。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_FOREIGN\_TABLE\_COLUMNS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nspname | name | schema名称。 |
| relname | name | 表名称。 |
| attname | name | 列名称。 |
| attfdwoptions | text[] | 外部数据封装器的属性选项，使用“keyword=value”格式的字符串。 |

### \_PG\_FOREIGN\_TABLES

存储所有的定义在本数据库的外部表信息。只显示当前用户有权访问的外部表信息。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_FOREIGN\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| foreign\_table\_catalog | information\_schema.sql\_identifier | 外部表所在的数据库名称（永远是当前数据库）。 |
| foreign\_table\_schema | name | 外部表的schema名称。 |
| foreign\_table\_name | name | 外部表的名称。 |
| ftoptions | text[] | 外部表的可选项。 |
| foreign\_server\_catalog | information\_schema.sql\_identifier | 外部服务器所在的数据库名称（永远是当前数据库）。 |
| foreign\_server\_name | information\_schema.sql\_identifier | 外部服务器的名称。 |
| authorization\_identifier | information\_schema.sql\_identifier | 所有者的角色名称。 |

### \_PG\_USER\_MAPPINGS

存储从本地用户到远程的映射。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_USER\_MAPPINGS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 从本地用户到远程的映射的oid。 |
| umoptions | text[] | 用户映射指定选项，使用“keyword=value”格式的字符串。 |
| umuser | oid | 被映射的本地用户的OID，如果用户映射是公共的则为0。 |
| authorization\_identifier | information\_schema.sql\_identifier | 本地用户角色名称。 |
| foreign\_server\_catalog | information\_schema.sql\_identifier | 外部服务器定义所在的database名称。 |
| foreign\_server\_name | information\_schema.sql\_identifier | 外部服务器名称。 |
| srvowner | information\_schema.sql\_identifier | 外部服务器所有者。 |

### INFORMATION\_SCHEMA\_CATALOG\_NAME

用来显示当前所在的database的名称。

**表 1** INFORMATION\_SCHEMA\_CATALOG\_NAME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| catalog\_name | information\_schema.sql\_identifier | 当前database的名称。 |

## DBE\_PLDEBUGGER Schema

DBE\_PLDEBUGGER下系统函数用于单机下调试存储过程，目前支持的接口及其描述如下所示。仅管理员有权限执行这些调试接口。

fig: **须知：**   
当在函数体中创建用户时，调用attach、next、continue、 info\_code、step、info\_breakpoint、backtrace、 finish中会返回密码的明文。因此不建议用户在函数体中创建用户。

对应权限角色为gs\_role\_pldebugger，可以由管理员用户通过如下命令将debugger权限赋权给该用户。

GRANT gs\_role\_pldebugger to user;

需要有两个客户端连接数据库，一个客户端负责执行调试接口作为debug端，另一个客户端执行调试函数，控制server端存储过程执行。示例如下。

* 准备调试
* 通过PG\_PROC，查找到待调试存储过程的oid，并执行DBE\_PLDEBUGGER.turn\_on(oid)。本客户端就会作为server端使用。

vastbase=# CREATE OR REPLACE PROCEDURE test\_debug ( IN x INT)   
AS   
BEGIN  
 INSERT INTO t1 (a) VALUES (x);  
 DELETE FROM t1 WHERE a = x;  
END;  
/  
  
vastbase=# SELECT OID FROM PG\_PROC WHERE PRONAME='test\_debug';  
 oid  
-------  
 16389  
(1 row)  
  
vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.turn\_on(16389);  
 nodename | port  
----------+------  
 datanode | 0  
(1 row)

* 开始调试

server端执行存储过程，会在存储过程内第一条SQL语句前hang住，等待debug端发送的调试消息。仅支持直接执行存储过程的调试，不支持通过trigger调用执行的存储过程调试。

call test\_debug(1);

再起一个客户端，作为debug端，通过turn\_on返回的数据，调用DBE\_PLDEBUGGER.attach关联到该存储过程上进行调试。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.attach('datanode',0);  
 funcoid | funcname | lineno | query  
---------+------------+--------+----------------------------------  
 16389 | test\_debug | 3 | INSERT INTO t1 (a) VALUES (x);  
(1 row)

在执行attach的客户端调试，执行下一条statement。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.next();  
 funcoid | funcname | lineno | query  
---------+------------+--------+----------------------  
 16389 | test\_debug | 0 | [EXECUTION FINISHED]  
(1 row)

在执行attach的客户端调试，可以执行以下变量操作

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.info\_locals(); --打印全部变量  
 varname | vartype | value | package\_name | isconst  
---------+---------+-------+--------------+---------  
 x | int4 | 1 | | f  
(1 row)  
  
vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.set\_var('x', 2); --变量赋值  
 set\_var  
---------  
 t  
(1 row)  
  
vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.print\_var('x'); --打印单个变量  
 varname | vartype | value | package\_name | isconst  
---------+---------+-------+--------------+---------  
 x | int4 | 2 | | f  
(1 row)

* 以下几种方式均可以完成当前调试。
* 直接执行完成当前正在调试的存储过程。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.continue();  
 funcoid | funcname | lineno | query  
---------+------------+--------+----------------------  
 16389 | test\_debug | 0 | [EXECUTION FINISHED]  
(1 row)

* 直接退出当前正在调试的存储过程，不执行尚未执行的语句。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.abort();  
 abort  
-------  
 t  
(1 row)

* client端操作：

1、查看代码信息并识别可以设置断点行号。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.info\_code(16389);  
lineno | query | canbreak  
 --------+-----------------------------------------------------------+----------  
 | CREATE OR REPLACE PROCEDURE public.test\_debug( IN x INT) | f  
 1 | AS DECLARE | f  
 2 | BEGIN | f  
 3 | INSERT INTO t1 (a) VALUES (x); | t  
 4 | DELETE FROM t1 WHERE a = x; | t  
 5 | END; | f  
 6 | / | f  
(7 rows)

2、设置断点。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.add\_breakpoint(16389,4);  
 lineno | query | canbreak  
 --------+-----------------------------------------------------------+----------  
 | CREATE OR REPLACE PROCEDURE public.test\_debug( IN x INT) | f  
 1 | AS DECLARE | f  
 2 | BEGIN | f  
 3 | INSERT INTO t1 (a) VALUES (x); | t  
 4 | DELETE FROM t1 WHERE a = x; | t  
 5 | END; | f  
 6 | / | f

(7 rows)

3、查看断点信息。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.info\_breakpoints();  
 breakpointno | funcoid | lineno | query | enable  
 --------------+---------+--------+---------------------------------+--------  
 0 | 16389 | 4 | DELETE FROM t1 WHERE a = x; | t  
(1 row)

4、执行至断点。

vastbase=# SELECT \* FROM DBE\_PLDEBUGGER.continue();  
 funcoid | funcname | lineno | query  
 ---------+------------+--------+---------------------------------  
 16389 | test\_debug | 4 | DELETE FROM t1 WHERE a = x;  
 (1 row)

存储过程执行结束后，调试会自动退出，再进行调试需要重新attach关联。如果server端不需要继续调试，可执行turn\_off关闭，或退出session。具体调试接口请见下面列表。

**表 1** DBE\_PLDEBUGGER

| **接口名称** | **描述** |
| --- | --- |
| DBE\_PLDEBUGGER.turn\_on] | server端调用，标记存储过程可以调试，调用后执行该存储过程时会hang住等待调试信息。 |
| DBE\_PLDEBUGGER.turn\_off | server端调用，标记存储过程关闭调试。 |
| DBE\_PLDEBUGGER.local\_debug\_server\_info | server端调用，打印本session内所有已turn\_on的存储过程。 |
| DBE\_PLDEBUGGER.attach | debug端调用，关联到正在调试存储过程。 |
| DBE\_PLDEBUGGER.info\_locals | debug端调用，打印正在调试的存储过程中的变量当前值。 |
| DBE\_PLDEBUGGER.next | debug端调用，单步执行。 |
| DBE\_PLDEBUGGER.continue | debug端调用，继续执行，直到断点或存储过程结束。 |
| DBE\_PLDEBUGGER.abort | debug端调用，停止调试，server端报错长跳转。 |
| DBE\_PLDEBUGGER.print\_var | debug端调用，打印正在调试的存储过程中指定的变量当前值。 |
| DBE\_PLDEBUGGER.info\_code | debug和server端都可以调用，打印指定存储过程的源语句和各行对应的行号。 |
| DBE\_PLDEBUGGER.step | debug端调用，单步进入执行。 |
| DBE\_PLDEBUGGER.add\_breakpoint | debug端调用，新增断点。 |
| DBE\_PLDEBUGGER.delete\_breakpoint | debug端调用，删除断点。 |
| DBE\_PLDEBUGGER.info\_breakpoints | debug端调用，查看当前的所有断点。 |
| DBE\_PLDEBUGGER.backtrace | debug端调用，查看当前的调用栈。 |
| DBE\_PLDEBUGGER.enable\_breakpoint | debug端调用，激活被禁用的断点。 |
| DBE\_PLDEBUGGER.disable\_breakpoint | debug端调用，禁用已激活的断点。 |
| DBE\_PLDEBUGGER.finish | debug端调用，继续调试，直到断点或返回上一层调用栈。 |
| DBE\_PLDEBUGGER.set\_var | debug端调用，为变量进行赋值操作。 |

### DBE\_PLDEBUGGER.turn\_on

该函数用于标记某一存储过程为可调试，执行turn\_on后server端可以执行该存储过程来进行调试。需要用户根据系统表PG\_PROC手动获取存储过程oid，传入函数中。turn\_on后本session内执行该存储过程会停在第一条sql前等待debug端的调试操作。该设置会在session断连后默认被清理掉。目前不支持对启用自治事务的存储过程/函数进行调试。

函数原型为：

DBE\_PLDEBUGGER.turn\_on(Oid)  
RETURN Record;

**表 1** turn\_on 入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| func\_oid | IN oid | 函数oid。 |
| nodename | OUT text | 节点名称。 |
| port | OUT integer | 连接端口号。 |

### DBE\_PLDEBUGGER.turn\_off

用于去掉turn\_on添加的调试标记，返回值表示成功或失败。可通过DBE\_PLDEBUGGER.local\_debug\_server\_info查找已经turn\_on的存储过程oid。

函数原型为：

DBE\_PLDEBUGGER.turn\_off(Oid)  
RETURN boolean;

**表 1** turn\_off 入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| func\_oid | IN oid | 函数oid。 |
| turn\_off | OUT boolean | turn off是否成功。 |

### DBE\_PLDEBUGGER.local\_debug\_server\_info

用于查找当前连接中已经turn\_on的存储过程oid。便于用户确认在调试哪些存储过程，需要通过funcoid和pg\_proc配合使用。

**表 1** local\_debug\_server\_info 返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | OUT text | 节点名称。 |
| port | OUT bigint | 端口号。 |
| funcoid | OUT oid | 存储过程oid。 |

### DBE\_PLDEBUGGER.attach

server端执行存储过程，停在第一条语句前，等待debug端关联。debug端调用attach，传入nodename和port，关联到该存储过程上。

如果调试过程中报错，attach会自动失效；如果调试过程中attach到其他存储过程上，当前attach的调试也会失效。

**表 1** attach 入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | IN text | 节点名称。 |
| port | IN integer | 连接端口号。 |
| funcoid | OUT oid | 函数id。 |
| funcname | OUT text | 函数名。 |
| lineno | OUT integer | 当前调试运行的下一行行号。 |
| query | OUT text | 当前调试的下一行函数源码。 |

### DBE\_PLDEBUGGER.info\_locals

debug端调试过程中，调用info\_locals，打印当前存储过程内变量。该函数入参frameno表示查询遍历的栈层数，支持无入参调用，缺省为查看最上层栈变量。

**表 1** info\_locals入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| frameno | IN integer （可选） | 指定的栈层数，缺省为最顶层 |
| varname | OUT text | 变量名 |
| vartype | OUT text | 变量类型 |
| value | OUT text | 变量值 |
| package\_name | OUT text | 变量对应的package名，非package时为空 |
| isconst | OUT boolean | 是否为常量 |

### DBE\_PLDEBUGGER.next

执行存储过程中当前的sql，返回执行的下一条的行数和对应query。

**表 1** next返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcoid | OUT oid | 函数id。 |
| funcname | OUT text | 函数名。 |
| lineno | OUT integer | 当前调试运行的下一行行号。 |
| query | OUT text | 当前调试的下一行函数源码。 |

### DBE\_PLDEBUGGER.continue

执行当前存储过程，直到下一个断点或结束。返回值表示执行的下一条的行数和对应query。

函数原型为：

DBE\_PLDEBUGGER.continue()  
RETURN Record;

**表 1** continue 返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcoid | OUT oid | 函数id。 |
| funcname | OUT text | 函数名。 |
| lineno | OUT integer | 当前调试运行的下一行行号。 |
| query | OUT text | 当前调试的下一行函数源码。 |

### DBE\_PLDEBUGGER.abort

令server端执行的存储过程报错跳出。返回值表示是否成功发送abort。

函数原型为：

DBE\_PLDEBUGGER.abort()  
RETURN boolean;

**表 1** abort 返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| abort | OUT boolean | 表示成功或失败。 |

### DBE\_PLDEBUGGER.print\_var

debug端调试过程中，调用print\_var，打印当前存储过程内变量中指定的变量名及其取值。该函数入参frameno表示查询遍历的栈层数，支持不加入该参数调用，缺省为查看最上层栈变量。

**表 1** print\_var入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| var\_name | IN text | 变量。 |
| frameno | IN integer（可选） | 指定的栈层数，缺省为最顶层。 |
| varname | OUT text | 变量名。 |
| vartype | OUT text | 变量类型。 |
| value | OUT text | 变量值。 |
| package\_name | OUT text | 变量对应的package名，预留使用，当前均为空。 |
| isconst | OUT boolean | 是否为常量。 |

### DBE\_PLDEBUGGER.info\_code

debug端调试过程中，调用info\_code，查看指定存储过程的源语句和各行对应的行号，行号从函数体开始，函数头部分行号为空。

**表 1** info\_code入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcoid | IN oid | 函数ID。 |
| lineno | OUT integer | 行号。 |
| query | OUT text | 源语句。 |
| canbreak | OUT bool | 当前行是否支持断点。 |

### DBE\_PLDEBUGGER.step

debug端调试过程中，如果当前执行的是一个存储过程，则进入该存储过程继续调试，返回该存储过程第一行的行号等信息，如果当前执行的不是存储过程，则和next行为一致，执行该sql后返回下一行的行号等信息。

**表 1** step入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcoid | OUT oid | 函数id。 |
| funcname | OUT text | 函数名。 |
| lineno | OUT integer | 当前调试运行的下一行行号。 |
| query | OUT text | 当前调试的下一行函数源码。 |

### DBE\_PLDEBUGGER.add\_breakpoint

debug端调试过程中，调用add\_breakpoint增加新的断点。如果返回-1则说明指定的断点不合法，请参考DBE\_PLDEBUGGER.info\_code的canbreak字段确定断点合适的位置。

**表 1** add\_breakpoint入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcoid | IN text | 函数ID。 |
| lineno | IN integer | 行号。 |
| breakpointno | OUT integer | 断点编号。 |

### DBE\_PLDEBUGGER.delete\_breakpoint

debug端调试过程中，调用delete\_breakpoint删除已有的断点。

**表 1** delete\_breakpoint入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| breakpointno | OUT integer | 断点编号。 |
| funcoid | OUT oid | 函数ID。 |
| lineno | OUT integer | 行号。 |
| query | OUT text | 断点内容。 |
| enable | OUT boolean | 是否有效 |

### DBE\_PLDEBUGGER.info\_breakpoints

debug端调试过程中，调用info\_breakpoints，查看当前的函数断点。

**表 1** info\_breakpoints返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| breakpointno | IN integer | 断点编号。 |
| result | OUT bool | 是否成功。 |

### DBE\_PLDEBUGGER.backtrace

debug端调试过程中，调用backtrace，查看当前的调用堆栈。

**表 1** backtrace返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| frameno | OUT integer | 调用栈编号。 |
| funcname | OUT oid | 函数名。 |
| lineno | OUT integer | 行号。 |
| query | OUT text | 断点内容。 |
| funcoid | OUT oid | 函数oid。 |

### DBE\_PLDEBUGGER.enable\_breakpoint

debug端调试过程中，调用enable\_breakpoint激活已被禁用的断点。

**表 1** enable\_breakpoint入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| breakpointno | IN integer | 断点编号 |
| result | OUT bool | 是否成功 |

### DBE\_PLDEBUGGER.disable\_breakpoint

debug端调试过程中，调用disable\_breakpoint禁用已被激活的断点。

**表 1** disable\_breakpoint入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| breakpointno | IN integer | 断点编号 |
| result | OUT bool | 是否成功 |

### DBE\_PLDEBUGGER.finish

执行存储过程中当前的SQL直到下一个断点触发或执行到上层栈的下一行。

**表 1** finish入参和返回值列表

| 名称 | 类型 | 描述 |
| --- | --- | --- |
| funcoid | OUT oid | 函数id。 |
| funcname | OUT text | 函数名。 |
| lineno | OUT integer | 当前调试运行的下一行行号。 |
| query | OUT text | 当前调试的下一行函数源码。 |

### DBE\_PLDEBUGGER.set\_var

将指定的调试的存储过程中最上层栈上的变量修改为入参的取值。如果存储过程中包含同名的变量，set\_var只支持第一个变量值的设置。

**表 1** set\_var入参和返回值列表

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| var\_name | IN text | 变量名。 |
| value | IN text | 修改值。 |
| result | OUT boolean | 结果，是否成功。 |

### 附录 D. SQL 一致性

本节试图概述Vastbase在多大程度上符合当前的 SQL 标准。以下信息不是完整的一致性声明，但它以对用户合理和有用的方式详细介绍了主要主题。

SQL 标准的正式名称是 ISO/IEC 9075 "Database Language SQL"。不时发布标准修订版；最近的更新出现在 2011 年。2011 版被称为 ISO/IEC 9075:2011，或简称为 SQL:2011。之前的版本是 SQL:2008、SQL:2003、SQL:1999 和 SQL-92。每个版本都替换了前一个版本，因此与早期版本一致的声明没有官方价值。Vastbase开发旨在与标准的最新官方版本保持一致，这种一致性不与传统特征或常识相矛盾。支持 SQL 标准所需的许多功能，但有时语法或功能略有不同。随着时间的推移，可以预期进一步朝着一致性迈进。

SQL-92为一致性定义了三个特性集：Entry、Intermediate 和 Full。大多数声称符合SQL标准的数据库管理系统仅在入门级别符合要求，因为中级和完整级别中的整套功能要么过于庞大，要么与遗留行为相冲突。

从SQL:1999开始，SQL 标准定义了一大组单独的特性，而不是 SQL-92中无效的宽泛的三个级别。这些特性的很大一部分代表了“核心” 特性，每个符合标准的 SQL 实现都必须提供这些特性。其余功能完全是可选的。一些可选特性组合在一起形成“包”，SQL 实现可以声明符合这些特性，从而声明符合特定的特性组。

以SQL:2003开头的标准版本也分为多个部分。每个都有一个简写名称。请注意，这些部分没有连续编号。

* ISO/IEC 9075-1 框架（SQL/框架）
* ISO/IEC 9075-2 基础（SQL/基础）
* ISO/IEC 9075-3 调用级接口 (SQL/CLI)
* ISO/IEC 9075-4 持久存储模块 (SQL/PSM)
* ISO/IEC 9075-9 外部数据管理 (SQL/MED)
* ISO/IEC 9075-10 对象语言绑定 (SQL/OLB)
* ISO/IEC 9075-11 信息和定义模式 (SQL/Schemata)
* ISO/IEC 9075-13 使用 Java 语言 (SQL/JRT) 的例程和类型
* ISO/IEC 9075-14 XML 相关规范 (SQL/XML)

Vastbase核心包含第1、2、9、11和 14 部分。第 3 部分由 ODBC 驱动程序覆盖，第 13 部分由 PL/Java 插件覆盖，但目前尚未验证这些组件的精确一致性. 目前没有针对 Vastbase的第 4 部分和第 10 部分的实现。

Vastbase 支持 SQL:2011 的大部分主要特性。在完全符合核心要求的 179 个强制性特性中，Vastbase 至少符合 160 个。此外，还有一长串支持的可选特性。可能值得注意的是，在撰写本文时，任何数据库管理系统的当前版本都没有声称完全符合 Core SQL:2011。

在接下来的两节中，我们提供了Vastbase 支持 的那些特性的列表，然后是SQL:2011中定义的Vastbase尚不支持的特性的列表。这两个列表都是近似的：对于列为受支持的功能，可能存在不符合要求的次要细节，并且实际上可能实现了大部分不受支持的功能。文档的主体总是包含最准确的信息，说明什么是有效的，什么是无效的。

## DBE\_PLDEVELOPER

DBE\_PLDEVELOPER下系统表用于记录PLPGSQL包、函数及存储过程编译过程中需要记录的信息。

### DBE\_PLDEVELOPER.gs\_source

用于记录PLPGSQL对象（存储过程、函数、包、包体）编译相关信息，具体内容见下列字段描述。

打开plsql\_show\_all\_error参数后，会把成功或失败的PLPGSQL对象编译信息记录在此表中，如果关闭plsql\_show\_all\_error参数则只会将正确的编译相关信息插入此表中。

**表 1** DBE\_PLDEVELOPER.gs\_source字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| id | oid | 对象的ID。 |
| owner | bigint | 对象创建用户ID。 |
| nspid | oid | 对象的模式ID。 |
| name | name | 对象名。 |
| type | text | 对象类型（procedure/function/package/package body）。 |
| status | boolean | 是否创建成功。 |
| src | text | 对象创建的原始语句。 |

### DBE\_PLDEVELOPER.gs\_errors

用于记录PLPGSQL对象（存储过程、函数、包、包体）编译过程中遇到的报错信息，具体内容见下列字段描述。

打开plsql\_show\_all\_error参数后，如果编译过程中存在报错，则会跳过报错继续编译并把报错信息记录在gs\_errors中，如果关闭plsql\_show\_all\_error参数则不会将相关信息插入此表中。

**表 1** DBE\_PLDEVELOPER.gs\_errors字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| id | oid | 对象的ID。 |
| owner | bigint | 对象创建用户ID。 |
| nspid | oid | 对象的模式ID。 |
| name | name | 对象名。 |
| type | text | 对象类型（procedure/function/package/package body）。 |
| line | integer | 行号。 |
| src | text | 对象创建的原始语句。 |

## DB4AI Schema

DB4AI模式在AI特性中主要是用来存储和管理数据集版本。模式中保存数据表的原始视图快照，每一个数据版本的更改记录以及版本快照的管理信息。模式面向普通用户，用户可在该模式下查找特性DB4AI.SNAPSHOT创建的快照版本信息。

### DB4AI.SNAPSHOT

SNAPSHOT表记录当前用户通过特性DB4AI.SNAPSHOT存储的快照。

**表 1** db4ai.snapshot表属性

| **名称** | **类型** | **描述** | **实例** |
| --- | --- | --- | --- |
| id | bigint | 当前快照的ID。 | 1 |
| parent\_id | bigint | 父快照的ID。 | 0 |
| matrix\_id | bigint | CSS模式下快照的矩阵ID，否则为NULL。 | 0 |
| root\_id | bigint | 初始快照的ID，通过db4ai.create\_snapshot()从操作数据构建。 | 0 |
| schema | name | 导出快照视图的模式。 | public |
| name | name | 快照的名称，包括版本后缀。 | example0@1.1.0 |
| owner | name | 创建此快照的用户的名称。 | nw |
| commands | text[] | 记录如何从其根快照生成到此快照的SQL语句的完整列表。 | {DELETE,“WHERE id > 7”} |
| comment | text | 快照说明。 | inherits from @1.0.0 |
| published | boolean | TRUE，当且仅当快照当前已发布。 | f |
| archived | boolean | TRUE，当且仅当快照当前已存档。 | f |
| created | timestamp without time zone | 快照创建日期的时间戳。 | 2021-08-25 10:59:52.955604 |
| row\_count | bigint | 此快照中的数据行数。 | 8 |

### DB4AI.CREATE\_SNAPSHOT

CREATE\_SNAPSHOT是DB4AI特性用于创建快照的接口函数。通过语法CREATE SNAPSHOT调用。

**表 1** DB4AI.CREATE\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字，默认值是当前用户或者PUBLIC。 |
| i\_name | IN NAME | 快照名称。 |
| i\_commands | IN TEXT[] | 定义数据获取的SQL命令。 |
| i\_vers | IN NAME | 版本后缀。 |
| i\_comment | IN TEXT | 快照描述。 |
| res | OUT db4ai.snapshot\_name | 结果。 |

### DB4AI.CREATE\_SNAPSHOT\_INTERNAL

CREATE\_SNAPSHOT\_INTERNAL是db4ai.create\_snapshot函数的内置执行函数。函数存在信息校验，无法直接调用。

**表 1** DB4AI.CREATE\_SNAPSHOT\_INTERNAL入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| s\_id | IN BIGINT | 快照ID。 |
| i\_schema | IN NAME | 快照存储的名字空间。 |
| i\_name | IN NAME | 快照名称。 |
| i\_commands | IN TEXT[] | 定义数据获取的SQL命令。 |
| i\_comment | IN TEXT | 快照描述。 |
| i\_owner | IN NAME | 快照拥有者。 |

### DB4AI.PREPARE\_SNAPSHOT

PREPARE\_SNAPSHOT是DB4AI特性中数据准备模型训练和解释快照进行协作。快照为所有应用更改的数据和文档提供了完整的序列。通过语法PREPARE SNAPSHOT调用。

**表 1** DB4AI.PREPARE\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字，默认值是当前用户或者PUBLIC。 |
| i\_parent | IN NAME | 父快照名称。 |
| i\_commands | IN TEXT[] | 定义快照修改的DDL和DML命令。 |
| i\_vers | IN NAME | 版本后缀。 |
| i\_comment | IN TEXT | 此数据策展单元的说明。 |
| res | OUT db4ai.snapshot\_name | 结果。 |

### DB4AI.PREPARE\_SNAPSHOT\_INTERNAL

PREPARE\_SNAPSHOT\_INTERNAL是db4ai.prepare\_snapshot函数的内置执行函数。函数存在信息校验，无法直接调用。

**表 1** DB4AI.PREPARE\_SNAPSHOT\_INTERNAL入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| s\_id | IN BIGINT | 快照ID。 |
| p\_id | IN BIGINT | 父快照ID。 |
| m\_id | IN BIGINT | 矩阵id。 |
| r\_id | IN BIGINT | 根快照ID。 |
| i\_schema | IN NAME | 快照模式。 |
| i\_name | IN NAME | 快照名称。 |
| i\_commands | IN TEXT[] | 定义快照修改的DDL和DML命令。 |
| i\_comment | IN TEXT | 快照描述。 |
| i\_owner | IN NAME | 快照所有者。 |
| i\_idx | INOUT INT | exec\_cmds的索引。 |
| i\_exec\_cmds | INOUT TEXT[] | 用于执行的DDL和DML。 |
| i\_mapping | IN NAME[] | 将用户列映射到备份列；如果不为NULL，则生成规则。 |

### DB4AI.ARCHIVE\_SNAPSHOT

ARCHIVE\_SNAPSHOT是DB4AI特性用于存档快照的接口函数。通过语法ARCHIVE SNAPSHOT调用。生效后的快照无法参数训练等任务。

**表 1** DB4AI.ARCHIVE\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字，默认值是当前用户 |
| i\_name | IN NAME | 快照名称 |
| res | OUT db4ai.snapshot\_name | 结果 |

### DB4AI.PUBLISH\_SNAPSHOT

PUBLISH\_SNAPSHOT是DB4AI特性用于发布快照的接口函数。通过语法PUBLISH SNAPSHOT调用。

**表 1** DB4AI.PUBLISH\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字，默认值是当前用户或者PUBLIC |
| i\_name | IN NAME | 快照名称 |
| res | OUT db4ai.snapshot\_name | 结果 |

### DB4AI.MANAGE\_SNAPSHOT\_INTERNAL

MANAGE\_SNAPSHOT\_INTERNAL是DB4AI.PUBLISH\_SNAPSHOT和DB4AI.ARCHIVE\_SNAPSHOT函数的内置执行函数。函数存在信息校验，无法直接调用。

**表 1** DB4AI.MANAGE\_SNAPSHOT\_INTERNAL入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字 |
| i\_name | IN NAME | 快照名称 |
| publish | IN BOOLEN | 是否是发布状态 |
| res | OUT db4ai.snapshot\_name | 结果 |

### DB4AI.SAMPLE\_SNAPSHOT

SAMPLE\_SNAPSHOT是DB4AI特性用于对基数据进行采样生成快照的接口函数。通过语法SAMPLE SNAPSHOT调用。

**表 1** DB4AI.SAMPLE\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字 |
| i\_parent | IN NAME | 父快照名称 |
| i\_sample\_infixes | IN NAME[] | 示例快照名称中缀 |
| i\_sample\_ratios | IN NUMBER[] | 每个样本的大小，作为父集的比率 |
| i\_stratify | IN NAME[] | 分层策略 |
| i\_sample\_comments | IN TEXT[] | 示例快照描述 |
| res | OUT db4ai.snapshot\_name | 结果 |

### DB4AI.PURGE\_SNAPSHOT

PURGE\_SNAPSHOT是DB4AI特性用于删除快照的接口函数。通过语法PURGE SNAPSHOT调用。

**表 1** DB4AI.PURGE\_SNAPSHOT入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字 |
| i\_name | IN NAME | 快照名称 |
| res | OUT db4ai.snapshot\_name | 结果 |

### DB4AI.PURGE\_SNAPSHOT\_INTERNAL

PURGE\_SNAPSHOT\_INTERNAL是DB4AI.PURGE\_SNAPSHOT函数的内置执行函数。函数存在信息校验，无法直接调用

**表 1** DB4AI.PURGE\_SNAPSHOT\_INTERNAL入参和返回值列表

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| i\_schema | IN NAME | 快照存储的模式名字 |
| i\_name | IN NAME | 快照名称 |

# SQL语法参考

本章主要介绍VastbaseG100支持的SQL语法。

## SQL基本要素

### VastbaseSQL

**什么是SQL**

SQL是用于访问和处理数据库的标准计算机语言。

SQL提供了各种任务的语句，包括：

* 查询数据。
* 在表中插入、更新和删除行。
* 创建、替换、更改和删除对象。
* 控制对数据库及其对象的访问。
* 保证数据库的一致性和完整性。

SQL语言由用于处理数据库和数据库对象的命令和函数组成。该语言还会强制实施有关数据类型、表达式和文本使用的规则。因此在章节《SQL语法参考》，除了SQL语法参考外，还会看到有关数据类型、表达式、函数和操作符等信息。

**SQL发展简史**

SQL发展简史如下：

* 1986年，ANSI X3.135-1986，ISO/IEC 9075:1986，SQL-86
* 1989年，ANSI X3.135-1989，ISO/IEC 9075:1989，SQL-89
* 1992年，ANSI X3.135-1992，ISO/IEC 9075:1992，SQL-92（SQL2）
* 1999年，ISO/IEC 9075:1999，SQL:1999（SQL3）
* 2003年，ISO/IEC 9075:2003，SQL:2003（SQL4）
* 2011年，ISO/IEC 9075:200N，SQL:2011（SQL5）

**Vastbase支持的SQL标准**

Vastbase默认支持SQL2、SQL3和SQL4的主要特性。

### 关键字

SQL里有保留字和非保留字之分。根据标准，保留字决不能用做其他标识符。非保留字只是在特定的环境里有特殊的含义，而在其他环境里是可以用做标识符的。

标识符的命名需要遵守如下规范：

* 标识符需要为字母、下划线、数字（0-9）或美元符号（$）。
* 标识符必须以字母（a-z）或下划线（\_）开头。
* fig: **说明：**
  + 此命名规范为建议项，非强制项。
  + 特殊情况下可以使用双引号规避特殊字符报错。

**表 1** SQL关键字

| **关键字** | **vastbase** | **SQL:1999** | **SQL-92** |
| --- | --- | --- | --- |
| ABORT | 非保留 | - | - |
| ABS | - | 非保留 | - |
| ABSOLUTE | 非保留 | 保留 | 保留 |
| ACCESS | 非保留 | - | - |
| ACCOUNT | 非保留 | - | - |
| ACTION | 非保留 | 保留 | 保留 |
| ADA | - | 非保留 | 非保留 |
| ADD | 非保留 | 保留 | 保留 |
| ADMIN | 非保留 | 保留 | - |
| AFTER | 非保留 | 保留 | - |
| AGGREGATE | 非保留 | 保留 | - |
| ALGORITHM | 非保留 | - | - |
| ALIAS | - | 保留 | - |
| ALL | 保留 | 保留 | 保留 |
| ALLOCATE | - | 保留 | 保留 |
| ALSO | 非保留 | - | - |
| ALTER | 非保留 | 保留 | 保留 |
| ALWAYS | 非保留 | - | - |
| ANALYSE | 保留 | - | - |
| ANALYZE | 保留 | - | - |
| AND | 保留 | 保留 | 保留 |
| ANY | 保留 | 保留 | 保留 |
| APP | 非保留 | - | - |
| APPEND | 非保留 | - | - |
| ARCHIVE | 非保留 | - | - |
| ARE | - | 保留 | 保留 |
| ARRAY | 保留 | 保留 | - |
| AS | 保留 | 保留 | 保留 |
| ASC | 保留 | 保留 | 保留 |
| ASENSITIVE | - | 非保留 | - |
| ASSERTION | 非保留 | 保留 | 保留 |
| ASSIGNMENT | 非保留 | 非保留 | - |
| ASYMMETRIC | 保留 | 非保留 | - |
| AT | 非保留 | 保留 | 保留 |
| ATOMIC | - | 非保留 | - |
| ATTRIBUTE | 非保留 | - | - |
| AUDIT | 非保留 | - | - |
| AUTHID | 保留 | - | - |
| AUTHORIZATION | 保留（可以是函数或类型） | 保留 | 保留 |
| AUTOEXTEND | 非保留 | - | - |
| AUTOMAPPED | 非保留 | - | - |
| AVG | - | 非保留 | 保留 |
| BACKWARD | 非保留 | - | - |
| BARRIER | 非保留 | - | - |
| BEFORE | 非保留 | 保留 | - |
| BEGIN | 非保留 | 保留 | 保留 |
| BEGIN\_NON\_ANOYBLOCK | 非保留 | - | - |
| BETWEEN | 非保留（不能是函数或类型） | 非保留 | 保留 |
| BIGINT | 非保留（不能是函数或类型） | - | - |
| BINARY | 保留（可以是函数或类型） | 保留 | - |
| BINARY\_DOUBLE | 非保留（不能是函数或类型） | - | - |
| BINARY\_INTEGER | 非保留（不能是函数或类型） | - | - |
| BIT | 非保留（不能是函数或类型） | 保留 | 保留 |
| BITVAR | - | 非保留 | - |
| BIT\_LENGTH | - | 非保留 | 保留 |
| BLANKS | 非保留 | - | - |
| BLOB | 非保留 | 保留 | - |
| BLOCKCHAIN | 非保留 | - | - |
| BODY | 非保留 | - | - |
| BOOLEAN | 非保留（不能是函数或类型） | 保留 | - |
| BOTH | 保留 | 保留 | 保留 |
| BUCKETCNT | 非保留（不能是函数或类型） | - | - |
| BUCKETS | 保留 | - | - |
| BREADTH | - | 保留 | - |
| BY | 非保留 | 保留 | 保留 |
| BYTEAWITHOUTORDER | 非保留（不能是函数或类型） | - | - |
| BYTEAWITHOUTORDERWITHEQUAL | 非保留（不能是函数或类型） | - | - |
| C | - | 非保留 | 非保留 |
| CACHE | 非保留 | - | - |
| CALL | 非保留 | 保留 | - |
| CALLED | 非保留 | 非保留 | - |
| CANCELABLE | 非保留 | - | - |
| CARDINALITY | - | 非保留 | - |
| CASCADE | 非保留 | 保留 | 保留 |
| CASCADED | 非保留 | 保留 | 保留 |
| CASE | 保留 | 保留 | 保留 |
| CAST | 保留 | 保留 | 保留 |
| CATALOG | 非保留 | 保留 | 保留 |
| CATALOG\_NAME | - | 非保留 | 非保留 |
| CHAIN | 非保留 | 非保留 | - |
| CHAR | 非保留（不能是函数或类型） | 保留 | 保留 |
| CHARACTER | 非保留（不能是函数或类型） | 保留 | 保留 |
| CHARACTERISTICS | 非保留 | - | - |
| CHARACTERSET | 非保留 | - | - |
| CHARACTER\_LENGTH | - | 非保留 | 保留 |
| CHARACTER\_SET\_CATALOG | - | 非保留 | 非保留 |
| CHARACTER\_SET\_NAME | - | 非保留 | 非保留 |
| CHARACTER\_SET\_SCHEMA | - | 非保留 | 非保留 |
| CHAR\_LENGTH | - | 非保留 | 保留 |
| CHECK | 保留 | 保留 | 保留 |
| CHECKED | - | 非保留 | - |
| CHECKPOINT | 非保留 | - | - |
| CLASS | 非保留 | 保留 | - |
| CLEAN | 非保留 | - | - |
| CLASS\_ORIGIN | - | 非保留 | 非保留 |
| CLIENT | 非保留 | - | - |
| CLIENT\_MASTER\_KEY | 非保留 | - | - |
| CLIENT\_MASTER\_KEYS | 非保留 | - | - |
| CLOB | 非保留 | 保留 | - |
| CLOSE | 非保留 | 保留 | 保留 |
| CLUSTER | 非保留 | - | - |
| COALESCE | 非保留（不能是函数或类型） | 非保留 | 保留 |
| COBOL | - | 非保留 | 非保留 |
| COLLATE | 保留 | 保留 | 保留 |
| COLLATION | 保留（可以是函数或类型） | 保留 | 保留 |
| COLLATION\_CATALOG | - | 非保留 | 非保留 |
| COLLATION\_NAME | - | 非保留 | 非保留 |
| COLLATION\_SCHEMA | - | 非保留 | 非保留 |
| COLUMN | 保留 | 保留 | 保留 |
| COLUMN\_ENCRYPTION\_KEY | 非保留 | - | - |
| COLUMN\_ENCRYPTION\_KEYS | 非保留 | - | - |
| COLUMN\_NAME | - | 非保留 | 非保留 |
| COMPACT | 保留（可以是函数或类型） | - | - |
| COMPATIBLE\_ILLEGAL\_CHARS | 非保留 | - | - |
| COMMAND\_FUNCTION | - | 非保留 | 非保留 |
| COMPLETE | 非保留 | - | - |
| COMMAND\_FUNCTION\_CODE | - | 非保留 | - |
| COMMENT | 非保留 | - | - |
| COMMENTS | 非保留 | - | - |
| COMMIT | 非保留 | 保留 | 保留 |
| COMMITTED | 非保留 | 非保留 | 非保留 |
| COMPRESS | 非保留 | - | - |
| COMPLETION | - | 保留 | - |
| CONCURRENTLY | 保留（可以是函数或类型） | - | - |
| CONDITION | 非保留 | - | - |
| CONDITION\_NUMBER | - | 非保留 | 非保留 |
| CONFIGURATION | 非保留 | - | - |
| CONNECT | 非保留 | 保留 | 保留 |
| CONNECTION | 非保留 | 保留 | 保留 |
| CONNECTION\_NAME | - | 非保留 | 非保留 |
| CONSTANT | 非保留 | - | - |
| CONSTRAINT | 保留 | 保留 | 保留 |
| CONSTRAINTS | 非保留 | 保留 | 保留 |
| CONSTRAINT\_CATALOG | - | 非保留 | 非保留 |
| CONSTRAINT\_NAME | - | 非保留 | 非保留 |
| CONSTRAINT\_SCHEMA | - | 非保留 | 非保留 |
| CONSTRUCTOR | - | 保留 | - |
| CONTAINS | - | 非保留 | - |
| CONTENT | 非保留 | - | - |
| CONTINUE | 非保留 | 保留 | 保留 |
| CONTVIEW | 非保留 | - | - |
| CONVERSION | 非保留 | - | - |
| CONVERT | - | 非保留 | 保留 |
| COORDINATOR | 非保留 | - | - |
| COORDINATORS | 非保留 | - | - |
| COPY | 非保留 | - | - |
| CORRESPONDING | - | 保留 | 保留 |
| COST | 非保留 | - | - |
| COUNT | - | 非保留 | 保留 |
| CREATE | 保留 | 保留 | 保留 |
| CROSS | 保留（可以是函数或类型） | 保留 | 保留 |
| CSN | 保留（可以是函数或类型） | - | - |
| CSV | 非保留 | - | - |
| CUBE | 非保留 | 保留 | - |
| CURRENT | 非保留 | 保留 | 保留 |
| CURRENT\_CATALOG | 保留 | - | - |
| CURRENT\_DATE | 保留 | 保留 | 保留 |
| CURRENT\_PATH | - | 保留 | - |
| CURRENT\_ROLE | 保留 | 保留 | - |
| CURRENT\_SCHEMA | 保留（可以是函数或类型） | - | - |
| CURRENT\_TIME | 保留 | 保留 | 保留 |
| CURRENT\_TIMESTAMP | 保留 | 保留 | 保留 |
| CURRENT\_USER | 保留 | 保留 | 保留 |
| CURSOR | 非保留 | 保留 | 保留 |
| CURSOR\_NAME | - | 非保留 | 非保留 |
| CYCLE | 非保留 | 保留 | - |
| DATA | 非保留 | 保留 | 非保留 |
| DATABASE | 非保留 | - | - |
| DATAFILE | 非保留 | - | - |
| DATANODE | 非保留 | - | - |
| DATANODES | 非保留 | - | - |
| DATE\_FORMAT | 非保留 | - | - |
| DATATYPE\_CL | 非保留 | - | - |
| DATE | 非保留（不能是函数或类型） | 保留 | 保留 |
| DELTAMERGE | 保留（可以是函数或类型） | - | - |
| DATETIME\_INTERVAL\_CODE | - | 非保留 | 非保留 |
| DATETIME\_INTERVAL\_PRECISION | - | 非保留 | 非保留 |
| DAY | 非保留 | 保留 | 保留 |
| DBCOMPATIBILITY | 非保留 | - | - |
| DEALLOCATE | 非保留 | 保留 | 保留 |
| DEC | 非保留（不能是函数或类型） | 保留 | 保留 |
| DECIMAL | 非保留（不能是函数或类型） | 保留 | 保留 |
| DECLARE | 非保留 | 保留 | 保留 |
| DECODE | 非保留（不能是函数或类型） | - | - |
| DEFAULT | 保留 | 保留 | 保留 |
| DEFAULTS | 非保留 | - | - |
| DEFERRABLE | 保留 | 保留 | 保留 |
| DEFERRED | 非保留 | 保留 | 保留 |
| DEFINED | - | 非保留 | - |
| DEFINER | 非保留 | 非保留 | - |
| DELETE | 非保留 | 保留 | 保留 |
| DELIMITER | 非保留 | - | - |
| DELIMITERS | 非保留 | - | - |
| DELTA | 非保留 | - | - |
| DEPTH | - | 保留 | - |
| DEREF | - | 保留 | - |
| DESC | 保留 | 保留 | 保留 |
| DESCRIBE | - | 保留 | 保留 |
| DESCRIPTOR | - | 保留 | 保留 |
| DESTROY | - | 保留 | - |
| DESTRUCTOR | - | 保留 | - |
| DETERMINISTIC | 非保留 | 保留 | - |
| DIAGNOSTICS | - | 保留 | 保留 |
| DICTIONARY | 非保留 | 保留 | - |
| DIRECT | 非保留 | - | - |
| DIRECTORY | 非保留 | - | - |
| DISABLE | 非保留 | - | - |
| DISCARD | 非保留 | - | - |
| DISCONNECT | 非保留 | 保留 | 保留 |
| DISPATCH | - | 非保留 | - |
| DISTINCT | 保留 | 保留 | 保留 |
| DISTRIBUTE | 非保留 | - | - |
| DISTRIBUTION | 非保留 | - | - |
| DO | 保留 | - | - |
| DOCUMENT | 非保留 | - | - |
| DOMAIN | 非保留 | 保留 | 保留 |
| DOUBLE | 非保留 | 保留 | 保留 |
| DROP | 非保留 | 保留 | 保留 |
| DUPLICATE | 非保留 | - | - |
| DYNAMIC | - | 保留 | - |
| DYNAMIC\_FUNCTION | - | 非保留 | 非保留 |
| DYNAMIC\_FUNCTION\_CODE | - | 非保留 | - |
| EACH | 非保留 | 保留 | - |
| ELSE | 保留 | 保留 | 保留 |
| ELASTIC | 非保留 | - | - |
| ENABLE | 非保留 | - | - |
| ENCLOSED | 非保留 | - | - |
| ENCODING | 非保留 | - | - |
| ENCRYPTED | 非保留 | - | - |
| ENCRYPTED\_VALUE | 非保留 | - | - |
| ENCRYPTION | 非保留 | - | - |
| ENCRYPTION\_TYPE | 非保留 | - | - |
| END | 保留 | 保留 | 保留 |
| END-EXEC | - | 保留 | 保留 |
| ENFORCED | 非保留 | - | - |
| ENUM | 非保留 | - | - |
| EOL | 非保留 | - | - |
| ERRORS | 非保留 | - | - |
| EQUALS | - | 保留 | - |
| ESCAPE | 非保留 | 保留 | 保留 |
| ESCAPING | 非保留 | - | - |
| EVERY | 非保留 | 保留 | - |
| EXCEPT | 保留 | 保留 | 保留 |
| EXCEPTION | - | 保留 | 保留 |
| EXCHANGE | 非保留 | - | - |
| EXCLUDE | 非保留 | - | - |
| EXCLUDED | 保留 | - | - |
| EXCLUDING | 非保留 | - | - |
| EXCLUSIVE | 非保留 | - | - |
| EXEC | - | 保留 | 保留 |
| EXECUTE | 非保留 | 保留 | 保留 |
| EXISTING | - | 非保留 | - |
| EXISTS | 非保留（不能是函数或类型） | 非保留 | 保留 |
| EXPIRED\_P | 非保留 | - | - |
| EXPLAIN | 非保留 | - | - |
| EXTENSION | 非保留 | - | - |
| EXTERNAL | 非保留 | 保留 | 保留 |
| EXTRACT | 非保留（不能是函数或类型） | 非保留 | 保留 |
| FALSE | 保留 | 保留 | 保留 |
| FAMILY | 非保留 | - | - |
| FAST | 非保留 | - | - |
| FEATURES | 非保留 | - | - |
| FETCH | 保留 | 保留 | 保留 |
| FENCED | 保留 | - | - |
| FIELDS | 非保留 | - | - |
| FILEHEADER | 非保留 | - | - |
| FILLER | 非保留 | - | - |
| FILTER | 非保留 | 保留 | 保留 |
| FINAL | - | 非保留 | - |
| FIRST | 非保留 | 保留 | 保留 |
| FIXED | 非保留 | 保留 | 保留 |
| FILL\_MISSING\_FIELDS | 非保留 | - | - |
| FLOAT | 非保留（不能是函数或类型） | 保留 | 保留 |
| FOLLOWING | 非保留 | - | - |
| FOR | 保留 | 保留 | 保留 |
| FORCE | 非保留 | - | - |
| FOREIGN | 保留 | 保留 | 保留 |
| FORMATTER | 非保留 | - | - |
| FORTRAN | - | 非保留 | 非保留 |
| FORWARD | 非保留 | - | - |
| FOUND | - | 保留 | 保留 |
| FREE | - | 保留 | - |
| FREEZE | 保留（可以是函数或类型） | - | - |
| FROM | 保留 | 保留 | 保留 |
| FULL | 保留（可以是函数或类型） | 保留 | 保留 |
| FUNCTION | 非保留 | 保留 | - |
| FUNCTIONS | 非保留 | - | - |
| G | - | 非保留 | - |
| GENERAL | - | 保留 | - |
| GENERATED | 非保留 | 非保留 | - |
| GET | - | 保留 | 保留 |
| GLOBAL | 非保留 | 保留 | 保留 |
| GO | - | 保留 | 保留 |
| GOTO | - | 保留 | 保留 |
| GRANT | 保留 | 保留 | 保留 |
| GRANTED | 非保留 | 非保留 | - |
| GREATEST | 非保留（不能是函数或类型） | - | - |
| GROUP | 保留 | 保留 | 保留 |
| GROUPING | 非保留（不能是函数或类型） | 保留 | - |
| GROUPPARENT | 保留 | - | - |
| HANDLER | 非保留 | - | - |
| HAVING | 保留 | 保留 | 保留 |
| HDFSDIRECTORY | 保留（可以是函数或类型） | - | - |
| HEADER | 非保留 | - | - |
| HIERARCHY | - | 非保留 | - |
| HOLD | 非保留 | 非保留 | - |
| HOST | - | 保留 | - |
| HOUR | 非保留 | 保留 | 保留 |
| IDENTIFIED | 非保留 | - | - |
| IDENTITY | 非保留 | 保留 | 保留 |
| IF | 非保留 | - | - |
| IGNORE | - | 保留 | - |
| IGNORE\_EXTRA\_DATA | 非保留 | - | - |
| ILIKE | 保留（可以是函数或类型） | - | - |
| IMMEDIATE | 非保留 | 保留 | 保留 |
| IMMUTABLE | 非保留 | - | - |
| IMPLEMENTATION | - | 非保留 | - |
| IMPLICIT | 非保留 | - | - |
| IN | 保留 | 保留 | 保留 |
| INTERNAL | 非保留 | - | - |
| INCLUDE | 非保留 | - | - |
| INCLUDING | 非保留 | - | - |
| INCREMENT | 非保留 | - | - |
| INCREMENTAL | 非保留 | - | - |
| INDEX | 非保留 | - | - |
| INDEXES | 非保留 | - | - |
| INDICATOR | - | 保留 | 保留 |
| INFILE | 非保留 | - | - |
| INFIX | - | 非保留 | - |
| INHERIT | 非保留 | - | - |
| INHERITS | 非保留 | - | - |
| INITIAL | 非保留 | - | - |
| INITIALIZE | - | 保留 | - |
| INITIALLY | 保留 | 保留 | 保留 |
| INITRANS | 非保留 | - | - |
| INLINE | 非保留 | - | - |
| INNER | 保留（可以是函数或类型） | 保留 | 保留 |
| INOUT | 非保留（不能是函数或类型） | 保留 | - |
| INPUT | 非保留 | 保留 | 保留 |
| INSENSITIVE | 非保留 | 非保留 | 保留 |
| INSERT | 非保留 | 保留 | 保留 |
| INSTANCE | - | 非保留 | - |
| INSTANTIABLE | - | 非保留 | - |
| INSTEAD | 非保留 | - | - |
| INT | 非保留（不能是函数或类型） | 保留 | 保留 |
| INTEGER | 非保留（不能是函数或类型） | 保留 | 保留 |
| INTERSECT | 保留 | 保留 | 保留 |
| INTERVAL | 非保留（不能是函数或类型） | 保留 | 保留 |
| INTO | 保留 | 保留 | 保留 |
| INVOKER | 非保留 | 非保留 | - |
| IP | 非保留 | - | - |
| IS | 保留 | 保留 | 保留 |
| ISNULL | 非保留 | - | - |
| ISOLATION | 非保留 | 保留 | 保留 |
| ITERATE | - | 保留 | - |
| JOIN | 保留（可以是函数或类型） | 保留 | 保留 |
| K | - | 非保留 | - |
| KEY | 非保留 | 保留 | 保留 |
| KEY\_PATH | 非保留 | - | - |
| KEY\_MEMBER | - | 非保留 | - |
| KEY\_STORE | 非保留 | - | - |
| KEY\_TYPE | - | 非保留 | - |
| KILL | 非保留 | - | - |
| LABEL | 非保留 | - | - |
| LANGUAGE | 非保留 | 保留 | 保留 |
| LARGE | 非保留 | 保留 | - |
| LAST | 非保留 | 保留 | 保留 |
| LATERAL | - | 保留 | - |
| LC\_COLLATE | 非保留 | - | - |
| LC\_CTYPE | 非保留 | - | - |
| LEADING | 保留 | 保留 | 保留 |
| LEAKPROOF | 非保留 | - | - |
| LEAST | 非保留（不能是函数或类型） | - | - |
| LEFT | 保留（可以是函数或类型） | 保留 | 保留 |
| LENGTH | - | 非保留 | 非保留 |
| LESS | 保留 | 保留 | - |
| LEVEL | 非保留 | 保留 | 保留 |
| LIKE | 保留（可以是函数或类型） | 保留 | 保留 |
| LIMIT | 保留 | 保留 | - |
| LIST | 非保留 | - | - |
| LISTEN | 非保留 | - | - |
| LOAD | 非保留 | - | - |
| LOCAL | 非保留 | 保留 | 保留 |
| LOCALTIME | 保留 | 保留 | - |
| LOCALTIMESTAMP | 保留 | 保留 | - |
| LOCATION | 非保留 | - | - |
| LOCATOR | - | 保留 | - |
| LOCK | 非保留 | - | - |
| LOG | 非保留 | - | - |
| LOGGING | 非保留 | - | - |
| LOGIN\_ANY | 非保留 | - | - |
| LOGIN\_FAILURE | 非保留 | - | - |
| LOGIN\_SUCCESS | 非保留 | - | - |
| LOGOUT | 非保留 | - | - |
| LOOP | 非保留 | - | - |
| LOWER | - | 非保留 | 保留 |
| MAP | - | 保留 | - |
| MAPPING | 非保留 | - | - |
| MASKING | 非保留 | - | - |
| MASTER | 非保留 | - | - |
| MATCH | 非保留 | 保留 | 保留 |
| MATCHED | 非保留 | - | - |
| MATERIALIZED | 非保留 | - | - |
| MAX | - | 非保留 | 保留 |
| MAXEXTENTS | 非保留 | - | - |
| MAXSIZE | 非保留 | - | - |
| MAXTRANS | 非保留 | - | - |
| MAXVALUE | 保留 | - | - |
| MERGE | 非保留 | - | - |
| MESSAGE\_LENGTH | - | 非保留 | 非保留 |
| MESSAGE\_OCTET\_LENGTH | - | 非保留 | 非保留 |
| MESSAGE\_TEXT | - | 非保留 | 非保留 |
| METHOD | - | 非保留 | - |
| MIN | - | 非保留 | 保留 |
| MINEXTENTS | 非保留 | - | - |
| MINUS | 保留 | - | - |
| MINUTE | 非保留 | 保留 | 保留 |
| MINVALUE | 非保留 | - | - |
| MOD | - | 非保留 | - |
| MODE | 非保留 | - | - |
| MODEL | 非保留 | - | - |
| MODIFIES | - | 保留 | - |
| MODIFY | 保留 | 保留 | - |
| MODULE | - | 保留 | 保留 |
| MONTH | 非保留 | 保留 | 保留 |
| MORE | - | 非保留 | 非保留 |
| MOVE | 非保留 | - | - |
| MOVEMENT | 非保留 | - | - |
| MUMPS | - | 非保留 | 非保留 |
| NAME | 非保留 | 非保留 | 非保留 |
| NAMES | 非保留 | 保留 | 保留 |
| NATIONAL | 非保留（不能是函数或类型） | 保留 | 保留 |
| NATURAL | 保留（可以是函数或类型） | 保留 | 保留 |
| NCHAR | 非保留（不能是函数或类型） | 保留 | 保留 |
| NCLOB | - | 保留 | - |
| NEW | - | 保留 | - |
| NEXT | 非保留 | 保留 | 保留 |
| NO | 非保留 | 保留 | 保留 |
| NOCOMPRESS | 非保留 | - | - |
| NOCYCLE | 非保留 | - | - |
| NODE | 非保留 | - | - |
| NOLOGGING | 非保留 | - | - |
| NOMAXVALUE | 非保留 | - | - |
| NOMINVALUE | 非保留 | - | - |
| NONE | 非保留（不能是函数或类型） | 保留 | - |
| NOT | 保留 | 保留 | 保留 |
| NOTHING | 非保留 | - | - |
| NOTIFY | 非保留 | - | - |
| NOTNULL | 保留（可以是函数或类型） | - | - |
| NOWAIT | 非保留 | - | - |
| NULL | 保留 | 保留 | 保留 |
| NULLABLE | - | 非保留 | 非保留 |
| NULLCOLS | 非保留 | - | - |
| NULLIF | 非保留（不能是函数或类型） | 非保留 | 保留 |
| NULLS | 非保留 | - | - |
| NUMBER | 非保留（不能是函数或类型） | 非保留 | 非保留 |
| NUMERIC | 非保留（不能是函数或类型） | 保留 | 保留 |
| NUMSTR | 非保留 | - | - |
| NVARCHAR | 非保留（不能是函数或类型） | - | - |
| NVARCHAR2 | 非保留（不能是函数或类型） | - | - |
| NVL | 非保留（不能是函数或类型） | - | - |
| OBJECT | 非保留 | 保留 | - |
| OCTET\_LENGTH | - | 非保留 | 保留 |
| OF | 非保留 | 保留 | 保留 |
| OFF | 非保留 | 保留 | - |
| OFFSET | 保留 | - | - |
| OIDS | 非保留 | - | - |
| OLD | - | 保留 | - |
| ON | 保留 | 保留 | 保留 |
| ONLY | 保留 | 保留 | 保留 |
| OPEN | - | 保留 | 保留 |
| OPERATION | - | 保留 | - |
| OPERATOR | 非保留 | - | - |
| OPTIMIZATION | 非保留 | - | - |
| OPTION | 非保留 | 保留 | 保留 |
| OPTIONALLY | 非保留 | - | - |
| OPTIONS | 非保留 | 非保留 | - |
| OR | 保留 | 保留 | 保留 |
| ORDER | 保留 | 保留 | 保留 |
| ORDINALITY | - | 保留 | - |
| OUT | 非保留（不能是函数或类型） | 保留 | - |
| OUTER | 保留（可以是函数或类型） | 保留 | 保留 |
| OUTPUT | - | 保留 | 保留 |
| OVER | 非保留 | - | - |
| OVERLAPS | 保留（可以是函数或类型） | 非保留 | 保留 |
| OVERLAY | 非保留（不能是函数或类型） | 非保留 | - |
| OVERRIDING | - | 非保留 | - |
| OWNED | 非保留 | - | - |
| OWNER | 非保留 | - | - |
| PACKAGE | 非保留 | - | - |
| PACKAGES | 非保留 | - | - |
| PAD | - | 保留 | 保留 |
| PARAMETER | - | 保留 | - |
| PARAMETERS | - | 保留 | - |
| PARAMETER\_MODE | - | 非保留 | - |
| PARAMETER\_NAME | - | 非保留 | - |
| PARAMETER\_ORDINAL\_POSITION | - | 非保留 | - |
| PARAMETER\_SPECIFIC\_CATALOG | - | 非保留 | - |
| PARAMETER\_SPECIFIC\_NAME | - | 非保留 | - |
| PARAMETER\_SPECIFIC\_SCHEMA | - | 非保留 | - |
| PARSER | 非保留 | - | - |
| PARTIAL | 非保留 | 保留 | 保留 |
| PARTITION | 非保留 | - | - |
| PARTITIONS | 非保留 | - | - |
| PASCAL | - | 非保留 | 非保留 |
| PASSING | 非保留 | - | - |
| PASSWORD | 非保留 | - | - |
| PATH | - | 保留 | - |
| PCTFREE | 非保留 | - | - |
| PER | 非保留 | - | - |
| PERM | 非保留 | - | - |
| PERCENT | 非保留 | - | - |
| PERFORMANCE | 保留 | - | - |
| PLACING | 保留 | - | - |
| PLAN | 非保留 | - | - |
| PLANS | 非保留 | - | - |
| PLI | - | 非保留 | 非保留 |
| POOL | 非保留 | - | - |
| POLICY | 非保留 | - | - |
| POSITION | 非保留（不能是函数或类型） | 非保留 | 保留 |
| POSTFIX | - | 保留 | - |
| PRECEDING | 非保留 | - | - |
| PRECISION | 非保留（不能是函数或类型） | 保留 | 保留 |
| PREDICT | 非保留 | - | - |
| PREFERRED | 非保留 | - | - |
| PREFIX | 非保留 | 保留 | - |
| PREORDER | - | 保留 | - |
| PREPARE | 非保留 | 保留 | 保留 |
| PREPARED | 非保留 | - | - |
| PRESERVE | 非保留 | 保留 | 保留 |
| PRIMARY | 保留 | 保留 | 保留 |
| PRIOR | 非保留 | 保留 | 保留 |
| PRIORER | 保留 | - | - |
| PRIVATE | 非保留 | - | - |
| PRIVILEGE | 非保留 | - | - |
| PRIVILEGES | 非保留 | 保留 | 保留 |
| PROCEDURAL | 非保留 | - | - |
| PROCEDURE | 保留 | 保留 | 保留 |
| PROFILE | 非保留 | - | - |
| PUBLIC | - | 保留 | 保留 |
| PUBLICATION | 非保留 | - | - |
| PUBLISH | 非保留 | - | - |
| PURGE | 非保留 | - | - |
| QUERY | 非保留 | - | - |
| QUOTE | 非保留 | - | - |
| RANDOMIZED | 非保留 | - | - |
| RANGE | 非保留 | - | - |
| RATIO | 非保留 | - | - |
| RAW | 非保留 | - | - |
| READ | 非保留 | 保留 | 保留 |
| READS | - | 保留 | - |
| REAL | 非保留（不能是函数或类型） | 保留 | 保留 |
| REASSIGN | 非保留 | - | - |
| REBUILD | 非保留 | - | - |
| RECHECK | 非保留 | - | - |
| RECURSIVE | 非保留 | 保留 | - |
| RECYCLEBIN | 保留（可以是函数或类型） | - | - |
| REDISANYVALUE | 非保留 | - | - |
| REF | 非保留 | 保留 | - |
| REFERENCES | 保留 | 保留 | 保留 |
| REFERENCING | - | 保留 | - |
| REFRESH | 非保留 | - | - |
| REINDEX | 非保留 | - | - |
| REJECT | 保留 | - | - |
| RELATIVE | 非保留 | 保留 | 保留 |
| RELEASE | 非保留 | - | - |
| RELOPTIONS | 非保留 | - | - |
| REMOTE | 非保留 | - | - |
| REMOVE | 非保留 | - | - |
| RENAME | 非保留 | - | - |
| REPEATABLE | 非保留 | 非保留 | 非保留 |
| REPLACE | 非保留 | - | - |
| REPLICA | 非保留 | - | - |
| RESET | 非保留 | - | - |
| RESIZE | 非保留 | - | - |
| RESOURCE | 非保留 | - | - |
| RESTART | 非保留 | - | - |
| RESTRICT | 非保留 | 保留 | 保留 |
| RESULT | - | 保留 | - |
| RETURN | 非保留 | 保留 | - |
| RETURNED\_LENGTH | - | 非保留 | 非保留 |
| RETURNED\_OCTET\_LENGTH | - | 非保留 | 非保留 |
| RETURNED\_SQLSTATE | - | 非保留 | 非保留 |
| RETURNING | 保留 | - | - |
| RETURNS | 非保留 | 保留 | - |
| REUSE | 非保留 | - | - |
| REVOKE | 非保留 | 保留 | 保留 |
| RIGHT | 保留（可以是函数或类型） | 保留 | 保留 |
| ROLE | 非保留 | 保留 | - |
| ROLES | 非保留 | - | - |
| ROLLBACK | 非保留 | 保留 | 保留 |
| ROLLUP | 非保留 | 保留 | - |
| ROTATION | 非保留 | - | - |
| ROUTINE | - | 保留 | - |
| ROUTINE\_CATALOG | - | 非保留 | - |
| ROUTINE\_NAME | - | 非保留 | - |
| ROUTINE\_SCHEMA | - | 非保留 | - |
| ROW | 非保留（不能是函数或类型） | 保留 | - |
| ROWS | 非保留 | 保留 | 保留 |
| ROWTYPE | 非保留 | - | - |
| ROW\_COUNT | - | 非保留 | 非保留 |
| RULE | 非保留 | - | - |
| ROWNUM | 保留 | 保留 | - |
| SAMPLE | 非保留 | - | - |
| SAVEPOINT | 非保留 | 保留 | - |
| SCALE | - | 非保留 | 非保留 |
| SCHEMA | 非保留 | 保留 | 保留 |
| SCHEMA\_NAME | - | 非保留 | 非保留 |
| SCOPE | - | 保留 | - |
| SCROLL | 非保留 | 保留 | 保留 |
| SEARCH | 非保留 | 保留 | - |
| SECOND | 非保留 | 保留 | 保留 |
| SECTION | - | 保留 | 保留 |
| SECURITY | 非保留 | 非保留 | - |
| SELECT | 保留 | 保留 | 保留 |
| SELF | - | 非保留 | - |
| SENSITIVE | - | 非保留 | - |
| SEQUENCE | 非保留 | 保留 | - |
| SEQUENCES | 非保留 | - | - |
| SERIALIZABLE | 非保留 | 非保留 | 非保留 |
| SERVER | 非保留 | - | - |
| SERVER\_NAME | - | 非保留 | 非保留 |
| SESSION | 非保留 | 保留 | 保留 |
| SESSION\_USER | 保留 | 保留 | 保留 |
| SET | 非保留 | 保留 | 保留 |
| SETOF | 非保留（不能是函数或类型） | - | - |
| SETS | 非保留 | 保留 | - |
| SHARE | 非保留 | - | - |
| SHIPPABLE | 非保留 | - | - |
| SHOW | 非保留 | - | - |
| SHUTDOWN | 非保留 | - | - |
| SIBLINGS | 非保留 | - | - |
| SIMILAR | 保留（可以是函数或类型） | 非保留 | - |
| SIMPLE | 非保留 | 非保留 | - |
| SIZE | 非保留 | 保留 | 保留 |
| SKIP | 非保留 | - | - |
| SLICE | 非保留 | - | - |
| SMALLDATETIME\_FORMAT | 非保留 | - | - |
| SMALLDATETIME | 非保留（不能是函数或类型） | - | - |
| SMALLINT | 非保留（不能是函数或类型） | 保留 | 保留 |
| SNAPSHOT | 非保留 | - | - |
| SOME | 保留 | 保留 | 保留 |
| SOURCE | 非保留 | 非保留 | - |
| SPACE | 非保留 | 保留 | 保留 |
| SPECIFIC | - | 保留 | - |
| SPECIFICTYPE | - | 保留 | - |
| SPECIFIC\_NAME | - | 非保留 | - |
| SPILL | 非保留 | - | - |
| SPLIT | 非保留 | - | - |
| SQL | - | 保留 | 保留 |
| SQLCODE | - | - | 保留 |
| SQLERROR | - | - | 保留 |
| SQLEXCEPTION | - | 保留 | - |
| SQLSTATE | - | 保留 | 保留 |
| SQLWARNING | - | 保留 | - |
| STABLE | 非保留 | - | - |
| STANDALONE | 非保留 | - | - |
| START | 非保留 | 保留 | - |
| STATE | - | 保留 | - |
| STATEMENT | 非保留 | 保留 | - |
| STATEMENT\_ID | 非保留 | - | - |
| STATIC | - | 保留 | - |
| STATISTICS | 非保留 | - | - |
| STDIN | 非保留 | - | - |
| STDOUT | 非保留 | - | - |
| STORAGE | 非保留 | - | - |
| STORE | 非保留 | - | - |
| STORED | 非保留 | - | - |
| STRATIFY | 非保留 | - | - |
| STREAM | 非保留 | - | - |
| STRICT | 非保留 | - | - |
| STRIP | 非保留 | - | - |
| STRUCTURE | - | 保留 | - |
| STYLE | - | 非保留 | - |
| SUBCLASS\_ORIGIN | - | 非保留 | 非保留 |
| SUBLIST | - | 非保留 | - |
| SUBPARTITION | 非保留 | - | - |
| SUBSCRIPTION | 非保留 | - | - |
| SUBSTRING | 非保留（不能是函数或类型） | 非保留 | 保留 |
| SUM | - | 非保留 | 保留 |
| SYMMETRIC | 保留 | 非保留 | - |
| SYNONYM | 非保留 | - | - |
| SYS\_REFCURSOR | 非保留 | - | - |
| SYSDATE | 保留 | - | - |
| SYSID | 非保留 | - | - |
| SYSTEM | 非保留 | 非保留 | - |
| SYSTEM\_USER | - | 保留 | 保留 |
| TABLE | 保留 | 保留 | 保留 |
| TABLES | 非保留 | - | - |
| TABLESAMPLE | 保留（可以是函数或类型） | - | - |
| TABLESPACE | 非保留 | - | - |
| TABLE\_NAME | - | 非保留 | 非保留 |
| TARGET | 非保留 | - | - |
| TIME\_FORMAT | 非保留 | - | - |
| TIMESTAMP\_FORMAT | 非保留 | - | - |
| TEMP | 非保留 | - | - |
| TEMPLATE | 非保留 | - | - |
| TEMPORARY | 非保留 | 保留 | 保留 |
| TERMINATE | - | 保留 | - |
| TERMINATED | 非保留 | - | - |
| TEXT | 非保留 | - | - |
| THAN | 非保留 | 保留 | - |
| THEN | 保留 | 保留 | 保留 |
| TIME | 非保留（不能是函数或类型） | 保留 | 保留 |
| TIMECAPSULE | 保留（可以是函数或类型） | - | - |
| TIMESTAMP | 非保留（不能是函数或类型） | 保留 | 保留 |
| TIMESTAMPDIFF | 非保留（不能是函数或类型） | - | - |
| TIMEZONE\_HOUR | - | 保留 | 保留 |
| TIMEZONE\_MINUTE | - | 保留 | 保留 |
| TINYINT | 非保留（不能是函数或类型） | - | - |
| TO | 保留 | 保留 | 保留 |
| TRAILING | 保留 | 保留 | 保留 |
| TRANSACTION | 非保留 | 保留 | 保留 |
| TRANSACTIONS\_COMMITTED | - | 非保留 | - |
| TRANSACTIONS\_ROLLED\_BACK | - | 非保留 | - |
| TRANSACTION\_ACTIVE | - | 非保留 | - |
| TRANSFORM | 非保留 | 非保留 | - |
| TRANSFORMS | - | 非保留 | - |
| TRANSLATE | - | 非保留 | 保留 |
| TRANSLATION | - | 保留 | 保留 |
| TREAT | 非保留（不能是函数或类型） | 保留 | - |
| TRIGGER | 非保留 | 保留 | - |
| TRIGGER\_CATALOG | - | 非保留 | - |
| TRIGGER\_NAME | - | 非保留 | - |
| TRIGGER\_SCHEMA | - | 非保留 | - |
| TRIM | 非保留（不能是函数或类型） | 非保留 | 保留 |
| TRUE | 保留 | 保留 | 保留 |
| TRUNCATE | 非保留 | - | - |
| TRUSTED | 非保留 | - | - |
| TSFIELD | 非保留 | - | - |
| TSTAG | 非保留 | - | - |
| TSTIME | 非保留 | - | - |
| TYPE | 非保留 | 非保留 | 非保留 |
| TYPES | 非保留 | - | - |
| UESCAPE | - | - | - |
| UNBOUNDED | 非保留 | - | - |
| UNCOMMITTED | 非保留 | 非保留 | 非保留 |
| UNDER | - | 保留 | - |
| UNENCRYPTED | 非保留 | - | - |
| UNION | 保留 | 保留 | 保留 |
| UNIQUE | 保留 | 保留 | 保留 |
| UNKNOWN | 非保留 | 保留 | 保留 |
| UNLIMITED | 非保留 | - | - |
| UNLISTEN | 非保留 | - | - |
| UNLOCK | 非保留 | - | - |
| UNLOGGED | 非保留 | - | - |
| UNNAMED | - | 非保留 | 非保留 |
| UNNEST | - | 保留 | - |
| UNTIL | 非保留 | - | - |
| UNUSABLE | 非保留 | - | - |
| UPDATE | 非保留 | 保留 | 保留 |
| UPPER | - | 非保留 | 保留 |
| USAGE | - | 保留 | 保留 |
| USEEOF | 非保留 | - | - |
| USER | 保留 | 保留 | 保留 |
| USER\_DEFINED\_TYPE\_CATALOG | - | 非保留 | - |
| USER\_DEFINED\_TYPE\_NAME | - | 非保留 | - |
| USER\_DEFINED\_TYPE\_SCHEMA | - | 非保留 | - |
| USING | 保留 | 保留 | 保留 |
| VACUUM | 非保留 | - | - |
| VALID | 非保留 | - | - |
| VALIDATE | 非保留 | - | - |
| VALIDATION | 非保留 | - | - |
| VALIDATOR | 非保留 | - | - |
| VALUE | 非保留 | 保留 | 保留 |
| VALUES | 非保留（不能是函数或类型） | 保留 | 保留 |
| VARCHAR | 非保留（不能是函数或类型） | 保留 | 保留 |
| VARCHAR2 | 非保留（不能是函数或类型） | - | - |
| VARIABLE | - | 保留 | - |
| VARIABLES | 非保留 | - | - |
| VARIADIC | 保留 | - | - |
| VARYING | 非保留 | 保留 | 保留 |
| VCGROUP | 非保留 | - | - |
| VERBOSE | 保留（可以是函数或类型） | - | - |
| VERSION | 非保留 | - | - |
| VERIFY | 保留 | - | - |
| VIEW | 非保留 | 保留 | 保留 |
| VOLATILE | 非保留 | - | - |
| WAIT | 非保留 | - | - |
| WEAK | 非保留 | - | - |
| WHEN | 保留 | 保留 | 保留 |
| WHENEVER | - | 保留 | 保留 |
| WHERE | 保留 | 保留 | 保留 |
| WHITESPACE | 非保留 | - | - |
| WINDOW | 保留 | - | - |
| WITH | 保留 | 保留 | 保留 |
| WITHIN | 非保留 | - | - |
| WITHOUT | 非保留 | 保留 | - |
| WORK | 非保留 | 保留 | 保留 |
| WORKLOAD | 非保留 | - | - |
| WRAPPER | 非保留 | - | - |
| WRITE | 非保留 | 保留 | 保留 |
| XML | 非保留 | - | - |
| XMLATTRIBUTES | 非保留（不能是函数或类型） | - | - |
| XMLCONCAT | 非保留（不能是函数或类型） | - | - |
| XMLELEMENT | 非保留（不能是函数或类型） | - | - |
| XMLEXISTS | 非保留（不能是函数或类型） | - | - |
| XMLFOREST | 非保留（不能是函数或类型） | - | - |
| XMLPARSE | 非保留（不能是函数或类型） | - | - |
| XMLPI | 非保留（不能是函数或类型） | - | - |
| XMLROOT | 非保留（不能是函数或类型） | - | - |
| XMLSERIALIZE | 非保留（不能是函数或类型） | - | - |
| YEAR | 非保留 | 保留 | 保留 |
| YES | 非保留 | - | - |
| ZONE | 非保留 | 保留 | 保留 |

### 数据类型

数据类型是一组值的集合以及定义在这个值集上的一组操作。Vastbase数据库是由表的集合组成的，而各表中的列定义了该表，每一列都属于一种数据类型，Vastbase根据数据类型有相应函数对其内容进行操作，例如Vastbase可对数值型数据进行加、减、乘、除操作。

#### 日期-时间类型

Vastbase支持的日期/时间类型请参见表1。该类型的操作符和内置函数请参考：SQL语法参考->函数和操作符->时间和日期处理函数和操作符章节。

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif**说明：**  
如果其他的数据库时间格式和vastbase的时间格式不一致，可通过修改配置参数DateStyle的值来保持一致。

**表 1** 日期/时间类型

| **名称** | **描述** | **存储空间** |
| --- | --- | --- |
| DATE | 日期和时间。 | 4字节（兼容模式A下存储空间大小为8字节） |
| TIME [(p)] [WITHOUT TIME ZONE] | 只用于一日内时间。  p表示小数点后的精度，取值范围为0~6。 | 8字节 |
| TIME [(p)] [WITH TIME ZONE] | 只用于一日内时间，带时区。别名为TIMETZ  p表示小数点后的精度，取值范围为0~6。 | 12字节 |
| TIMESTAMP[(p)] [WITHOUT TIME ZONE] | 日期和时间。  p表示小数点后的精度，取值范围为0~6。 | 8字节 |
| TIMESTAMP[(p)][WITH TIME ZONE] | 日期和时间，带时区。别名为TIMESTAMPTZ。  p表示小数点后的精度，取值范围为0~6。 | 8字节 |
| SMALLDATETIME | 日期和时间，不带时区。  精确到分钟，秒位大于等于30秒进一位。 | 8字节 |
| INTERVAL DAY (l) TO SECOND (p) | 时间间隔，X天X小时X分X秒。  l：天数的精度，取值范围为0~6。兼容性考虑，目前未实现具体功能。  p：秒数的精度，取值范围为0~6。小数末尾的零不显示。 | 16字节 |
| INTERVAL [FIELDS] [ (p) ] | 时间间隔。  fields：可以是YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、DAY TO HOUR、DAY TO MINUTE、DAY TO SECOND、HOUR TO MINUTE、HOUR TO SECOND、MINUTE TO SECOND。  p：秒数的精度，取值范围为0~6，且fields为SECOND、DAY TO SECOND、HOUR TO SECOND或MINUTE TO SECOND时，参数p才有效。小数末尾的零不显示。 | 12字节 |
| reltime | 相对时间间隔。格式为：  X years X mons X days XX:XX:XX。  采用儒略历计时，规定一年为365.25天，一个月为30天，计算输入值对应的相对时间间隔，输出采用POSTGRES格式。 | 4字节 |
| abstime | 日期和时间。格式为：  YYYY-MM-DD hh:mm:ss+timezone  取值范围为1901-12-13 20:45:53 GMT~2038-01-18 23:59:59 GMT，精度为秒。 | 4字节 |

**示例1：创建date类型数据。**

1. 创建表。

CREATE TABLE date\_type\_tab(coll date);

2、插入数据。

INSERT INTO date\_type\_tab VALUES (date '12-10-2010');

3、查看数据。

SELECT \* FROM date\_type\_tab;

结果返回如下：

coll   
---------------------  
 2010-12-10 00:00:00  
(1 row)

4、删除表。

DROP TABLE date\_type\_tab;

**示例2：创建time without time zone、time with time zone、timestamp without time zone、timestamp with time zone、smalldatetime五中类型数据。**

1、创建表。

CREATE TABLE time\_type\_tab (da time without time zone ,dai time with time zone,dfgh timestamp without time zone,dfga timestamp with time zone, vbg smalldatetime);

2、插入数据。

INSERT INTO time\_type\_tab VALUES ('21:21:21','21:21:21 pst','2010-12-12','2013-12-11 pst','2003-04-12 04:05:06');

3、查看数据。

SELECT \* FROM time\_type\_tab;

结果返回如下

da | dai | dfgh | dfga | vbg   
----------+-------------+---------------------+------------------------+---------------------  
 21:21:21 | 21:21:21-08 | 2010-12-12 00:00:00 | 2013-12-11 16:00:00+08 | 2003-04-12 04:05:00  
(1 row)

4、删除表。

DROP TABLE time\_type\_tab;

**示例3：创建天数间隔INTERVAL类型数据。**

1. 创建表。

CREATE TABLE day\_type\_tab (a int,b INTERVAL DAY(3) TO SECOND (4));

2、插入数据。

INSERT INTO day\_type\_tab VALUES (1, INTERVAL '3' DAY);

3、查看数据。

SELECT \* FROM day\_type\_tab;

结果返回如下：

a | b   
---+--------  
 1 | 3 days  
(1 row)

4、删除表。

DROP TABLE day\_type\_tab;

**示例4：创建年份间隔INTERVAL类型数据。**

1、创建表。

CREATE TABLE year\_type\_tab(a int, b interval year (6));

1. 插入数据。

INSERT INTO year\_type\_tab VALUES(1,interval '2' year);

3、查看数据。

SELECT \* FROM year\_type\_tab;

结果返回如下：

a | b   
---+---------  
 1 | 2 years  
(1 row)

4、删除表。

DROP TABLE year\_type\_tab;

**日期输入**

日期和时间的输入几乎可以是任何合理的格式，包括ISO-8601格式、SQL-兼容格式、传统POSTGRES格式或者其它的形式。系统支持按照日、月、年的顺序自定义日期输入。如果把DateStyle参数设置为MDY就按照“月-日-年”解析，设置为DMY就按照“日-月-年”解析，设置为YMD就按照“年-月-日”解析。

日期的文本输入需要加单引号包围，语法如下：

type [ ( p ) ] 'value'

可选的精度声明中的p是一个整数，表示在秒域中小数部分的位数。表2显示了date类型的输入方式。

**表 2** 日期输入方式

| **例子** | **描述** |
| --- | --- |
| 1999-01-08 | ISO 8601格式（建议格式），任何方式下都是1999年1月8号。 |
| January 8, 1999 | 在任何datestyle输入模式下都无歧义。 |
| 1/8/1999 | 有歧义，在MDY模式下是一月八号，在DMY模式下是八月一号。 |
| 1/18/1999 | MDY模式下是一月十八日，其它模式下被拒绝。 |
| 01/02/03 | MDY模式下的2003年1月2日。  DMY模式下的2003年2月1日。  YMD模式下的2001年2月3日。 |
| 1999-Jan-08 | 任何模式下都是1月8日。 |
| Jan-08-1999 | 任何模式下都是1月8日。 |
| 08-Jan-1999 | 任何模式下都是1月8日。 |
| 99-Jan-08 | YMD模式下是1月8日，否则错误。 |
| 08-Jan-99 | 一月八日，除了在YMD模式下是错误的之外。 |
| Jan-08-99 | 一月八日，除了在YMD模式下是错误的之外。 |
| 19990108 | ISO 8601；任何模式下都是1999年1月8日。 |
| 990108 | ISO 8601；任何模式下都是1999年1月8日。 |
| 1999.008 | 年和年里的第几天。 |
| J2451187 | 儒略日。 |
| January 8, 99 BC | 公元前99年。 |

示例：

* --创建表。  
  vastbase=# CREATE TABLE date\_type\_tab(coll date);  
    
  --插入数据。  
  vastbase=# INSERT INTO date\_type\_tab VALUES (date '12-10-2010');  
    
  --查看数据。  
  vastbase=# SELECT \* FROM date\_type\_tab;  
   coll   
  ---------------------  
   2010-12-10 00:00:00  
  (1 row)  
    
  --查看日期格式。  
  vastbase=# SHOW datestyle;  
   DateStyle   
  -----------  
   ISO, MDY  
  (1 row)  
    
  --设置日期格式。  
  vastbase=# SET datestyle='YMD';  
  SET  
    
  --插入数据。  
  vastbase=# INSERT INTO date\_type\_tab VALUES(date '2010-12-11');  
    
  --查看数据。  
  vastbase=# SELECT \* FROM date\_type\_tab;  
   coll   
  ---------------------  
   2010-12-10 00:00:00  
   2010-12-11 00:00:00  
  (2 rows)  
    
  --删除表。  
  vastbase=# DROP TABLE date\_type\_tab;

**时间**

时间类型包括time [ (p) ] without time zone和time [ (p) ] with time zone。如果只写time等效于time without time zone。

如果在time without time zone类型的输入中声明了时区，则会忽略这个时区。

时间输入类型的详细信息请参见表3，时区输入类型的详细信息请参加表4。

**表 3** 时间输入

| **例子** | **描述** |
| --- | --- |
| 05:06.8 | ISO 8601 |
| 4:05:06 | ISO 8601 |
| 4:05 | ISO 8601 |
| 40506 | ISO 8601 |
| 4:05 AM | 与04:05一样，AM不影响数值 |
| 4:05 PM | 与16:05一样，输入小时数必须<= 12 |
| 04:05:06.789-8 | ISO 8601 |
| 04:05:06-08:00 | ISO 8601 |
| 04:05-08:00 | ISO 8601 |
| 040506-08 | ISO 8601 |
| 04:05:06 PST | 缩写的时区 |
| 2003-04-12 04:05:06 America/New\_York | 用名称声明的时区 |

**表 4** 时区输入

| **例子** | **描述** |
| --- | --- |
| PST | 太平洋标准时间（Pacific Standard Time） |
| America/New\_York | 完整时区名称 |
| -8:00 | ISO 8601与PST的偏移 |
| -800 | ISO 8601与PST的偏移 |
| -8 | ISO 8601与PST的偏移 |

示例：

* vastbase=# SELECT time '04:05:06';  
   time   
  ----------  
   04:05:06  
  (1 row)  
    
  vastbase=# SELECT time '04:05:06 PST';  
   time   
  ----------  
   04:05:06  
  (1 row)  
    
  vastbase=# SELECT time with time zone '04:05:06 PST';  
   timetz   
  -------------  
   04:05:06-08  
  (1 row)

**特殊值**

vastbase支持几个特殊值，在读取的时候将被转换成普通的日期/时间值，请参考表5。

**表 5** 特殊值

| **输入字符串** | **适用类型** | **描述** |
| --- | --- | --- |
| epoch | date、timestamp | 1970-01-01 00:00:00+00 （Unix系统零时） |
| infinity | timestamp | 比任何其他时间戳都晚 |
| -infinity | timestamp | 比任何其他时间戳都早 |
| now | date、time、timestamp | 当前事务的开始时间 |
| today | date、timestamp | 今日午夜 |
| tomorrow | date、timestamp | 明日午夜 |
| yesterday | date、timestamp | 昨日午夜 |
| allballs | time | 00:00:00.00 UTC |

**时间段输入**

reltime的输入方式可以采用任何合法的时间段文本格式，包括数字形式（含负数和小数）及时间形式，其中时间形式的输入支持SQL标准格式、ISO-8601格式、POSTGRES格式等。另外，文本输入需要加单引号。

时间段输入的详细信息请参考表6。

**表 6** 时间段输入

| **输入示例** | **输出结果** | **描述** |
| --- | --- | --- |
| 60 | 2 mons | 采用数字表示时间段，默认单位是day，可以是小数或负数。特别的，负数时间段，在语义上，可以理解为“早于多久”。 |
| 31.25 | 1 mons 1 days 06:00:00 |
| -365 | -12 mons -5 days |
| 1 years 1 mons 8 days 12:00:00 | 1 years 1 mons 8 days 12:00:00 | 采用POSTGRES格式表示时间段，可以正负混用，不区分大小写，输出结果为将输入时间段计算并转换得到的简化POSTGRES格式时间段。 |
| -13 months -10 hours | -1 years -25 days -04:00:00 |
| -2 YEARS +5 MONTHS 10 DAYS | -1 years -6 mons -25 days -06:00:00 |
| P-1.1Y10M | -3 mons -5 days -06:00:00 | 采用ISO-8601格式表示时间段，可以正负混用，"H"不区分大小写，输出结果为将输入时间段计算并转换得到的简化POSTGRES格式时间段。 |
| -12H | -12:00:00 |

示例：

* --创建表。  
  vastbase=# CREATE TABLE reltime\_type\_tab(col1 character(30), col2 reltime);  
    
  --插入数据。  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('90', '90');  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('-366', '-366');  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('1975.25', '1975.25');  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('-2 YEARS +5 MONTHS 10 DAYS', '-2 YEARS +5 MONTHS 10 DAYS');  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('30 DAYS 12:00:00', '30 DAYS 12:00:00');  
  vastbase=# INSERT INTO reltime\_type\_tab VALUES ('P-1.1Y10M', 'P-1.1Y10M');  
    
  --查看数据。  
  vastbase=# SELECT \* FROM reltime\_type\_tab;  
   col1 | col2   
  --------------------------------+-------------------------------------  
   1975.25 | 5 years 4 mons 29 days  
   -2 YEARS +5 MONTHS 10 DAYS | -1 years -6 mons -25 days -06:00:00  
   P-1.1Y10M | -3 mons -5 days -06:00:00  
   -366 | -1 years -18:00:00  
   90 | 3 mons  
   30 DAYS 12:00:00 | 1 mon 12:00:00  
  (6 rows)  
    
  --删除表。  
  vastbase=# DROP TABLE reltime\_type\_tab;

#### 数值类型

##### 整数类型

本文列出了所有的可用的数值类型（整数类型、任意精度型、浮点类型、序列整型）。数字操作符和相关的内置函数请参见：SQL语法参考->SQL基本要素->函数和操作符->数字操作函数和操作符。

**表 1** 整数类型

| **名称** | **描述** | **存储空间** | **范围** |
| --- | --- | --- | --- |
| TINYINT | 微整数，别名为INT1。 | 1字节 | 0 ~ 255 |
| SMALLINT | 小范围整数，别名为INT2。 | 2字节 | -32,768 ~ +32,767 |
| INTEGER | 常用的整数，别名为INT4。 | 4字节 | -2,147,483,648 ~ +2,147,483,647 |
| BINARY\_INTEGER | 常用的整数INTEGER的别名。 | 4字节 | -2,147,483,648 ~ +2,147,483,647 |
| BIGINT | 大范围的整数，别名为INT8。 | 8字节 | -9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807 |
| int16 | 十六字节的大范围证书，目前不支持用户用于建表等使用。 | 16字节 | -170,141,183,460,469,231,731,687,303,715,884,105,728 ~ +170,141,183,460,469,231,731,687,303,715,884,105,727 |

* image1TINYINT、SMALLINT、INTEGER、BIGINT和INT16类型存储各种范围的数字，也就是整数。试图存储超出范围以外的数值将会导致错误。
* 常用的类型是INTEGER，因为它提供了在范围、存储空间、性能之间的最佳平衡。一般只有取值范围确定不超过SMALLINT的情况下，才会使用SMALLINT类型。而只有在INTEGER的范围不够的时候才使用BIGINT，因为前者相对快得多。

**示例**

**示例1：**创建具有TINYINT类型数据的表。

1. 创建测试表。

CREATE TABLE int\_type\_t1(IT\_COL1 TINYINT);

1. 向表中插入数据。

INSERT INTO int\_type\_t1 VALUES(10);

1. 查看数据。

SELECT \* FROM int\_type\_t1;

返回结果为：

it\_col1

---------

10

(1 row)

1. 删除表。

DROP TABLE int\_type\_t1;

**示例2：**创建具有TINYINT,INTEGER,BIGINT类型数据的表。

1、创建测试表。

CREATE TABLE int\_type\_t2 (

a TINYINT,

b TINYINT,

c INTEGER,

d BIGINT

);

2、向表中插入数据。

INSERT INTO int\_type\_t2 VALUES(100, 10, 1000, 10000);

3、查看数据。

SELECT \* FROM int\_type\_t2;

返回结果为：

a | b | c | d

-----+----+------+-------

100 | 10 | 1000 | 10000

(1 row)

4、删除表。

DROP TABLE int\_type\_t2;

##### 任意精度型

**表 2** 任意精度型

| **名称** | **描述** | **存储空间** | **范围** |
| --- | --- | --- | --- |
| NUMERIC[(p[,s])],  DECIMAL[(p[,s])] | 精度p取值范围为[1,1000]，标度s取值范围为[0,p]。  说明：  p为总位数，s为小数位数。 | 用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。 | 未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。 |
| NUMBER[(p[,s])] | NUMERIC类型的别名。 | 用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。 | 未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。 |

**示例**

**示例1：**创建具有DECIMAL类型数据的表。

1、创建测试表。

CREATE TABLE decimal\_type\_t1 ( DT\_COL1 DECIMAL(10,4));

2、向表中插入数据。

INSERT INTO decimal\_type\_t1 VALUES(123456.122331);

1. 查询表中的数据。

SELECT \* FROM decimal\_type\_t1;

返回结果为：

dt\_col1

-------------

123456.1223

(1 row)

4、删除表。

DROP TABLE decimal\_type\_t1;

**示例2：**创建具有NUMERIC类型数据的表。

1、创建测试表。

CREATE TABLE numeric\_type\_t1 (NT\_COL1 NUMERIC(10,4));

2、向表中插入数据。

INSERT INTO numeric\_type\_t1 VALUES(123456.12354);

3、查看数据。

SELECT \* FROM numeric\_type\_t1;

返回结果为：

nt\_col1

-------------

123456.1235

(1 row)

4、删除表。

DROP TABLE numeric\_type\_t1;

* image1与整数类型相比，任意精度类型需要更大的存储空间，其存储效率、运算效率以及压缩比效果都要差一些。在进行数值类型定义时，优先选择整数类型。当且仅当数值超出整数可表示最大范围时，再选用任意精度类型。
* 使用Numeric/Decimal进行列定义时，建议指定该列的精度p以及标度s。

##### 序列整型

**表 3** 序列整型

| **名称** | **描述** | **存储空间** | **范围** |
| --- | --- | --- | --- |
| SMALLSERIAL | 二字节序列整型。 | 2字节 | -32,768 ~ +32,767 |
| SERIAL | 四字节序列整型。 | 4字节 | -2,147,483,648 ~ +2,147,483,647 |
| BIGSERIAL | 八字节序列整型。 | 8字节 | -9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807 |
| LARGESERIAL | 默认插入十六字节序列整形，实际数值类型和numeric相同。 | 变长类型，每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。 | 小数点前最大131,072位，小数点后最大16,383位。 |

image1SMALLSERIAL、SERIAL、BIGSERIAL和LARGESERIAL类型不是真正的类型，只是为在表中设置唯一标识做的概念上的便利。因此，创建一个整数字段，并且把它的缺省数值安排为从一个序列发生器读取。应用了一个NOT NULL约束以确保NULL不会被插入。在大多数情况下用户可能还希望附加一个UNIQUE或PRIMARY KEY约束避免意外地插入重复的数值，但这个不是自动的。最后，将序列发生器从属于那个字段，这样当该字段或表被删除的时候也一并删除它。目前只支持在创建表时候指定SERIAL列，不可以在已有的表中，增加SERIAL列。另外临时表也不支持创建SERIAL列。因为SERIAL不是真正的类型，也不可以将表中存在的列类型转化为SERIAL。

**示例**

**示例1：**创建具有SMALLSERIAL类型数据的表。

1. 创建表。

CREATE TABLE smallserial\_type\_tab(a SMALLSERIAL);

1. 插入数据。

INSERT INTO smallserial\_type\_tab VALUES(default);

INSERT INTO smallserial\_type\_tab VALUES(default);

1. 查看数据。

SELECT \* FROM smallserial\_type\_tab;

返回结果为：

a

---

1

2

(2 rows)

**示例2：**创建具有 SERIAL类型数据的表。

1. 创建表。

CREATE TABLE serial\_type\_tab(b SERIAL);

1. 插入数据。

INSERT INTO serial\_type\_tab VALUES(default);

INSERT INTO serial\_type\_tab VALUES(default);

1. 查看结果。

SELECT \* FROM serial\_type\_tab;

返回结果为：

b

---

1

2

(2 rows)

**示例3：**创建具有 BIGSERIAL类型数据的表。

1. 创建表。

CREATE TABLE bigserial\_type\_tab(c BIGSERIAL);

1. 插入数据。

INSERT INTO serial\_type\_tab VALUES(default);

INSERT INTO serial\_type\_tab VALUES(default);

1. 查看数据。

SELECT \* FROM serial\_type\_tab;

返回结果：

b

---

1

2

(2 rows)

**示例4：**创建具有LARGESERIAL类型数据的表。

1. 创建表。

CREATE TABLE largeserial\_type\_tab(c LARGESERIAL);

1. 插入数据。

INSERT INTO largeserial\_type\_tab VALUES(default);

INSERT INTO largeserial\_type\_tab VALUES(default);

1. 查看数据。

SELECT \* FROM largeserial\_type\_tab;

返回结果为：

c

---

1

2

(2 rows)

##### 浮点类型

**表 4** 浮点类型

| **名称** | **描述** | **存储空间** | **范围** |
| --- | --- | --- | --- |
| REAL | 单精度浮点数，不精准。 | 4字节 | -3.402E+38~3.402E+38，6位十进制数字精度。 |
| FLOAT4 |
| DOUBLE PRECISION，别名BINARY\_DOUBLE | 双精度浮点数，不精准。 | 8字节 | -1.79E+308~1.79E+308，15位十进制数字精度。 |
| FLOAT8 |
| FLOAT[(p)] | 浮点数，不精准。精度p取值范围为[1,53]。说明：p为精度，表示总位数。 | 4字节或8字节 | 根据精度p不同选择REAL或DOUBLE PRECISION作为内部表示。如不指定精度，内部用DOUBLE PRECISION表示。 |
| BINARY\_DOUBLE | 是DOUBLE PRECISION的别名。 | 8字节 | -1.79E+308~1.79E+308，15位十进制数字精度。 |
| DEC[(p[,s])] | 精度p取值范围为[1,1000]，标度s取值范围为[0,p]。说明：p为总位数，s为小数位位数。 | 用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。 | 未指定精度的情况下，小数点前最大131,072位，小数点后最大16,383位。 |
| INTEGER[(p[,s])] | 精度p取值范围为[1,1000]，标度s取值范围为[0,p]。 | 用户声明精度。每四位（十进制位）占用两个字节，然后在整个数据上加上八个字节的额外开销。 | - |

**示例**

1. 创建测试表。

CREATE TABLE float\_type\_t2

(

FT\_COL1 INTEGER,

FT\_COL2 FLOAT4,

FT\_COL3 FLOAT8,

FT\_COL4 FLOAT(3),

FT\_COL5 BINARY\_DOUBLE,

FT\_COL6 DECIMAL(10,4),

FT\_COL7 INTEGER(6,3)

);

1. 插入数据。

INSERT INTO float\_type\_t2 VALUES(10,10.365456,123456.1234,10.3214, 321.321, 123.123654, 123.123654);

1. 查看数据。

SELECT \* FROM float\_type\_t2 ;

返回结果为：

ft\_col1 | ft\_col2 | ft\_col3 | ft\_col4 | ft\_col5 | ft\_col6 | ft\_col7

---------+---------+-------------+---------+---------+----------+---------

10 | 10.3655 | 123456.1234 | 10.3214 | 321.321 | 123.1237 | 123.124

(1 row)

1. 删除表。

DROP TABLE float\_type\_t2;

#### 货币类型

货币类型存储带有固定小数精度的货币金额。

表1中显示的范围假设有两位小数。可以以任意格式输入，包括整型、浮点型或者典型的货币格式（如“$1,000.00”）。根据区域字符集，输出一般是最后一种形式。

**表 1** 货币类型

| **名称** | **存储容量** | **描述** | **范围** |
| --- | --- | --- | --- |
| money | 8 字节 | 货币金额 | -92233720368547758.08 到 +92233720368547758.07 |

numeric、int和bigint类型的值可以转化为money类型。如果从real和double precision类型转换到money类型，可以先转化为numeric类型，再转化为money类型，例如：

* vastbase=# SELECT '12.34'::float8::numeric::money;

这种用法是不推荐使用的。浮点数不应该用来处理货币类型，因为小数点的位数可能会导致错误。

money类型的值可以转换为numeric类型而不丢失精度。转换为其他类型可能丢失精度，并且必须通过以下两步来完成：

* vastbase=# SELECT '52093.89'::money::numeric::float8;

当一个money类型的值除以另一个money类型的值时，结果是double precision（也就是一个纯数字，而不是money类型）；在运算过程中货币单位相互抵消。

#### 布尔类型

**表 1** 布尔类型

| **名称** | **描述** | **存储空间** | **取值** |
| --- | --- | --- | --- |
| BOOLEAN | 布尔类型 | 1字节。 | true：真false：假null：未知（unknown） |

* “真”值的有效文本值是：
* TRUE、't'、'true'、'y'、'yes'、'1' 、'TRUE'、true、整数范围内1~2^63-1、整数范围内-1~-2^63。
* “假”值的有效文本值是：
* FALSE、'f'、'false'、'n'、'no'、'0'、0、'FALSE'、false。

使用TRUE和FALSE是比较规范的用法（也是SQL兼容的用法）。

**示例**

用字母t和f输出Boolean值。

* --创建表。  
  vastbase=# CREATE TABLE bool\_type\_t1   
  (  
   BT\_COL1 BOOLEAN,  
   BT\_COL2 TEXT  
  );  
    
  --插入数据。  
  vastbase=# INSERT INTO bool\_type\_t1 VALUES (TRUE, 'sic est');  
    
  vastbase=# INSERT INTO bool\_type\_t1 VALUES (FALSE, 'non est');  
    
  --查看数据。  
  vastbase=# SELECT \* FROM bool\_type\_t1;  
   bt\_col1 | bt\_col2   
  ---------+---------  
   t | sic est  
   f | non est  
  (2 rows)  
    
  vastbase=# SELECT \* FROM bool\_type\_t1 WHERE bt\_col1 = 't';  
   bt\_col1 | bt\_col2   
  ---------+---------  
   t | sic est  
  (1 row)  
    
  --删除表。  
  vastbase=# DROP TABLE bool\_type\_t1;

#### 字符类型

Vastbase支持的字符类型请参见表1。字符串操作符和相关的内置函数请参考：SQL语法参考->SQL基本要素->函数和操作符->字符处理函数和操作符。

**表 1** 字符类型

| **名称** | **描述** | **存储空间** |
| --- | --- | --- |
| CHAR(n)CHARACTER(n)NCHAR(n) | 定长字符串，不足补空格。n是指字节长度，如不带精度n，默认精度为1。 | 最大为10MB。 |
| VARCHAR(n)CHARACTER VARYING(n) | 变长字符串。PG兼容模式下，n是字符长度。其他兼容模式下，n是指字节长度。 | 最大为10MB。 |
| VARCHAR2(n) | 变长字符串。是VARCHAR(n)类型的别名。n是指字节长度。 | 最大为10MB。 |
| NVARCHAR2(n) | 变长字符串。n是指字符长度。 | 最大为10MB。 |
| NVARCHAR(n) | 变长字符串。是NVARCHAR2(n)类型的别名。n是指字符长度。 | 最大为10MB。 |
| TEXT | 变长字符串。 | 最大为1GB-1，但还需要考虑到列描述头信息的大小， 以及列所在元组的大小限制（也小于1GB-1），因此TEXT类型最大大小可能小于1GB-1。 |
| CLOB | 文本大对象。是TEXT类型的别名。 | 最大为1GB-1，但还需要考虑到列描述头信息的大小， 以及列所在元组的大小限制（也小于1GB-1），因此CLOB类型最大大小可能小于1GB-1。 |
| bpchar | 无具体含义，是varchar类型的别名。 | |

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

1. 除了每列的大小限制以外，每个元组的总大小也不可超过1GB-1字节，主要受列的控制头信息、元组控制头信息以及元组中是否存在NULL字段等影响。
2. NCHAR为bpchar类型的别名，NCHAR(n)为bpchar(n)类型的别名。
3. 超过1GB的clob只有dbe\_lob相关高级包支持，系统函数不支持大于1GBclob。

在Vastbase里另外还有两种定长字符类型。在表2里显示。name类型只用在内部系统表中，作为存储标识符，不建议普通用户使用。该类型长度当前定为64字节（63可用字符加结束符）。类型“char”只用了一个字节的存储空间。他在系统内部主要用于系统表，主要作为简单化的枚举类型使用。

**表 2** 特殊字符类型

| **名称** | **描述** | **存储空间** |
| --- | --- | --- |
| name | 用于对象名的内部类型。 | 64字节。 |
| "char" | 单字节内部类型。 | 1字节。 |

**示例1：创建数据类型为CHARACTER()的字段。**

1. 创建表。

CREATE TABLE char\_type\_t1   
(  
 CT\_COL1 CHARACTER(4)  
);

1. 插入数据。

INSERT INTO char\_type\_t1 VALUES ('ok');

3、查询表中的数据。

SELECT ct\_col1, char\_length(ct\_col1) FROM char\_type\_t1;

结果返回如下：

ct\_col1 | char\_length   
---------+-------------  
 ok | 4  
(1 row)

4、删除表。

DROP TABLE char\_type\_t1;

**示例2：创建数据类型为VARCHAR()的字段。**

1. 创建表。

CREATE TABLE char\_type\_t2   
(  
 CT\_COL1 VARCHAR(5)  
);

1. 插入数据。

INSERT INTO char\_type\_t2 VALUES ('ok');

INSERT INTO char\_type\_t2 VALUES ('good');

3、插入的数据长度超过类型规定的长度报错。  
INSERT INTO char\_type\_t2 VALUES ('too long');   
结果返回如下：

ERROR: value too long for type character varying(5)  
CONTEXT: referenced column: ct\_col1

4、明确类型的长度，超过数据类型长度后会自动截断。

INSERT INTO char\_type\_t2 VALUES ('too long'::varchar(5));

5、查询数据。

SELECT ct\_col1, char\_length(ct\_col1) FROM char\_type\_t2;

结果返回如下：

ct\_col1 | char\_length   
---------+-------------  
 ok | 2  
 good | 4  
 too l | 5  
(3 rows)

6、删除数据。

DROP TABLE char\_type\_t2;

#### 二进制类型

Vastbase支持的二进制类型请参见表1。

**表 1** 二进制类型

| **名称** | **描述** | **存储空间** |
| --- | --- | --- |
| BLOB | 二进制大对象。说明：列存不支持BLOB类型。 | 最大为1GB-8203字节（即1073733621字节）。 |
| RAW | 变长的十六进制类型。说明：列存不支持RAW类型。 | 4字节加上实际的十六进制字符串。最大为1GB-8203字节（即1073733621字节）。 |
| BYTEA | 变长的二进制字符串。 | 4字节加上实际的二进制字符串。最大为1GB-8203字节（即1073733621字节）。 |
| BYTEAWITHOUTORDERWITHEQUALCOL | 变长的二进制字符串（密态特性新增的类型，如果加密列的加密类型指定为确定性加密，则该列的实际类型为BYTEAWITHOUTORDERWITHEQUALCOL），元命令打印加密表将显示原始数据类型。 | 4字节加上实际的二进制字符串。最大为1GB减去53字节（即1073741771字节）。 |
| BYTEAWITHOUTORDERCOL | 变长的二进制字符串（密态特性新增的类型，如果加密列的加密类型指定为随机加密，则该列的实际类型为BYTEAWITHOUTORDERCOL），元命令打印加密表将显示原始数据类型。 | 4字节加上实际的二进制字符串。最大为1GB减去53字节（即1073741771字节）。 |
| \_BYTEAWITHOUTORDERWITHEQUALCOL | 变长的二进制字符串，密态特性新增的类型。 | 4字节加上实际的二进制字符串。最大为1GB减去53字节（即1073741771字节）。 |
| \_BYTEAWITHOUTORDERCOL | 变长的二进制字符串，密态特性新增的类型。 | 4字节加上实际的二进制字符串。最大为1GB减去53字节（即1073741771字节）。 |

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif**说明：**

* 除了每列的大小限制以外，每个元组的总大小也不可超过1GB-8203字节（即1073733621字节）。
* 不支持直接使用BYTEAWITHOUTORDERWITHEQUALCOL、BYTEAWITHOUTORDERCOL、*BYTEAWITHOUTORDERWITHEQUALCOL和*BYTEAWITHOUTORDERCOL类型创建表。

示例:

* --创建表。  
  vastbase=# CREATE TABLE blob\_type\_t1   
  (  
   BT\_COL1 INTEGER,  
   BT\_COL2 BLOB,  
   BT\_COL3 RAW,  
   BT\_COL4 BYTEA  
  ) ;  
    
  --插入数据。  
  vastbase=# INSERT INTO blob\_type\_t1 VALUES(10,empty\_blob(),  
  HEXTORAW('DEADBEEF'),E'xDEADBEEF');  
    
  --查询表中的数据。  
  vastbase=# SELECT \* FROM blob\_type\_t1;  
   bt\_col1 | bt\_col2 | bt\_col3 | bt\_col4   
  ---------+---------+----------+------------  
   10 | | DEADBEEF | xdeadbeef  
  (1 row)  
    
  --删除表。  
  vastbase=# DROP TABLE blob\_type\_t1;

#### 几何类型

Vastbase支持的几何类型请参见表1。最基本的类型：点，是其它类型的基础。

**表 1** 几何类型

| **名称** | **存储空间** | **说明** | **表现形式** |
| --- | --- | --- | --- |
| point | 16字节 | 平面中的点 | (x,y) |
| line | 32字节 | （无限长）的线 | {A,B,C} |
| lseg | 32字节 | （有限）线段 | ((x1,y1),(x2,y2)) |
| box | 32字节 | 矩形 | ((x1,y1),(x2,y2)) |
| path | 16+16n字节 | 闭合路径（与多边形类似） | ((x1,y1),...) |
| path | 16+16n字节 | 开放路径 | [(x1,y1),...] |
| polygon | 40+16n字节 | 多边形（与闭合路径相似） | ((x1,y1),...) |
| circle | 24 字节 | 圆 | <(x,y),r> （圆心和半径） |

Vastbase提供了一系列的函数和操作符用来进行各种几何计算，如拉伸、转换、旋转、计算相交等。详细信息请参见：SQL语法参考->SQL基本要素->函数和操作符->几何函数和操作符。

**点**

点是几何类型的基本二维构造单位。用下面语法描述point的数值：

( x , y )  
x , y

x和y是用浮点数表示的点的坐标。

点输出使用第一种语法。

示例段（lseg）是用

SELECT point(23.4, -44.5) AS RESULT;

返回结果为：

result

--------------

(23.4,-44.5)

(1 row)

**线**

线是由线的方程式Ax + By + C = 0 表示的，这里的A和B都不是0。 类型line的值以下列的形式输入和输出：

{ A, B, C }

作为选择，下列的任意一种格式都可以用作输入：

[ ( x1 , y1 ) , ( x2 , y2 ) ]  
( ( x1 , y1 ) , ( x2 , y2 ) )  
( x1 , y1 ) , ( x2 , y2 )  
x1 , y1 , x2 , y2

**线段**

线一对点来代表的。用下面的语法描述lseg的数值：

[ ( x1 , y1 ) , ( x2 , y2 ) ]  
( ( x1 , y1 ) , ( x2 , y2 ) )  
( x1 , y1 ) , ( x2 , y2 )  
x1 , y1 , x2 , y2

(x1,y1)和(x2,y2)表示线段的端点。

线段输出使用第一种语法。

**矩形**

矩形是用一对对角点来表示的。用下面的语法描述box的值：

( ( x1 , y1 ) , ( x2 , y2 ) )  
( x1 , y1 ) , ( x2 , y2 )  
x1 , y1 , x2 , y2

(x1,y1)和(x2,y2)表示矩形的一对对角点。

矩形的输出使用第二种语法。

任何两个对角都可以出现在输入中，但按照那样的顺序，右上角和左下角的值会被重新排序以存储。

**示例**

将点转换成矩形。

SELECT box(point '(0,0)', point '(1,1)') AS RESULT;

返回结果为：

result

-------------

(1,1),(0,0)

(1 row)

**路径**

路径由一系列连接的点组成。路径可能是开放的，也就是认为列表中第一个点和最后一个点没有连接，也可能是闭合的，这时认为第一个和最后一个点连接起来。

用下面的语法描述path的数值：

[ ( x1 , y1 ) , ... , ( xn , yn ) ]  
( ( x1 , y1 ) , ... , ( xn , yn ) )  
( x1 , y1 ) , ... , ( xn , yn )  
( x1 , y1 , ... , xn , yn )  
x1 , y1 , ... , xn , yn

点表示组成路径的线段的端点。方括弧（[]）表明一个开放的路径，圆括弧（()）表明一个闭合的路径。当最外层的括号被省略，如在第三至第五语法，会假定一个封闭的路径。

路径的输出使用第一种或第二种语法输出。

**示例**

将[(0,0),(1,1),(2,0)]开放的路径转换为闭合路径。

SELECT pclose(path '[(0,0),(1,1),(2,0)]') AS RESULT;

返回结果为：

result

---------------------

((0,0),(1,1),(2,0))

(1 row)

**多边形**

多边形由一系列点代表（多边形的顶点）。多边形可以认为与闭合路径一样，但是存储方式不一样而且有自己的一套支持函数。

用下面的语法描述polygon的数值：

( ( x1 , y1 ) , ... , ( xn , yn ) )  
( x1 , y1 ) , ... , ( xn , yn )  
( x1 , y1 , ... , xn , yn )  
x1 , y1 , ... , xn , yn

点表示多边形的端点。

多边形输出使用第一种语法。

**示例**

将路径转换成多边形。

SELECT polygon(path '((0,0),(1,1),(2,0))') AS RESULT;

返回结果为：

result

---------------------

((0,0),(1,1),(2,0))

(1 row)

**圆**

圆由一个圆心和半径标识。用下面的语法描述circle的数值：

< ( x , y ) , r >  
( ( x , y ) , r )  
( x , y ) , r  
x , y , r

(x,y)表示圆心，r表示半径。

圆的输出用第一种格式。

**示例**

将圆转换成12点多边形。

SELECT polygon(circle '((0,0),2.0)') AS RESULT;

result

---------------------------------------------------------------------------------------------------------------------------

((-2,0),(-1.73205080756888,1),(-1,1.73205080756888),(-1.22464679914735e-16,2),(1,1.73205080756888),(1.返回结果为：

73205080756888,1),(2

,2.44929359829471e-16),(1.73205080756888,-0.999999999999999),(1,-1.73205080756888),(3.67394039744206e-16,-2),(-0.9999999999

99999,-1.73205080756888),(-1.73205080756888,-1))

(1 row)

#### 网络地址类型

Vastbase提供用于存储IPv4、IPv6、MAC地址的数据类型。

用这些数据类型存储网络地址比用纯文本类型好，因为这些类型提供输入错误检查和特殊的操作和功能（请参考：SQL语法参考->SQL基本要素->函数和操作符->网络地址函数和操作符）。

**表 1** 网络地址类型

| **名称** | **存储空间** | **描述** |
| --- | --- | --- |
| cidr | 7或19字节 | IPv4或IPv6网络 |
| inet | 7或19字节 | IPv4或IPv6主机和网络 |
| macaddr | 6字节 | MAC地址 |

在对inet或cidr数据类型进行排序的时候，IPv4地址总是排在IPv6地址前面，包括那些封装或者是映射在IPv6地址里的IPv4地址，比如::10.2.3.4或::ffff:10.4.3.2。

**cidr**

cidr（无类别域间路由，Classless Inter-Domain Routing）类型，保存一个IPv4或IPv6网络地址。声明网络格式为address/y，address表示IPv4或者IPv6地址，y表示子网掩码的二进制位数。如果省略y，则掩码部分使用已有类别的网络编号系统进行计算，但要求输入的数据已经包括了确定掩码所需的所有字节。

**表 2** cidr类型输入举例

| **cidr输入** | **cidr输出** | **abbrev（cidr）** |
| --- | --- | --- |
| 192.168.100.128/25 | 192.168.100.128/25 | 192.168.100.128/25 |
| 192.168/24 | 192.168.0.0/24 | 192.168.0/24 |
| 192.168/25 | 192.168.0.0/25 | 192.168.0.0/25 |
| 192.168.1 | 192.168.1.0/24 | 192.168.1/24 |
| 192.168 | 192.168.0.0/24 | 192.168.0/24 |
| 10.1.2 | 10.1.2.0/24 | 10.1.2/24 |
| 10.1 | 10.1.0.0/16 | 10.1/16 |
| 10 | 10.0.0.0/8 | 10/8 |
| 10.1.2.3/32 | 10.1.2.3/32 | 10.1.2.3/32 |
| 2001:4f8:3:ba::/64 | 2001:4f8:3:ba::/64 | 2001:4f8:3:ba::/64 |
| 2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128 | 2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128 | 2001:4f8:3:ba:2e0:81ff:fe22:d1f1 |
| ::ffff:1.2.3.0/120 | ::ffff:1.2.3.0/120 | ::ffff:1.2.3/120 |
| ::ffff:1.2.3.0/128 | ::ffff:1.2.3.0/128 | ::ffff:1.2.3.0/128 |

**inet**

inet类型在一个数据区域内保存主机的IPv4或IPv6地址，以及一个可选子网。主机地址中网络地址的位数表示子网（“子网掩码”）。如果子网掩码是32并且地址是IPv4，则这个值不表示任何子网，只表示一台主机。在IPv6里，地址长度是128位，因此128位表示唯一的主机地址。

该类型的输入格式是address/y，address表示IPv4或者IPv6地址，y是子网掩码的二进制位数。如果省略/y，则子网掩码对IPv4是32，对IPv6是128，所以该值表示只有一台主机。如果该值表示只有一台主机，/y将不会显示。

inet和cidr类型之间的基本区别是inet接受子网掩码，而cidr不接受。

**macaddr**

macaddr类型存储MAC地址，也就是以太网卡硬件地址（尽管MAC地址还用于其它用途）。可以接受下列格式：

* '08:00:2b:01:02:03'  
  '08-00-2b-01-02-03'  
  '08002b:010203'  
  '08002b-010203'  
  '0800.2b01.0203'  
  '08002b010203'

这些示例都表示同一个地址。对于数据位a到f，大小写都行。输出时都是以第一种形式展示。

#### 位串类型

位串就是一串1和0的字符串。它们可以用于存储位掩码。

Vastbase支持两种位串类型：bit(n)和bit varying(n)，这里的n是一个正整数。

bit类型的数据必须准确匹配长度n，如果存储短或者长的数据都会报错。bit varying类型的数据是最长为n的变长类型，超过n的类型会被拒绝。一个没有长度的bit等效于bit(1)，没有长度的bit varying表示没有长度限制。

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

如果用户明确地把一个位串值转换成bit(n)，则此位串右边的内容将被截断或者在右边补齐零，直到刚好n位，而不会抛出任何错误。

如果用户明确地把一个位串数值转换成bit varying(n)，如果它超过了n位，则它的右边将被截断。

* --创建表。  
  vastbase=# CREATE TABLE bit\_type\_t1   
  (  
   BT\_COL1 INTEGER,  
   BT\_COL2 BIT(3),  
   BT\_COL3 BIT VARYING(5)  
  ) ;  
    
  --插入数据。  
  vastbase=# INSERT INTO bit\_type\_t1 VALUES(1, B'101', B'00');  
    
  --插入数据的长度不符合类型的标准会报错。  
  vastbase=# INSERT INTO bit\_type\_t1 VALUES(2, B'10', B'101');  
  ERROR: bit string length 2 does not match type bit(3)  
  CONTEXT: referenced column: bt\_col2  
    
  --将不符合类型长度的数据进行转换。  
  vastbase=# INSERT INTO bit\_type\_t1 VALUES(2, B'10'::bit(3), B'101');  
    
  --查看数据。  
  vastbase=# SELECT \* FROM bit\_type\_t1;  
   bt\_col1 | bt\_col2 | bt\_col3   
  ---------+---------+---------  
   1 | 101 | 00  
   2 | 100 | 101  
  (2 rows)  
    
  --删除表。  
  vastbase=# DROP TABLE bit\_type\_t1;

#### 文本搜索类型

Vastbase提供了两种数据类型用于支持全文检索。tsvector类型表示为文本搜索优化的文件格式，tsquery类型表示文本查询。

**tsvector**

tsvector类型表示一个检索单元，通常是一个数据库表中一行的文本字段或者这些字段的组合，tsvector类型的值是一个标准词位的有序列表，标准词位就是把同一个词的变型体都标准化成相同的，在输入的同时会自动排序和消除重复。to\_tsvector函数通常用于解析和标准化文档字符串。

tsvector的值是唯一分词的分类列表，把一句话的词格式化为不同的词条，在进行分词处理的时候tsvector会自动去掉分词中重复的词条，按照一定的顺序录入。如：

* vastbase=#SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector;  
   tsvector   
  ----------------------------------------------------  
   'a' 'and' 'ate' 'cat' 'fat' 'mat' 'on' 'rat' 'sat'  
  (1 row)

从上面的例子可以看出，通过tsvector把一个字符串按照空格进行分词，分词的顺序是按照长短和字母排序的。但是如果词条中需要包含空格或标点符号，可以用引号标记：

* vastbase=#SELECT $$the lexeme ' ' contains spaces$$::tsvector;  
   tsvector   
  -------------------------------------------  
   ' ' 'contains' 'lexeme' 'spaces' 'the'  
  (1 row)

如果在词条中使用引号，可以使用双$$符号作为标记：

* vastbase=#SELECT $$the lexeme 'Joe''s' contains a quote$$::tsvector;  
   tsvector   
  ------------------------------------------------  
   'Joe''s' 'a' 'contains' 'lexeme' 'quote' 'the'  
  (1 row)

词条位置常量也可以放到词汇中：

* vastbase=#SELECT 'a:1 fat:2 cat:3 sat:4 on:5 a:6 mat:7 and:8 ate:9 a:10 fat:11 rat:12'::tsvector;  
   tsvector   
  -------------------------------------------------------------------------------  
   'a':1,6,10 'and':8 'ate':9 'cat':3 'fat':2,11 'mat':7 'on':5 'rat':12 'sat':4  
  (1 row)

位置常量通常表示文档中源字的位置。位置信息可以用于进行排名。位置常量的范围是1到16383，最大值默认是16383。相同词的重复位会被忽略掉。

拥有位置的词汇甚至可以用一个权来标记，这个权可以是A、B、C或D。默认的是D，因此输出中不会出现：

* vastbase=#SELECT 'a:1A fat:2B,4C cat:5D'::tsvector;  
   tsvector   
  ----------------------------  
   'a':1A 'cat':5 'fat':2B,4C  
  (1 row)

权可以用来反映文档结构，如：标记标题与主体文字的区别。全文检索排序函数可以为不同的权标记分配不同的优先级。

下面的示例是tsvector类型标准用法。如：

* vastbase=#SELECT 'The Fat Rats'::tsvector;  
   tsvector   
  --------------------  
   'Fat' 'Rats' 'The'  
  (1 row)

但是对于英文全文检索应用来说，上面的单词会被认为非规范化的，所以需要通过to\_tsvector函数对这些单词进行规范化处理：

* vastbase=#SELECT to\_tsvector('english', 'The Fat Rats');  
   to\_tsvector   
  -----------------  
   'fat':2 'rat':3  
  (1 row)

**tsquery**

tsquery类型表示一个检索条件，存储用于检索的词汇，并且使用布尔操作符&（AND），|（OR）和!（NOT）来组合他们，括号用来强调操作符的分组。to\_tsquery函数及plainto\_tsquery函数会将单词转换为tsquery类型前进行规范化处理。

* vastbase=#SELECT 'fat & rat'::tsquery;  
   tsquery   
  ---------------  
   'fat' & 'rat'  
  (1 row)  
    
  vastbase=#SELECT 'fat & (rat | cat)'::tsquery;  
   tsquery   
  ---------------------------  
   'fat' & ( 'rat' | 'cat' )  
  (1 row)  
    
  vastbase=#SELECT 'fat & rat & ! cat'::tsquery;  
   tsquery   
  ------------------------  
   'fat' & 'rat' & !'cat'  
  (1 row)

在没有括号的情况下，!（非）结合的最紧密，而&（和）结合的比|（或）紧密。

tsquery中的词汇可以用一个或多个权字母来标记，这些权字母限制这次词汇只能与带有匹配权的tsvector词汇进行匹配。

* vastbase=#SELECT 'fat:ab & cat'::tsquery;  
   tsquery   
  ------------------  
   'fat':AB & 'cat'  
  (1 row)

同样，tsquery中的词汇可以用\*标记来指定前缀匹配：

* vastbase=#SELECT 'super:\*'::tsquery;  
   tsquery   
  -----------  
   'super':\*  
  (1 row)

这个查询可以匹配tsvector中以“super”开始的任意单词。

请注意，前缀首先被文本搜索分词器处理，这也就意味着下面的结果为真：

* vastbase=#SELECT to\_tsvector( 'postgraduate' ) @@ to\_tsquery( 'postgres:\*' ) AS RESULT;  
   result   
  ----------  
   t  
  (1 row)

因为postgres经过处理后得到postgr：

* vastbase=#SELECT to\_tsquery('postgres:\*');  
   to\_tsquery   
  ------------  
   'postgr':\*  
  (1 row)

这样就匹配postgraduate了。

'Fat:ab & Cats'规范化转为tsquery类型结果如下：

* vastbase=#SELECT to\_tsquery('Fat:ab & Cats');  
   to\_tsquery   
  ------------------  
   'fat':AB & 'cat'  
  (1 row)

#### UUID类型

UUID数据类型用来存储RFC 4122，ISO/IEF 9834-8:2005以及相关标准定义的通用唯一标识符（UUID）。这个标识符是一个由算法产生的128位标识符，确保它不可能使用相同算法在已知的模块中产生的相同标识符。

UUID是一个小写十六进制数字的序列，由分字符分成几组，一组8位数字+三组4位数字+一组12位数字，总共32个数字代表128位，标准的UUID示例如下：

* a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11

Vastbase同样支持以其他方式输入：大写字母和数字、由花括号包围的标准格式、省略部分或所有连字符、在任意一组四位数字之后加一个连字符。示例：

* A0EEBC99-9C0B-4EF8-BB6D-6BB9BD380A11  
  {a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11}  
  a0eebc999c0b4ef8bb6d6bb9bd380a11  
  a0ee-bc99-9c0b-4ef8-bb6d-6bb9-bd38-0a11

一般是以标准格式输出。

#### HLL数据类型

HLL（HyperLoglog）是统计数据集中唯一值个数的高效近似算法。它有着计算速度快、节省空间的特点，不需要直接存储集合本身，而是存储一种名为HLL的数据结构。每当有新数据加入进行统计时，只需要把数据经过哈希计算并插入到HLL中，最后根据HLL就可以得到结果。

HLL与其他算法的比较请参见表1。

**表 1** HLL与其他算法比较

| **项目** | **Sort算法** | **Hash算法** | **HLL** |
| --- | --- | --- | --- |
| 时间复杂度 | O(nlogn) | O(n) | O(n) |
| 空间复杂度 | O(n) | O(n) | log(logn) |
| 误差率 | 0 | 0 | ≈0.8% |
| 所需存储空间 | 原始数据大小 | 原始数据大小 | 默认规格下最大16KB |

HLL在计算速度和所占存储空间上都占优势。在时间复杂度上，Sort算法需要排序至少O(nlogn)的时间，虽说Hash算法和HLL一样扫描一次全表O(n)的时间就可以得出结果，但是存储空间上，Sort算法和Hash算法都需要先把原始数据存起来再进行统计，会导致存储空间消耗巨大，而对HLL来说不需要存原始数据，只需要维护HLL数据结构，故占用空间有很大的压缩，默认规格下HLL数据结构的最大空间约为16KB。

fig: **须知：**

* 当前默认规格下可计算最大distinct值的数量约为1.1e+15个，误差率为0.8%。用户应注意如果计算结果超过当前规格下distinct最大值会导致计算结果误差率变大，或导致计算结果失败并报错。
* 用户在首次使用该特性时，应该对业务的distinct value做评估，选取适当的配置参数并做验证，以确保精度符合要求：
  + 当前默认参数下，可以计算的distinct值为1.1e+15，如果计算得到的distinct值为NaN，需要调整log2m，或者采用其他算法计算distinct值。
  + 虽然hash算法存在极低的hash collision概率，但是建议用户在首次使用时，选取2-3个hash seed验证，如果得到的distinct value相差不大，则可以从该组seed中任选一个作为hash seed。

HLL中主要的数据结构，请参见表2。

**表 2** HyperLogLog中主要数据结构

| **数据类型** | **功能描述** |
| --- | --- |
| hll | hll头部为27字节长度字段，默认规格下数据段长度0~16KB，可直接计算得到distinct值。 |

创建HLL数据类型时，可以支持0~4个参数入参，具体的参数含义与参数规格同函数hll\_empty一致。第一个参数为log2m，表示分桶数的对数值，取值范围10~16；第二个参数为log2explicit，表示Explicit模式的阈值大小，取值范围0~12；第三个参数为log2sparse，表示Sparse模式的阈值大小，取值范围0~14；第四个参数为duplicatecheck，表示是否启用duplicatecheck，取值范围为0~1。当入参输入值为-1时，会采用默认值设定HLL的参数。可以通过d或d+查看HLL类型的参数。

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

创建HLL数据类型时，根据入参的行为不同，结果不同：

* 创建HLL类型时对应入参不输入或输入-1，采用默认值设定对应的HLL参数。
* 输入合法范围的入参，对应HLL参数采用输入值。
* 输入不合法范围的入参，创建HLL类型报错。
* -- 创建hll类型的表，不指定入参  
  vastbase=# create table t1 (id integer, set hll);  
  vastbase=# d t1  
   Table "public.t1"  
   Column | Type | Modifiers  
  --------+---------+-----------  
   id | integer |  
   set | hll |  
    
  -- 创建hll类型的表，指定前两个入参，后两个采用默认值  
  vastbase=# create table t2 (id integer, set hll(12,4));  
  vastbase=# d t2  
   Table "public.t2"  
   Column | Type | Modifiers  
  --------+----------------+-----------  
   id | integer |  
   set | hll(12,4,12,0) |  
    
  --创建hll类型的表，指定第三个入参，其余采用默认值  
  vastbase=# create table t3(id int, set hll(-1,-1,8,-1));  
  vastbase=# d t3  
   Table "public.t3"  
   Column | Type | Modifiers  
  --------+----------------+-----------  
   id | integer |  
   set | hll(14,10,8,0) |  
    
  --创建hll类型的表，指定入参不合法报错  
  vastbase=# create table t4(id int, set hll(5,-1));  
  ERROR: log2m = 5 is out of range, it should be in range 10 to 16, or set -1 as default

对含有HLL类型的表插入HLL对象时，HLL类型的设定参数须同插入对象的设定参数一致，否则报错。

* -- 创建带有hll类型的表  
  vastbase=# create table t1(id integer, set hll(14));  
     
  -- 向表中插入hll对象,参数一致，成功  
  vastbase=# insert into t1 values (1, hll\_empty(14,-1));  
    
  -- 向表中插入hll对象，参数不一致，失败  
  vastbase=# insert into t1(id, set) values (1, hll\_empty(14,5));  
  ERROR: log2explicit does not match: source is 5 and dest is 10

HLL的应用场景。

* 场景1：“Hello World”
* 通过下面的示例说明如何使用hll数据类型：
* -- 创建带有hll类型的表  
  vastbase=# create table helloworld (id integer, set hll);  
     
  -- 向表中插入空的hll  
  vastbase=# insert into helloworld(id, set) values (1, hll\_empty());  
     
  -- 把整数经过哈希计算加入到hll中  
  vastbase=# update helloworld set set = hll\_add(set, hll\_hash\_integer(12345)) where id = 1;  
     
  -- 把字符串经过哈希计算加入到hll中  
  vastbase=# update helloworld set set = hll\_add(set, hll\_hash\_text('hello world')) where id = 1;  
     
  -- 得到hll中的distinct值  
  vastbase=# select hll\_cardinality(set) from helloworld where id = 1;  
   hll\_cardinality   
   -----------------  
   2  
   (1 row)  
     
  -- 删除表  
  vastbase=# drop table helloworld;
* 场景2：“网站访客数量统计”
* 通过下面的示例说明hll如何统计在一段时间内访问网站的不同用户数量：
* -- 创建原始数据表，表示某个用户在某个时间访问过网站。  
  vastbase=# create table facts (  
   date date,  
   user\_id integer  
   );  
     
   -- 构造数据，表示一天中有哪些用户访问过网站。  
  vastbase=# insert into facts values ('2019-02-20', generate\_series(1,100));  
  vastbase=# insert into facts values ('2019-02-21', generate\_series(1,200));  
  vastbase=# insert into facts values ('2019-02-22', generate\_series(1,300));  
  vastbase=# insert into facts values ('2019-02-23', generate\_series(1,400));  
  vastbase=# insert into facts values ('2019-02-24', generate\_series(1,500));  
  vastbase=# insert into facts values ('2019-02-25', generate\_series(1,600));  
  vastbase=# insert into facts values ('2019-02-26', generate\_series(1,700));  
  vastbase=# insert into facts values ('2019-02-27', generate\_series(1,800));  
     
  -- 创建表并指定列为hll。  
  vastbase=# create table daily\_uniques (  
   date date UNIQUE,  
   users hll  
   );  
     
  -- 根据日期把数据分组，并把数据插入到hll中。  
  vastbase=# insert into daily\_uniques(date, users)  
   select date, hll\_add\_agg(hll\_hash\_integer(user\_id))  
   from facts  
   group by 1;  
     
  -- 计算每一天访问网站不同用户数量  
  vastbase=# select date, hll\_cardinality(users) from daily\_uniques order by date;  
   date | hll\_cardinality  
   ------------+------------------  
   2019-02-20 | 100  
   2019-02-21 | 200.217913059312  
   2019-02-22 | 301.76494508014  
   2019-02-23 | 400.862858326446  
   2019-02-24 | 502.626933349694  
   2019-02-25 | 601.922606454213  
   2019-02-26 | 696.602316769498  
   2019-02-27 | 798.111731634412  
   (8 rows)  
     
  -- 计算在2019.02.20到2019.02.26一周中有多少不同用户访问过网站  
  vastbase=# select hll\_cardinality(hll\_union\_agg(users)) from daily\_uniques where date >= '2019-02-20'::date and date <= '2019-02-26'::date;  
   hll\_cardinality   
   ------------------  
   702.941844662509  
   (1 row)  
     
  -- 计算昨天访问过网站而今天没访问网站的用户数量。  
  vastbase=# SELECT date, (#hll\_union\_agg(users) OVER two\_days) - #users AS lost\_uniques FROM daily\_uniques WINDOW two\_days AS (ORDER BY date ASC ROWS 1 PRECEDING);   
   date | lost\_uniques  
   ------------+--------------  
   2019-02-20 | 0  
   2019-02-21 | 0  
   2019-02-22 | 0  
   2019-02-23 | 0  
   2019-02-24 | 0  
   2019-02-25 | 0  
   2019-02-26 | 0  
   2019-02-27 | 0  
   (8 rows)  
     
  -- 删除表  
  vastbase=# drop table facts;  
  vastbase=# drop table daily\_uniques;

场景3：“插入数据不满足hll数据结构要求”

当用户给hll类型的字段插入数据的时候，必须保证插入的数据满足hll数据结构要求，如果解析后不满足就会报错。如下示例中： 插入数据'E1234'时，该数据不满足hll数据结构，不能解析成功因此失败报错。

* vastbase=# create table test(id integer, set hll);  
  vastbase=# insert into test values(1, 'E1234');  
  ERROR: not a hll type, size=6 is not enough  
  vastbase=# drop table test;

#### 范围类型

范围类型是表达某种元素类型（称为范围的*subtype*）的一个值的范围的数据类型。例如，timestamp的范围可以被用来表达一个会议室被保留的时间范围。在这种情况下，数据类型是tsrange（“timestamp range”的简写）而timestamp是 subtype。subtype 必须具有一种总体的顺序，这样对于元素值是在一个范围值之内、之前或之后就是界线清楚的。

范围类型非常有用，因为它们可以表达一种单一范围值中的多个元素值，并且可以很清晰地表达诸如范围重叠等概念。用于时间安排的时间和日期范围是最清晰的例子；但是价格范围、一种仪器的量程等等也都有用。

**内建范围类型**

有下列内建范围类型：

* int4range — integer的范围
* int8range — bigint的范围
* numrange — numeric的范围
* tsrange — 不带时区的 timestamp的范围
* tstzrange — 带时区的 timestamp的范围
* daterange — date的范围

此外，你可以定义自己的范围类型，参考：SQL语法参考->SQL语法->CREATE TYPE章节。

**例子**

* CREATE TABLE reservation (room int, during tsrange);   
  INSERT INTO reservation VALUES (1108, '[2010-01-01 14:30, 2010-01-01 15:30)');   
  -- 包含   
  SELECT int4range(10, 20) @> 3;   
  -- 重叠   
  SELECT numrange(11.1, 22.2) && numrange(20.0, 30.0);   
  -- 抽取上界   
  SELECT upper(int8range(15, 25));   
  -- 计算交集   
  SELECT int4range(10, 20) \* int4range(15, 25);   
  -- 范围为空吗？   
  SELECT isempty(numrange(1, 5));

范围类型上的操作符和函数的完整列表可参考：SQL语法参考->SQL基本要素->函数和操作符->范围函数和操作符章节。

**包含和排除边界**

每一个非空范围都有两个界限，下界和上界。上下界之间的所有值都被包括在范围内。一个包含界限意味着边界点本身也被包括在范围内，而一个排除边界意味着边界点不被包括在范围内。

在一个范围的文本形式中，一个包含下界被表达为“[”而一个排除下界被表达为“(”。同样，一个包含上界被表达为“]”而一个排除上界被表达为“)”（详见"范围输入/输出"）。

函数lower\_inc和upper\_inc分别测试一个范围值的上下界。

**无限（无界）范围**

一个范围的下界可以被忽略，意味着所有小于上界的值都被包括在范围中，例如(,3]。 同样，如果范围的上界被忽略，那么所有比上界大的值都被包括在范围中。如果上下界都被忽略，该元素类型的所有值都被认为在该范围中。 规定缺失的包括界限自动转换为排除，例如，[,] 转换为 (,)。 你可以认为这些缺失值为 +/- 无穷大，但它们是特殊范围类型值，并且被视为超出任何范围元素类型的 +/- 无穷大值。

具有“infinity”概念的元素类型可以用它们作为显式边界值。例如，在时间戳范围，[today,infinity)不包括特殊的timestamp值infinity，尽管 [today,infinity] 包括它，就好比 [today,) 和 [today,]。

函数lower\_inf和upper\_inf分别测试一个范围的无限上下界。

**范围输入/输出**

一个范围值的输入必须遵循下列模式之一：

* (lower-bound,upper-bound)   
  (lower-bound,upper-bound]   
  [lower-bound,upper-bound)   
  [lower-bound,upper-bound]   
  empty

圆括号或方括号指示上下界是否为排除的或者包含的。注意最后一个模式是empty，它表示一个空范围（一个不包含点的范围）。

lower-bound可以是作为subtype的合法输入的一个字符串，或者是空表示没有下界。同样，upper-bound可以是作为 subtype 的合法输入的一个字符串，或者是空表示没有上界。

每个界限值可以使用"（双引号）字符引用。如果界限值包含圆括号、方括号、逗号、双引号或反斜线时，这样做是必须的，因为否则那些字符会被认作范围语法的一部分。要把一个双引号或反斜线放在一个被引用的界限值中，就在它前面放一个反斜线（还有，在一个双引号引用的界限值中的一对双引号表示一个双引号字符，这与 SQL 字符串中的单引号规则类似）。此外，你可以避免引用并且使用反斜线转义来保护所有数据字符，否则它们会被当做返回语法的一部分。还有，要写一个是空字符串的界限值，则可以写成""，因为什么都不写表示一个无限界限。

范围值前后允许有空格，但是圆括号或方括号之间的任何空格会被当做上下界值的一部分（取决于元素类型，它可能是也可能不是有意义的）。

例子：

* -- 包括 3，不包括 7，并且包括 3 和 7 之间的所有点   
  SELECT '[3,7)'::int4range;   
  -- 既不包括 3 也不包括 7，但是包括之间的所有点   
  SELECT '(3,7)'::int4range;   
  -- 只包括单独一个点 4   
  SELECT '[4,4]'::int4range;   
  -- 不包括点（并且将被标准化为 '空'）   
  SELECT '[4,4)'::int4range;

**构造范围**

每一种范围类型都有一个与其同名的构造器函数。使用构造器函数常常比写一个范围文字常数更方便，因为它避免了对界限值的额外引用。构造器函数接受两个或三个参数。两个参数的形式以标准的形式构造一个范围（下界是包含的，上界是排除的），而三个参数的形式按照第三个参数指定的界限形式构造一个范围。第三个参数必须是下列字符串之一： “()”、 “(]”、 “[)”或者 “[]”。 例如：

* -- 完整形式是：下界、上界以及指示界限包含性/排除性的文本参数。   
  SELECT numrange(1.0, 14.0, '(]');   
  -- 如果第三个参数被忽略，则假定为 '[)'。   
  SELECT numrange(1.0, 14.0);   
  -- 尽管这里指定了 '(]'，显示时该值将被转换成标准形式，因为 int8range 是一种离散范围类型（见下文）。   
  SELECT int8range(1, 14, '(]');   
  -- 为一个界限使用 NULL 导致范围在那一边是无界的。   
  SELECT numrange(NULL, 2.2);

**离散范围类型**

一种范围的元素类型具有一个良定义的“步长”，例如integer或date。在这些类型中，如果两个元素之间没有合法值，它们可以被说成是相邻。这与连续范围相反，连续范围中总是（或者几乎总是）可以在两个给定值之间标识其他元素值。例如，numeric类型之上的一个范围就是连续的，timestamp上的范围也是（尽管timestamp具有有限的精度，并且在理论上可以被当做离散的，最好认为它是连续的，因为通常并不关心它的步长）。

另一种考虑离散范围类型的方法是对每一个元素值都有一种清晰的“下一个”或“上一个”值。了解了这种思想之后，通过选择原来给定的下一个或上一个元素值来取代它，就可以在一个范围界限的包含和排除表达之间转换。例如，在一个整数范围类型中，[4,8]和(3,9)表示相同的值集合，但是对于 numeric 上的范围就不是这样。

一个离散范围类型应该具有一个*正规化*函数，它知道元素类型期望的步长。正规化函数负责把范围类型的相等值转换成具有相同的表达，特别是与包含或者排除界限一致。如果没有指定一个正规化函数，那么具有不同格式的范围将总是会被当作不等，即使它们实际上是表达相同的一组值。

内建的范围类型int4range、int8range和daterange都使用一种正规的形式，该形式包括下界并且排除上界，也就是[)。不过，用户定义的范围类型可以使用其他习惯。

**定义新的范围类型**

用户可以定义他们自己的范围类型。这样做最常见的原因是为了使用内建范围类型中没有提供的 subtype 上的范围。例如，要创建一个 subtype float8的范围类型：

* CREATE TYPE floatrange AS RANGE (  
   subtype = float8,  
   subtype\_diff = float8mi   
  );  
  SELECT '[1.234, 5.678]'::floatrange;

因为float8没有有意义的“步长”，我们在这个例子中没有定义一个正规化函数。

定义自己的范围类型也允许你指定使用一个不同的子类型 B-树操作符类或者集合， 以便更改排序顺序来决定哪些值会落入到给定的范围中。

如果 subtype 被认为是具有离散值而不是连续值，CREATE TYPE命令应当指定一个canonical函数。正规化函数接收一个输入的范围值，并且必须返回一个可能具有不同界限和格式的等价的范围值。对于两个表示相同值集合的范围（例如[1, 7]和[1, 8)），正规的输出必须一样。选择哪一种表达作为正规的没有关系，只要两个具有不同格式的等价值总是能被映射到具有相同格式的相同值就行。除了调整包含/排除界限格式外，假使期望的补偿比 subtype 能够存储的要大，一个正规化函数可能会舍入边界值。例如，一个timestamp之上的范围类型可能被定义为具有一个一小时的步长，这样正规化函数可能需要对不是一小时的倍数的界限进行舍入，或者可能直接抛出一个错误。

另外，任何打算要和 GiST 或 SP-GiST 索引一起使用的范围类型应当定一个 subtype 差异或subtype\_diff函数（没有subtype\_diff时索引仍然能工作，但是可能效率不如提供了差异函数时高）。subtype 差异函数采用两个 subtype 输入值，并且返回表示为一个float8值的差（即*X*减*Y*）。在我们上面的例子中，可以使用常规float8减法操作符之下的函数。但是对于任何其他 subtype，可能需要某种类型转换。还可能需要一些关于如何把差异表达为数字的创新型想法。为了最大的可扩展性，subtype\_diff函数应该同意选中的操作符类和排序规则所蕴含的排序顺序，也就是说，只要它的第一个参数根据排序顺序大于第二个参数，它的结果就应该是正值。

subtype\_diff函数的一个不那么过度简化的例子：

* CREATE FUNCTION time\_subtype\_diff(x time, y time) RETURNS float8 AS 'SELECT EXTRACT(EPOCH FROM (x - y))' LANGUAGE sql STRICT IMMUTABLE;   
  CREATE TYPE timerange AS RANGE (   
   subtype = time,   
   subtype\_diff = time\_subtype\_diff  
   );   
  SELECT '[11:10, 23:00]'::timerange;

更多关于创建范围类型的信息请参考：SQL语法参考->SQL语法->CREATE TYPE章节。

**索引**

可以为范围类型的表列创建 GiST 和 SP-GiST 索引。例如，要创建一个 GiST 索引：

* CREATE INDEX reservation\_idx ON reservation USING GIST (during);

一个 GiST 或 SP-GiST 索引可以加速涉及以下范围操作符的查询： =、 &&、 <@、 @>、 <<、 >>、 -|-、 &<以及 &> （参考：SQL语法参考->SQL基本要素->函数和操作符->范围函数和操作符章节）。

此外，B-树和哈希索引可以在范围类型的表列上创建。对于这些索引类型，基本上唯一有用的范围操作就是等值。使用相应的< 和 >操作符，对于范围值定义有一种 B-树排序顺序，但是该顺序相当任意并且在真实世界中通常不怎么有用。范围类型的 B-树和哈希支持主要是为了允许在查询内部进行排序和哈希，而不是创建真正的索引。

#### 对象标识符类型

Vastbase在内部使用对象标识符（OID）作为各种系统表的主键。系统不会给用户创建的表增加一个OID系统字段，OID类型代表一个对象标识符。

目前OID类型用一个四字节的无符号整数实现。因此不建议在创建的表中使用OID字段做主键。

**表 1** 对象标识符类型

| **名称** | **引用** | **描述** | **示例** |
| --- | --- | --- | --- |
| OID | - | 数字化的对象标识符。 | 564182 |
| CID | - | 命令标识符。它是系统字段cmin和cmax的数据类型。命令标识符是32位的量。 | - |
| XID | - | 事务标识符。它是系统字段xmin和xmax的数据类型。事务标识符也是64位的量。 | - |
| TID | - | 行标识符。它是系统表字段ctid的数据类型。行ID是一对数值（块号，块内的行索引），它标识该行在其所在表内的物理位置。 | - |
| REGCONFIG | pg\_ts\_config | 文本搜索配置。 | english |
| REGDICTIONARY | pg\_ts\_dict | 文本搜索字典。 | simple |
| REGOPER | pg\_operator | 操作符名。 | - |
| REGOPERATOR | pg\_operator | 带参数类型的操作符。 | \*(integer,integer)或-(NONE,integer) |
| REGPROC | pg\_proc | 函数名称。 | sum |
| REGPROCEDURE | pg\_proc | 带参数类型的函数。 | sum(int4) |
| REGCLASS | pg\_class | 关系名。 | pg\_type |
| REGTYPE | pg\_type | 数据类型名。 | integer |

OID类型：主要作为数据库系统表中字段使用。

**示例：**

* vastbase=# SELECT oid FROM pg\_class WHERE relname = 'pg\_type';  
   oid   
  ------  
   1247  
  (1 row)

OID别名类型REGCLASS：主要用于对象OID值的简化查找。

**示例：**

* vastbase=# SELECT attrelid,attname,atttypid,attstattarget FROM pg\_attribute WHERE attrelid = 'pg\_type'::REGCLASS;  
   attrelid | attname | atttypid | attstattarget   
  ----------+------------+----------+---------------  
   1247 | xc\_node\_id | 23 | 0  
   1247 | tableoid | 26 | 0  
   1247 | cmax | 29 | 0  
   1247 | xmax | 28 | 0  
   1247 | cmin | 29 | 0  
   1247 | xmin | 28 | 0  
   1247 | oid | 26 | 0  
   1247 | ctid | 27 | 0  
   1247 | typname | 19 | -1  
   1247 | typnamespace | 26 | -1  
   1247 | typowner | 26 | -1  
   1247 | typlen | 21 | -1  
   1247 | typbyval | 16 | -1  
   1247 | typtype | 18 | -1  
   1247 | typcategory | 18 | -1  
   1247 | typispreferred | 16 | -1  
   1247 | typisdefined | 16 | -1  
   1247 | typdelim | 18 | -1  
   1247 | typrelid | 26 | -1  
   1247 | typelem | 26 | -1  
   1247 | typarray | 26 | -1  
   1247 | typinput | 24 | -1  
   1247 | typoutput | 24 | -1  
   1247 | typreceive | 24 | -1  
   1247 | typsend | 24 | -1  
   1247 | typmodin | 24 | -1  
   1247 | typmodout | 24 | -1  
   1247 | typanalyze | 24 | -1  
   1247 | typalign | 18 | -1  
   1247 | typstorage | 18 | -1  
   1247 | typnotnull | 16 | -1  
   1247 | typbasetype | 26 | -1  
   1247 | typtypmod | 23 | -1  
   1247 | typndims | 23 | -1  
   1247 | typcollation | 26 | -1  
   1247 | typdefaultbin | 194 | -1  
   1247 | typdefault | 25 | -1  
   1247 | typacl | 1034 | -1  
  (38 rows)

#### 伪类型

Vastbase数据类型中包含一系列特殊用途的类型，这些类型按照类别被称为伪类型。伪类型不能作为字段的数据类型，但是可以用于声明函数的参数或者结果类型。

当一个函数不仅是简单地接受并返回某种SQL数据类型的情况下伪类型是很有用的。表1列出了所有的伪类型。

**表 1** 伪类型

| **名称** | **描述** |
| --- | --- |
| any | 表示函数接受任何输入数据类型。 |
| anyelement | 表示函数接受任何数据类型。 |
| anyarray | 表示函数接受任意数组数据类型。 |
| anynonarray | 表示函数接受任意非数组数据类型。 |
| anyenum | 表示函数接受任意枚举数据类型。 |
| anyrange | 表示函数接受任意范围数据类型。 |
| cstring | 表示函数接受或者返回一个空结尾的C字符串。 |
| internal | 表示函数接受或者返回一种服务器内部的数据类型。 |
| language\_handler | 声明一个过程语言调用句柄返回language\_handler。 |
| fdw\_handler | 声明一个外部数据封装器返回fdw\_handler。 |
| record | 标识函数返回一个未声明的行类型。 |
| trigger | 声明一个触发器函数返回trigger。 |
| void | 表示函数不返回数值。 |
| opaque | 一个已经过时的类型，以前用于所有上面这些用途。 |

声明用C编写的函数（不管是内置的还是动态装载的）都可以接受或者返回任何这样的伪数据类型。当伪类型作为参数类型使用时，用户需要保证函数的正常运行。

用过程语言编写的函数只能使用实现语言允许的伪类型。目前，过程语言都不允许使用作为参数类型的伪类型，并且只允许使用void和record作为结果类型。一些多态的函数还支持使用anyelement、anyarray、anynonarray anyenum和anyrange类型。

伪类型internal用于声明那种只能在数据库系统内部调用的函数，他们不能直接在SQL查询里调用。如果函数至少有一个internal类型的参数，则不能从SQL里调用他。建议不要创建任何声明返回internal的函数，除非他至少有一个internal类型的参数。

示例：

* --创建表  
  vastbase=# create table t1 (a int);  
    
  --插入两条数据  
  vastbase=# insert into t1 values(1),(2);  
    
  --创建函数showall()。  
  vastbase=# CREATE OR REPLACE FUNCTION showall() RETURNS SETOF record  
  AS $$ SELECT count(\*) from t1; $$  
  LANGUAGE SQL;  
    
  --调用函数showall()。  
  vastbase=# SELECT showall();  
   showall   
  ---------  
   (2)  
  (1 row)  
    
  --删除函数。  
  vastbase=# DROP FUNCTION showall();  
    
  --删除表  
  vastbase=# drop table t1;

#### 列存表支持的数据类型

列存表支持的数据类型如表1所示。

**表 1** 列存表支持的数据类型

| **类别** | **数据类型** | **长度** | **是否支持** |
| --- | --- | --- | --- |
| Numeric Types | smallint | 2 | 支持 |
| integer | 4 | 支持 |  |
| bigint | 8 | 支持 |  |
| decimal | -1 | 支持 |  |
| numeric | -1 | 支持 |  |
| real | 4 | 支持 |  |
| double precision | 8 | 支持 |  |
| smallserial | 2 | 支持 |  |
| serial | 4 | 支持 |  |
| bigserial | 8 | 支持 |  |
| largeerial | 8 | 支持 |  |
| Monetary Types | money | 8 | 支持 |
| Character Types | character varying(n), varchar(n) | -1 | 支持 |
| character(n), char(n) | n | 支持 |  |
| character、char | 1 | 支持 |  |
| text | -1 | 支持 |  |
| nvarchar | -1 | 支持 |  |
| nvarchar2 | -1 | 支持 |  |
| name | 64 | 不支持 |  |
| Date/Time Types | timestamp with time zone | 8 | 支持 |
| timestamp without time zone | 8 | 支持 |  |
| date | 4 | 支持 |  |
| time without time zone | 8 | 支持 |  |
| time with time zone | 12 | 支持 |  |
| interval | 16 | 支持 |  |
| big object | clob | -1 | 支持 |
| blob | -1 | 不支持 |  |
| other types | … | … | 不支持 |

#### XML类型

Vastbase支持XML类型，使用示例如下。

* Vastbase= CREATE TABLE xmltest ( id int, data xml );   
  Vastbase= INSERT INTO xmltest VALUES (1, 'one');  
  Vastbase= INSERT INTO xmltest VALUES (2, 'two');   
  Vastbase= SELECT \* FROM xmltest ORDER BY 1;  
   id | data   
  ----+--------------------  
  1 | one   
  2 | two   
  (2 rows)  
  Vastbase= SELECT xmlconcat('', NULL, '');   
  xmlconcat  
  (1 row)  
  Vastbase= SELECT xmlconcat('', NULL, '');   
  xmlconcat  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

* 该功能默认未开启，如需使用，需要重新使用build.sh脚本编译数据库，修改./configure配置参数，在其中加入--with-libxml参数。
* 在执行编译之前，需要先执行yum install -y libxml2-devel，否则会有“configure: error: library 'xml2' (version >= 2.6.23) is required for XML support”的报错。
* 在执行编译之前，需要三方库二进制文件中dependeny操作系统环境/libobs/comm/lib加入到系统环境变量LD\_LIBRARY\_PATH中，否则会报错“libiconv.so不存在”。

#### 账本数据库使用的数据类型

账本数据库使用HASH16数据类型来存储行级hash摘要或表级hash摘要，使用HASH32数据类型来存储全局hash摘要或者历史表校验hash。

**表 1** 账本数据库HASH类型

| **名称** | **描述** | **存储空间** | **范围** |
| --- | --- | --- | --- |
| HASH16 | 以无符号64位整数存储。 | 8字节 | 0 ~ +18446744073709551615 |
| HASH32 | 以包含16个的无符号整形元素数的组存储。 | 16字节 | 16个元素的无符号整形数组能够包含的取值范围 |

HASH16数据类型用来在账本数据库中存储行级或表级hash摘要，在获得长度为16个字符串的十六进制字符串的hash序列后，系统将调用hash16in函数将该序列转换为一个无符号64位整数存储进HASH16类型变量中。示例如下：

* 十六进制字符串：e697da2eaa3a775b   
  对应的无符号64位整数：16615989244166043483  
  十六进制字符串：ffffffffffffffff   
  对应的无符号64位整数：18446744073709551615

HASH32数据类型用来在账本数据库中存储全局hash摘要或者历史表校验hash，在获得长度为32个字符串的十六进制字符串的hash序列后，系统将调用hash32in函数将该序列转换到一个包含16个无符号整形元素的数组中。示例如下：

* 十六进制字符串：685847ed1fe38e18f6b0e2b18c00edee   
  对应的HASH32数组：[104,88,71,237,31,227,142,24,246,176,226,177,140,0,237,238]

#### 事务ID和快照数据类型txid\_snapshot

**功能描述**

Vastbase通过txid\_snapshot类型记录事务ID和快照信息。

**语法格式**

txid\_snapshot的文本表示为：

xmin:xmax:xip\_list

**参数解释**

|  |  |
| --- | --- |
| **名称** | **描述** |
| xmin | 仍然活动的最早的事务ID（txid）。所有较早事务将是已经提交可见的，或者是直接回滚。 |
| xmax | 作为尚未分配的txid。所有大于或等于此txids的都是尚未开始的快照时间，因此不可见。 |
| xip\_list | 当前快照中活动的txids。这个列表只包含在xmin和xmax之间活动的txids；有可能活动的txids高于xmax。介于大于等于xmin、小于xmax，并且不在这个列表中的txid，在这个时间快照已经完成的，因此按照提交状态查看他是可见还是回滚。这个列表不包含子事务的txids。 |

**示例**

获取当前快照

select txid\_current\_snapshot();

结果返回如下：

txid\_current\_snapshot  
-----------------------  
 25091:25101:  
(1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

txid\_current\_snapshot()函数返回类型为txid\_snapshot。

#### JSON/JSONB类型

JSON（JavaScript Object Notation）数据，可以是单独的一个标量，也可以是一个数组，也可以是一个键值对象，其中数组和对象可以统称容器（container）：

* 标量（scalar）：单一的数字、bool、string、null都可以叫做标量。
* 数组（array）：[]结构，里面存放的元素可以是任意类型的JSON，并且不要求数组内所有元素都是同一类型。
* 对象（object）：{}结构，存储key:value的键值对，其键只能是用“”包裹起来的字符串，值可以是任意类型的JSON，对于重复的键，按最后一个键值对为准。

Vastbase内存在两种数据类型JSON和JSONB，可以用来存储JSON数据。其中JSON是对输入的字符串的完整拷贝，使用时再去解析，所以它会保留输入的空格、重复键以及顺序等；JSONB解析输入后保存的二进制，它在解析时会删除语义无关的细节和重复的键，对键值也会进行排序，使用时不用再次解析。

因此可以发现，两者其实都是JSON，它们接受相同的字符串作为输入。它们实际的主要差别是效率。JSON数据类型存储输入文本的精确拷贝，处理函数必须在每个执行上重新解析； 而JSONB数据以分解的二进制格式存储，这使得它由于添加了转换机制而在输入上稍微慢些，但是在处理上明显更快，因为不需要重新解析。同时由于JSONB类型存在解析后的格式归一化等操作，同等的语义下只会有一种格式，因此可以更好更强大的支持很多其他额外的操作，比如按照一定的规则进行大小比较等。JSONB也支持索引，这也是一个明显的优势。

**输入格式示例**

输入必须是一个符合JSON数据格式的字符串，此字符串用单引号 ' ' 声明。

* null（null-json）：仅null，全小写。
* select 'null'::json;

结果显示如下：

* json   
  --------------  
   null  
  (1 row)
* 数字（num-json）：正负整数、小数、0，支持科学计数法，不支持多余的前导0，正数的+号，以及NaN和infinity。
* select '1'::json,'-1.5'::json,'-1.5e-5'::jsonb, '-1.5e+2'::jsonb;

结果显示如下：

* json | json | jsonb | jsonb   
  ------+------+----------+-------  
   1 | -1.5 | -.000015 | -150  
  (1 row)
* 布尔（bool-json）：仅true、false，全小写。
* select 'true'::json;  
  select 'false'::jsonb;

结果显示如下：

* json   
   -----------  
   true  
  (1 row)  
   jsonb   
   -----------  
   false  
  (1 row)
* 字符串（str-json）：必须是加双引号的字符串。
* select '"a"'::json;  
  select '"abc"'::jsonb;

结果显示如下：

* json   
  ---------  
   "a"  
  (1 row)  
   jsonb   
  ----------  
   "abc"  
  (1 row)
* 数组（array-json）：使用中括号[]包裹，满足数组书写条件。数组内元素类型可以是任意合法的JSON，且不要求类型一致。
* select '[1, 2, "foo", null]'::json;select '[]'::json;  
  select '[1, 2, "foo", null, [[]], {}]'::jsonb;

结果显示如下：

* json   
  ---------------------  
   [1, 2, "foo", null]  
  (1 row)  
   json  
  -----------  
   []  
  (1 row)  
   jsonb  
  --------------------------------  
   [1, 2, "foo", null, [[]], {}]  
  (1 row)
* 对象（object-json）：使用大括号{}包裹，键必须是满足JSON字符串规则的字符串，值可以是任意合法的JSON。
* select '{}'::json;select '{"a": 1, "b": {"a": 2, "b": null}}'::json;  
  select '{"foo": [true, "bar"], "tags": {"a": 1, "b": null}}'::jsonb;

结果显示如下：

* json   
  ------  
   {}  
  (1 row)  
   json   
  -------------------------------------  
   {"a": 1, "b": {"a": 2, "b": null}}  
  (1 row)  
   jsonb   
  -----------------------------------------------------  
   {"foo": [true, "bar"], "tags": {"a": 1, "b": null}}  
  (1 row)
* 区分 'null'::json 和 null::json 是两个不同的概念，类似于字符串 str=“” 和 str=null。
* 对于数字，当使用科学计数法的时候，jsonb类型会将其展开，而json会精准拷贝输入。

**JSONB高级特性**

JSON和JSONB的主要差异在于存储方式上的不同，JSONB存储的是解析后的二进制，能够体现JSON的层次结构，更方便直接访问等，因此JSONB会有很多JSON所不具有的高级特性。

* 注意事项
  + 不支持列存。
  + 不支持作为分区键。
  + 不支持外表、mot。
* 格式归一化，对于输入的object-json字符串，解析成jsonb二进制后，会天然的丢弃语义上无关紧要的细节，比如空格：
* select ' [1, " a ", {"a" :1 }] '::jsonb;

结果显示如下：

* jsonb   
  ----------------------  
   [1, " a ", {"a": 1}]  
  (1 row)
* 对于object-json，键值会重新进行排序，排序规则：长度长的在后、长度相等则ascii码大的在后，如：
* select '{"aa" : 1, "b" : 2, "a" : 3}'::jsonb;

结果显示如下：

* jsonb   
  ---------------------------  
   {"a": 3, "b": 2, "aa": 1}  
  (1 row)
* 对于object-json，会删除重复的键值，只保留最后一个出现的，如：
* select '{"a" : 1, "a" : 2}'::jsonb;

结果显示如下：

* jsonb   
  ----------  
   {"a": 2}  
  (1 row)
* 大小比较：由于经过了格式归一化，保证了同一种语义下的jsonb只会有一种存在形式，因此按照制定的规则，可以比较大小。
  + 首先比较类型：object-jsonb > array-jsonb > bool-jsonb > num-jsonb > str-jsonb > null-jsonb
  + 同类型则比较内容：
    - str-json类型：依据text比较的方法，使用数据库默认排序规则进行比较，返回值正数代表大于，负数代表小于，0表示相等。
    - num-json类型：数值比较。
    - bool-json类型：true > false。
    - array-jsonb类型：长度长的 > 长度短的，长度相等则依次比较每个元素。
    - object-jsonb类型：长度长的 > 长度短的，长度相等则依次比较每个键值对，先比较键，在比较值。
  + object-jsonb类型内比较时，比较时使用的是格式整理后的最终结果进行比较，因此相对于我们直接的输入未必会很直观。
* 创建索引、主外键
* BTREE索引
* jsonb类型支持创建btree索引，支持创建主键、外键。
* GIN索引
  + - GIN索引可以用来有效的搜索出现在大量jsonb文档（datums）中的键或者键/值对。提供了两个GIN操作符类（jsonb\_ops、jsonb\_hash\_ops），提供了不同的性能和灵活性取舍。缺省的GIN操作符类支持使用@>、<@、?、 ?&和?|操作符查询，非缺省的GIN操作符类jsonb\_path\_ops只支持索引@>、<@操作符。
    - 相关的操作符请参见JSON/JSONB函数和操作符
* 包含存在

查询一个JSON之中是否包含某些元素，或者某些元素是否存在于某个JSON中是jsonb的一个重要能力。简单的标量/原始值只包含相同的值。

* SELECT '"foo"'::jsonb @> '"foo"'::jsonb;

结果显示如下：

* ?column?   
  ----------  
   t  
  (1 row)
* 左侧数组包含了右侧字符串。
* SELECT '[1, "aa", 3]'::jsonb ? 'aa';

结果显示如下：

* ?column?   
  ----------  
   t  
  (1 row)
* 左侧数组包含了右侧的数组所有元素，顺序、重复不重要。
* SELECT '[1, 2, 3]'::jsonb @> '[1, 3, 1]'::jsonb;

结果显示如下：

* ?column?   
  ----------  
   t  
  (1 row)
* 左侧object-json包含了右侧object-json的所有键值对。
* SELECT '{"product": "PostgreSQL", "version": 9.4, "jsonb":true}'::jsonb @> '{"version":9.4}'::jsonb;

结果显示如下：

* ?column?   
  ----------  
   t  
  (1 row)
* 左侧数组并没有包含右侧的数组所有元素，因为左侧数组的三个元素为1、2、[1,3]，右侧的为1、3。
* SELECT '[1, 2, [1, 3]]'::jsonb @> '[1, 3]'::jsonb;

结果显示如下：

* ?column?   
  ----------  
   f  
  (1 row)

相关的操作符请参见JSON/JSONB函数和操作符。

* 函数和操作符
* json/jsonb类型相关支持的函数和操作符请参见JSON/JSONB函数和操作符。

#### ROWID数据类型

**功能描述**

Vastbase数据库的表中的每一行数据都有一个唯一的标识符，或者称为rowid，rowid是一个伪列在Vastbase内部通常就是使用它来访问数据。该值表明了该行在数据库中的物理具体位置。可以在一个查询中使用rowid来表明查询结果中包含该值。

ROWID可以分为物理rowid和逻辑rowid两种。普通的表中的rowid是物理rowid，索引组织表(IOT)的rowid是逻辑rowid。

**示例**

1、创建测试表ridtest。

CREATE TABLE ridtest (id int,name varchar(10));

2、插入测试数据。

INSERT INTO ridtest values(1,'zhangsan');

3、查询rowid。

SELECT rowid,id FROM ridtest;

返回结果如下：

rowid | id  
----------------------+----  
 BWEAAA==AAAAAA==AQA= | 1  
(1 row)

#### BFILE数据类型

**功能描述**

bfile用于存储服务器文件系统中的物理二进制文件数据对象。bfile文件存储在操作系统文件中，而不是在数据库中，bfile列存储对操作系统文件的引用。操作系统访问的任何存储设备都可以保存bfile数据，包括硬盘驱动器、CD-ROM、PhotoCD和DVD。如果操作系统支持对操作系统文件的流模式访问，则数据库可以访问bfile。

bfile类型的数据是：

* 在应用程序运行时不会更改的二进制数据，例如图形。
* 加载到其他大型对象类型（例如BLOB或CLOB）中的数据，可以在其中操作数据。
* 适用于字节流访问的数据，例如多媒体数据。

在对bfile类型的SQL操作中，需要支持bfilename函数。bfilename函数返回一个bfile文件定位器，该定位器指向操作系统中一个物理文件。

bfilename函数主要用于bfile类型数据的构造，构造的数据源为一个directory对象和一个操作系统物理二进制文件名，函数语法格式为：

BFILENAME('directory', 'filename')

* directory对象：指向一个操作系统的物理目录。该目录不应为数据库数据目录，以及操作系统控制文件、日志文件和其他系统文件所在的目录。必须在bfilename函数调用前创建，并且对名区分大小写。
* filename：文件名，操作系统上一个物理二进制文件，由于bfile为只读数据类型，必须在访问文件数据前创建该文件。

bfile数据类型主要用于对bfilename函数构造的数据进行存储，然后调用dbms\_lob内置包的接口对该数据进行操作。bfile数据类型的功能主要分为三个部分：

* bfile数据的存储和打印。
* bfile数据的使用，主要为对操作系统物理文件的读取过程，即分为以下三部分：
* 打开操作系统物理文件。bfile为只读数据类型，因此打开文件的模式只能为只读。
* 对文件进行读取。可根据参数设置来决定读取文件的字节数，如需读取整个文件，可先调用dbms\_lob包的getlength，函数获取文件的大小。需注意的是，单次读取文件的字节数不能超过32767字节，如文件过大，可通过设置参数对文件进行多次读取。
* 关闭文件。对打开的物理文件进行关闭。

bfile数据的修改：对于bfile数据的持久引用（如存储在表中），bfile数据可以作为表的列值进行修改。

**注意事项**

bfilename函数依赖vastbase自带的directory功能。

**使用流程**

1、创建directory对象，创建需访问的文件并将其置于directory对象所指向的操作系统目录中。

2、创建包含bfile数据类型列的表。

3、调用bfilename函数对数据进行构造并插入到表中进行存储。

**示例**

1、操作系统环境下创建文件，并输入数据。

vi /home/vastbase/bfile.data

cat /home/vastbase/bfile.data

文件内容为：

张三

李四

2、在数据库中执行如下命令创建目录和表。

drop directory d\_bfile;

drop table testbfile1;

create directory d\_bfile as '/home/vastbase';

create table testbfile1(id number,bfile\_name bfile);

3、插入数据。

insert into testbfile1 values (1,bfilename('d\_bfile','bfile.data'));

4、查询数据。

select \* from testbfile1;

返回结果为：

id | bfile\_name

----+-----------------------------------

1 | bfilename('d\_bfile','bfile.data')

#### ACLITEM权限类型

**功能描述**

Vastbase中对象的访问权限使用aclitem类型进行记录。

**语法格式**

rolename=privileges/rolegrant

**参数说明**

* rolename：被授权者。
* privileges：授予的权限，权限列表如下所示：
  + r：SELECT
  + w：UPDATE
  + a：INSERT
  + d：DELETE
  + D：TRUNCATE
  + x：REFERENCES
  + t：TRIGGER
  + X：EXECUTE
  + U：USAGE
  + C：CREATE
  + c：CONNECT
  + T：TEMPORARY
  + arwdDxt：ALL PRIVILEGES
  + \*：授予优先权选项
* rolegrant：授权者。

**示例**

系统表中记录了对象的访问权限，如表1所示。

**表1**系统表中的ACL列

|  |  |
| --- | --- |
| **系统表** | **字段名** |
| pg\_database | datacl |
| pg\_tablespace | spcacl |
| pg\_class | relacl |
| pg\_attribute | attacl |
| pg\_type | typacl |
| pg\_proc | proacl |
| pg\_language | lanacl |

查询pg\_database中记录的aclitem类型字段

select datname,datacl from pg\_database;

结果显示如下：

datname | datacl  
-----------+-------------------------------------  
 template1 | {=c/vastbase,vastbase=CTc/vastbase}  
 db\_oracle |  
 my\_test |  
 template0 | {=c/vastbase,vastbase=CTc/vastbase}  
 vastbase |  
 postgres |  
(6 rows)

### 常量与宏

Vastbase支持的常量和宏请参见表1。

**表 1** 常量和宏

| **参数** | **描述** | **示例** |
| --- | --- | --- |
| CURRENT\_CATALOG | 当前数据库 | vastbase=# SELECT CURRENT\_CATALOG;  current\_database  ------------------  Vastbase  (1 row) |
| CURRENT\_ROLE | 当前用户 | vastbase=# SELECT CURRENT\_ROLE;  current\_user  --------------  vastbase  (1 row) |
| CURRENT\_SCHEMA | 当前数据库模式 | vastbase=# SELECT CURRENT\_SCHEMA;  current\_schema  ----------------  public  (1 row) |
| CURRENT\_USER | 当前用户 | vastbase=# SELECT CURRENT\_USER;  current\_user  --------------  vastbase  (1 row) |
| LOCALTIMESTAMP | 当前会话时间（无时区） | vastbase=# SELECT LOCALTIMESTAMP;  timestamp  ----------------------------  2015-10-10 15:37:30.968538  (1 row) |
| NULL | 空值 | - |
| SESSION\_USER | 当前系统用户 | vastbase=# SELECT SESSION\_USER;  session\_user  --------------  vastbase  (1 row) |
| SYSDATE | 当前系统日期 | vastbase=# SELECT SYSDATE;  sysdate  ---------------------  2015-10-10 15:48:53  (1 row) |
| USER | 当前用户，此用户为CURRENT\_USER的别名 | vastbase=# SELECT USER;  current\_user  --------------  vastbase  (1 row) |

### 函数和操作符

#### 时间和日期处理函数和操作符

**时间日期操作符**

fig: **警告：**

用户在使用时间和日期操作符时，对应的操作数请使用明确的类型前缀修饰，以确保数据库在解析操作数的时候能够与用户预期一致，不会产生用户非预期的结果。比如下面示例没有明确数据类型就会出现异常错误，例如 ：

* SELECT date '2001-10-01' - '7' AS RESULT;

报错如下：

* ERROR: invalid input syntax for type oradate: "7"  
  LINE 1: SELECT date '2001-10-01' - '7' AS RESULT;  
   ^  
  CONTEXT: referenced column: result

下面介绍4中操作符，分别是“+”，“-”，“\*”，“/”

* **“'+”**

示例如下：

SELECT date '2001-9-28' + integer '7' AS RESULT;  
 result  
------------  
 2001-10-05  
(1 row)  
  
SELECT date '2001-09-28' + interval '1 hour' AS RESULT;  
 result   
---------------------  
 2001-09-28 01:00:00  
(1 row)  
  
 SELECT date '2001-09-28' + time '03:00' AS RESULT;  
 result   
---------------------  
 2001-09-28 03:00:00  
(1 row)  
  
 SELECT interval '1 day' + interval '1 hour' AS RESULT;  
 result   
----------------  
 1 day 01:00:00  
(1 row)  
  
SELECT timestamp '2001-09-28 01:00' + interval '23 hours' AS RESULT;  
 result   
---------------------  
 2001-09-29 00:00:00  
(1 row)  
  
SELECT time '01:00' + interval '3 hours' AS RESULT;  
 result   
----------  
 04:00:00  
(1 row)

* “-”

示例如下：

SELECT date '2001-10-01' - date '2001-09-28' AS RESULT;  
 result  
--------  
 3days  
(1 row)  
  
SELECT date '2001-10-01' - integer '7' AS RESULT;  
 result   
---------------------  
 2001-09-24 00:00:00  
(1 row)  
  
 SELECT date '2001-09-28' - interval '1 hour' AS RESULT;  
 result   
---------------------  
 2001-09-27 23:00:00  
(1 row)  
  
SELECT time '05:00' - time '03:00' AS RESULT;  
 result   
----------  
 02:00:00  
(1 row)  
  
 SELECT time '05:00' - interval '2 hours' AS RESULT;  
 result   
----------  
 03:00:00  
(1 row)  
  
SELECT timestamp '2001-09-28 23:00' - interval '23 hours' AS RESULT;  
 result   
---------------------  
 2001-09-28 00:00:00  
(1 row)  
  
SELECT interval '1 day' - interval '1 hour' AS RESULT;  
 result   
----------  
 23:00:00  
(1 row)  
  
SELECT timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00' AS RESULT;  
 result   
----------------  
 1 day 15:00:00  
(1 row)

* “-”

示例如下：

SELECT 900 \* interval '1 second' AS RESULT;  
 result   
----------  
 00:15:00  
(1 row)  
  
 SELECT 21 \* interval '1 day' AS RESULT;  
 result   
---------  
 21 days  
(1 row)  
  
 SELECT double precision '3.5' \* interval '1 hour' AS RESULT;  
 result   
----------  
 03:30:00  
(1 row)

* /“”

示例如下：

* SELECT interval '1 hour' / double precision '1.5' AS RESULT;  
   result   
  ----------  
   00:40:00  
  (1 row)

**时间/日期函数**

* age(timestamp, timestamp)

描述：将两个参数相减，并以年、月、日作为返回值。若相减值为负，则函数返回亦为负，入参可以都带timezone或都不带timezone。

返回值类型：interval

示例：

SELECT age(timestamp '2001-04-10', timestamp '1957-06-13');  
   
 age  
-------------------------  
 43 years 9 mons 27 days  
(1 row)

* age(timestamp)

描述：当前时间和参数相减，入参可以带或者不带timezone。

返回值类型：interval

示例：

* SELECT age(timestamp '1957-06-13');  
     
   age   
  -------------------------  
   60 years 2 mons 18 days  
  (1 row)
* clock\_timestamp()

描述：实时时钟的当前时间戳。

返回值类型：timestamp with time zone

示例：

* SELECT clock\_timestamp();  
     
   clock\_timestamp  
  -------------------------------  
   2022-06-23 16:26:09.726396+08  
  (1 row)
* current\_date

描述：当前日期。  
  
返回值类型：date

示例：

* SELECT current\_date;  
   date   
  ------------  
  2022-06-23  
  (1 row)
* current\_time

描述：当前时间。  
 返回值类型：time with time zone

示例：

* SELECT current\_time;  
   timetz   
   --------------------  
   16:26:37.704326+08  
  (1 row)
* current\_timestamp

描述：当前日期及时间。  
 返回值类型：timestamp with time zone

示例：

* SELECT current\_timestamp;  
   pg\_systimestamp   
   ------------------------------  
   2022-06-23 16:27:29.336502+08  
  (1 row)
* date\_part(text, timestamp)

描述：获取日期/时间值中子域的值，例如年或者小时的值。等效于 extract(field from timestamp)。  
 timestamp类型：abstime、date、interval、reltime、time with time zone、 time without time zone、timestamp with time zone、timestamp without time zone。  
 返回值类型：double precision

示例：

* SELECT date\_part('hour', timestamp '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   20  
  (1 row)
* date\_part(text, interval)

描述：获取日期/时间值中子域的值。获取月份值时，如果月份值大于12，则 取与12的模。等效于extract(field from timestamp)。  
  
 返回值类型：double precision

示例：

* SELECT date\_part('month', interval '2 years 3 months');  
   date\_part   
  -----------  
   3  
  (1 row)
* date\_trunc(text, timestamp)

描述：截取到参数text指定的精度。  
 返回值类型：interval、timestamp with time zone、timestamp without time zone

示例：

* SELECT date\_trunc('hour', timestamp '2001-02-16 20:38:40');  
   date\_trunc   
   ---------------------  
   2001-02-16 20:00:00  
  (1 row)
* trunc(timestamp)

描述：默认按天截取。

示例：

* SELECT trunc(timestamp '2001-02-16 20:38:40'); trunc  
   ---------------------  
   2001-02-16 00:00:00  
  (1 row)
* daterange(arg1, arg2)

描述：获取时间边界信息。arg1和arg2的类型为date。  
 返回值类型：daterange

示例：

* SELECT daterange('2000-05-06','2000-08-08');  
   daterange   
   -------------------------  
   [2000-05-06,2000-08-08)  
  (1 row)
* daterange(arg1, arg2, text)

描述：获取时间边界信息。arg1和arg2的类型为date，text类型为text。  
 返回值类型：daterange

示例：

* SELECT daterange('2000-05-06','2000-08-08','[]');  
   daterange   
   -------------------------  
   [2000-05-06,2000-08-09)  
  (1 row)
* extract(field from timestamp)

描述：获取小时的值。  
 返回值类型：double precision

示例：

* SELECT extract(hour from timestamp '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   20  
  (1 row)
* extract(field from interval)

描述：获取月份的值。如果大于12，则取与12的模。  
 返回值类型：double precision

示例：

* SELECT extract(month from interval '2 years 3 months');  
   date\_part   
  -----------  
   3  
  (1 row)
* isfinite(date)

描述：测试是否为有效日期。  
 返回值类型：Boolean

示例：

* SELECT isfinite(date '2001-02-16');  
   isfinite   
  ----------  
   t  
  (1 row)
* isfinite(timestamp)

描述：测试判断是否为有效时间。  
  
 返回值类型：Boolean

示例：

* SELECT isfinite(timestamp '2001-02-16 21:28:30');  
   isfinite   
  ----------  
   t  
  (1 row)
* isfinite(interval)

描述：测试是否为有效区间。  
 返回值类型：Boolean

示例：

* SELECT isfinite(interval '4 hours');  
   isfinite   
  ----------  
   t  
  (1 row)
* justify\_days(interval)

描述：将时间间隔以月（30天为一月）为单位。  
 返回值类型：interval

示例：

* SELECT justify\_days(interval '35 days');  
   justify\_days   
  --------------  
   1 mon 5 days  
  (1 row)
* justify\_hours(interval)

描述：将时间间隔以天（24小时为一天）为单位。  
 返回值类型：interval

示例：

* SELECT JUSTIFY\_HOURS(INTERVAL '27 HOURS');  
  justify\_hours   
  ----------------  
   1 day 03:00:00  
  (1 row)
* justify\_interval(interval)

描述：结合justify\_days和justify\_hours，调整interval。  
 返回值类型：interval

示例：

* SELECT JUSTIFY\_INTERVAL(INTERVAL '1 MON -1 HOUR');  
  justify\_interval   
  ------------------  
   29 days 23:00:00
* localtime

描述：当前时间。  
 返回值类型：time

示例：

* SELECT localtime AS RESULT;  
  result  
  ----------------  
   16:05:55.664681  
  (1 row)
* localtimestamp

描述：当前日期及时间。  
 返回值类型：timestamp

示例：

* SELECT localtimestamp;  
   timestamp   
  ----------------------------  
   2017-09-01 17:03:30.781902  
  (1 row)
* now()
* 描述：当前日期及时间。  
  返回值类型：timestamp with time zone

示例：

* SELECT now();  
   now  
  -------------------------------  
   2022-08-01 18:00:22.753185+08  
  (1 row)
* timenow
* 描述：当前日期及时间。  
  返回值类型：timestamp with time zone

示例：

* select timenow();  
   timenow  
  ------------------------  
   2022-08-01 18:00:50+08  
  (1 row)
* numtodsinterval(num, interval\_unit)
* 描述：将数字转换为interval类型。num为numeric类型数字，interval\_unit为固定格式字符串（'DAY' | 'HOUR' | 'MINUTE' | 'SECOND'）。  
  可以通过设置参数IntervalStyle为a，兼容该函数interval输出格式。
* 示例：
* SELECT numtodsinterval(100, 'HOUR');  
   numtodsinterval   
  -----------------  
   100:00:00  
  (1 row)  
     
  SET intervalstyle = a;  
   SET  
     
  SELECT numtodsinterval(100, 'HOUR');  
   numtodsinterval  
  -------------------------------  
   +000000004 04:00:00.000000000  
  (1 row)
* pg\_sleep(seconds)
* 描述：服务器线程延迟时间，单位为秒。  
  返回值类型：void

示例：

* SELECT pg\_sleep(10);  
   pg\_sleep   
   ----------  
     
  (1 row)
* statement\_timestamp()
* 描述：当前日期及时间。  
  返回值类型：timestamp with time zone

示例：

* SELECT statement\_timestamp();  
   statement\_timestamp  
   ------------------------------  
   2022-08-01 17:26:08.15684+08  
  (1 row)
* sysdate
* 描述：当前日期及时间。  
  返回值类型：timestamp

示例：

* SELECT sysdate;  
   sysdate  
   ---------------------  
   2022-08-01 16:49:22  
  (1 row)
* timeofday()
* 描述：当前日期及时间（像clock\_timestamp，但是返回时为text）。  
  返回值类型：text

示例：

* SELECT timeofday();  
     
   timeofday  
  -------------------------------------  
   Mon Aug 01 16:49:48.255352 2022 CST  
  (1 row)
* transaction\_timestamp()
* 描述：当前日期及时间，与current\_timestamp等效。  
  返回值类型：timestamp with time zone

示例：

* SELECT transaction\_timestamp();  
   transaction\_timestamp  
   -------------------------------  
   2022-08-01 16:50:05.538612+08  
  (1 row)
* add\_months(d,n)
* 描述：用于计算时间点d再加上n个月的时间。  
  返回值类型：timestamp

示例：

* SELECT add\_months(to\_date('2017-5-29', 'yyyy-mm-dd'), 11) FROM sys\_dummy;  
   add\_months   
   ---------------------  
   2018-04-29 00:00:00  
  (1 row)
* last\_day(d)
* 描述：用于计算时间点d当月最后一天的时间。  
  返回值类型：timestamp

示例：

* select last\_day(to\_date('2017-01-01', 'YYYY-MM-DD')) AS cal\_result;  
   cal\_result   
   ---------------------  
   2017-01-31 00:00:00  
  (1 row)
* next\_day(x,y)
* 描述：用于计算时间点x开始的下一个星期几（y）的时间。  
  返回值类型：timestamp

示例：

* select next\_day(timestamp '2017-05-25 00:00:00','Sunday')AS cal\_result;  
   cal\_result   
   ---------------------  
   2017-05-28 00:00:00  
  (1 row)
* tinterval(abstime, abstime )
* 描述：用两个绝对时间创建时间间隔。  
  返回值类型：tinterval

示例：

call tinterval(abstime 'May 10, 1947 23:59:12', abstime 'Mon May 1 00:30:30 1995');  
 tinterval  
-----------------------------------------------------  
 ["1947-05-10 23:59:12+09" "1995-05-01 00:30:30+08"]  
(1 row)

* tintervalend(tinterval)
* 描述：返回tinteval的结束时间。  
  返回值类型：abstime

示例：

* select tintervalend('["Sep 4, 1983 23:59:12" "Oct4, 1983 23:59:12"]');  
   tintervalend  
  ------------------------  
   1983-10-04 23:59:12+08  
  (1 row)
* tintervalrel(tinterval)
* 描述：计算并返回tinterval的相对时间。  
  返回值类型：reltime

示例：

* select tintervalrel('["Sep 4, 1983 23:59:12" "Oct4, 1983 23:59:12"]');  
   tintervalrel  
  --------------  
   1 mon
* smalldatetime\_ge
* 描述：判断是否第一个参数大于等于第二个参数。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean
* smalldatetime\_cmp
* 描述：对比smalldatetime是否相等。
* 参数：smalldatetime, smalldatetime
* 返回值类型：integer
* smalldatetime\_eq
* 描述：对比smalldatetime是否相等。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean。
* smalldatetime\_gt
* 描述：判断是否第一个参数大于第二个参数。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean
* smalldatetime\_hash
* 描述：计算timestamp对应的哈希值。
* 参数：smalldatetime
* 返回值类型：integer
* smalldatetime\_in
* 描述：输入timestamp。
* 参数：cstring, oid, integer
* 返回值类型：smalldatetime
* smalldatetime\_larger
* 描述：返回较大的timestamp。
* 参数：smalldatetime, smalldatetime
* 返回值类型：smalldatetime
* smalldatetime\_le
* 描述：判断是否第一个参数小于等于第二个参数。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean
* smalldatetime\_lt
* 描述：判断是否第一个参数小于第二个参数。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean
* smalldatetime\_ne
* 描述：比较两个timestamp是否不相等。
* 参数：smalldatetime, smalldatetime
* 返回值类型：boolean
* smalldatetime\_out
* 描述：timestamp转换为外部形式。
* 参数：smalldatetime
* 返回值类型：cstring
* smalldatetime\_send
* 描述：timestamp转换为二进制格式。
* 参数：smalldatetime
* 返回值类型：bytea
* smalldatetime\_smaller
* 描述：返回较小的一个smalldatetime。
* 参数：smalldatetime, smalldatetime
* 返回值类型：smalldatetime
* smalldatetime\_to\_abstime
* 描述：smalldatetime转换为abstime。
* 参数：smalldatetime
* 返回值类型：abstime
* smalldatetime\_to\_time
* 描述：smalldatetime转换为time。
* 参数：smalldatetime
* 返回值类型：time without time zone
* smalldatetime\_to\_timestamp
* 描述：smalldatetime转换为timestamp。
* 参数：smalldatetime
* 返回值类型：timestamp without time zone
* smalldatetime\_to\_timestamptz
* 描述：smalldatetime转换为timestamptz。
* 参数：smalldatetime
* 返回值类型：timestamp with time zone
* smalldatetime\_to\_varchar2
* 描述：smalldatetime转换为varchar2。
* 参数：smalldatetime
* 返回值类型：character varying

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

* 获取当前时间有多种方式，请根据实际业务从场景选择合适的接口：
* 1、以下接口按照当前事务的开始时刻返回值：
* CURRENT\_DATE CURRENT\_TIME CURRENT\_TIME(precision) CURRENT\_TIMESTAMP(precision) LOCALTIME LOCALTIMESTAMP LOCALTIME(precision) LOCALTIMESTAMP(precision)

其中CURRENT\_TIME和CURRENT\_TIMESTAMP(precision)传递带有时区的值；LOCALTIME和LOCALTIMESTAMP传递的值不带时区。CURRENT\_TIME、LOCALTIME和 LOCALTIMESTAMP可以有选择地接受一个精度参数， 该精度导致结果的秒域被园整为指定小数位。如果没有精度参数，结果将被给予所能得到的全部精度。  
 因为这些函数全部都按照当前事务的开始时刻返回结果，所以它们的值在事务运行的整个期间内都不改变。 我们认为这是一个特性：目的是为了允许一个事务在“当前”时间上有一致的概念， 这样在同一个事务里的多个修改可以保持同样的时间戳。

2、以下接口返回当前语句开始时间：

transaction\_timestamp() statement\_timestamp() now()

其中transaction\_timestamp()等价于CURRENT\_TIMESTAMP(precision)，但是其命名清楚地反映了它的返回值。statement\_timestamp()返回当前语句的开始时刻（更准确的说是收到 客户端最后一条命令的时间）。statement\_timestamp()和transaction\_timestamp()在一个事务的第一条命令期间返回值相同，但是在随后的命令中却不一定相同。

now()等效于transaction\_timestamp()。

3、以下接口返回函数被调用时的真实当前时间：

* clock\_timestamp() timeofday()

clock\_timestamp()返回真正的当前时间，因此它的值甚至在同一条 SQL 命令中都会变化。timeofday()和clock\_timestamp()相似，timeofday()也返回真实的当前时间，但是它的结果是一个格式化的text串，而不是timestamp with time zone值。

**EXTRACT**

* EXTRACT(field FROM source)

extract函数从日期或时间的数值里抽取子域，比如年、小时等。source必须是一个timestamp、time或interval类型的值表达式（类型为date的表达式转换为timestamp，因此也可以用）。field是一个标识符或者字符串，它指定从源数据中抽取的域。extract函数返回类型为double precision的数值。field的取值范围如下所示。

* century

世纪。

* 第一个世纪从0001-01-01 00:00:00 AD开始。这个定义适用于所有使用阳历的国家。没有0世纪，直接从公元前1世纪到公元1世纪。
* 示例：
* SELECT EXTRACT(CENTURY FROM TIMESTAMP '2000-12-16 12:21:13');  
   date\_part   
  -----------  
   20  
  (1 row)
* day

如果source为timestamp，表示月份里的日期（1-31）。

* SELECT EXTRACT(DAY FROM TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
   -----------  
   16  
   (1 row)

如果source为interval，表示天数。

* SELECT EXTRACT(DAY FROM INTERVAL '40 days 1 minute');  
   date\_part   
   -----------  
   40  
   (1 row)
* decade
* 年份除以10。
* SELECT EXTRACT(DECADE FROM TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
   -----------  
   200  
  (1 row)
* dow
* 每周的星期几，星期天（0）到星期六（6）。
* SELECT EXTRACT(DOW FROM TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   5  
  (1 row)
* doy
* 一年的第几天（1~365/366）。
* SELECT EXTRACT(DOY FROM TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   47  
  (1 row)
* epoch
* 如果source为timestamp with time zone，表示自1970-01-01 00:00:00-00 UTC以来的秒数（结果可能是负数）；
* 如果source为date和timestamp，表示自1970-01-01 00:00:00-00当地时间以来的秒数；
* 如果source为interval，表示时间间隔的总秒数。
* SELECT EXTRACT(EPOCH FROM TIMESTAMP WITH TIME ZONE '2001-02-16 20:38:40.12-08');  
   date\_part   
   --------------  
   982384720.12  
   (1 row)  
    
  SELECT EXTRACT(EPOCH FROM INTERVAL '5 days 3 hours');  
   date\_part   
  -----------  
   442800  
   (1 row)

将epoch值转换为时间戳的方法。

* SELECT TIMESTAMP WITH TIME ZONE 'epoch' + 982384720.12 \* INTERVAL '1 second' AS RESULT;  
   result   
  ---------------------------  
   2001-02-17 12:38:40.12+08  
   (1 row)
* hour
* 小时域（0-23）。
* SELECT EXTRACT(HOUR FROM TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   20  
  (1 row)
* isodow
* 一周的第几天（1-7）。
* 星期一为1，星期天为7。

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

除了星期天外，都与dow相同。

* SELECT EXTRACT(ISODOW FROM TIMESTAMP '2001-02-18 20:38:40');  
   date\_part   
   -----------  
   7  
  (1 row)
* isoyear
* 日期中的ISO 8601标准年（不适用于间隔）。
* 每个带有星期一开始的周中包含1月4日的ISO年，所以在年初的1月或12月下旬的ISO年可能会不同于阳历的年。详细信息请参见后续的week描述。

SELECT EXTRACT(ISOYEAR FROM DATE '2006-01-01');  
 date\_part   
-----------  
 2005  
(1 row)

SELECT EXTRACT(ISOYEAR FROM DATE '2006-01-02');  
  
 date\_part   
-----------  
  
 2006  
 (1 row)

* microseconds
* 秒域（包括小数部分）乘以1,000,000。

SELECT EXTRACT(MICROSECONDS FROM TIME '17:12:28.5');  
 date\_part   
-----------  
 28500000  
(1 row)

* millennium
* 千年。
* 20世纪（19xx年）里面的年份在第二个千年里。第三个千年从2001年1月1日零时开始。

SELECT EXTRACT(MILLENNIUM FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
-----------  
 3  
(1 row)

* milliseconds
* 秒域（包括小数部分）乘以1000。请注意它包括完整的秒。

SELECT EXTRACT(MILLISECONDS FROM TIME '17:12:28.5');  
 date\_part   
-----------  
 28500  
(1 row)

* minute
* 分钟域（0-59）。

SELECT EXTRACT(MINUTE FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
-----------  
 38  
(1 row)

* month
* 如果source为timestamp，表示一年里的月份数（1-12）。

SELECT EXTRACT(MONTH FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
-----------  
 2  
(1 row)

如果source为interval，表示月的数目，然后对12取模（0-11）。

SELECT EXTRACT(MONTH FROM INTERVAL '2 years 13 months');  
 date\_part   
-----------  
 1  
(1 row)

* quarter
* 该天所在的该年的季度（1-4）。

SELECT EXTRACT(QUARTER FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
-----------  
 1  
(1 row)

* second
* 秒域，包括小数部分（0-59）。

SELECT EXTRACT(SECOND FROM TIME '17:12:28.5');  
 date\_part   
 -----------  
 28.5  
(1 row)

* timezone
* 与UTC的时区偏移量，单位为秒。正数对应UTC东边的时区，负数对应UTC西边的时区。
* timezone\_hour
* 时区偏移量的小时部分。
* timezone\_minute
* 时区偏移量的分钟部分。
* week
* 该天在所在的年份里是第几周。ISO 8601定义一年的第一周包含该年的一月四日（ISO-8601 的周从星期一开始）。换句话说，一年的第一个星期四在第一周。
* 在ISO定义里，一月的头几天可能是前一年的第52或者第53周，十二月的后几天可能是下一年第一周。比如，2005-01-01是2004年的第53周，而2006-01-01是2005年的第52周，2012-12-31是2013年的第一周。建议isoyear字段和week一起使用以得到一致的结果。

SELECT EXTRACT(WEEK FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
 -----------  
 7  
(1 row)

* year
* 年份域。

SELECT EXTRACT(YEAR FROM TIMESTAMP '2001-02-16 20:38:40');  
 date\_part   
 -----------  
 2001  
(1 row)

**date\_part**

date\_part函数是在传统的Ingres函数的基础上制作的（该函数等效于SQL标准函数extract）：

* **date\_part**('field', source)

这里的field参数必须是一个字符串，而不是一个名称。有效的field与extract一样，详细信息请参见EXTRACT。

示例：

* SELECT date\_part('day', TIMESTAMP '2001-02-16 20:38:40');  
   date\_part   
  -----------  
   16  
  (1 row)
* SELECT date\_part('hour', INTERVAL '4 hours 3 minutes');  
   date\_part   
  -----------  
   4  
  (1 row)

表3显示了可以用于格式化日期和时间值的模版。

**表 3** 用于日期/时间格式化的模式

| **类别** | **模式** | **描述** |
| --- | --- | --- |
| 小时 | HH | 一天的小时数（01-12） |
| HH12 | 一天的小时数（01-12） |
| HH24 | 一天的小时数（00-23） |
| 分钟 | MI | 分钟（00-59） |
| 秒 | SS | 秒（00-59） |
| FF | 微秒（000000-999999） |
| SSSSS | 午夜后的秒（0-86399） |
| 上、下午 | AM或A.M. | 上午标识 |
| PM或P.M. | 下午标识 |
| 年 | Y,YYY | 带逗号的年（4和更多位） |
| SYYYY | 公元前四位年 |
| YYYY | 年（4和更多位） |
| YYY | 年的后三位 |
| YY | 年的后两位 |
| Y | 年的最后一位 |
| IYYY | ISO年（4位或更多位） |
| IYY | ISO年的最后三位 |
| IY | ISO年的最后两位 |
| I | ISO年的最后一位 |
| RR | 年的后两位（可在21世纪存储20世纪的年份） |
| RRRR | 可接收4位年或两位年。若是两位，则和RR的返回值相同，若是四位，则和YYYY相同。 |
| BC或B.C.  AD或A.D. | 纪元标识。BC（公元前），AD（公元后）。 |
| 月 | MONTH | 全长大写月份名（空白填充为9字符） |
| MON | 大写缩写月份名（3字符） |
| MM | 月份数（01-12） |
| RM | 罗马数字的月份（I-XII ；I=JAN）（大写） |
| 天 | DAY | 全长大写日期名（空白填充为9字符） |
| DY | 缩写大写日期名（3字符） |
| DDD | 一年里的日（001-366） |
| DD | 一个月里的日（01-31） |
| D | 一周里的日（1-7 ；周日是 1） |
| 周 | W | 一个月里的周数（1-5）（第一周从该月第一天开始） |
| WW | 一年里的周数（1-53）（第一周从该年的第一天开始） |
| IW | ISO一年里的周数（第一个星期四在第一周里） |
| 世纪 | CC | 世纪（2位）（21 世纪从 2001-01-01 开始） |
| 儒略日 | J | 儒略日（自公元前 4712 年 1 月 1 日来的天数） |
| 季度 | Q | 季度 |

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

上表中RR计算年的规则如下：

* 输入的两位年份在00~49之间：当前年份的后两位在00~49之间，返回值年份的前两位和当前年份的前两位相同；当前年份的后两位在50~99之间，返回值年份的前两位是当前年份的前两位加1。
* 输入的两位年份在50~99之间：当前年份的后两位在00~49之间，返回值年份的前两位是当前年份的前两位减1；当前年份的后两位在50~99之间，返回值年份的前两位和当前年份的前两位相同。

#### JSON/JSONB函数和操作符

JSON/JSONB通用操作符参考表1，JSONB额外支持操作符参考表2。

**表 1** JSON/JSONB通用操作符

| **操作符** | **左操作数类型** | **右操作数类型** | **返回类型** | **描述** | **例子** | **例子结果** |
| --- | --- | --- | --- | --- | --- | --- |
| -> | Array-json(b) | int | json(b) | 获得array-json元素。下标不存在返回空。 | '[{“a”:“foo”},{“b”:“bar”},{“c”:“baz”}]'::json->2 | {“c”:“baz”} |
| -> | object-json(b) | text | json(b) | 通过键获得值。不存在则返回空。 | '{“a”: {“b”:“foo”}}'::json->'a' | {“b”:“foo”} |
| ->> | Array-json(b) | int | text | 获得 JSON 数组元素。下标不存在返回空。 | '[1,2,3]'::json->>2 | 3 |
| ->> | object-json(b) | text | text | 通过键获得值。不存在则返回空。 | '{“a”:1,“b”:2}'::json->>'b' | 2 |
| #> | container-json (b) | text[] | json(b) | 获取在指定路径的 JSON 对象，路径不存在则返回空。 | '{“a”: {“b”:{“c”: “foo”}}}'::json #>'{a,b}' | {“c”: “foo”} |
| #>> | container-json (b) | text[] | text | 获取在指定路径的 JSON 对象，路径不存在则返回空。 | '{“a”:[1,2,3],“b”:[4,5,6]}'::json #>>'{a,2}' | 3 |

对于 #> 和 #>> 操作符，当给出的路径无法查找到数据时，不会报错，会返回空。

**表 2** JSONB额外支持操作符

| **操作符** | **右操作数类型** | **描述** | **例子** |
| --- | --- | --- | --- |
| @> | jsonb | 左边的 JSON的顶层是否包含右边JSON的顶层所有项。 | '{“a”:1, “b”:2}'::jsonb @> '{“b”:2}'::jsonb |
| <@ | jsonb | 左边的 JSON的所有项是否全部存在于右边JSON的顶层。 | '{“b”:2}'::jsonb <@ '{“a”:1, “b”:2}'::jsonb |
| ? | text | 键/元素的字符串是否存在于 JSON 值的顶层。 | '{“a”:1, “b”:2}'::jsonb ? 'b' |
| ? |  | text[] | 这些数组字符串中的任何一个是否做为顶层键存在。 |
| ?& | text[] | 是否所有这些数组字符串都作为顶层键存在。 | '[“a”, “b”]'::jsonb ?& array['a', 'b'] |
| = | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_eq。 | / |
| <> | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_ne。 | / |
| < | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_lt。 | / |
| > | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_gt。 | / |
| <= | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_le。 | / |
| >= | jsonb | 判断两个jsonb的大小关系，同函数jsonb\_ge。 | / |

**JSON/JSONB支持的函数**

* array\_to\_json(anyarray [, pretty\_bool])
* 描述：返回JSON类型的数组。一个多维数组成为一个JSON数组的数组。如果pretty\_bool为true，将在一维元素之间添加换行符。
* 返回类型：json
* 示例：
* SELECT array\_to\_json('{{1,5},{99,100}}'::int[]);

当结果显示如下信息，则表示函数调用成功。

* array\_to\_json   
   ------------------  
   [[1,5],[99,100]]  
   (1 row)
* row\_to\_json(record [, pretty\_bool])
* 描述：返回JSON类型的行。如果pretty\_bool为true，将在第一级元素之间添加换行符。
* 返回类型：json
* 示例：
* SELECT row\_to\_json(row(1,'foo'));

当结果显示如下信息，则表示函数调用成功。

* row\_to\_json   
   ---------------------  
   {"f1":1,"f2":"foo"}   
  (1 row)
* json\_array\_element(array-json, integer)、jsonb\_array\_element(array-jsonb, integer)
* 描述：同操作符'->', 返回数组中指定下标的元素。
* 返回类型：json、jsonb
* 示例：
* select json\_array\_element('[1,true,[1,[2,3]],null]',2);

当结果显示如下信息，则表示函数调用成功。

* json\_array\_element  
  --------------------  
   [1,[2,3]]  
  (1 row)
* json\_array\_element\_text(array-json, integer)、jsonb\_array\_element\_text(array-jsonb, integer)
* 描述：同操作符'->>', 返回数组中指定下标的元素。
* 返回类型：text、text
* 示例：
* select json\_array\_element\_text('[1,true,[1,[2,3]],null]',2);

当结果显示如下信息，则表示函数调用成功。

* json\_array\_element\_text  
   -----------------------  
   [1,[2,3]]  
  (1 row)
* json\_object\_field(object-json, text)、jsonb\_object\_field(object-jsonb, text)
* 描述：同操作符'->', 返回对象中指定键对应的值。
* 返回类型：json、json
* 示例：
* select json\_object\_field('{"a": {"b":"foo"}}','a');

当结果显示如下信息，则表示函数调用成功。

* json\_object\_field  
   -------------------  
   {"b":"foo"}  
   (1 row)
* json\_object\_field\_text(object-json, text)、jsonb\_object\_field\_text(object-jsonb, text)

描述：同操作符'->>', 返回对象中指定键对应的值。

返回类型：text、text

示例：

* select json\_object\_field\_text('{"a": {"b":"foo"}}','a');

当结果显示如下信息，则表示函数调用成功。

* json\_object\_field\_text  
  ----------------------  
   {"b":"foo"}  
  (1 row)
* json\_extract\_path(json, VARIADIC text[])、jsonb\_extract\_path((jsonb, VARIADIC text[])
* 描述：等价于操作符'#>'。根据$2所指的路径，查找json，并返回。
* 返回类型：json、jsonb
* 示例：
* select json\_extract\_path('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');

当结果显示如下信息，则表示函数调用成功。

* json\_extract\_path  
  -------------------  
   "stringy"  
  (1 row)
* json\_extract\_path\_text(json, VARIADIC text[])、jsonb\_extract\_path\_text((jsonb, VARIADIC text[])
* 描述：等价于操作符'#>>'。根据$2所指的路径，查找json，并返回。
* 返回类型：text、text
* 示例：
* select json\_extract\_path\_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"stringy"}}', 'f4','f6');

当结果显示如下信息，则表示函数调用成功。

* json\_extract\_path\_text  
   -----------------------  
   "stringy"  
  (1 row)
* json\_array\_elements(array-json)、jsonb\_array\_elements(array-jsonb)
* 描述：拆分数组，每一个元素返回一行。
* 返回类型：json、jsonb
* 示例：
* select json\_array\_elements('[1,true,[1,[2,3]],null]');

当结果显示如下信息，则表示函数调用成功。

* json\_array\_elements  
   ---------------------  
   1  
   true  
   [1,[2,3]]  
   null  
   (4 rows)
* json\_array\_elements\_text(array-json)、jsonb\_array\_elements\_text(array-jsonb)
* 描述：拆分数组，每一个元素返回一行。
* 返回类型：text、text
* 示例：
* select \* from json\_array\_elements\_text('[1,true,[1,[2,3]],null]');

当结果显示如下信息，则表示函数调用成功。

* value  
   -----------  
   1  
   true  
   [1,[2,3]]   
   (4 rows)
* json\_array\_length(array-json)、jsonb\_array\_length(array-jsonb)
* 描述：返回数组长度。
* 返回类型：integer
* 示例：
* SELECT json\_array\_length('[1,2,3,{"f1":1,"f2":[5,6]},4,null]');

当结果显示如下信息，则表示函数调用成功。

* json\_array\_length  
   -------------------  
   6  
   (1 row)
* json\_each(object-json)、jsonb\_each(object-jsonb)
* 描述：将对象的每个键值对拆分转换成一行两列。
* 返回类型：setof(key text, value json)、setof(key text, value jsonb)
* 示例：
* select \* from json\_each('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');

当结果显示如下信息，则表示函数调用成功。

* key | value  
  -----+----------  
   f1 | [1,2,3]  
   f2 | {"f3":1}  
   f4 | null  
  (3 rows)
* json\_each\_text(object-json)、jsonb\_each\_text(object-jsonb)
* 描述：将对象的每个键值对拆分转换成一行两列。
* 返回类型：setof(key text, value text)、setof(key text, value text)
* 示例：
* select \* from json\_each\_text('{"f1":[1,2,3],"f2":{"f3":1},"f4":null}');

当结果显示如下信息，则表示函数调用成功。

* key | value  
  -----+----------  
   f1 | [1,2,3]  
   f2 | {"f3":1}  
   f4 |  
  (3 rows)
* json\_object\_keys(object-json)、jsonb\_object\_keys(object-jsonb)
* 描述：返回对象中顶层的所有键。
* 返回类型：SETOF text
* 示例：
* select json\_object\_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}, "f1":"abcd"}');

当结果显示如下信息，则表示函数调用成功。

* json\_object\_keys  
  ------------------  
   f1  
   f2  
   f1  
  (3 rows)
* jsonb中会有去重操作
* select jsonb\_object\_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}, "f1":"abcd"}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_object\_keys  
  -------------------  
   f1  
   f2  
   (2 rows)
* json\_populate\_record(anyelement, object-json [, bool])、jsonb\_populate\_record(anyelement, object-jsonb [, bool])
* 描述：$1必须是一个复合类型的参数。将会把object-json里的每个对键值进行拆分，以键当做列名，与$1中的列名进行匹配查找，并填充到$1的格式中。
* 返回类型：anyelement、anyelement
* 示例：
* 1、创建类型jpop。
* create type jpop as (a text, b int, c bool);

2、查询验证。

* select \* from json\_populate\_record(null::jpop,'{"a":"blurfl","x":43.2}');

当结果显示如下信息，则表示函数调用成功。

* a | b | c  
  --------+---+---  
   blurfl | |  
  (1 row)

3、查询验证。

* select \* from json\_populate\_record((1,1,null)::jpop,'{"a":"blurfl","x":43.2}');

当结果显示如下信息，则表示函数调用成功。

* a | b | c  
  --------+---+---  
   blurfl | 1 |  
  (1 row)
* json\_populate\_record\_set(anyelement, array-json [, bool])、jsonb\_populate\_record\_set(anyelement, array-jsonb [, bool])
* 描述：参考上述函数json\_populate\_record、jsonb\_populate\_record，对$2数组的每一个元素进行上述参数函数的操作，因此这也要求$2数组的每个元素都是object-json类型的。
* 返回类型：setof anyelement、setof anyelement
* 示例：
* 1、创建类型jpop
* create type jpop as (a text, b int, c bool);

2、查询验证

* select \* from json\_populate\_recordset(null::jpop, '[{"a":1,"b":2},{"a":3,"b":4}]');

当结果显示如下信息，则表示函数调用成功。

* a | b | c  
   ---+---+---  
   1 | 2 |  
   3 | 4 |  
  (2 rows)
* json\_typeof(json)、jsonb\_typeof(jsonb)
* 描述：检测json类型
* 返回类型：text、text
* 示例：
* select value, json\_typeof(value) from (values (json '123.4'), (json '"foo"'), (json 'true'), (json 'null'), (json '[1, 2, 3]'), (json '{"x":"foo", "y":123}'), (NULL::json)) as data(value);

当结果显示如下信息，则表示函数调用成功。

* value | json\_typeof  
  ----------------------+-------------  
   123.4 | number  
   "foo" | string  
   true | boolean  
   null | null  
   [1, 2, 3] | array  
   {"x":"foo", "y":123} | object  
   |  
   (7 rows)
* json\_build\_array( [VARIADIC “any”] )
* 描述：从一个可变参数列表构造出一个JSON数组。
* 返回类型：array-json
* 示例：
* select json\_build\_array('a',1,'b',1.2,'c',true,'d',null,'e',json '{"x": 3, "y": [1,2,3]}','');

当结果显示如下信息，则表示函数调用成功。

* json\_build\_array   
   -----------------------------------------------------------------------------  
   ["a", 1, "b", 1.2, "c", true, "d", null, "e", {"x": 3, "y": [1,2,3]}, null]  
   (1 row)
* json\_build\_object( [VARIADIC “any”] )
* 描述：从一个可变参数列表构造出一个JSON对象，其入参必须为偶数个，两两一组组成键值对。注意键不可为null。
* 返回类型：object-json
* 示例：
* select json\_build\_object(1,2);

当结果显示如下信息，则表示函数调用成功。

* json\_build\_object  
  -------------------  
   {"1" : 2}  
  (1 row)
* json\_to\_record(object-json, bool)
* 描述：正如所有返回record 的函数一样，调用者必须用一个AS子句显式地定义记录的结构。会将object-json的键值对进行拆分重组，把键当做列名，去匹配填充as显示指定的记录的结构。
* 返回类型：record
* 示例：
* select \* from json\_to\_record('{"a":1,"b":"foo","c":"bar"}',true) as x(a int, b text, d text);

当结果显示如下信息，则表示函数调用成功。

* a | b | d  
   ---+-----+---  
   1 | foo |  
  (1 row)
* json\_to\_recordset(array-json, bool)
* 描述：参考函数json\_to\_record，对数组内个每个元素，执行上述函数的操作，因此这要求数组内的每个元素都得是object-json。
* 返回类型：setof record
* 示例：
* select \* from json\_to\_recordset('[{"a":1,"b":"foo","d":false},{"a":2,"b":"bar","c":true}]',false) as x(a int, b text, c boolean);

当结果显示如下信息，则表示函数调用成功。

* a | b | c  
   ---+-----+---  
   1 | foo |  
   2 | bar | t  
  (2 rows)
* json\_object(text[])、json\_object(text[], text[])
* 描述：从一个文本数组构造一个object-json。这是个重载函数，当入参为一个文本数组的时候，其数组长度必须为偶数，成员被当做交替出现的键/值对。两个文本数组的时候，第一个数组认为是键，第二个认为是值，两个数组长度必须相等。键不可为null。
* 返回类型：object-json

示例1：

* select json\_object('{a,1,b,2,3,NULL,"d e f","a b c"}');

当结果显示如下信息，则表示函数调用成功。

* json\_object  
  -------------------------------------------------------  
   {"a" : "1", "b" : "2", "3" : null, "d e f" : "a b c"}  
  (1 row)

示例2：

* select json\_object('{a,b,"a b c"}', '{a,1,1}');

当结果显示如下信息，则表示函数调用成功。

* json\_object  
  ---------------------------------------  
   {"a" : "a", "b" : "1", "a b c" : "1"}  
  (1 row)
* json\_agg(any)
* 描述：将值聚集为json数组。
* 返回类型：array-json
* 示例：

1、创建测试表class。

* CREATE TABLE class(name varchar,score int);  
  INSERT INTO class(name,score) values ('A' ,'2');  
  INSERT INTO class(name,score) values ('A' ,'3');  
  INSERT INTO class(name,score) values ('D' ,'5');  
  INSERT INTO class(name,score) values ('D' ,'');

2、调用函数进行验证。

* select name, json\_agg(score),score from classes group by name order by name;

当结果显示如下信息，则表示函数调用成功。

* name | json\_agg   
   ------+-----------  
   A | [2, 3]  
   D | [5, null]  
   (2 rows)
* json\_object\_agg(any, any)
* 描述：将值聚集为json对象。
* 返回类型：object-json
* 示例：

1、创建测试表class。

* CREATE TABLE class(name varchar,score int);  
  INSERT INTO class(name,score) values ('A' ,'2');  
  INSERT INTO class(name,score) values ('A' ,'3');  
  INSERT INTO class(name,score) values ('D' ,'5');  
  INSERT INTO class(name,score) values ('D' ,'');

2、调用函数进行验证。

* select json\_object\_agg(name, score) from class group by name order by name;

当结果显示如下信息，则表示函数调用成功。

* json\_object\_agg  
  -------------------------  
   { "A" : 2, "A" : 3 }  
   { "D" : 5, "D" : null }  
  (2 rows)
* jsonb\_contained(jsonb, jsonb)
* 描述：同操作符 '<@', 判断$1中的所有元素是否在$2的顶层存在。
* 返回类型：bool
* 示例：
* select jsonb\_contained('[1,2,3]', '[1,2,3,4]');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_contained  
  -----------------  
   t  
  (1 row)
* jsonb\_contains(jsonb, jsonb)
* 描述：同操作符 '@>', 判断$1中的顶层所有元素是否包含在$2的所有元素。
* 返回类型：bool
* 示例：
* select jsonb\_contains('[1,2,3,4]', '[1,2,3]');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_contains  
  ----------------  
   t  
  (1 row)
* jsonb\_exists(jsonb, text)
* 描述：同操作符 '?', 字符串$2是否存在$1的顶层以keyelemscalar的形式存在。
* 返回类型：bool
* 示例：
* select jsonb\_exists('["1",2,3]', '1');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_exists  
  --------------  
   t  
  (1 row)
* jsonb\_exists\_all(jsonb, text[])
* 描述：同操作符 '?&', 字符串数组$2里面，是否所有的元素，都在$1的顶层以keyelemscalar的形式存在。
* 返回类型：bool
* 示例：
* select jsonb\_exists\_all('["1","2",3]', '{1, 2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_exists\_all  
  ------------------  
   t  
  (1 row)
* jsonb\_exists\_any(jsonb, text[])
* 描述：同操作符 '?|', 字符串数组$2里面，是否存在的元素，在$1的顶层以keyelemscalar的形式存在。
* 返回类型：bool
* 示例：
* select jsonb\_exists\_any('["1","2",3]', '{1, 2, 4}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_exists\_any  
  ------------------  
   t  
  (1 row)
* jsonb\_cmp(jsonb, jsonb)
* 描述：比较大小，正数代表大于，负数代表小于，0表示相等。
* 返回类型：integer
* 示例：
* select jsonb\_cmp('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_cmp  
   -----------  
   -1  
  (1 row)
* jsonb\_eq(jsonb, jsonb)
* 描述：同操作符 '=', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_eq('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_eq  
  ----------  
   f  
  (1 row)
* jsonb\_ne(jsonb, jsonb)
* 描述：同操作符 '<>', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_ne('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_ne  
  ----------  
   t  
  (1 row)
* jsonb\_gt(jsonb, jsonb)
* 描述：同操作符 '>', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_gt('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_gt  
  ----------  
   f  
  (1 row)
* jsonb\_ge(jsonb, jsonb)
* 描述：同操作符 '>=', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_ge('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_ge  
  ----------  
   f  
  (1 row)
* jsonb\_lt(jsonb, jsonb)
* 描述：同操作符 '<', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_lt('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_lt  
  ----------  
   t  
  (1 row)
* jsonb\_le(jsonb, jsonb)
* 描述：同操作符 '<=', 比较两个值的大小。
* 返回类型：bool
* 示例：
* select jsonb\_le('["a", "b"]', '{"a":1, "b":2}');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_le  
  ----------  
   t  
  (1 row)
* to\_json(anyelement)
* 描述：把参数转换为'json'。
* 返回类型：json
* 示例：
* select to\_json('{1,5}'::text[]);

当结果显示如下信息，则表示函数调用成功。

* to\_json  
  -----------  
   ["1","5"]  
  (1 row)
* jsonb\_hash(jsonb)
* 描述：对jsonb进行hash运算。
* 返回类型：integer
* 示例：
* select jsonb\_hash('[1,2,3]');

当结果显示如下信息，则表示函数调用成功。

* jsonb\_hash  
   ------------  
   -559968547  
   (1 row)
* 其他函数
* 描述：gin索引以及jsonjsonb聚集函数所用到的内部函数，功能不过多赘述。
* gin\_compare\_jsonb  
  gin\_consistent\_jsonb  
  gin\_consistent\_jsonb\_hash  
  gin\_extract\_jsonb  
  gin\_extract\_jsonb\_hash  
  gin\_extract\_jsonb\_query  
  gin\_extract\_jsonb\_query\_hash  
  gin\_triconsistent\_jsonb  
  gin\_triconsistent\_jsonb\_hash  
    
  json\_agg\_transfn  
  json\_agg\_finalfn  
  json\_object\_agg\_transfn  
  json\_object\_agg\_finalfn

#### 安全函数

**安全函数**

* gs\_encrypt\_aes128(encryptstr,keystr)
* 描述：以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。keystr的长度范围为8~16字节，至少包含3种字符（大写字母、小写字母、数字、特殊字符）。
* 返回值类型：text
* 返回值长度：至少为92字节，不超过4\*[(Len+68)/3]字节，其中Len为加密前数据长度（单位为字节）。
* 示例：
* vastbase=# SELECT gs\_encrypt\_aes128('MPPDB','Asdf1234');  
     
   gs\_encrypt\_aes128  
  ---------------------------------------------------------------------------------------  
   gwditQLQG8NhFw4OuoKhhQJoXojhFlYkjeG0aYdSCtLCnIUgkNwvYI04KbuhmcGZp8jWizBdR1vU9CspjuzI0lbz12A=  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

由于该函数的执行过程需要传入解密口令，为了安全起见，vsql工具不会将该函数记录入执行历史。即无法在vsql里通过上下翻页功能找到该函数的执行历史。

* gs\_encrypt(encryptstr,keystr,encrypttype)
* 描述：根据encrypttype，以keystr为密钥对encryptstr字符串进行加密，返回加密后的字符串。keystr的长度范围为8~16字节，至少包含3种字符（大写字母、小写字母、数字、特殊字符），encrypttype可以是aes128或sm4。
* 返回值类型：text
* 示例：
* vastbase=# SELECT gs\_encrypt('MPPDB','Asdf1234','sm4');   
   gs\_encrypt   
   --------------------------------  
   ZBzOmaGA4Bb+coyucJ0B8AkIShqc  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

由于该函数的执行过程需要传入解密口令，为了安全起见，vsql工具不会将该函数记录入执行历史。即无法在vsql里通过上下翻页功能找到该函数的执行历史。

* gs\_decrypt\_aes128(decryptstr,keystr)
* 描述：以keystr为密钥对decrypt字符串进行解密，返回解密后的字符串。解密使用的keystr必须保证与加密时使用的keystr一致才能正常解密。keystr不得为空。

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

* 此参数需要结合gs\_encrypt\_aes128加密函数共同使用。
* 返回值类型：text
* 示例：
* vastbase=# SELECT gs\_decrypt\_aes128('gwditQLQG8NhFw4OuoKhhQJoXojhFlYkjeG0aYdSCtLCnIUgkNwvYI04KbuhmcGZp8jWizBdR1vU9CspjuzI0lbz12A=','1234');  
   gs\_decrypt\_aes128   
  ---------------------  
   MPPDB  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

由于该函数的执行过程需要传入解密口令，为了安全起见，vsql工具不会将该函数记录入执行历史；即无法在vsql里通过上下翻页功能找到该函数的执行历史。

* gs\_decrypt(decryptstr,keystr,decrypttype)
* 描述：根据decrypttype，以keystr为密钥对decrypt字符串进行解密，返回解密后的字符串。解密使用的decrypttype及keystr必须保证与加密时使用的encrypttype及keystr一致才能正常解密。keystr不得为空。decrypttype可以是aes128或sm4。
* 此函数需要结合gs\_encrypt加密函数共同使用。
* 返回值类型：text
* 示例：
* vastbase=# select gs\_decrypt('ZBzOmaGA4Bb+coyucJ0B8AkIShqc','Asdf1234','sm4');  
   gs\_decrypt   
  --------------  
   MPPDB  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

由于该函数的执行过程需要传入解密口令，为了安全起见，vsql工具不会将该函数记录入执行历史；即无法在vsql里通过上下翻页功能找到该函数的执行历史。

* gs\_password\_deadline
* 描述：显示当前帐户密码离过期还距离多少天。
* 返回值类型：interval
* 示例：
* vastbase=# SELECT gs\_password\_deadline();  
   gs\_password\_deadline   
  ---------------------------  
   83 days 17:44:32.196094  
  (1 row)
* gs\_password\_notifytime
* 描述：显示帐户密码到期前提醒的天数。
* 返回值类型：int32
* login\_audit\_messages
* 描述：查看登录用户的登录信息。
* 返回值类型：元组
* 示例：
  + 查看上一次登录认证通过的日期、时间和IP等信息。
* vastbase=> select \* from login\_audit\_messages(true);  
  username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+---------------+--------+-----------------  
  omm | openGauss | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\]  
  (1 row)
* vastbase=> select \* from login\_audit\_messages(true);  
  username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+---------------+--------+-----------------  
  omm | vastbase | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\]  
  (1 row)
* vastbase=> select \* from login\_audit\_messages(true);  
   username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+---------------+--------+-----------------  
   omm | openGauss | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\]  
  (1 row)
* 查看上一次登录认证失败的日期、时间和IP等信息。
* vastbase=> select \* from login\_audit\_messages(false) ORDER BY logintime desc limit 1;  
   username | database | logintime | mytype | result | client\_conninfo------------+----------+------------------------+--------------+--------+-------------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
   (1 row)
* vastbase=> select \* from login\_audit\_messages(false) ORDER BY logintime desc limit 1;  
   username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+--------------+--------+-------------------  
   omm | vastbase | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
  (1 row)
* vastbase=> select \* from login\_audit\_messages(false) ORDER BY logintime desc limit 1;  
  username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+--------------+--------+-------------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
  (1 row)
* vastbase=> select \* from login\_audit\_messages(false);  
   username | database | logintime | mytype | result | client\_conninfo  
   ------------+----------+------------------------+--------------+--------+-------------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
   omm | openGauss | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\]  
  (2 rows)
* vastbase=> select \* from login\_audit\_messages(false);  
   username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+--------------+--------+-------------------  
   omm | vastbase | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
  omm | vastbase | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\]  
  (2 rows)
* 查看自从最后一次认证通过以来失败的尝试次数、日期和时间。
* vastbase=> select \* from login\_audit\_messages(false);  
   username | database | logintime | mytype | result | client\_conninfo  
  ------------+----------+------------------------+--------------+--------+-------------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\]  
   omm | openGauss | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\]  
   (2 rows)
* login\_audit\_messages\_pid
* 描述：查看登录用户的登录信息。与login\_audit\_messages的区别在于结果基于当前backendid向前查找。所以不会因为同一用户的后续登录，而影响本次登录的查询结果。也就是查询不到该用户后续登录的信息。

返回值类型：元组

* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(true);  
   username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+---------------+--------+-----------------+-----------------  
   omm | openGauss | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\] | 139823109633792  
  (1 row)
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(true);  
  username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+---------------+--------+-----------------+-----------------  
  omm | vastbase | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\] | 139823109633792  
  (1 row)

示例：

* 查看上一次登录认证通过的日期、时间和IP等信息。
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false) ORDER BY logintime desc limit 1;  
  username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
  omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
  (1 row)
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false) ORDER BY logintime desc limit 1;  
  username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
  omm | vastbase | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
  (1 row)
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(true);  
   username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+---------------+--------+-----------------+-----------------  
   omm | openGauss | 2020-06-29 21:56:40+08 | login\_success | ok | vsql@\[local\] | 139823109633792  
  (1 row)
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false);  
  username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
  omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
  omm | openGauss | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
   (2 rows)
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false);  
  username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
  omm | vastbase | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
  omm | vastbase | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
   (2 rows)
* 查看上一次登录认证失败的日期、时间和IP等信息。
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false) ORDER BY logintime desc limit 1;  
   username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
  (1 row)
* 查看自从最后一次认证通过以来失败的尝试次数、日期和时间。
* vastbase=> SELECT \* FROM login\_audit\_messages\_pid(false);  
   username | database | logintime | mytype | result | client\_conninfo | backendid  
  ------------+----------+------------------------+--------------+--------+-------------------+-----------------  
   omm | openGauss | 2020-06-29 21:57:55+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
   omm | openGauss | 2020-06-29 21:57:53+08 | login\_failed | failed | [unknown]@\[local\] | 139823109633792  
   (2 rows)
* inet\_server\_addr
* 描述：显示服务器IP信息。
* 返回值类型：inet
* 示例：
* vastbase=# SELECT inet\_server\_addr();  
   inet\_server\_addr  
  --------------------  
   10.10.0.13  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

* 上面是以客户端在10.10.0.50上，服务器端在10.10.0.13上为例。
* 如果是通过本地连接，使用此接口显示为空。
* inet\_client\_addr
* 描述：显示客户端IP信息。
* 返回值类型：inet
* 示例：
* vastbase=# SELECT inet\_client\_addr();  
   inet\_client\_addr  
  --------------------  
   10.10.0.50  
  (1 row)

D:\build9\v2.2%E6%89%8B%E5%86%8C\%E5%90%88%E5%B9%B6%E6%96%87%E6%A1%A3\public_sys-resources\icon-note.gif **说明：**

* 上面是以客户端在10.10.0.50上，服务器端在10.10.0.13上为例。
* 如果是通过本地连接，使用此接口显示为空。
* pg\_query\_audit
* 描述：查看数据库主节点审计日志。
* 返回值类型：record
* 函数返回字段如下：

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| time | timestamp with time zone | 操作时间 |
| type | text | 操作类型 |
| result | text | 操作结果 |
| userid | oid | 用户id |
| username | text | 执行操作的用户名 |
| database | text | 数据库名称 |
| client\_conninfo | text | 客户端连接信息 |
| object\_name | text | 操作对象名称 |
| detail\_info | text | 执行操作详细信息 |
| node\_name | text | 节点名称 |
| thread\_id | text | 线程id |
| local\_port | text | 本地端口 |
| remote\_port | text | 远端端口 |

* ​函数使用方法及示例请参考：《[VASTDATA]Vastbase G100 V2.2管理员指南->日志参考->审计日志->查看审计结果》章节。
* pg\_delete\_audit
* 描述：删除指定时间段的审计日志。
* 返回值类型：void
* 函数使用方法及示例请参考：《[VASTDATA]Vastbase G100 V2.2管理员指南->日志参考->审计日志->维护审计日志》
* gs\_sm3(text)

描述：用来计算输入字符串的SM3摘要。

返回值类型：SM3哈希值的十六进制字符串，长度固定为64。

示例：

select gs\_sm3('abcdef');

结果显示如下：

gs\_sm3

------------------------------------------------------------------

5d60e23c9fe29b5e62517e144ad67541c6eb132c8926637b6393fe8d9b62b3bf

(1 row)

#### 文本检索函数和操作符

**文本检索操作符**

* @@

描述：tsvector类型的词汇与tsquery类型的词汇是否匹配

示例：

* SELECT to\_tsvector('fat cats ate rats') @@ to\_tsquery('cat & rat') AS RESULT;  
   result   
  --------  
   t  
  (1 row)
* @@@

描述：@@的同义词

示例：

* SELECT to\_tsvector('fat cats ate rats') @@@ to\_tsquery('cat & rat') AS RESULT;  
   result   
  --------  
   t  
  (1 row)
* ||

描述：连接两个tsvector类型的词汇

示例：

* SELECT 'a:1 b:2'::tsvector || 'c:1 d:2 b:3'::tsvector AS RESULT;  
   result   
  ---------------------------  
   'a':1 'b':2,5 'c':3 'd':4  
  (1 row)
* &&

描述：将两个tsquery类型的词汇进行“与”操作

示例：

* SELECT 'fat | rat'::tsquery && 'cat'::tsquery AS RESULT;  
   result   
  ---------------------------  
   ( 'fat' | 'rat' ) & 'cat'  
  (1 row)
* ||

描述：将两个tsquery类型的词汇进行“或”操作

示例：

* SELECT 'fat | rat'::tsquery || 'cat'::tsquery AS RESULT;  
   result   
  ---------------------------  
   ( 'fat' | 'rat' ) | 'cat'  
  (1 row)
* !!

描述：tsquery类型词汇的非关系

示例：

* SELECT !! 'cat'::tsquery AS RESULT;  
   result   
  --------  
   !'cat'  
  (1 row)
* @>

描述：一个tsquery类型的词汇是否包含另一个tsquery类型的词汇

示例：

* SELECT 'cat'::tsquery @> 'cat & rat'::tsquery AS RESULT;  
   result   
  --------  
   f  
  (1 row)
* <@

描述：一个tsquery类型的词汇是否被包含另一个tsquery类型的词汇

示例：

* SELECT 'cat'::tsquery <@ 'cat & rat'::tsquery AS RESULT;  
   result   
  --------  
   t  
  (1 row)

除了上述的操作符，还为tsvector类型和tsquery类型的数据定义了普通的B-tree比较操作符（=、<等）。

**文本检索函数**

* get\_current\_ts\_config()

描述：获取文本检索的默认配置。

返回类型：regconfig

示例：

* SELECT get\_current\_ts\_config();  
   get\_current\_ts\_config   
  -----------------------  
   english  
  (1 row)
* length(tsvector)

描述：tsvector类型词汇的单词数。

返回类型：integer

示例：

* SELECT length('fat:2,4 cat:3 rat:5A'::tsvector);  
   length   
  --------  
   3  
  (1 row)
* numnode(tsquery)

描述：tsquery类型的单词加上操作符的数量。

返回类型：integer

示例：

* SELECT numnode('(fat & rat) | cat'::tsquery);  
   numnode   
  ---------  
   5  
  (1 row)
* plainto\_tsquery([ config regconfig , ] query text)

描述：产生tsquery类型的词汇，并忽略标点。

返回类型：tsquery

示例：

* SELECT plainto\_tsquery('english', 'The Fat Rats');  
   plainto\_tsquery   
  -----------------  
   'fat' & 'rat'  
  (1 row)
* querytree(query tsquery)

描述：获取tsquery类型的词汇可加索引的部分。

返回类型：text

示例：

* SELECT querytree('foo & ! bar'::tsquery);  
   querytree   
  -----------  
   'foo'  
  (1 row)
* setweight(tsvector, "char")

描述：给tsvector类型的每个元素分配权值。

返回类型：tsvector

示例：

* SELECT setweight('fat:2,4 cat:3 rat:5B'::tsvector, 'A');  
   setweight   
  -------------------------------  
   'cat':3A 'fat':2A,4A 'rat':5A  
  (1 row)
* strip(tsvector)

描述：删除tsvector类型单词中的position和权值。

返回类型：tsvector

示例：

* SELECT strip('fat:2,4 cat:3 rat:5A'::tsvector);  
   strip   
  -------------------  
   'cat' 'fat' 'rat'  
  (1 row)
* to\_tsquery([ config regconfig , ] query text)

描述：标准化单词，并转换为tsquery类型。

返回类型：tsquery

示例：

* SELECT to\_tsquery('english', 'The & Fat & Rats');  
   to\_tsquery   
  ---------------  
   'fat' & 'rat'  
  (1 row)
* to\_tsvector([ config regconfig , ] document text)

描述：去除文件信息，并转换为tsvector类型。

返回类型：tsvector

示例：

* SELECT to\_tsvector('english', 'The Fat Rats');  
   to\_tsvector   
  -----------------  
   'fat':2 'rat':3  
  (1 row)
* to\_tsvector\_for\_batch([ config regconfig , ] document text)

描述：去除文件信息，并转换为tsvector类型。

返回类型：tsvector

示例：

* SELECT to\_tsvector\_for\_batch('english', 'The Fat Rats');  
   to\_tsvector   
  -----------------  
   'fat':2 'rat':3  
  (1 row)
* ts\_headline([ config regconfig, ] document text, query tsquery [, options text ])

描述：高亮显示查询的匹配项。

返回类型：text

示例：

* SELECT ts\_headline('x y z', 'z'::tsquery);  
   ts\_headline   
  --------------  
   x y <b>z</b>  
  (1 row)
* ts\_ rank([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ])

描述：文档查询排名。

返回类型：float4

示例：

* SELECT ts\_rank ('hello world'::tsvector, 'world'::tsquery);  
   ts\_rank   
  ----------  
   .0607927  
  (1 row)
* ts\_ rank\_cd([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ])

描述：排序文件查询使用覆盖密度。

返回类型：float4

示例：

* SELECT ts\_rank\_cd('hello world'::tsvector, 'world'::tsquery);  
   ts\_rank\_cd  
  ------------  
   0  
  (1 row)
* ts\_rewrite(query tsquery, target tsquery, substitute tsquery)

描述：替换目标tsquery类型的单词。

返回类型：tsquery

示例：

* SELECT ts\_rewrite('a & b'::tsquery, 'a'::tsquery, 'foo|bar'::tsquery);  
   ts\_rewrite   
  -------------------------  
   'b' & ( 'foo' | 'bar' )  
  (1 row)
* ts\_rewrite(query tsquery, select text)

描述：使用SELECT命令的结果替代目标中tsquery类型的单词。

返回类型：tsquery

示例：

* SELECT ts\_rewrite('world'::tsquery, 'select ''world''::tsquery, ''hello''::tsquery');  
   ts\_rewrite   
  ------------  
   'hello'  
  (1 row)

**文本检索调试函数**

* ts\_debug([ config regconfig, ] document text, OUT alias text, OUT description text, OUT token text, OUT dictionaries regdictionary[], OUT dictionary regdictionary, OUT lexemes text[])

描述：测试一个配置。

返回类型：setof record

示例：

* SELECT ts\_debug('english', 'The Brightest supernovaes');  
   ts\_debug   
  -----------------------------------------------------------------------------------  
   (asciiword,"Word, all ASCII",The,{english\_stem},english\_stem,{})  
   (blank,"Space symbols"," ",{},,)  
   (asciiword,"Word, all ASCII",Brightest,{english\_stem},english\_stem,{brightest})  
   (blank,"Space symbols"," ",{},,)  
   (asciiword,"Word, all ASCII",supernovaes,{english\_stem},english\_stem,{supernova})  
  (5 rows)
* ts\_lexize(dict regdictionary, token text)

描述：测试一个数据字典。

返回类型：text[]

示例：

* SELECT ts\_lexize('english\_stem', 'stars');  
   ts\_lexize   
  -----------  
   {star}  
  (1 row)
* ts\_parse(parser\_name text, document text, OUT tokid integer, OUT token text)

描述：测试一个解析。

返回类型：setof record

示例：

* SELECT ts\_parse('default', 'foo - bar');  
   ts\_parse   
  -----------  
   (1,foo)  
   (12," ")  
   (12,"- ")  
   (1,bar)  
  (4 rows)
* ts\_parse(parser\_oid oid, document text, OUT tokid integer, OUT token text)

描述：测试一个解析。

返回类型：setof record

示例：

* SELECT ts\_parse(3722, 'foo - bar');  
   ts\_parse   
  -----------  
   (1,foo)  
   (12," ")  
   (12,"- ")  
   (1,bar)  
  (4 rows)
* ts\_token\_type(parser\_name text, OUT tokid integer, OUT alias text, OUT description text)

描述：获取分析器定义的记号类型。

返回类型：setof record

示例：

* SELECT ts\_token\_type('default');  
   ts\_token\_type   
  --------------------------------------------------------------  
   (1,asciiword,"Word, all ASCII")  
   (2,word,"Word, all letters")  
   (3,numword,"Word, letters and digits")  
   (4,email,"Email address")  
   (5,url,URL)  
   (6,host,Host)  
   (7,sfloat,"Scientific notation")  
   (8,version,"Version number")  
   (9,hword\_numpart,"Hyphenated word part, letters and digits")  
   (10,hword\_part,"Hyphenated word part, all letters")  
   (11,hword\_asciipart,"Hyphenated word part, all ASCII")  
   (12,blank,"Space symbols")  
   (13,tag,"XML tag")  
   (14,protocol,"Protocol head")  
   (15,numhword,"Hyphenated word, letters and digits")  
   (16,asciihword,"Hyphenated word, all ASCII")  
   (17,hword,"Hyphenated word, all letters")  
   (18,url\_path,"URL path")  
   (19,file,"File or path name")  
   (20,float,"Decimal notation")  
   (21,int,"Signed integer")  
   (22,uint,"Unsigned integer")  
   (23,entity,"XML entity")  
  (23 rows)
* ts\_token\_type(parser\_oid oid, OUT tokid integer, OUT alias text, OUT description text)

描述：获取分析器定义的记号类型。

返回类型：setof record

示例：

* SELECT ts\_token\_type(3722);  
   ts\_token\_type   
  --------------------------------------------------------------  
   (1,asciiword,"Word, all ASCII")  
   (2,word,"Word, all letters")  
   (3,numword,"Word, letters and digits")  
   (4,email,"Email address")  
   (5,url,URL)  
   (6,host,Host)  
   (7,sfloat,"Scientific notation")  
   (8,version,"Version number")  
   (9,hword\_numpart,"Hyphenated word part, letters and digits")  
   (10,hword\_part,"Hyphenated word part, all letters")  
   (11,hword\_asciipart,"Hyphenated word part, all ASCII")  
   (12,blank,"Space symbols")  
   (13,tag,"XML tag")  
   (14,protocol,"Protocol head")  
   (15,numhword,"Hyphenated word, letters and digits")  
   (16,asciihword,"Hyphenated word, all ASCII")  
   (17,hword,"Hyphenated word, all letters")  
   (18,url\_path,"URL path")  
   (19,file,"File or path name")  
   (20,float,"Decimal notation")  
   (21,int,"Signed integer")  
   (22,uint,"Unsigned integer")  
   (23,entity,"XML entity")  
  (23 rows)
* ts\_stat(sqlquery text, [ weights text, ] OUT word text, OUT ndoc integer, OUT nentry integer)

描述：获取tsvector列的统计数据。

返回类型：setof record

示例：

* SELECT ts\_stat('select ''hello world''::tsvector');  
   ts\_stat   
  -------------  
   (world,1,1)  
   (hello,1,1)  
  (2 rows)

#### 类型转换函数

**类型转换函数**

* cash\_words(money)

描述：类型转换函数，将money转换成text。

示例：

* SELECT cash\_words('1.23');  
   cash\_words  
  -----------------------------------  
   One dollar and twenty three cents  
  (1 row)
* cast(x as y)

描述：类型转换函数，将x转换成y指定的类型。

示例：

* SELECT cast('22-oct-1997' as timestamp);  
   timestamp   
  ---------------------  
   1997-10-22 00:00:00  
  (1 row)
* hextoraw(raw)

描述：将一个十六进制构成的字符串转换为raw类型。

返回值类型：raw

示例：

* SELECT hextoraw('7D');  
   hextoraw   
  ----------  
   7D  
  (1 row)
* numtoday(numeric)

描述：将数字类型的值转换为指定格式的时间戳。

返回值类型：timestamp

示例：

* SELECT numtoday(2);  
   numtoday  
  ----------  
   2 days  
  (1 row)
* pg\_systimestamp()

描述：获取系统时间戳。

返回值类型：timestamp with time zone

示例：

* SELECT pg\_systimestamp();  
   pg\_systimestamp  
  -------------------------------  
   2022-08-02 11:57:31.266877+08  
  (1 row)
* rawtohex(string)

描述：将一个二进制构成的字符串转换为十六进制的字符串。

结果为输入字符的ACSII码，以十六进制表示。

返回值类型：varchar

示例：

* SELECT rawtohex('1234567');  
   rawtohex   
  ----------------  
   31323334353637  
  (1 row)
* to\_bigint(varchar)

描述：将字符类型转换为bigint类型。

返回值类型：bigint

示例：

* SELECT to\_bigint('123364545554455');  
   to\_bigint   
  ----------------  
   123364545554455  
  (1 row)
* to\_char(datetime/interval [, fmt])
* 描述：将一个DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE或者TIMESTAMP WITH LOCAL TIME ZONE类型的DATETIME或者INTERVAL值按照fmt指定的格式转换为VARCHAR类型。
  + 可选参数fmt可以为以下几类：日期、时间、星期、季度和世纪。每类都可以有不同的模板，模板之间可以合理组合，常见的模板有：HH、MI、SS、YYYY、MM、DD。
  + 模板可以有修饰词，常用的修饰词是FM，可以用来抑制前导的零或尾随的空白。
* 返回值类型：varchar

示例：

* SELECT to\_char(current\_timestamp,'HH12:MI:SS');  
   to\_char   
  ----------  
   10:19:26  
  (1 row)
* SELECT to\_char(current\_timestamp,'FMHH12:FMMI:FMSS');  
   to\_char   
  ----------  
   10:19:46  
  (1 row)
* to\_char(double precision/real, text)

描述：将浮点类型的值转换为指定格式的字符串。

返回值类型：text

示例：

* SELECT to\_char(125.8::real, '999D99');  
   to\_char   
  ---------  
   125.80  
  (1 row)
* to\_char(numeric/smallint/integer/bigint/double precision/real[, fmt])
* 描述：将一个整型或者浮点类型的值转换为指定格式的字符串。
  + 可选参数fmt可以为以下几类：十进制字符、“分组”符、正负号和货币符号，每类都可以有不同的模板，模板之间可以合理组合，常见的模板有：9、0、,（千分隔符）、.（小数点）。
  + 模板可以有类似FM的修饰词，但FM不抑制由模板0指定而输出的0。
  + 要将整型类型的值转换成对应16进制值的字符串，使用模板X或x。
* 返回值类型：varchar

示例：

* SELECT to\_char(1485,'9,999');  
   to\_char   
  ---------  
   1,485  
  (1 row)
* SELECT to\_char( 1148.5,'9,999.999');  
   to\_char  
  ----------  
   1,148.5  
  (1 row)
* SELECT to\_char(148.5,'990999.909');  
    
   to\_char  
  -----------  
   0148.5  
  (1 row)
* SELECT to\_char(123,'XXX');  
   to\_char   
  ---------  
   7B  
  (1 row)
* to\_char(interval, text)

描述：将时间间隔类型的值转换为指定格式的字符串。

返回值类型：text

示例：

* SELECT to\_char(interval '15h 2m 12s', 'HH24:MI:SS');  
   to\_char  
  ----------  
   15:02:12  
  (1 row)
* to\_char(int, text)

描述：将整数类型的值转换为指定格式的字符串。

返回值类型：text

示例：

* SELECT to\_char(125, '999');  
   to\_char  
  ---------  
   125  
  (1 row)
* to\_char(numeric, text)

描述：将数字类型的值转换为指定格式的字符串。

返回值类型：text

示例：

* SELECT to\_char(-125.8, '999D99S');  
   to\_char  
  ---------  
   125.8  
  (1 row)
* to\_char(string)

描述：将CHAR、VARCHAR、VARCHAR2、CLOB类型转换为VARCHAR类型。

如使用该函数对CLOB类型进行转换，且待转换CLOB类型的值超出目标类型的范围，则返回错误。

返回值类型：varchar

示例：

* SELECT to\_char('01110');  
   to\_char  
  ---------  
   01110  
  (1 row)
* to\_char(timestamp, text)

描述：将时间戳类型的值转换为指定格式的字符串。

返回值类型：text

示例：

* SELECT to\_char(current\_timestamp, 'HH12:MI:SS');  
   to\_char  
  ----------  
   10:55:59  
  (1 row)
* to\_clob(char/nchar/varchar/varchar2/nvarchar/nvarchar2/text/raw)

描述：将RAW类型或者文本字符集类型CHAR、NCHAR、VARCHAR、VARCHAR2、NVARCHAR、NVARCHAR2、TEXT转成CLOB类型。

返回值类型：clob

示例：

* SELECT to\_clob('ABCDEF'::RAW(10));  
   to\_clob   
  ---------  
   ABCDEF  
  (1 row)
* SELECT to\_clob('hello111'::CHAR(15));  
   to\_clob   
  ----------  
   hello111  
  (1 row)
* SELECT to\_clob('gauss123'::NCHAR(10));  
   to\_clob   
  ----------  
   gauss123  
  (1 row)
* SELECT to\_clob('gauss234'::VARCHAR(10));  
   to\_clob   
  ----------  
   gauss234  
  (1 row)
* SELECT to\_clob('gauss345'::VARCHAR2(10));  
   to\_clob   
  ----------  
   gauss345  
  (1 row)
* SELECT to\_clob('gauss456'::NVARCHAR2(10));  
   to\_clob   
  ----------  
   gauss456  
  (1 row)
* SELECT to\_clob('World222!'::TEXT);  
   to\_clob   
  -----------  
   World222!  
  (1 row)
* to\_date(text)
* 描述：将文本类型的值转换为指定格式的时间戳。目前只支持两类格式。
  + 格式一：无分隔符日期，如20150814，需要包括完整的年月日。
  + 格式二：带分隔符日期，如2014-08-14，分隔符可以是单个任意非数字字符。
* 返回值类型：timestamp without time zone

示例：

* SELECT to\_date('2015-08-14');  
   to\_date  
  ---------------------  
   2015-08-14 00:00:00  
  (1 row)
* to\_date(text, text)

描述：将字符串类型的值转换为指定格式的日期。

返回值类型：timestamp without time zone

示例：

* SELECT to\_date('05 Dec 2000', 'DD Mon YYYY');  
   to\_date  
  ---------------------  
   2000-12-05 00:00:00  
  (1 row)
* to\_number ( expr [, fmt])

描述：将expr按指定格式转换为一个NUMBER类型的值。

类型转换格式请参考表1。

转换十六进制字符串为十进制数字时，最多支持16个字节的十六进制字符串转换为无符号数。

转换十六进制字符串为十进制数字时，格式字符串中不允许出现除'x'或'X'以外的其他字符，否则报错。

返回值类型：number

示例：

* SELECT to\_number('12,454.8-', '99G999D9S');  
   to\_number   
  -----------  
   -12454.8  
  (1 row)
* to\_number(text, text)

描述：将字符串类型的值转换为指定格式的数字。

返回值类型：numeric

示例：

* SELECT to\_number('12,454.8-', '99G999D9S');  
   to\_number  
  -----------  
   -12454.8  
  (1 row)
* to\_timestamp(double precision)

描述：把UNIX纪元转换成时间戳。

返回值类型：timestamp with time zone

示例：

* SELECT to\_timestamp(1284352323);  
   to\_timestamp   
  ------------------------  
   2010-09-13 12:32:03+08  
  (1 row)
* to\_timestamp(string [,fmt])
* 描述：将字符串string按fmt指定的格式转换成时间戳类型的值。不指定fmt时，按参数nls\_timestamp\_format所指定的格式转换。
* openGauss的to\_timestamp中，
  + 如果输入的年份YYYY=0，系统报错。
  + 如果输入的年份YYYY<0，在fmt中指定SYYYY，则正确输出公元前绝对值n的年份。

fmt中出现的字符必须与日期/时间格式化的模式相匹配，否则报错。

返回值类型：timestamp without time zone

示例：

* SHOW nls\_timestamp\_format;  
   nls\_timestamp\_format   
  ----------------------------  
   DD-Mon-YYYY HH:MI:SS.FF AM  
  (1 row)  
    
  SELECT to\_timestamp('12-sep-2014');  
   to\_timestamp   
  ---------------------  
   2014-09-12 00:00:00  
  (1 row)
* SELECT to\_timestamp('12-Sep-10 14:10:10.123000','DD-Mon-YY HH24:MI:SS.FF');  
   to\_timestamp   
  -------------------------  
   2010-09-12 14:10:10.123  
  (1 row)
* SELECT to\_timestamp('-1','SYYYY');  
   to\_timestamp   
  ------------------------  
   0001-01-01 00:00:00 BC  
  (1 row)
* SELECT to\_timestamp('98','RR');  
   to\_timestamp   
  ---------------------  
   1998-01-01 00:00:00  
  (1 row)
* SELECT to\_timestamp('01','RR');  
   to\_timestamp   
  ---------------------  
   2001-01-01 00:00:00  
  (1 row)
* to\_timestamp(text, text)

描述：将字符串类型的值转换为指定格式的时间戳。

返回值类型：timestamp

示例：

* SELECT to\_timestamp('05 Dec 2000', 'DD Mon YYYY');  
   to\_timestamp  
  ---------------------  
   2000-12-05 00:00:00  
  (1 row)

**表 1** 数值格式化的模版模式

| **模式** | **描述** |
| --- | --- |
| 9 | 带有指定数值位数的值 |
| 0 | 带前导零的值 |
| .（句点） | 小数点 |
| ,（逗号） | 分组（千）分隔符 |
| PR | 尖括号内负值 |
| S | 带符号的数值（使用区域设置） |
| L | 货币符号（使用区域设置） |
| D | 小数点（使用区域设置） |
| G | 分组分隔符（使用区域设置） |
| MI | 在指明的位置的负号（如果数字 < 0） |
| PL | 在指明的位置的正号（如果数字 > 0） |
| SG | 在指明的位置的正/负号 |
| RN | 罗马数字（输入在 1 和 3999 之间） |
| TH或th | 序数后缀 |
| V | 移动指定位（小数） |

* abstime\_text
* 描述：将abstime类型转为text类型输出。
* 参数：abstime
* 返回值类型：text
* abstime\_to\_smalldatetime
* 描述：将abstime类型转为smalldatatime类型。
* 参数：abstime
* 返回值类型：smalldatetime
* bigint\_tid
* 描述：将bigint转为tid。
* 参数：bigint
* 返回值类型：tid
* bool\_int1
* 描述：将bool转为int1。
* 参数：boolean
* 返回值类型：tinyint
* bool\_int2
* 描述：将bool转为int2。
* 参数：boolean
* 返回值类型：smallint
* bool\_int8
* 描述：将bool转为int8。
* 参数：boolean
* 返回值类型：bigint
* bpchar\_date
* 描述：将字符串转为日期。
* 参数：character
* 返回值类型：date
* bpchar\_float4
* 描述：将字符串转为float4。
* 参数：character
* 返回值类型：real
* bpchar\_float8
* 描述：将字符串转为float8。
* 参数：character
* 返回值类型：double precision
* bpchar\_int4
* 描述：将字符串转为int4。
* 参数：character
* 返回值类型：integer
* bpchar\_int8
* 描述：将字符串转为int8。
* 参数：character
* 返回值类型：bigint
* bpchar\_numeric
* 描述：将字符串转为numeric。
* 参数：character
* 返回值类型：numeric
* bpchar\_timestamp
* 描述：将字符串转为时间戳。
* 参数：character
* 返回值类型：timestamp without time zone
* bpchar\_to\_smalldatetime
* 描述：将字符串转为smalldatetime。
* 参数：character
* 返回值类型：smalldatetime
* cupointer\_bigint
* 描述：将列存CU指针类型转为bigint类型。
* 参数：text
* 返回值类型：bigint
* date\_bpchar
* 描述：将date类型转换为bpchar类型。
* 参数：date
* 返回值类型：character
* date\_text
* 描述：将date类型转换为text类型。
* 参数：date
* 返回值类型：text
* date\_varchar
* 描述：将date类型转换为varchar类型。
* 参数：date
* 返回值类型：character varying
* f4toi1
* 描述：把float4类型强转为uint8类型。
* 参数：real
* 返回值类型：tinyint
* f8toi1
* 描述：把float8类型强转为uint8类型。
* 参数：double precision
* 返回值类型：tinyint
* float4\_bpchar
* 描述：float4转换为bpchar。
* 参数：real
* 返回值类型：character
* float4\_text
* 描述：float4转换为text。
* 参数：real
* 返回值类型：text
* float4\_varchar
* 描述：float4转换为varchar。
* 参数：real
* 返回值类型：character varying
* float8\_bpchar
* 描述：float8转换为bpchar。
* 参数：double precision
* 返回值类型：character
* float8\_interval
* 描述：float8转换为interval。
* 参数：double precision
* 返回值类型：interval
* float8\_text
* 描述：float8转换为text。
* 参数：double precision
* 返回值类型：text
* float8\_varchar
* 描述：float8转换为varchar。
* 参数：double precision
* 返回值类型：character varying
* i1tof4
* 描述：uint8转换为float4。
* 参数：tinyint
* 返回值类型：real
* i1tof8
* 描述：uint8转换为float8。
* 参数：tinyint
* 返回值类型：double precision
* i1toi2
* 描述：uint8转换为int16。
* 参数：tinyint
* 返回值类型：smallint
* i1toi4
* 描述：uint8转换为int32。
* 参数：tinyint
* 返回值类型：integer
* i1toi8
* 描述：uint8转换为int64。
* 参数：tinyint
* 返回值类型：bigint
* i2toi1
* 描述：int16转换为uint8。
* 参数：smallint
* 返回值类型：tinyint
* i4toi1
* 描述：int32转换为uint8。
* 参数：integer
* 返回值类型：tinyint
* i8toi1
* 描述：int64转换为uint8。
* 参数：bigint
* 返回值类型：tinyint
* int1\_avg\_accum
* 描述：将第二个uint8类型参数，加入到第一个参数中，一个参数为bigint类型数组。
* 参数：bigint[], tinyint
* 返回值类型：bigint[]
* int1\_bool
* 描述：uint8转换为bool。
* 参数：tinyint
* 返回值类型：boolean
* int1\_bpchar
* 描述：uint8转换为bpchar。
* 参数：tinyint
* 返回值类型：character
* int1\_mul\_cash
* 描述：返回一个int8类型参数和一个cash类型参数的乘积，返回值为cash类型。
* 参数：tinyint, money
* 返回值类型：money
* int1\_numeric
* 描述：uint8转换为numeric。
* 参数：tinyint
* 返回值类型：numeric
* int1\_nvarchar2
* 描述：uint8转换为nvarchar2。
* 参数：tinyint
* 返回值类型：nvarchar2
* int1\_text
* 描述：uint8转换为text。
* 参数：tinyint
* 返回值类型：text
* int1\_varchar
* 描述：uint8转换为varchar。
* 参数：tinyint
* 返回值类型：character varying
* int1in
* 描述：字符串转化为无符号一字节整数。
* 参数：cstring
* 返回值类型：tinyint
* int1out
* 描述：无符号一字节整数转化为字符串。
* 参数：tinyint
* 返回值类型：cstring
* int1up
* 描述：输入整数转化为无符号一字节整数。
* 参数：tinyint
* 返回值类型：tinyint
* int2\_bool
* 描述：将有符号二字节整数转化为bool型。
* 参数：smallint
* 返回值类型：boolean
* int2\_bpchar
* 描述：将有符号二字节整数转化为BpChar。
* 参数：smallint
* 返回值类型：character
* int2\_text
* 描述：有符号二字节整数转化为text类型。
* 参数：smallint
* 返回值类型：text
* int2\_varchar
* 描述：有符号二字节整数转化为varchar类型。
* 参数：smallint
* 返回值类型：character varying
* int8\_text
* 描述：有符号八字节整数转化为text类型。
* 参数：bigint
* 返回值类型：text
* int8\_varchar
* 描述：有符号八字节整数转化为varchar。
* 参数：bigint
* 返回值类型：character varying
* intervaltonum
* 描述：将内部数据类型日期转化为numeric类型。
* 参数：interval
* 返回值类型：numeric
* numeric\_bpchar
* 描述：numeric转化为bpchar。
* 参数：numeric
* 返回值类型：character
* numeric\_int1
* 描述：numeric转化为有符号1字节整数。
* 参数：numeric
* 返回值类型：tinyint
* numeric\_text
* 描述：numeric转化为text。
* 参数：numeric
* 返回值类型：text
* numeric\_varchar
* 描述：numeric转化为varchar。
* 参数：numeric
* 返回值类型：character varying
* nvarchar2in
* 描述：将c字符串转化为varchar。
* 参数：cstring, oid, integer
* 返回值类型：nvarchar2
* nvarchar2out
* 描述：将text转化为c字符串。
* 参数：nvarchar2
* 返回值类型：cstring
* nvarchar2send
* 描述：将varchar转化为二进制。
* 参数：nvarchar2
* 返回值类型：bytea
* oidvectorin\_extend
* 描述：将字符串转化为oidvector。
* 参数：cstring
* 返回值类型：oidvector\_extend
* oidvectorout\_extend
* 描述：将oidvector转化为字符串。
* 参数：oidvector\_extend
* 返回值类型：cstring
* oidvectorsend\_extend
* 描述：将oidvector转化为字符串。
* 参数：oidvector\_extend
* 返回值类型：bytea
* reltime\_text
* 描述：reltime转换为text。
* 参数：reltime
* 返回值类型：text
* text\_date
* 描述：text类型转换为date类型。
* 参数：text
* 返回值类型：date
* text\_float4
* 描述：text类型转换为float4类型。
* 参数：text
* 返回值类型：real
* text\_float8
* 描述：text类型转换为float8类型。
* 参数：text
* 返回值类型：double precision
* text\_int1
* 描述：text类型转换为int1类型。
* 参数：text
* 返回值类型：tinyint
* text\_int2
* 描述：text类型转换为int2类型。
* 参数：text
* 返回值类型：smallint
* text\_int4
* 描述：text类型转换为int4类型。
* 参数：text
* 返回值类型：integer
* text\_int8
* 描述：text类型转换为int8类型。
* 参数：text
* 返回值类型：bigint
* text\_numeric
* 描述：text类型转换为numeric类型。
* 参数：text
* 返回值类型：numeric
* text\_timestamp
* 描述：text类型转换为timestamp类型。
* 参数：text
* 返回值类型：timestamp without time zone
* time\_text
* 描述：time类型转换为text类型。
* 参数：time without time zone
* 返回值类型：text
* timestamp\_text
* 描述：timestamp类型转换为text类型。
* 参数：timestamp without time zone
* 返回值类型：text
* timestamp\_to\_smalldatetime
* 描述：timestamp类型转换为smalldatetime类型。
* 参数：timestamp without time zone
* 返回值类型：smalldatetime
* timestamp\_varchar
* 描述：timestamp类型转换为varchar类型。
* 参数：timestamp without time zone
* 返回值类型：character varying
* timestamptz\_to\_smalldatetime
* 描述：timestamptz类型转换为smalldatetime。
* 参数：timestamp with time zone
* 返回值类型：smalldatetime
* timestampzone\_text
* 描述：timestampzone类型转换为text类型。
* 参数：timestamp with time zone
* 返回值类型：text
* timetz\_text
* 描述：timetz类型转换为text类型。
* 参数：time with time zone
* 返回值类型：text
* to\_integer
* 描述：转换为integer类型。
* 参数：character varying
* 返回值类型：integer
* to\_interval
* 描述：转换为interval类型。
* 参数：character varying
* 返回值类型：interval
* to\_numeric
* 描述：转换为numeric类型。
* 参数：character varying
* 返回值类型：numeric
* to\_nvarchar2
* 描述：转换为nvarchar2类型。
* 参数：numeric
* 返回值类型：nvarchar2
* to\_text
* 描述：转换为text类型。
* 参数：smallint
* 返回值类型：text
* to\_ts
* 描述：转换为ts类型。
* 参数：character varying
* 返回值类型：timestamp without time zone
* to\_varchar2
* 描述：转换为varchar2类型。
* 参数：timestamp without time zone
* 返回值类型：character varying
* varchar\_date
* 描述：varchar类型转换为date。
* 参数：character varying
* 返回值类型：date
* varchar\_float4
* 描述：varchar类型转换为float4。
* 参数：character varying
* 返回值类型：real
* varchar\_float8
* 描述：varchar类型转换为float8。
* 参数：character varying
* 返回值类型：double precision
* varchar\_int4
* 描述：varchar类型转换为int4。
* 参数：character varying
* 返回值类型：integer
* varchar\_int8
* 描述：varchar类型转换为int8。
* 参数：character varying
* 返回值类型：bigint
* varchar\_numeric
* 描述：varchar类型转换为numeric。
* 参数：character varying
* 返回值类型：numeric
* varchar\_timestamp
* 描述：varchar类型转换为timestamp。
* 参数：character varying
* 返回值类型：timestamp without time zone
* varchar2\_to\_smlldatetime
* 描述：varchar2类型转换为smlldatetime。
* 参数：character varying
* 返回值类型：smalldatetime
* xidout4
* 描述：xid输出为4字节数字。
* 参数：xid32
* 返回值类型：cstring
* xidsend4
* 描述：xid转换为二进制格式。
* 参数：xid32
* 返回值类型：bytea

**编码类型转换**

* convert\_to\_nocase(text, text)

描述：将字符串转换为指定的编码类型。

返回值类型：bytea

示例：

* SELECT convert\_to\_nocase('12345', 'GBK');  
   convert\_to\_nocase   
  -------------------  
   x3132333435  
  (1 row)

#### 数字操作函数和操作符

**数字操作符**

* +
* 描述：加
* 示例：
* SELECT 2+3 AS RESULT;
* result
* --------
* 5
* (1 row)
* -
* 描述：减
* 示例：
* SELECT 2-3 AS RESULT;
* result
* --------
* -1
* (1 row)
* \*
* 描述：乘
* 示例：
* SELECT 2\*3 AS RESULT;
* result
* --------
* 6
* (1 row)
* /
* 描述：除（除法操作符不会取整）
* 示例：
* SELECT 4/2 AS RESULT;
* result
* --------
* 2
* (1 row)
* SELECT 4/3 AS RESULT;
* result
* ------------------
* 1.33333333333333
* (1 row)
* +/-
* 描述：正/负
* 示例：
* SELECT -2 AS RESULT;
* result
* --------
* -2
* (1 row)
* %
* 描述：模（求余）
* 示例：
* SELECT 5%4 AS RESULT;
* result
* --------
* 1
* (1 row)
* @
* 描述：绝对值
* 示例：
* SELECT @ -5.0 AS RESULT;
* result
* --------
* 5.0
* (1 row)
* ^
* 描述：幂（指数运算）
* 示例：
* SELECT 2.0^3.0 AS RESULT;
* result
* --------------------
* 8.0000000000000000
* (1 row)
* |/
* 描述：平方根
* 示例：
* SELECT |/ 25.0 AS RESULT;
* result
* --------
* 5
* (1 row)
* ||/
* 描述：立方根
* 示例：
* SELECT ||/ 27.0 AS RESULT;
* result
* --------
* 3
* (1 row)
* !
* 描述：阶乘
* 示例：
* SELECT 5! AS RESULT;
* result
* --------
* 120
* (1 row)
* !!
* 描述：阶乘（前缀操作符）
* 示例：
* SELECT !!5 AS RESULT;
* result
* --------
* 120
* (1 row)
* &
* 描述：二进制AND
* 示例：
* SELECT 91&15 AS RESULT;
* result
* --------
* 11
* (1 row)
* |
* 描述：二进制OR
* 示例：
* SELECT 32|3 AS RESULT;
* result
* --------
* 35
* (1 row)
* #
* 描述：二进制XOR
* 示例：
* SELECT 17#5 AS RESULT;
* result
* --------
* 20
* (1 row)
* ~
* 描述：二进制NOT
* 示例：
* SELECT ~1 AS RESULT;
* result
* --------
* -2
* (1 row)
* <<
* 描述：二进制左移
* 示例：
* SELECT 1<<4 AS RESULT;
* result
* --------
* 16
* (1 row)
* >>
* 描述：二进制右移
* 示例：

SELECT 8>>2 AS RESULT;

result

--------

2

(1 row)

**数字操作函数**

* abs(x)
* 描述：绝对值。
* 返回值类型：和输入相同。
* 示例：
* SELECT abs(-17.4);
* abs
* ------
* 17.4
* (1 row)
* acos(x)
* 描述：反余弦。
* 返回值类型：double precision
* 示例：
* SELECT acos(-1);
* acos
* ------------------
* 3.14159265358979
* (1 row)
* asin(x)
* 描述：反正弦。
* 返回值类型：double precision
* 示例：
* SELECT asin(0.5);
* asin
* ------------------
* .523598775598299
* (1 row)
* atan(x)
* 描述：反正切。
* 返回值类型：double precision
* 示例：
* SELECT atan(1);
* atan
* ------------------
* .785398163397448
* (1 row)
* atan2(y, x)
* 描述：y/x的反正切。
* 返回值类型：double precision
* 示例：
* SELECT atan2(2, 1);
* atan2
* ------------------
* 1.10714871779409
* (1 row)
* bitand(integer, integer)
* 描述：计算两个数字与运算(&)的结果。
* 返回值类型：bigint类型数字。
* 示例：
* SELECT bitand(127, 63);
* bitand
* --------
* 63
* (1 row)
* cbrt(dp)
* 描述：立方根。
* 返回值类型：double precision
* 示例：
* SELECT cbrt(27.0);
* cbrt
* ------
* 3
* (1 row)
* ceil(x)
* 描述：不小于参数的最小的整数。
* 返回值类型：整数。
* 示例：
* SELECT ceil(-42.8);
* ceil
* ------
* -42
* (1 row)
* ceiling(dp or numeric)
* 描述：不小于参数的最小整数（ceil的别名）。
* 返回值类型：dp or numeric，不考虑隐式类型转换的情况下与输入相同。
* 示例：
* SELECT ceiling(-95.3);
* ceiling
* ---------
* -95
* (1 row)
* cos(x)
* 描述：余弦。
* 返回值类型：double precision
* 示例：
* SELECT cos(-3.1415927);
* cos
* -------------------
* -.999999999999999
* (1 row)
* cot(x)
* 描述：余切。
* 返回值类型：double precision
* 示例：
* SELECT cot(1);
* cot
* ------------------
* .642092615934331
* (1 row)
* degrees(dp)
* 描述：把弧度转为角度。
* 返回值类型：double precision
* 示例：
* SELECT degrees(0.5);
* degrees
* ------------------
* 28.6478897565412
* (1 row)
* div(y numeric, x numeric)
* 描述：y除以x的商的整数部分。
* 返回值类型：numeric
* 示例：
* SELECT div(9,4);
* div
* -----
* 2
* (1 row)
* exp(x)
* 描述：自然指数。
* 返回值类型：dp or numeric，不考虑隐式类型转换的情况下与输入相同。
* 示例：
* SELECT exp(1.0);
* exp
* --------------------
* 2.7182818284590452
* (1 row)
* floor(x)
* 描述：不大于参数的最大整数。
* 返回值类型：与输入相同。
* 示例：
* SELECT floor(-42.8);
* floor
* -------
* -43
* (1 row)
* int1(in)
* 描述：将传入的text参数转换为int1类型值并返回。
* 返回值类型：int1
* 取值范围：0 ~ 255
* 示例：
* select int1('123');
* int1
* ------
* 123
* (1 row)
* int2(in)
* 描述：将传入参数转换为int2类型值并返回。
* 支持的入参类型包括float4、float8、int16、numeric、text。
* 返回值类型：int2
* 取值范围：-32,768 ~ +32,767
* 示例：
* select int2('1234');
* int2
* ------
* 1234
* (1 row)
* select int2(25.3);
* int2
* ------
* 25
* (1 row)
* int4(in)
* 描述：将传入参数转换为int4类型值并返回。
* 支持的入参类型包括bit、boolean、char、duoble precision、int16、numeric、real、smallint、text。
* 返回值类型：int4
* 取值范围：-2,147,483,648 ~ +2,147,483,647
* 示例：
* select int4('789');
* int4
* ------
* 789
* (1 row)
* select int4(99.9);
* int4
* ------
* 100
* (1 row)
* float4(in)
* 描述：将传入参数转换为float4类型值并返回。支持的入参类型包括：bigint，duoble precision，int16, integer, numeric，smallint，text。
* 返回值类型：float4
* 示例：
* select float4('789');
* float4
* --------
* 789
* (1 row)
* select float4(99.9);
* float4
* --------
* 99.9
* (1 row)
* float8(in)
* 描述：将传入参数转换为float8类型值并返回。支持的入参类型包括：bigint，int16, integer, numeric，real，smallint，text。
* 返回值类型：float8
* 示例：
* select float8('789');
* float8
* --------
* 789
* (1 row)
* select float8(99.9);
* float8
* --------
* 99.9
* (1 row)
* int16(in)
* 描述：将传入参数转换为int16类型值并返回。支持的入参类型包括：bigint，boolean，double precision，integer，numeric，oid，real，smallint，tinyint。
* 返回值类型：int16
* 示例：
* select int16('789');
* int16
* --------
* 789
* (1 row)
* select int16(99.9);
* int16
* -------
* 100
* (1 row)
* numeric(in)
* 描述：将传入参数转换为numeric类型值并返回。支持的入参类型包括：bigint，boolean，double precision，int16，integer，money，real，smallint。
* 返回值类型：numeric
* 示例：
* select "numeric"('789');
* numeric
* ---------
* 789
* (1 row)
* select "numeric"(99.9);
* numeric
* ---------
* 99.9
* (1 row)
* oid(in)
* 描述：将传入参数转换为oid类型值并返回。支持的入参类型包括：bigint，int16。
* 返回值类型：oid
* radians(dp)
* 描述：把角度转为弧度。
* 返回值类型：double precision
* 示例：
* SELECT radians(45.0);
* radians
* ------------------
* .785398163397448
* (1 row)
* random()
* 描述：0.0到1.0之间的随机数。
* 返回值类型：double precision
* 示例：
* SELECT random();
* random
* ------------------
* .824823560658842
* (1 row)
* multiply(x double precision or text, y double precision or text)
* 描述：x和y的乘积。
* 返回值类型：double precision
* 示例：
* SELECT multiply(9.0, '3.0');
* multiply
* -------------------
* 27
* (1 row)
* SELECT multiply('9.0', 3.0);
* multiply
* -------------------
* 27
* (1 row)
* ln(x)
* 描述：自然对数。
* 返回值类型：dp or numeric，不考虑隐式类型转换的情况下与输入相同。
* 示例：
* SELECT ln(2.0);
* ln
* -------------------
* .6931471805599453
* (1 row)
* log(x)
* 描述：以10为底的对数。
* 返回值类型：与输入相同。
* 示例：
* SELECT log(100.0);
* log
* -----
* 2
* (1 row)
* log(b numeric, x numeric)
* 描述：以b为底的对数。
* 返回值类型：numeric
* 示例：
* SELECT log(2.0, 64.0);
* log
* -----
* 6
* (1 row)
* mod(x,y)
* 描述：x/y的余数（模）。如果x是0，则返回0。
* 返回值类型：与参数类型相同。
* 示例：
* SELECT mod(9,4);
* mod
* -----
* 1
* (1 row)
* SELECT mod(9,0);
* mod
* -----
* 9
* (1 row)
* pi()
* 描述：“π”常量。
* 返回值类型：double precision
* 示例：
* SELECT pi();
* pi
* ------------------
* 3.14159265358979
* (1 row)
* power(a double precision, b double precision)
* 描述：a的b次幂。
* 返回值类型：double precision
* 示例：
* SELECT power(9.0, 3.0);
* power
* -------
* 729
* (1 row)
* round(x)
* 描述：离输入参数最近的整数。
* 返回值类型：与输入相同。
* 示例：
* SELECT round(42.4);
* round
* -------
* 42
* (1 row)
* SELECT round(42.6);
* round
* -------
* 43
* (1 row)
* round(v numeric, s int)
* 描述：保留小数点后s位，s后一位进行四舍五入。
* 返回值类型：numeric
* 示例：
* SELECT round(42.4382, 2);
* round
* -------
* 42.44
* (1 row)
* setseed(dp)
* 描述：为随后的random()调用设置种子(-1.0到1.0之间，包含)。
* 返回值类型：void
* 示例：
* SELECT setseed(0.54823);
* setseed
* ---------
* (1 row)
* sign(x)
* 描述：输出此参数的符号。
* 返回值类型：-1表示负数，0表示0，1表示正数。
* 示例：
* SELECT sign(-8.4);
* sign
* ------
* -1
* (1 row)
* sin(x)
* 描述：正弦。
* 返回值类型：double precision
* 示例：
* SELECT sin(1.57079);
* sin
* ------------------
* .999999999979986
* (1 row)
* sqrt(x)
* 描述：平方根。
* 返回值类型：dp or numeric，不考虑隐式类型转换的情况下与输入相同。
* 示例：
* SELECT sqrt(2.0);
* sqrt
* -------------------
* 1.414213562373095
* (1 row)
* tan(x)
* 描述：正切。
* 返回值类型：double precision
* 示例：
* SELECT tan(20);
* tan
* ------------------
* 2.23716094422474
* (1 row)
* trunc(x)
* 描述：截断（取整数部分）。
* 返回值类型：与输入相同。
* 示例：
* SELECT trunc(42.8);
* trunc
* -------
* 42
* (1 row)
* trunc(v numeric, s int)
* 描述：截断为s位小数。
* 返回值类型：numeric
* 示例：
* SELECT trunc(42.4382, 2);
* trunc
* -------
* 42.43
* (1 row)
* smgrne(a smgr, b smgr)
* 描述：比较两个smgr类型整数是否不相等。
* 返回值类型：bool
* smgreq(a smgr, b smgr)
* 描述：比较两个smgr类型整数是否相等。
* 返回值类型：bool
* int1abs
* 描述：返回uint8类型数据的绝对值。
* 参数：tinyint
* 返回值类型：tinyint
* int1and
* 描述：返回两个uint8类型数据按位与的结果。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1cmp
* 描述：返回两个uint8类型数据比较的结果，若第一个参数大，则返回1；若第二个参数大，则返回-1；若相等，则返回0。
* 参数：tinyint, tinyint
* 返回值类型：integer
* int1div
* 描述：返回两个uint8类型数据相除的结果，结果为float8类型。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1eq
* 描述：比较两个uint8类型数据是否相等。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1ge
* 描述：判断两个uint8类型数据是否第一个参数大于等于第二个参数。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1gt
* 描述：无符号1字节整数做大于运算。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1larger
* 描述：无符号1字节整数求最大值。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1le
* 描述：无符号1字节整数做小于等于运算。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1lt
* 描述：无符号1字节整数做小于运算。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1smaller
* 描述：无符号1字节整数求最小算。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1inc
* 描述：无符号1字节整数加一。
* 参数：tinyint
* 返回值类型：tinyint
* int1mi
* 描述：无符号一字节整数做差运算。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1mod
* 描述：无符号一字节整数做取余运算。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1mul
* 描述：无符号一字节整数做乘法运算。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1ne
* 描述：无符号一字节整数不等于运算。
* 参数：tinyint, tinyint
* 返回值类型：boolean
* int1pl
* 描述：无符号一字节整数加法。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1um
* 描述：无符号一字节数去相反数并返回有符号二字节整数。
* 参数：tinyint
* 返回值类型：smallint
* int1xor
* 描述：无符号一字节整数异或操作。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* cash\_div\_int1
* 描述：对money类型进行除法运算。
* 参数：money, tinyint
* 返回值类型：money
* cash\_mul\_int1
* 描述：对money类型进行乘法运算。
* 参数：money, tinyint
* 返回值类型：money
* int1not
* 描述：无符号一字节整数二进制位翻转。
* 参数：tinyint
* 返回值类型：tinyint
* int1or
* 描述：无符号一字节整数或运算。
* 参数：tinyint, tinyint
* 返回值类型：tinyint
* int1shl
* 描述：无符号一字节整数左移指定位数。
* 参数：tinyint, integer
* 返回值类型：tinyint
* width\_bucket(op numeric, b1 numeric, b2 numeric, count int)
* 描述：返回一个桶，这个桶是在一个有count个桶，上界为b1下界为b2的等深柱图中operand将被赋予的那个桶。
* 返回值类型：int
* 示例：
* SELECT width\_bucket(5.35, 0.024, 10.06, 5);
* width\_bucket
* --------------
* 3
* (1 row)
* width\_bucket(op dp, b1 dp, b2 dp, count int)
* 描述：返回一个桶，这个桶是在一个有count个桶，上界为b1下界为b2的等深柱图中operand将被赋予的那个桶。
* 返回值类型：int
* 示例：

SELECT width\_bucket(5.35, 0.024, 10.06, 5);

width\_bucket

--------------

3

(1 row)

#### 系统信息函数

**会话信息函数**

* current\_catalog
* 描述：当前数据库的名称（在标准SQL中称“catalog”）。
* 返回值类型：name

示例：

vastbase=# SELECT current\_catalog;

current\_database

------------------

vastbase

(1 row)

* current\_database()
* 描述：当前数据库的名称。
* 返回值类型：name

示例：

vastbase=# SELECT current\_database();

current\_database

------------------

vastbase

(1 row)

* current\_query()
* 描述：由客户端提交的当前执行语句（可能包含多个声明）。
* 返回值类型：text

示例：

vastbase=# SELECT current\_query();

current\_query

-------------------------

SELECT current\_query();

(1 row)

* current\_schema[()]
* 描述：当前模式的名称。
* 返回值类型：name

示例：

vastbase=# SELECT current\_schema();

current\_schema

----------------

public

(1 row)

备注：current\_schema返回在搜索路径中第一个顺位有效的模式名。（如果搜索路径为空则返回NULL，没有有效的模式名也返回NULL）。如果创建表或者其他命名对象时没有声明目标模式，则将使用这些对象的模式。

* current\_schemas(Boolean)
* 描述：搜索路径中的模式名称。
* 返回值类型：name[]

示例：

vastbase=# SELECT current\_schemas(true);

current\_schemas

---------------------

{pg\_catalog,public}

(1 row)

备注：

current\_schemas(Boolean)返回搜索路径中所有模式名称的数组。布尔选项决定像pg\_catalog这样隐含包含的系统模式是否包含在返回的搜索路径中。

搜索路径可以通过运行时设置更改。命令是：

SET search\_path TO schema [, schema, ...]

* current\_user
* 描述：当前执行环境下的用户名。
* 返回值类型：name

示例：

vastbase=# SELECT current\_user;

current\_user

--------------

vastbase

(1 row)

备注：current\_user是用于权限检查的用户标识。通常，他表示会话用户，但是可以通过参考：开发者指南->SQL语法参考->SQL语法章节改变他。在函数执行的过程中随着属性SECURITY DEFINER的改变，其值也会改变。

* definer\_current\_user
* 描述：当前执行环境下的用户名。
* 返回值类型：name

示例：

vastbase=# SELECT definer\_current\_user();

definer\_current\_user

----------------------

vastbase

(1 row)

* pg\_current\_sessionid()
* 描述：当前执行环境下的会话ID。
* 返回值类型：text

示例：

vastbase=# SELECT pg\_current\_sessionid();

pg\_current\_sessionid

----------------------------

1659507037.139786426119936

(1 row)

备注：pg\_current\_sessionid()是用于获取当前执行环境下的会话ID。其 组成结构为：时间戳.会话ID，当线程池模式开启 （enable\_thread\_pool=on）时，会话ID为SessionID；而线程池模式关 闭时，会话ID为ThreadID。

* pg\_current\_sessid
* 描述：当前执行环境下的会话ID。
* 返回值类型：text

示例：

vastbase=# select pg\_current\_sessid();

pg\_current\_sessid

-------------------

140308875015936

(1 row)

备注：在线程池模式下获得当前会话的会话ID，非线程池模式下获得当 前会话对应的后台线程ID。

* pg\_current\_userid
* 描述：当前用户ID。
* 返回值类型：text

vastbase=# SELECT pg\_current\_userid();

pg\_current\_userid

-------------------

10

(1 row)

* working\_version\_num()
* 描述：版本序号信息。返回一个系统兼容性有关的版本序号。
* 返回值类型：int

示例：

vastbase=# SELECT working\_version\_num();

working\_version\_num

---------------------

92231

(1 row)

* tablespace\_oid\_name()
* 描述: 根据表空间oid，查找表空间名称。
* 返回值类型：text

示例：

vastbase=# select tablespace\_oid\_name(1663);

tablespace\_oid\_name

---------------------

pg\_default

(1 row)

* inet\_client\_addr()
* 描述：连接的远端地址。inet\_client\_addr返回当前客户端的IP地址。

fig: **说明：**

此函数只有在远程连接模式下有效。

返回值类型：inet

示例：

vastbase=# SELECT inet\_client\_addr();

inet\_client\_addr

------------------

(1 row)

* inet\_client\_port()
* 描述：连接的远端端口。inet\_client\_port返回当前客户端的端口号。

fig: **说明：**

此函数只有在远程连接模式下有效。

返回值类型：int

示例：

vastbase=# SELECT inet\_client\_port();

inet\_client\_port

------------------

33143

(1 row)

* inet\_server\_addr()
* 描述：连接的本地地址。inet\_server\_addr返回服务器接收当前连接用的IP地址。

fig: **说明：**

此函数只有在远程连接模式下有效。

返回值类型：inet

示例：

vastbase=# SELECT inet\_server\_addr();

inet\_server\_addr

------------------

10.10.0.13

(1 row)

* inet\_server\_port()
* 描述：连接的本地端口。inet\_server\_port返回接收当前连接的端口号。如果是通过Unix-domain socket连接的，则所有这些函数都返回NULL。

fig: **说明：**

此函数只有在远程连接模式下有效。

返回值类型：int

示例：

vastbase=# SELECT inet\_server\_port();

inet\_server\_port

------------------

(1 row)

* pg\_backend\_pid()
* 描述：当前会话连接的服务进程的进程ID。
* 返回值类型：int

示例：

vastbase=# SELECT pg\_backend\_pid();

pg\_backend\_pid

-----------------

140229352617744

(1 row)

* pg\_conf\_load\_time()
* 描述：配置加载时间。pg\_conf\_load\_time返回最后加载服务器配置文件的时间戳。
* 返回值类型：timestamp with time zone

示例：

vastbase=# SELECT pg\_conf\_load\_time();

pg\_conf\_load\_time

-----------------------------

2022-08-03 10:27:43.7626+08

(1 row)

* pg\_my\_temp\_schema()
* 描述：会话的临时模式的OID，不存在则为0。

返回值类型：oid

示例：

vastbase=# SELECT pg\_my\_temp\_schema();

pg\_my\_temp\_schema

-------------------

0

(1 row)

备注：pg\_my\_temp\_schema返回当前会话中临时模式的OID，如果不存 在（没有创建临时表）的话则返回0。如果给定的OID是其它会话中临 时模式的OID，pg\_is\_other\_temp\_schema则返回true。

* pg\_is\_other\_temp\_schema(oid)
* 描述：是否为另一个会话的临时模式。
* 返回值类型：Boolean

示例：

vastbase=# SELECT pg\_is\_other\_temp\_schema(25356);

pg\_is\_other\_temp\_schema

-------------------------

f

(1 row)

* pg\_listening\_channels()
* 描述：会话正在侦听的信道名称。
* 返回值类型：setof text

示例：

vastbase=# SELECT pg\_listening\_channels();

pg\_listening\_channels

-----------------------

(0 rows)

备注：pg\_listening\_channels返回当前会话正在侦听的一组信道名称。

* pg\_postmaster\_start\_time()
* 描述：服务器启动时间。pg\_postmaster\_start\_time返回服务器启动时的timestamp with time zone。
* 返回值类型：timestamp with time zone

示例：

vastbase=# SELECT pg\_postmaster\_start\_time();

pg\_postmaster\_start\_time

------------------------------

2022-08-03 10:27:44.54975+08

(1 row)

* pg\_get\_ruledef(rule\_oid)
* 描述：获取规则的CREATE RULE命令。
* 返回值类型：text

示例：

vastbase=# select \* from pg\_get\_ruledef(24828);

pg\_get\_ruledef

-------------------------------------------------------------------

CREATE RULE t1\_ins AS ON INSERT TO t1 DO INSTEAD INSERT INTO t2 (id) VALUES (new.id);

(1 row)

* sessionid2pid()
* 描述: 从sessionid中得到pid信息（例如，gs\_session\_stat中sessid列）。
* 返回值类型: int8

示例：

vastbase=# select sessionid2pid(sessid::cstring) from gs\_session\_stat limit 2;

sessionid2pid

-----------------

139973107902208

139973107902208

(2 rows)

* pg\_trigger\_depth()
* 描述：触发器的嵌套层次。
* 返回值类型：int

示例：

vastbase=# SELECT pg\_trigger\_depth();

pg\_trigger\_depth

------------------

0

(1 row)

* session\_user
* 描述：会话用户名。
* 返回值类型：name

示例：

vastbase=# SELECT session\_user;

session\_user

--------------

omm

(1 row)

备注：session\_user通常是连接当前数据库的初始用户，不过系统管理 员可以用开发者指南->SQL语法参考->SQL语法->SET-SESSION- AUTHORIZATION章节修改这个设置。

* user
* 描述：等价于current\_user。
* 返回值类型：name

示例：

vastbase=# SELECT user;

current\_user

--------------

vastbase

(1 row)

* getpgusername()
* 描述：获取数据库用户名。
* 返回值类型：name

示例：

vastbase=# select getpgusername();

getpgusername

---------------

vastbase

(1 row)

* getdatabaseencoding()
* 描述：获取数据库编码方式。
* 返回值类型：name

示例：

vastbase=# select getdatabaseencoding();

getdatabaseencoding

---------------------

UTF8

(1 row)

* version()
* 描述：版本信息。version返回一个描述服务器版本信息的字符串。
* 返回值类型：text

示例：

vastbase=# select version();

version

-----------------------------------------------------------------------------------------------------------------------------------------

--------------------

(Vastbase G100 V2.2 (Build 9) Alpha) compiled at 2022-06-10 02:16:40 commit 7387 last mr on x86\_64-unknown-linux-gnu, compiled by g++

(GCC) 7.3.0, 64-bit

(1 row)

* gs\_deployment()
* 描述：当前系统的部署形态信息。
* 返回值类型：text

示例：

vastbase=# select gs\_deployment();

gs\_deployment

-----------------------

OpenSourceCentralized

(1 row)

* get\_hostname()
* 描述：返回当前节点的hostname。
* 返回值类型：text

示例：

vastbase=# SELECT get\_hostname();

get\_hostname

-----------------------

localhost.localdomain

(1 row)

* get\_nodename()
* 描述：返回当前节点的名字。
* 返回值类型：text

示例：

vastbase=# SELECT get\_nodename();

get\_nodename

--------------

node1

(1 row)

* get\_schema\_oid(cstring)
* 描述：返回查询schema的oid。
* 返回值类型：oid

示例：

vastbase=# SELECT get\_schema\_oid('public');

get\_schema\_oid

----------------

2200

(1 row)

* get\_client\_info()
* 描述：返回客户端信息。
* 返回值类型：record

**访问权限查询函数**

DDL类权限ALTER、DROP、COMMENT、INDEX、VACUUM属于所有者固有的权限，隐式拥有。

* has\_any\_column\_privilege(user, table, privilege)
* 描述：指定用户是否有访问表任何列的权限。
* **表 1** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| table | text, oid |
| privilege | text |

返回类型：Boolean

* has\_any\_column\_privilege(table, privilege)
* 描述：当前用户是否有访问表任何列的权限，合法参数类型见表1。
* 返回类型：Boolean
* 备注：has\_any\_column\_privilege检查用户是否以特定方式访问表的任何列。其参数可能与has\_table\_privilege类似，除了访问权限类型必须是SELECT、INSERT、UPDATE、COMMENT或REFERENCES的一些组合。
* fig: **说明：**
* 拥有表的表级别权限则隐含的拥有该表每列的列级权限，因此如果与has\_table\_privilege参数相同，has\_any\_column\_privilege总是返回true。但是如果授予至少一列的列级权限也返回成功。
* has\_column\_privilege(user, table, column, privilege)
* 描述：指定用户是否有访问列的权限。
* **表 2** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| table | text, oid |
| column | text, smallint |
| privilege | text |

返回类型：Boolean

* has\_column\_privilege(table, column, privilege)
* 描述：当前用户是否有访问列的权限，合法参数类型见表2。
* 返回类型：Boolean
* 备注：has\_column\_privilege检查用户是否以特定方式访问一列。其参数类似于has\_table\_privilege，可以通过列名或属性号添加列。想要的访问权限类型必须是SELECT、INSERT、UPDATE、COMMENT或REFERENCES的一些组合。
* fig: **说明：**
* 拥有表的表级别权限则隐含的拥有该表每列的列级权限。
* has\_cek\_privilege(user, cek, privilege)
* 描述：指定用户是否有访问列加密密钥CEK的权限。参数说明如下。
* **表 3** 参数类型说明

| **参数名** | **合法入参类型** | **描述** | **取值范围** |
| --- | --- | --- | --- |
| user | name，oid | 用户 | 用户名字或id。 |
| cek | text，oid | 列加密密钥 | 列加密密钥名称或id。 |
| privilege | text | 权限 | * USAGE：允许使用指定列加密密钥。 * DROP：允许删除指定列加密密钥。 |

返回类型：Boolean

* has\_cmk\_privilege(user, cmk, privilege)
* 描述：指定用户是否有访问客户端加密主密钥CMK的权限。参数说明如下。
* **表 4** 参数类型说明

| **参数名** | **合法入参类型** | **描述** | **取值范围** |
| --- | --- | --- | --- |
| user | name，oid | 用户 | 用户名字或id。 |
| cmk | text，oid | 客户端加密主密钥 | 客户端加密主密钥名称或id。 |
| privilege | text | 权限 | * USAGE：允许使用指定客户端加密主密钥。 * DROP：允许删除指定客户端加密主密钥。 |

返回类型：Boolean

* has\_database\_privilege(user, database, privilege)
* 描述：指定用户是否有访问数据库的权限。参数说明如下。

**表 5** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| database | text, oid |
| privilege | text |

返回类型：Boolean

* has\_database\_privilege(database, privilege)
* 描述：当前用户是否有访问数据库的权限，合法参数类型请参见[表5](#zh-cn_topic_0283136950_table111152337017)。
* 返回类型：Boolean
* 备注：has\_database\_privilege检查用户是否能以在特定方式访问数据库。其参数类似has\_table\_privilege。访问权限类型必须是CREATE、CONNECT、TEMPORARY、ALTER、DROP、COMMENT或TEMP（等价于TEMPORARY）的一些组合。
* has\_directory\_privilege(user, directory, privilege)
* 描述：指定用户是否有访问directory的权限。
* **表 6** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| database | text, oid |
| privilege | text |

返回类型：Boolean

* has\_directory\_privilege(directory, privilege)
* 描述：当前用户是否有访问directory的权限，合法参数类型请参见[表6](#zh-cn_topic_0283136950_table111483362025)。
* 返回类型：Boolean
* has\_foreign\_data\_wrapper\_privilege(user, fdw, privilege)
* 描述：指定用户是否有访问外部数据封装器的权限。
* **表 7** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| fdw | text, oid |
| privilege | text |

返回类型：Boolean

* has\_foreign\_data\_wrapper\_privilege(fdw, privilege)
* 描述：当前用户是否有访问外部数据封装器的权限。合法参数类型请参见[表7](#zh-cn_topic_0283136950_table3176631131)。
* 返回类型：Boolean
* 备注：has\_foreign\_data\_wrapper\_privilege检查用户是否能以特定方式访问外部数据封装器。其参数类似has\_table\_privilege。访问权限类型必须是USAGE。
* has\_function\_privilege(user, function, privilege)
* 描述：指定用户是否有访问函数的权限。
* **表 8** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| function | text, oid |
| privilege | text |

返回类型：Boolean

* has\_function\_privilege(function, privilege)
* 描述：当前用户是否有访问函数的权限。合法参数类型请参见[表8](#zh-cn_topic_0283136950_table169651367619)。
* 返回类型：Boolean
* 备注：has\_function\_privilege检查一个用户是否能以指定方式访问一个函数。其参数类似has\_table\_privilege。使用文本字符而不是OID声明一个函数时，允许输入的类型和regprocedure数据类型一样（请参考开发者指南->SQL语法参考->SQL基本要素->数据类型->对象标识符类型章节）。访问权限类型必须是EXECUTE、ALTER、DROP或COMMENT。
* has\_language\_privilege(user, language, privilege)
* 描述：指定用户是否有访问语言的权限。
* **表 9** 参数类型说明
* 返回类型：Boolean

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| language | text, oid |
| privilege | text |

* has\_language\_privilege(language, privilege)
* 描述：当前用户是否有访问语言的权限。合法参数类型请参见表9。
* 返回类型：Boolean
* 备注：has\_language\_privilege检查用户是否能以特定方式访问一个过程语言。其参数类似has\_table\_privilege。访问权限类型必须是USAGE。
* has\_nodegroup\_privilege(user, nodegroup, privilege)
* 描述：检查用户是否有数据库节点访问权限。
* 返回类型：Boolean
* **表 10** 参数类型说明

| **参数名** | **合法入参类型** |
| --- | --- |
| user | name, oid |
| nodegroup | text, oid |
| privilege | text |

* has\_nodegroup\_privilege(nodegroup, privilege)
* 描述：检查用户是否有数据库节点访问权限。参数与has\_table\_privilege类似。访问权限类型必须是USAGE、CREATE、COMPUTE、ALTER或CROP。
* 返回类型：Boolean
* has\_schema\_privilege(user, schema, privilege)
* 描述：指定用户是否有访问模式的权限。
* 返回类型：Boolean
* has\_schema\_privilege(schema, privilege)
* 描述：当前用户是否有访问模式的权限。
* 返回类型：Boolean
* 备注：has\_schema\_privilege检查用户是否能以特定方式访问一个模式。其参数类似has\_table\_privilege。访问权限类型必须是CREATE、USAGE、ALTER、DROP或COMMENT的一些组合。
* has\_server\_privilege(user, server, privilege)
* 描述：指定用户是否有访问外部服务的权限。
* 返回类型：Boolean
* has\_server\_privilege(server, privilege)
* 描述：当前用户是否有访问外部服务的权限。
* 返回类型：Boolean
* 备注：has\_server\_privilege检查用户是否能以指定方式访问一个外部服务器。其参数类似has\_table\_privilege。访问权限类型必须是USAGE、ALTER、DROP或COMMENT之一的值。
* has\_table\_privilege(user, table, privilege)
* 描述：指定用户是否有访问表的权限。
* 返回类型：Boolean
* has\_table\_privilege(table, privilege)
* 描述：当前用户是否有访问表的权限。
* 返回类型：Boolean
* 备注：has\_table\_privilege检查用户是否以特定方式访问表。用户可以通过名称或OID（pg\_authid.oid）来指定，public表明PUBLIC伪角色，或如果缺省该参数，则使用current\_user。该表可以通过名称或者OID声明。如果用名称声明，则在必要时可以用模式进行修饰。如果使用文本字符串来声明所希望的权限类型，这个文本字符串必须是SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、TRIGGER、ALTER、DROP、COMMENT、INDEX或VACUUM之一的值。可以给权限类型添加WITH GRANT OPTION，用来测试权限是否拥有授权选项。也可以用逗号分隔列出的多个权限类型，如果拥有任何所列出的权限，则结果便为true。
* has\_tablespace\_privilege(user, tablespace, privilege)
* 描述：指定用户是否有访问表空间的权限。
* 返回类型：Boolean
* has\_tablespace\_privilege(tablespace, privilege)
* 描述：当前用户是否有访问表空间的权限。
* 返回类型：Boolean
* 备注：has\_tablespace\_privilege检查用户是否能以特定方式访问一个表空间。其参数类似has\_table\_privilege。访问权限类型必须是CREATE、ALTER、DROP或COMMENT之一的值。
* pg\_has\_role(user, role, privilege)
* 描述：指定用户是否有角色的权限。
* 返回类型：Boolean
* pg\_has\_role(role, privilege)
* 描述：当前用户是否有角色的权限。
* 返回类型：Boolean
* 备注：pg\_has\_role检查用户是否能以特定方式访问一个角色。其参数类似has\_table\_privilege，除了public不能用做用户名。访问权限类型必须是MEMBER或USAGE的一些组合。 MEMBER表示的是角色中的直接或间接成员关系（也就是SET ROLE的权限），而USAGE表示无需通过SET ROLE也直接拥有角色的使用权限。
* has\_any\_privilege(user, privilege)
* 描述：指定用户是否有某项ANY权限，若同时查询多个权限，只要具有其中一个则返回true。
* 返回类型：Boolean

**表 11** 参数类型说明

| **参数名** | **合法入参类型** | **描述** | **取值范围** |
| --- | --- | --- | --- |
| user | name | 用户 | 已存在的用户名。 |
| privilege | text | ANY权限 | 可选取值：  CREATE ANY TABLE [WITH ADMIN OPTION]  ALTER ANY TABLE [WITH ADMIN OPTION]  DROP ANY TABLE [WITH ADMIN OPTION]  SELECT ANY TABLE [WITH ADMIN OPTION]  INSERT ANY TABLE [WITH ADMIN OPTION]  UPDATE ANY TABLE [WITH ADMIN OPTION]  DELETE ANY TABLE [WITH ADMIN OPTION]  CREATE ANY SEQUENCE [WITH ADMIN OPTION]  CREATE ANY INDEX [WITH ADMIN OPTION]  CREATE ANY FUNCTION [WITH ADMIN OPTION]  EXECUTE ANY FUNCTION [WITH ADMIN OPTION]  CREATE ANY PACKAGE [WITH ADMIN OPTION]  EXECUTE ANY PACKAGE [WITH ADMIN OPTION]  CREATE ANY TYPE [WITH ADMIN OPTION] |

**模式可见性查询函数**

每个函数执行检查数据库对象类型的可见性。对于函数和操作符，如果在前面的搜索路径中没有相同的对象名称和参数的数据类型，则此对象是可见的。对于操作符类，则要同时考虑名称和相关索引的访问方法。

所有这些函数都需要使用OID来标识要需要检查的对象。如果用户想通过名称测试对象，则使用OID别名类型（regclass、regtype、regprocedure、regoperator、regconfig或regdictionary）将会很方便。

比如，如果一个表所在的模式在搜索路径中，并且在前面的搜索路径中没有同名的表，则这个表是可见的。它等效于表可以不带明确模式修饰进行引用。比如，要列出所有可见表的名称：

vastbase=# SELECT relname FROM pg\_class WHERE pg\_table\_is\_visible(oid);

* pg\_collation\_is\_visible(collation\_oid)
* 描述：该排序是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_conversion\_is\_visible(conversion\_oid)
* 描述：该转换是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_function\_is\_visible(function\_oid)
* 描述：该函数是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_opclass\_is\_visible(opclass\_oid)
* 描述：该操作符类是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_operator\_is\_visible(operator\_oid)
* 描述：该操作符是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_opfamily\_is\_visible(opclass\_oid)
* 描述：该操作符族是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_table\_is\_visible(table\_oid)
* 描述：该表是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_ts\_config\_is\_visible(config\_oid)
* 描述：该文本检索配置是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_ts\_dict\_is\_visible(dict\_oid)
* 描述：该文本检索词典是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_ts\_parser\_is\_visible(parser\_oid)
* 描述：该文本搜索解析是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_ts\_template\_is\_visible(template\_oid)
* 描述：该文本检索模板是否在搜索路径中可见。
* 返回类型：Boolean
* pg\_type\_is\_visible(type\_oid)
* 描述：该类型（或域）是否在搜索路径中可见。
* 返回类型：Boolean

**系统表信息函数**

* format\_type(type\_oid, typemod)
* 描述：获取数据类型的SQL名称。
* 返回类型：text
* 备注：format\_type通过某个数据类型的类型OID以及可能的类型修饰词，返回其SQL名称。如果不知道具体的修饰词，则在类型修饰词的位置传入NULL。类型修饰词一般只对有长度限制的数据类型有意义。format\_type所返回的SQL名称中包含数据类型的长度值，其大小是：实际存储长度len - sizeof(int32)，单位字节。原因是数据存储时需要32位的空间来存储用户对数据类型的自定义长度信息，即实际存储长度要比用户定义长度多4个字节。在下例中，format\_type返回的SQL名称为“character varying(6)”，6表示varchar类型的长度值是6字节，因此该类型的实际存储长度为10字节。

vastbase=# SELECT format\_type((SELECT oid FROM pg\_type WHERE typname='varchar'), 10);

format\_type

-------------

varchar(6)

(1 row)

* getdistributekey(table\_name)
* 描述：获取一个hash表的分布列。单机环境下不支持分布，该函数返回为空。
* pg\_check\_authid(role\_oid)
* 描述：检查是否存在给定oid的角色名。
* 返回类型：Boolean

示例：

vastbase=# select pg\_check\_authid(1);

pg\_check\_authid

-----------------

f

(1 row)

* pg\_describe\_object(catalog\_id, object\_id, object\_sub\_id)
* 描述：获取数据库对象的描述。
* 返回类型：text

备注：pg\_describe\_object返回由目录OID，对象OID和一个（或许0个）子对象ID指定的数据库对象的描述。这有助于确认存储在pg\_depend系统表中对象的身份。

* pg\_get\_constraintdef(constraint\_oid)
* 描述：获取约束的定义。
* 返回类型：text
* pg\_get\_constraintdef(constraint\_oid, pretty\_bool)
* 描述：获取约束的定义。
* 返回类型：text
* 备注：pg\_get\_constraintdef和pg\_get\_indexdef分别从约束或索引上使用创建命令进行重构。
* pg\_get\_expr(pg\_node\_tree, relation\_oid)
* 描述：反编译表达式的内部形式，假设其中的任何Vars都引用第二个参数指定的关系。
* 返回类型：text
* pg\_get\_expr(pg\_node\_tree, relation\_oid, pretty\_bool)
* 描述：反编译表达式的内部形式，假设其中的任何Vars都引用第二个参数指定的关系。
* 返回类型：text
* 备注：pg\_get\_expr反编译一个独立表达式的内部形式，比如一个字段的缺省值。在检查系统表的内容的时候很有用。如果表达式可能包含关键字，则指定他们引用相关的OID作为第二个参数；如果没有关键字，零就足够了。
* pg\_get\_functiondef(func\_oid)
* 描述：获取函数的定义。
* 返回类型：text

示例：

vastbase=# select \* from pg\_get\_functiondef(598);

headerlines | definition

-------------+----------------------------------------------------

4 | CREATE OR REPLACE FUNCTION pg\_catalog.abbrev(inet)+

| RETURNS text +

| LANGUAGE internal +

| IMMUTABLE STRICT NOT FENCED NOT SHIPPABLE +

| AS $function$inet\_abbrev$function$ +

|

(1 row)

* pg\_get\_function\_arguments(func\_oid)
* 描述：获取函数定义的参数列表（带默认值）。
* 返回类型：text
* 备注：pg\_get\_function\_arguments返回一个函数的参数列表，需要在CREATE FUNCTION中使用这种格式。
* pg\_get\_function\_identity\_arguments(func\_oid)
* 描述：获取参数列表来确定一个函数（不带默认值）。
* 返回类型：text
* 备注：pg\_get\_function\_identity\_arguments返回需要的参数列表用来标识函数，这种形式需要在ALTER FUNCTION中使用，并且这种形式省略了默认值。
* pg\_get\_function\_result(func\_oid)
* 描述：获取函数的RETURNS子句。
* 返回类型：text
* 备注：pg\_get\_function\_result为函数返回适当的RETURNS子句。
* pg\_get\_indexdef(index\_oid)
* 描述：获取索引的CREATE INDEX命令。
* 返回类型：text

示例：

vastbase=# select \* from pg\_get\_indexdef(16416);

pg\_get\_indexdef

-------------------------------------------------------------------------

CREATE INDEX test3\_b\_idx ON test3 USING btree (b) TABLESPACE pg\_default

(1 row)

* pg\_get\_indexdef(index\_oid, dump\_schema\_only)
* 描述：获取索引的CREATE INDEX命令，仅用于dump场景。对于包含local索引的间隔分区表，当dump\_schema\_only为true时，返回的创建索引语句中不包含自动创建的分区的local索引信息；当dump\_schema\_only为false时，返回的创建索引语句中包含自动创建的分区的local索引信息。对于非间隔分区表或者不包含local索引的间隔分区分区表，dump\_schema\_only参数取值不影响函数返回结果。
* 返回类型：text

示例：

vastbase=# CREATE TABLE sales

(prod\_id NUMBER(6),

cust\_id NUMBER,

time\_id DATE,

channel\_id CHAR(1),

promo\_id NUMBER(6),

quantity\_sold NUMBER(3),

amount\_sold NUMBER(10,2)

)

PARTITION BY RANGE( time\_id) INTERVAL('1 day')

(

partition p1 VALUES LESS THAN ('2019-02-01 00:00:00'),

partition p2 VALUES LESS THAN ('2019-02-02 00:00:00')

);

vastbase=# create index index\_sales on sales(prod\_id) local (PARTITION idx\_p1 ,PARTITION idx\_p2);

-- 插入数据没有匹配的分区，新创建一个分区，并将数据插入该分区

vastbase=# INSERT INTO sales VALUES(1, 12, '2019-02-05 00:00:00', 'a', 1, 1, 1);

vastbase=# select oid from pg\_class where relname = 'index\_sales';

oid

-------

16645

(1 row)

vastbase=# select \* from pg\_get\_indexdef(16645, true);

pg\_get\_indexdef

--------------------------------------------------------------------------------------------------------------------------

CREATE INDEX index\_sales ON sales USING btree (prod\_id) LOCAL(PARTITION idx\_p1, PARTITION idx\_p2) TABLESPACE pg\_default

(1 row)

vastbase=# select \* from pg\_get\_indexdef(16645, false);

pg\_get\_indexdef

-----------------------------------------------------------------------------------------------------------------------------------------

---------------

CREATE INDEX index\_sales ON sales USING btree (prod\_id) LOCAL(PARTITION idx\_p1, PARTITION idx\_p2, PARTITION sys\_p1\_prod\_id\_idx) TABLESP

ACE pg\_default

(1 row)

* pg\_get\_indexdef(index\_oid, column\_no, pretty\_bool)
* 描述：获取索引的CREATE INDEX命令，或者如果column\_no不为零，则只获取一个索引字段的定义。

示例：

vastbase=# select \* from pg\_get\_indexdef(16645, 0, false);

pg\_get\_indexdef

-------------------------------------------------------------------------------------

CREATE INDEX index\_sales ON sales USING btree (prod\_id) LOCAL TABLESPACE pg\_default

(1 row)

vastbase=# select \* from pg\_get\_indexdef(16645, 1, false);

pg\_get\_indexdef

-----------------

prod\_id

(1 row)

返回类型：text

备注：pg\_get\_functiondef为函数返回一个完整的CREATE OR REPLACE FUNCTION语句。

* pg\_get\_keywords()
* 描述：获取SQL关键字和类别列表。
* 返回类型：setof record
* 备注：pg\_get\_keywords返回一组关于描述服务器识别SQL关键字的记录。word列包含关键字。catcode列包含一个分类代码：U表示通用的，C表示列名，T表示类型或函数名，或R表示保留。catdesc列包含了一个可能本地化描述分类的字符串。
* pg\_get\_userbyid(role\_oid)
* 描述：获取给定OID的角色名。
* 返回类型：name
* 备注：pg\_get\_userbyid通过角色的OID抽取对应的用户名。
* pg\_check\_authid(role\_id)
* 描述：通过role\_id检查用户是否存在。
* 返回类型：text

示例：

vastbase=# select pg\_check\_authid(20);

pg\_check\_authid

-----------------

f

(1 row)

* pg\_get\_viewdef(view\_name)
* 描述：为视图获取底层的SELECT命令。
* 返回类型：text
* pg\_get\_viewdef(view\_name, pretty\_bool)
* 描述：为视图获取底层的SELECT命令，如果pretty\_bool为true，行字段可以包含80列。
* 返回类型：text
* 备注：pg\_get\_viewdef重构出定义视图的SELECT查询。这些函数大多数都有两种形式，其中带有pretty\_bool参数，且参数为true时，是"适合打印"的结果，这种格式更容易读。另一种是缺省的格式，更有可能被将来的不同版本用同样的方法解释。如果是用于转储，那么尽可能避免使用适合打印的格式。给pretty-print参数传递false生成的结果和没有这个参数的变种生成的结果是完全一样。
* pg\_get\_viewdef(view\_oid)
* 描述：为视图获取底层的SELECT命令。
* 返回类型：text
* pg\_get\_viewdef(view\_oid, pretty\_bool)
* 描述：为视图获取底层的SELECT命令，如果pretty\_bool为true，行字段可以包含80列。
* 返回类型：text
* pg\_get\_viewdef(view\_oid, wrap\_column\_int)
* 描述：为视图获取底层的SELECT命令；行字段被换到指定的列数，打印是隐含的。
* 返回类型：text
* pg\_get\_tabledef(table\_oid)
* 描述：根据table\_oid获取表定义

示例：

vastbase=# select \* from pg\_get\_tabledef(16384);

pg\_get\_tabledef

-------------------------------------------------------

SET search\_path = public; +

CREATE TABLE t1 ( +

c1 bigint DEFAULT nextval('serial'::regclass)+

) +

WITH (orientation=row, compression=no) +

TO GROUP group1;

(1 row)

返回类型：text

* pg\_get\_tabledef(table\_name)
* 描述：根据table\_name获取表定义。

示例：

vastbase=# select \* from pg\_get\_tabledef('t1');

pg\_get\_tabledef

--------------------------------------------------------

SET search\_path = public; +

CREATE TABLE t1 ( +

c1 integer, +

c2 integer +

) +

WITH (orientation=row, compression=no, fillfactor=80);

(1 row)

返回类型：text

备注：pg\_get\_tabledef重构出表定义的CREATE语句，包含了表定义本身、索引信息、comments信息。对于表对象依赖的group、schema、tablespace、server等信息，需要用户自己去创建，表定义里不会有这些对象的创建语句。

* pg\_options\_to\_table(reloptions)
* 描述：获取存储选项名称/值对的集合。
* 返回类型：setof record
* 备注：pg\_options\_to\_table当通过pg\_class.reloptions或pg\_attribute.attoptions时返回存储选项名称/值对（option\_name/option\_value）的集合。
* pg\_tablespace\_databases(tablespace\_oid)
* 描述：获取在指定的表空间中有对象的数据库OID集合。
* 返回类型：setof oid
* 备注：pg\_tablespace\_databases允许检查表空间的状况，返回在该表空间中保存了对象的数据库OID集合。如果这个函数返回数据行，则该表空间就是非空的，因此不能删除。要显示该表空间中的特定对象，用户需要连接pg\_tablespace\_databases标识的数据库与查询pg\_class系统表。
* pg\_tablespace\_location(tablespace\_oid)
* 描述：获取表空间所在的文件系统的路径。
* 返回类型：text
* pg\_typeof(any)
* 描述：获取任何值的数据类型。
* 返回类型：regtype
* 备注：pg\_typeof返回传递给他的值的数据类型OID。这可能有助于故障排除或动态构造SQL查询。声明此函数返回regtype，这是一个OID别名类型（请参考开发者指南->SQL语法参考->SQL基本要素->数据类型->对象标识符类型章节）；这意味着它是一个为了比较而显示类型名称的OID。

示例：

vastbase=# SELECT pg\_typeof(33);

pg\_typeof

-----------

integer

(1 row)

vastbase=# SELECT typlen FROM pg\_type WHERE oid = pg\_typeof(33);

typlen

--------

4

(1 row)

* collation for (any)
* 描述：获取参数的排序。
* 返回类型：text
* 备注：表达式collation for返回传递给他的值的排序。

示例：

vastbase=# SELECT collation for (description) FROM pg\_description LIMIT 1;

pg\_collation\_for

------------------

"default"

(1 row)

值可能是引号括起来的并且模式限制的。如果没有为参数表达式排序，则返回一个null值。如果参数不是排序的类型，则抛出一个错误。

* pg\_extension\_update\_paths(name)
* 描述：返回指定扩展的版本更新路径。
* 返回类型：text(source text), text(path text), text(target text)
* pg\_get\_serial\_sequence(tablename, colname)
* 描述：获取对应表名和列名上的序列。
* 返回类型：text

示例：

vastbase=# select \* from pg\_get\_serial\_sequence('t2', 'c1');

pg\_get\_serial\_sequence

------------------------

public.serial

(1 row)

* pg\_sequence\_parameters(sequence\_oid)
* 描述：获取指定sequence的参数，包含起始值，最小值和最大值，递增值等。
* 返回类型：int16, int16,int16, bigint, Boolean

示例：

vastbase=# select \* from pg\_sequence\_parameters(16643);

start\_value | minimum\_value | maximum\_value | increment | cycle\_option

-------------+---------------+---------------------+-----------+--------------

101 | 1 | 9223372036854775807 | 1 | f

(1 row)

**注释信息函数**

* col\_description(table\_oid, column\_number)
* 描述：获取一个表字段的注释
* 返回类型：text
* 备注：col\_description返回一个表中字段的注释，通过表OID和字段号来声明。
* obj\_description(object\_oid, catalog\_name)
* 描述：获取一个数据库对象的注释
* 返回类型：text
* 备注：带有两个参数的obj\_description返回一个数据库对象的注释，该对象是通过其OID和其所属的系统表名称声明。比如，obj\_description(123456,'pg\_class')将返回OID为123456的表的注释。只带一个参数的obj\_description只要求对象OID。
* obj\_description不能用于表字段，因为字段没有自己的OID。
* obj\_description(object\_oid)
* 描述：获取一个数据库对象的注释
* 返回类型：text
* shobj\_description(object\_oid, catalog\_name)
* 描述：获取一个共享数据库对象的注释
* 返回类型：text
* 备注：shobj\_description和obj\_description差不多，不同之处仅在于前者用于共享对象。一些系统表是通用于vastbase中所有数据库的全局表，因此这些表的注释也是全局存储的。

**事务ID和快照**

内部事务ID类型（xid）是64位。这些函数使用的数据类型txid\_snapshot，存储在特定时刻事务ID可见性的信息。其组件描述在表11。

**表 12** 快照组件

| **名称** | **描述** |
| --- | --- |
| xmin | 最早的事务ID（txid）仍然活动。所有较早事务将是已经提交可见的，或者是直接回滚。 |
| xmax | 作为尚未分配的txid。所有大于或等于此txids的都是尚未开始的快照时间，因此不可见。 |
| xip\_list | 当前快照中活动的txids。这个列表只包含在xmin和xmax之间活动的txids；有可能活动的txids高于xmax。介于大于等于xmin、小于xmax，并且不在这个列表中的txid，在这个时间快照已经完成的，因此按照提交状态查看他是可见还是回滚。这个列表不包含子事务的txids。 |

txid\_snapshot的文本表示为：xmin:xmax:xip\_list。

示例：10:20:10,14,15意思为：xmin=10, xmax=20, xip\_list=10, 14, 15。

以下的函数在一个输出形式中提供服务器事务信息。这些函数的主要用途是为了确定在两个快照之间有哪个事务提交。

* txid\_current()
* 描述：获取当前事务ID。
* 返回类型：bigint
* gs\_txid\_oldestxmin()
* 描述：获取当前最小事务id的值oldesxmin。
* 返回类型：bigint
* txid\_current\_snapshot()
* 描述：获取当前快照。
* 返回类型：txid\_snapshot
* txid\_snapshot\_xip(txid\_snapshot)
* 描述：在快照中获取正在进行的事务ID。
* 返回类型：setof bigint
* txid\_snapshot\_xmax(txid\_snapshot)
* 描述：获取快照的xmax。
* 返回类型：bigint
* txid\_snapshot\_xmin(txid\_snapshot)
* 描述：获取快照的xmin。
* 返回类型：bigint
* txid\_visible\_in\_snapshot(bigint, txid\_snapshot)
* 描述：在快照中事务ID是否可见（不使用子事务ID）。
* 返回类型：Boolean
* get\_local\_prepared\_xact()
* 描述：获取当前节点两阶段残留事务信息，包括事务id，两阶段gid名称，prepared的时间，owner的oid，database的oid及当前节点的node\_name。
* 返回类型：xid, text, timestamptz, oid, oid，text
* get\_remote\_prepared\_xacts()
* 描述：获取所有远程节点两阶段残留事务信息，包括事务id，两阶段gid名称，prepared的时间，owner的名称，database的名称及node\_name。
* 返回类型：xid, text, timestamptz, name, name，text
* global\_clean\_prepared\_xacts(text, text)
* 描述：并发清理两阶段残留事务，仅gs\_clean工具可以调用清理，其他用户调用均返回false。
* 返回类型：Boolean
* gs\_get\_next\_xid\_csn()
* 描述：返回全局所有节点上的next\_xid和next\_csn值。
* 返回值如下：
* **表 13** gs\_get\_next\_xid\_csn返回参数说明

| **字段名** | **描述** |
| --- | --- |
| nodename | 节点名称。 |
| next\_xid | 当前节点下一个事务id号。 |
| next\_csn | 当前节点下一个csn号。 |

* pg\_control\_system()
* 描述：返回系统控制文件状态。
* 返回类型：SETOF record
* pg\_control\_checkpoint()
* 描述：返回系统检查点状态。
* 返回类型：SETOF record
* pv\_builtin\_functions
* 描述：查看所有内置系统函数信息。
* 参数：nan
* 返回值类型：proname name, pronamespace oid, proowner oid, prolang oid, procost real, prorows real, provariadic oid, protransform regproc, proisagg boolean, proiswindow boolean, prosecdef boolean, proleakproof boolean, proisstrict boolean, proretset boolean, provolatile "char", pronargs smallint, pronargdefaults smallint, prorettype oid, proargtypes oidvector, proallargtypes integer[], proargmodes "char"[], proargnames text[], proargdefaults pg\_node\_tree, prosrc text, probin text, proconfig text[], proacl aclitem[], prodefaultargpos int2vector, fencedmode boolean, proshippable boolean, propackage boolean, oid oid
* pv\_thread\_memory\_detail
* 描述：返回各线程的内存信息。
* 参数：nan
* 返回值类型：threadid text, tid bigint, thrdtype text, contextname text, level smallint, parent text, totalsize bigint, freesize bigint, usedsize bigint
* pg\_relation\_compression\_ratio
* 描述：查询表压缩率，默认返回1.0。
* 参数：text
* 返回值类型：real
* pg\_relation\_with\_compression
* 描述：查询表是否压缩。
* 参数：text
* 返回值类型：boolean
* pg\_stat\_file\_recursive
* 描述：列出路径下所有文件。
* 参数：location text
* pg\_shared\_memory\_detail
* 描述：返回所有已产生的共享内存上下文的使用信息，各列描述请参考开发者指南->schema->DBE\_PERF Schema->Memory->GS\_SHARED\_MEMORY\_DETAIL章节。
* 参数：nan
* 返回值类型：contextname text, level smallint, parent text, totalsize bigint, freesize bigint, usedsize bigint
* get\_gtm\_lite\_status
* 描述：返回GTM上的backupXid和csn号，用来支持问题定位，GTM-FREE模式下不支持使用本系统函数。
* gs\_stat\_get\_wlm\_plan\_operator\_info
* 描述：从内部哈希表中获取算子计划信息。
* 参数：oid
* 返回值类型：datname text, queryid int8, plan\_node\_id int4, startup\_time int8, total\_time int8, actual\_rows int8, max\_peak\_memory int4, query\_dop int4, parent\_node\_id int4, left\_child\_id int4, right\_child\_id int4, operation text, orientation text, strategy text, options text, condition text, projection text
* pg\_stat\_get\_partition\_tuples\_hot\_updated
* 描述：返回给定分区id的分区热更新元组数的统计。
* 参数：oid
* 返回值类型：bigint
* gs\_session\_memory\_detail\_tp
* 描述：返回会话的内存使用情况，参考gs\_session\_memory\_detail。
* 参数：nan
* 返回值类型：sessid text, sesstype text, contextname text, level smallint, parent text, totalsize bigint, freesize bigint, usedsize bigint
* gs\_thread\_memory\_detail
* 描述：返回各线程的内存信息。
* 参数：nan
* 返回值类型：threadid text, tid bigint, thrdtype text, contextname text, level smallint, parent text, totalsize bigint, freesize bigint, usedsize bigint
* pg\_stat\_get\_wlm\_realtime\_operator\_info
* 描述：从内部哈希表中获取实时执行计划算子信息。
* 参数：nan
* 返回值类型：queryid bigint, pid bigint, plan\_node\_id integer, plan\_node\_name text, start\_time timestamp with time zone, duration bigint, status text, query\_dop integer, estimated\_rows bigint, tuple\_processed bigint, min\_peak\_memory integer, max\_peak\_memory integer, average\_peak\_memory integer, memory\_skew\_percent integer, min\_spill\_size integer, max\_spill\_size integer, average\_spill\_size integer, spill\_skew\_percent integer, min\_cpu\_time bigint, max\_cpu\_time bigint, total\_cpu\_time bigint, cpu\_skew\_percent integer, warning text
* pg\_stat\_get\_wlm\_realtime\_ec\_operator\_info
* 描述：从内部哈希表中获取EC执行计划算子信息。
* 参数：nan
* 返回值类型：queryid bigint, plan\_node\_id integer, plan\_node\_name text, start\_time timestamp with time zone, ec\_operator integer, ec\_status text, ec\_execute\_datanode text, ec\_dsn text, ec\_username text, ec\_query text, ec\_libodbc\_type text, ec\_fetch\_count bigint
* pg\_stat\_get\_wlm\_operator\_info
* 描述：从内部哈希表中获取执行计划算子信息。
* 参数：nan
* 返回值类型：queryid bigint, pid bigint, plan\_node\_id integer, plan\_node\_name text, start\_time timestamp with time zone, duration bigint, query\_dop integer, estimated\_rows bigint, tuple\_processed bigint, min\_peak\_memory integer, max\_peak\_memory integer, average\_peak\_memory integer, memory\_skew\_percent integer, min\_spill\_size integer, max\_spill\_size integer, average\_spill\_size integer, spill\_skew\_percent integer, min\_cpu\_time bigint, max\_cpu\_time bigint, total\_cpu\_time bigint, cpu\_skew\_percent integer, warning text
* pg\_stat\_get\_wlm\_node\_resource\_info
* 描述：获取当前节点资源信息。
* 参数：nan
* 返回值类型：min\_mem\_util integer, max\_mem\_util integer, min\_cpu\_util integer, max\_cpu\_util integer, min\_io\_util integer, max\_io\_util integer, used\_mem\_rate integer
* pg\_stat\_get\_session\_wlmstat
* 描述：返回当前会话负载信息。
* 参数：pid integer
* 返回值类型：datid oid, threadid bigint, sessionid bigint, threadpid integer, usesysid oid, appname text, query text, priority bigint, block\_time bigint, elapsed\_time bigint, total\_cpu\_time bigint, skew\_percent integer, statement\_mem integer, active\_points integer, dop\_value integer, current\_cgroup text, current\_status text, enqueue\_state text, attribute text, is\_plana boolean, node\_group text, srespool name
* pg\_stat\_get\_wlm\_ec\_operator\_info
* 描述：从内部哈希表中获取EC执行计划算子信息。
* 参数：nan
* 返回值类型：queryid bigint, plan\_node\_id integer, plan\_node\_name text, start\_time timestamp with time zone, duration bigint, tuple\_processed bigint, min\_peak\_memory integer, max\_peak\_memory integer, average\_peak\_memory integer, ec\_operator integer, ec\_status text, ec\_execute\_datanode text, ec\_dsn text, ec\_username text, ec\_query text, ec\_libodbc\_type text, ec\_fetch\_count bigint
* pg\_stat\_get\_wlm\_instance\_info
* 描述：返回当前实例负载信息。
* 参数：nan
* 返回值类型：instancename text, timestamp timestamp with time zone, used\_cpu integer, free\_memory integer, used\_memory integer, io\_await double precision, io\_util double precision, disk\_read double precision, disk\_write double precision, process\_read bigint, process\_write bigint, logical\_read bigint, logical\_write bigint, read\_counts bigint, write\_counts bigint
* pg\_stat\_get\_wlm\_instance\_info\_with\_cleanup
* 描述：返回当前实例负载信息，并且保存到系统表中。
* 参数：nan
* 返回值类型：instancename text, timestamp timestamp with time zone, used\_cpu integer, free\_memory integer, used\_memory integer, io\_await double precision, io\_util double precision, disk\_read double precision, disk\_write double precision, process\_read bigint, process\_write bigint, logical\_read bigint, logical\_write bigint, read\_counts bigint, write\_counts bigint
* pg\_stat\_get\_wlm\_realtime\_session\_info
* 描述：返回实时会话负载信息。
* 参数：nan
* 返回值类型：nodename text, threadid bigint, block\_time bigint, duration bigint, estimate\_total\_time bigint, estimate\_left\_time bigint, schemaname text, query\_band text, spill\_info text, control\_group text, estimate\_memory integer, min\_peak\_memory integer, max\_peak\_memory integer, average\_peak\_memory integer, memory\_skew\_percent integer, min\_spill\_size integer, max\_spill\_size integer, average\_spill\_size integer, spill\_skew\_percent integer, min\_dn\_time bigint, max\_dn\_time bigint, average\_dn\_time bigint, dntime\_skew\_percent integer, min\_cpu\_time bigint, max\_cpu\_time bigint, total\_cpu\_time bigint, cpu\_skew\_percent integer, min\_peak\_iops integer, max\_peak\_iops integer, average\_peak\_iops integer, iops\_skew\_percent integer, warning text, query text, query\_plan text, cpu\_top1\_node\_name text, cpu\_top2\_node\_name text, cpu\_top3\_node\_name text, cpu\_top4\_node\_name text, cpu\_top5\_node\_name text, mem\_top1\_node\_name text, mem\_top2\_node\_name text, mem\_top3\_node\_name text, mem\_top4\_node\_name text, mem\_top5\_node\_name text, cpu\_top1\_value bigint, cpu\_top2\_value bigint, cpu\_top3\_value bigint, cpu\_top4\_value bigint, cpu\_top5\_value bigint, mem\_top1\_value bigint, mem\_top2\_value bigint, mem\_top3\_value bigint, mem\_top4\_value bigint, mem\_top5\_value bigint, top\_mem\_dn text, top\_cpu\_dn text
* pg\_stat\_get\_wlm\_session\_iostat\_info
* 描述：返回会话负载IO信息。
* 参数：nan
* 返回值类型：threadid bigint, maxcurr\_iops integer, mincurr\_iops integer, maxpeak\_iops integer, minpeak\_iops integer, iops\_limits integer, io\_priority integer, curr\_io\_limits integer
* pg\_stat\_get\_wlm\_statistics
* 描述：返回会话负载统计数据。
* 参数：nan
* 返回值类型：statement text, block\_time bigint, elapsed\_time bigint, total\_cpu\_time bigint, qualification\_time bigint, skew\_percent integer, control\_group text, status text, action text

#### 字符处理函数和操作符

Vastbase提供的字符处理函数和操作符主要用于字符串与字符串、字符串与非字符串之间的连接，以及字符串的模式匹配操作。注意：字符串处理函数除了length相关函数，其他函数和操作符不支持大于1GB clob作为参数。

* bit\_length(string)
* 描述：字符串的位数。
* 返回值类型：int
* 示例：

vastbase=# SELECT bit\_length('world');

bit\_length

------------

40

(1 row)

* btrim(string text [, characters text])
* 描述：从string开头和结尾删除只包含characters中字符（缺省是空白）的最长字符串。
* 返回值类型：text
* 示例：

vastbase=# SELECT btrim('sring' , 'ing');

btrim

-------

sr

(1 row)

* char\_length(string)或character\_length(string)
* 描述：字符串中的字符个数。
* 返回值类型：int
* 示例：

vastbase=# SELECT char\_length('hello');

char\_length

-------------

5

(1 row)

* instr(text,text,int,int)
* 描述：instr(string1,string2,int1,int2)返回在string1中从int1位置开始匹配到第int2次string2的位置，第一个int表示开始匹配起始位置，第二个int表示匹配的次数。
* 返回值类型：int
* 示例：

vastbase=# SELECT instr( 'abcdabcdabcd', 'bcd', 2, 2 );

instr

-------

6

(1 row)

* lengthb(text/bpchar)
* 描述：获取指定字符串的字节数。
* 返回值类型：int
* 示例：

vastbase=# SELECT lengthb('hello');

lengthb

---------

5

(1 row)

* left(str text, n int)
* 描述：返回字符串的前n个字符。当n是负数时，返回除最后|n|个字符以外的所有字符。
* 返回值类型：text
* 示例：

vastbase=# SELECT left('abcde', 2);

left

------

ab

(1 row)

* length(string bytea, encoding name )
* 描述：指定encoding编码格式的string的字符数。在这个编码格式中，string必须是有效的。
* 返回值类型：int
* 示例：

vastbase=# SELECT length('jose', 'UTF8');

length

--------

4

(1 row)

fig: **说明：**

如果是查询bytea类型的长度，指定utf8编码时，最大长度只能为536870888。

* lpad(string text, length int [, fill text])
* 描述：通过填充字符fill（缺省时为空白），把string填充为length长度。如果string已经比length长则将其尾部截断。
* 返回值类型：text
* 示例：

vastbase=# SELECT lpad('hi', 5, 'xyza');

lpad

-------

xyzhi

(1 row)

* notlike(x bytea name text, y bytea text)
* 描述：比较x和y是否不一致。
* 返回值类型：bool
* 示例：

vastbase=# SELECT notlike(1,2);

notlike

--------------

t

(1 row)

vastbase=# SELECT notlike(1,1);

notlike

--------------

f

(1 row)

* octet\_length(string)
* 描述：字符串中的字节数。
* 返回值类型：int
* 示例：

vastbase=# SELECT octet\_length('jose');

octet\_length

--------------

4

(1 row)

* overlay(string placing string FROM int [for int])
* 描述：替换子字符串。FROM int表示从第一个string的第几个字符开始替换，for int表示替换第一个string的字符数目。
* 返回值类型：text
* 示例：

vastbase=# SELECT overlay('hello' placing 'world' from 2 for 3 );

overlay

---------

hworldo

(1 row)

* position(substring in string)
* 描述：指定子字符串的位置。字符串区分大小写。
* 返回值类型：int，字符串不存在时返回0。
* 示例：

vastbase=# SELECT position('ing' in 'string');

position

----------

4

(1 row)

* pg\_client\_encoding()
* 描述：当前客户端编码名称。
* 返回值类型：name
* 示例：

vastbase=# SELECT pg\_client\_encoding();

pg\_client\_encoding

--------------------

UTF8

(1 row)

* quote\_ident(string text)
* 描述：返回适用于SQL语句的标识符形式（使用适当的引号进行界定）。只有在必要的时候才会添加引号（字符串包含非标识符字符或者会转换大小写的字符）。返回值中嵌入的引号都写了两次。
* 返回值类型：text
* 示例：

vastbase=# SELECT quote\_ident('hello world');

quote\_ident

--------------

"hello world"

(1 row)

* quote\_literal(string text)
* 描述：返回适用于在SQL语句里当作文本使用的形式（使用适当的引号进行界定）。
* 返回值类型：text
* 示例：

vastbase=# SELECT quote\_literal('hello');

quote\_literal

---------------

'hello'

(1 row)

如果出现如下写法，text文本将进行转义。

vastbase=# SELECT quote\_literal(E'O'hello');

quote\_literal

---------------

'O''hello'

(1 row)

如果出现如下写法，反斜杠会写入两次。

vastbase=# SELECT quote\_literal('Ohello');

quote\_literal

---------------

E'Ohello'

(1 row)

如果参数为NULL，返回空。如果参数可能为null，通常使用函数quote\_nullable更适用。

vastbase=# SELECT quote\_literal(NULL);

quote\_literal

---------------

(1 row)

* quote\_literal(value anyelement)
* 描述：将给定的值强制转换为text，加上引号作为文本。
* 返回值类型：text
* 示例：

vastbase=# SELECT quote\_literal(42.5);

quote\_literal

---------------

'42.5'

(1 row)

如果出现如下写法，定值将进行转义。

vastbase=# SELECT quote\_literal(E'O'42.5');

quote\_literal

---------------

'0''42.5'

(1 row)

如果出现如下写法，反斜杠会写入两次。

vastbase=# SELECT quote\_literal('O42.5');

quote\_literal

---------------

E'O42.5'

(1 row)

* quote\_nullable(string text)
* 描述：返回适用于在SQL语句里当作字符串使用的形式（使用适当的引号进行界定）。
* 返回值类型：text
* 示例：

vastbase=# SELECT quote\_nullable('hello');

quote\_nullable

----------------

'hello'

(1 row)

如果出现如下写法，text文本将进行转义。

vastbase=# SELECT quote\_nullable(E'O'hello');

quote\_nullable

----------------

'O''hello'

(1 row)

如果出现如下写法，反斜杠会写入两次。

vastbase=# SELECT quote\_nullable('Ohello');

quote\_nullable

----------------

E'Ohello'

(1 row)

如果参数为NULL，返回NULL。

vastbase=# SELECT quote\_nullable(NULL);

quote\_nullable

----------------

NULL

(1 row)

* quote\_nullable(value anyelement)
* 描述：将给定的参数值转化为text，加上引号作为文本。
* 返回值类型：text
* 示例：

vastbase=# SELECT quote\_nullable(42.5);

quote\_nullable

----------------

'42.5'

(1 row)

如果出现如下写法，定值将进行转义。

vastbase=# SELECT quote\_nullable(E'O'42.5');

quote\_nullable

----------------

'O''42.5'

(1 row)

如果出现如下写法，反斜杠会写入两次。

vastbase=# SELECT quote\_nullable('O42.5');

quote\_nullable

----------------

E'O42.5'

(1 row)

如果参数为NULL，返回NULL。

vastbase=# SELECT quote\_nullable(NULL);

quote\_nullable

----------------

NULL

(1 row)

* substring\_inner(string [from int] [for int])
* 描述：截取子字符串，from int表示从第几个字符开始截取，for int表示截取几个字节。
* 返回值类型：text
* 示例：

vastbase=# select substring\_inner('adcde', 2,3);

substring\_inner

-----------------

dcd

(1 row)

* substring(string [from int] [for int])
* 描述：截取子字符串，from int表示从第几个字符开始截取，for int表示截取几个字节。
* 返回值类型：text
* 示例：

vastbase=# SELECT substring('Thomas' from 2 for 3);

substring

-----------

hom

(1 row)

* substring(string from pattern)
* 描述：截取匹配POSIX正则表达式的子字符串。如果没有匹配它返回空值，否则返回文本中匹配模式的那部分。
* 返回值类型：text
* 示例：

vastbase=# SELECT substring('Thomas' from '...$');

substring

-----------

mas

(1 row)

vastbase=# SELECT substring('foobar' from 'o(.)b');

result

--------

o

(1 row)

vastbase=# SELECT substring('foobar' from '(o(.)b)');

result

--------

oob

(1 row)

fig: **说明：**

如果POSIX正则表达式模式包含任何圆括号，那么将返回匹配第一对子表达式（对应第一个左圆括号的）的文本。如果你想在表达式里使用圆括号而又不想导致这个例外，那么你可以在整个表达式外边放上一对圆括号。

* substring(string from pattern for escape)
* 描述：截取匹配SQL正则表达式的子字符串。声明的模式必须匹配整个数据串，否则函数失败并返回空值。为了标识在成功的时候应该返回的模式部分，模式必须包含逃逸字符的两次出现，并且后面要跟上双引号（"）。匹配这两个标记之间的模式的文本将被返回。
* 返回值类型：text
* 示例：

vastbase=# SELECT substring('Thomas' from '%#"o\_a#"\_' for '#');

substring

-----------

oma

(1 row)

* rawcat(raw,raw)
* 描述：字符串拼接函数。
* 返回值类型：raw
* 示例：

vastbase=# SELECT rawcat('ab','cd');

rawcat

--------

ABCD

(1 row)

* regexp\_like(text,text,text)
* 描述：正则表达式的模式匹配函数。
* 返回值类型：bool
* 示例：

vastbase=# SELECT regexp\_like('str','[ac]');

regexp\_like

-------------

f

(1 row)

* regexp\_substr(string text, pattern text [, position int [, occurrence int [, flags text]]])
* 描述：正则表达式的抽取子串函数。与substr功能相似，正则表达式出现多个并列的括号时，也全部处理。
* 参数说明：
* string：用于匹配的源字符串。
* pattern：用于匹配的正则表达式模式串。
* position：可选参数，表示从源字符串的第几个字符开始匹配，默认值为1。
* occurrence：可选参数，表示抽取第几个满足匹配的子串，为，默认值为1。
* flags：可选参数，包含零个或多个改变函数匹配行为的单字母标记。其中：m表示按照多行模式匹配。SQL语法兼容A和B的情况下，n选项在GUC参数behavior\_compat\_options值包含aformat\_regexp\_match时，表示 . 能够匹配 'n' 字符，flags中没有指定n时，默认.不能匹配 'n' 字符；值不包含aformat\_regexp\_match时，. 默认能匹配'n'字符。n选项的含义与m选项一致。
* 返回值类型：text
* 示例：

vastbase=# SELECT regexp\_substr('str','[ac]');

regexp\_substr

---------------

(1 row)

vastbase=# SELECT regexp\_substr('foobarbaz', 'b(..)', 3, 2) AS RESULT;

result

--------

baz

(1 row)

* lregexp\_count(string text, pattern text [, position int [, flags text]])
* 描述：获取满足匹配的子串个数。
* 参数说明：
* string：用于匹配的源字符串。
* pattern：用于匹配的正则表达式模式串。
* position：表示从源字符串的第几个字符开始匹配，为可选参数，默认值为1。
* flags：可选参数，包含零个或多个改变函数匹配行为的单字母标记。其中：m表示按照多行模式匹配。SQL语法兼容A和B的情况下，n选项在GUC参数behavior\_compat\_options值包含aformat\_regexp\_match时，表示 . 能够匹配 'n' 字符，flags中没有指定n时，默认.不能匹配 'n' 字符；值不包含aformat\_regexp\_match时，. 默认能匹配'n'字符。n选项的含义与m选项一致。
* 返回值类型：int
* 示例：

vastbase=# SELECT regexp\_count('foobarbaz','b(..)', 5) AS RESULT;

result

--------

1

(1 row)

* l regexp\_instr(string text, pattern text [, position int [, occurrence int [, return\_opt int [, flags text]]]])
* 描述：获取满足匹配条件的子串位置（从1开始）。如果没有匹配的子串，则返回0。
* 参数说明：
* string：用于匹配的源字符串。
* pattern：用于匹配的正则表达式模式串。
* position：可选参数，表示从源字符串的第几个字符开始匹配，默认值为1。
* occurrence：可选参数，表示获取第occurrence个匹配子串的位置，默认值为1。
* return\_opt：可选参数，用于控制返回匹配子串的首字符位置还是尾字符位置。取值为0时，返回匹配子串的第一个字符的位置（从1开始计算），取值为大于0的值时，返回匹配子串的尾字符的下一个字符的位置。默认值为0。
* flags：可选参数，包含零个或多个改变函数匹配行为的单字母标记。其中：m表示按照多行模式匹配。SQL语法兼容A和B的情况下，n选项在GUC参数behavior\_compat\_options值包含aformat\_regexp\_match时，表示 . 能够匹配 'n' 字符，flags中没有指定n时，默认.不能匹配 'n' 字符；值不包含aformat\_regexp\_match时，. 默认能匹配'n'字符。n选项的含义与m选项一致。
* 返回值类型：int
* 示例：

vastbase=# SELECT regexp\_instr('foobarbaz','b(..)', 1, 1, 0) AS RESULT;

result

--------

4

(1 row)

vastbase=# SELECT regexp\_instr('foobarbaz','b(..)', 1, 2, 0) AS RESULT;

result

--------

7

(1 row)

* regexp\_matches(string text, pattern text [, flags text])
* 描述：返回string中所有匹配POSIX正则表达式的子字符串。如果pattern不匹配，该函数不返回行。如果模式不包含圆括号子表达式，则每一个被返回的行都是一个单一元素的文本数组，其中包括匹配整个模式的子串。如果模式包含圆括号子表达式，该函数返回一个文本数组，它的第n个元素是匹配模式的第n个圆括号子表达式的子串。
* flags参数为可选参数，包含零个或多个改变函数行为的单字母标记。i表示进行大小写无关的匹配，g表示替换每一个匹配的子字符串而不仅仅是第一个。
* fig: **须知：**
* 如果提供了最后一个参数，但参数值是空字符串（''），且数据库SQL兼容模式设置为A的情况下，会导致返回结果为空集。这是因为A兼容模式将''作为NULL处理，避免此类行为的方式有如下几种：
* 将数据库SQL兼容模式改为C；
* 不提供最后一个参数，或最后一个参数不为空字符串。
* 返回值类型：setof text[]
* 示例：

vastbase=# SELECT regexp\_matches('foobarbequebaz', '(bar)(beque)');

regexp\_matches

----------------

{bar,beque}

(1 row)

vastbase=# SELECT regexp\_matches('foobarbequebaz', 'barbeque');

regexp\_matches

----------------

{barbeque}

(1 row)

vastbase=# SELECT regexp\_matches('foobarbequebazilbarfbonk', '(b[^b]+)(b[^b]+)', 'g');

result

--------------

{bar,beque}

{bazil,barf}

(2 rows)

* regexp\_split\_to\_array(string text, pattern text [, flags text ])
* 描述：用POSIX正则表达式作为分隔符，分隔string。和regexp\_split\_to\_table相同，不过regexp\_split\_to\_array会把它的结果以一个text数组的形式返回。
* 返回值类型：text[]
* 示例：

vastbase=# SELECT regexp\_split\_to\_array('hello world', E's+');

regexp\_split\_to\_array

-----------------------

{hello,world}

(1 row)

* regexp\_split\_to\_table(string text, pattern text [, flags text])
* 描述：用POSIX正则表达式作为分隔符，分隔string。如果没有与pattern的匹配，该函数返回string。如果有至少有一个匹配，对每一个匹配它都返回从上一个匹配的末尾（或者串的开头）到这次匹配开头之间的文本。当没有更多匹配时，它返回从上一次匹配的末尾到串末尾之间的文本。
* flags参数包含零个或多个改变函数行为的单字母标记。i表示进行大小写无关的匹配。
* 返回值类型：setof text
* 示例：

vastbase=# SELECT regexp\_split\_to\_table('hello world', E's+');

regexp\_split\_to\_table

-----------------------

hello

world

(2 rows)

* repeat(string text, number int )
* 描述：将string重复number次。
* 返回值类型：text。
* 示例：

vastbase=# SELECT repeat('Pg', 4);

repeat

----------

PgPgPgPg

(1 row)

fig: **说明：**

由于数据库内存分配机制限制单次内存分配不可超过1GB，因此number最大值不应超过(1G-x)/lengthb(string) - 1。x为头信息长度，通常大于4字节，其具体值在不同的场景下存在差异。

* replace(string text, from text, to text)
* 描述：把字符串string里出现地所有子字符串from的内容替换成子字符串to的内容。
* 返回值类型：text
* 示例：

vastbase=# SELECT replace('abcdefabcdef', 'cd', 'XXX');

replace

----------------

abXXXefabXXXef

(1 row)

* replace(string, substring)
* 描述：删除字符串string里出现的所有子字符串substring的内容。
* string类型：text
* substring类型：text
* 返回值类型：text
* 示例：

vastbase=# SELECT replace('abcdefabcdef', 'cd');

replace

----------------

abefabef

(1 row)

* reverse(str)
* 描述：返回颠倒的字符串。
* 返回值类型：text
* 示例：

vastbase=# SELECT reverse('abcde');

reverse

---------

edcba

(1 row)

* right(str text, n int)
* 描述：返回字符串中的后n个字符。当n是负值时，返回除前|n|个字符以外的所有字符。
* 返回值类型：text
* 示例：

vastbase=# SELECT right('abcde', 2);

right

-------

de

(1 row)

vastbase=# SELECT right('abcde', -2);

right

-------

cde

(1 row)

* rpad(string text, length int [, fill text])
* 描述：使用填充字符fill（缺省时为空白），把string填充到length长度。如果string已经比length长则将其从尾部截断。
* 返回值类型：text
* 示例：

vastbase=# SELECT rpad('hi', 5, 'xy');

rpad

-------

hixyx

(1 row)

* rtrim(string text[,characters text])

描述：从字符串string的结尾删除只包含characters中字符（缺省是个 空白）的最长的字符串。

返回值类型：text

示例：

vastbase=# SELECT rtrim('trimxxxx', 'x');

rtrim

-------

trim

(1 row)

* substrb(text,int,int)
* 描述：提取子字符串，第一个int表示提取的起始位置，第二个表示提取几位字符。
* 返回值类型：text
* 示例：

vastbase=# SELECT substrb('string',2,3);

substrb

---------

tri

(1 row)

* substrb(text,int)
* 描述：提取子字符串，int表示提取的起始位置。
* 返回值类型：text
* 示例：

vastbase=# SELECT substrb('string',2);

substrb

---------

tring

(1 row)

* substr(bytea,from,count)
* 描述：从参数bytea中抽取子字符串。from表示抽取的起始位置，count表示抽取的子字符串长度。
* 返回值类型：text
* 示例：

vastbase=# SELECT substr('string',2,3);

substr

--------

tri

(1 row)

* string || string
* 描述：连接字符串。
* 返回值类型：text
* 示例：

vastbase=# SELECT 'MPP'||'DB' AS RESULT;

result

--------

MPPDB

(1 row)

* string || non-string或non-string || string
* 描述：连接字符串和非字符串。
* 返回值类型：text
* 示例：

vastbase=# SELECT 'Value: '||42 AS RESULT;

result

-----------

Value: 42

(1 row)

* split\_part(string text, delimiter text, field int)
* 描述：根据delimiter分隔string返回生成的第field个子字符串（从出现第一个delimiter的text为基础）。
* 返回值类型：text
* 示例：

vastbase=# SELECT split\_part('abc~@~def~@~ghi', '~@~', 2);

split\_part

------------

def

(1 row)

* strpos(string, substring)
* 描述：指定的子字符串的位置。和position(substring in string)一样，不过参数顺序相反。
* 返回值类型：int
* 示例：

vastbase=# SELECT strpos('source', 'rc');

strpos

--------

4

(1 row)

* to\_hex(number int or bigint)
* 描述：把number转换成十六进制表现形式。
* 返回值类型：text
* 示例：

vastbase=# SELECT to\_hex(2147483647);

to\_hex

----------

7fffffff

(1 row)

* translate(string text, from text, to text)
* 描述：把在string中包含的任何匹配from中字符的字符转化为对应的在to中的字符。如果from比to长，删掉在from中出现的额外的字符。
* 返回值类型：text
* 示例：

vastbase=# SELECT translate('12345', '143', 'ax');

translate

-----------

a2x5

(1 row)

* length(string)
* 描述：获取参数string中字符的数目。
* 返回值类型：integer
* 示例：

vastbase=# SELECT length('abcd');

length

--------

4

(1 row)

* lengthb(string)
* 描述：获取参数string中字节的数目。与字符集有关，同样的中文字符，在GBK与UTF8中，返回的字节数不同。
* 返回值类型：integer
* 示例：

vastbase=# SELECT lengthb('Chinese');

lengthb

---------

7

(1 row)

* substr(string,from)
* 描述：
* 从参数string中抽取子字符串。
* from表示抽取的起始位置。
* from为0时，按1处理。
* from为正数时，抽取从from到末尾的所有字符。
* from为负数时，抽取字符串的后n个字符，n为from的绝对值。
* 返回值类型：varchar
* 示例：
* from为正数时：

vastbase=# SELECT substr('ABCDEF',2);

substr

--------

BCDEF

(1 row)

From

为负数时：

vastbase=# SELECT substr('ABCDEF',-2);

substr

--------

EF

(1 row)

* substr(string,from,count)
* 描述：
* 从参数string中抽取子字符串。
* from表示抽取的起始位置。
* count表示抽取的子字符串长度。
* from为0时，按1处理。
* from为正数时，抽取从from开始的count个字符。
* from为负数时，抽取从倒数第n个开始的count个字符，n为from的绝对值。
* count小于1时，返回null。
* 返回值类型：varchar
* 示例：
* from为正数时：

vastbase=# SELECT substr('ABCDEF',2,2);

substr

--------

BC

(1 row)

from

为负数时：

vastbase=# SELECT substr('ABCDEF',-3,2);

substr

--------

DE

(1 row)

* substrb(string,from)
* 描述：该函数和SUBSTR(string,from)函数功能一致，但是计算单位为字节。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT substrb('ABCDEF',-2);

substrb

---------

EF

(1 row)

* substrb(string,from,count)
* 描述：该函数和SUBSTR(string,from,count)函数功能一致，但是计算单位为字节。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT substrb('ABCDEF',2,2);

substrb

---------

BC

(1 row)

* trim([leading |trailing |both] [characters] from string)
* 描述：从字符串string的开头、结尾或两边删除只包含characters中字符（缺省是一个空白）的最长的字符串。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT trim(BOTH 'x' FROM 'xTomxx');

btrim

-------

Tom

(1 row)

vastbase=# SELECT trim(LEADING 'x' FROM 'xTomxx');

ltrim

-------

Tomxx

(1 row)

vastbase=# SELECT trim(TRAILING 'x' FROM 'xTomxx');

rtrim

-------

xTom

(1 row)

* rtrim([string [, characters])

描述：从字符串string的结尾删除只包含characters中字符（缺省是个 空白）的最长的字符串。

返回值类型：varchar

示例：

vastbase=# SELECT rtrim('TRIMxxxx','x');

rtrim

-------

TRIM

(1 row)

* ltrim(string [, characters])
* 描述：从字符串string的开头删除只包含characters中字符（缺省是一个空白）的最长的字符串。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT ltrim('xxxxTRIM','x');

ltrim

-------

TRIM

(1 row)

* upper(string)
* 描述：把字符串转化为大写。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT upper('tom');

upper

-------

TOM

(1 row)

* lower(string)
* 描述：把字符串转化为小写。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT lower('TOM');

lower

-------

tom

(1 row)

* rpad(string varchar, length int [, fill varchar])
* 描述：使用填充字符fill（缺省时为空白），把string填充到length长度。如果string已经比length长则将其从尾部截断。
* length参数在vastbase中表示字符长度。一个汉字长度计算为一个字符。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT rpad('hi',5,'xyza');

rpad

-------

hixyz

(1 row)

vastbase=# SELECT rpad('hi',5,'abcdefg');

rpad

-------

hiabc

(1 row)

* instr(string,substring[,position,occurrence])
* 描述：从字符串string的position（缺省时为1）所指的位置开始查找并返回第occurrence（缺省时为1）次出现子串substring的位置的值。
* 当position为0时，返回0。
* 当position为负数时，从字符串倒数第n个字符往前逆向搜索。n为position的绝对值。
* 本函数以字符为计算单位，如一个汉字为一个字符。
* 返回值类型：integer
* 示例：

vastbase=# SELECT instr('corporate floor','or', 3);

instr

-------

5

(1 row)

vastbase=# SELECT instr('corporate floor','or',-3,2);

instr

-------

2

(1 row)

* initcap(string)
* 描述：将字符串中的每个单词的首字母转化为大写，其他字母转化为小写。
* 返回值类型：text
* 示例：

vastbase=# SELECT initcap('hi THOMAS');

initcap

-----------

Hi Thomas

(1 row)

* ascii(string)
* 描述：参数string的第一个字符的ASCII码。
* 返回值类型：integer
* 示例：

vastbase=# SELECT ascii('xyz');

ascii

-------

120

(1 row)

* replace(string varchar, search\_string varchar, replacement\_string varchar)
* 描述：把字符串string中所有子字符串search\_string替换成子字符串replacement\_string。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT replace('jack and jue','j','bl');

replace

----------------

black and blue

(1 row)

* lpad(string varchar, length int[, repeat\_string varchar])
* 描述：在string的左侧添上一系列的repeat\_string（缺省为空白）来组成一个总长度为n的新字符串。
* 如果string本身的长度比指定的长度length长，则本函数将把string截断并把前面长度为length的字符串内容返回。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT lpad('PAGE 1',15,'\*.');

lpad

-----------------

\*.\*.\*.\*.\*PAGE 1

(1 row)

vastbase=# SELECT lpad('hello world',5,'abcd');

lpad

-------

hello

(1 row)

* concat(str1,str2)
* 描述：将字符串str1和str2连接并返回。
* fig: **须知：**
* 数据库SQL兼容模式设置为MY的情况下，参数str1或str2为NULL会导致返回结果为NULL。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT concat('Hello', ' World!');

concat

--------------

Hello World!

(1 row)

vastbase=# SELECT concat('Hello', NULL);

concat

--------

Hello

(1 row)

* chr(integer)
* 描述：给出ASCII码的字符。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT chr(65);

chr

-----

A

(1 row)

* regexp\_substr(source\_char, pattern)
* 描述：正则表达式的抽取子串函数。SQL语法兼容A和B的情况下，GUC参数behavior\_compat\_options的值包含aformat\_regexp\_match时，. 不能匹配 'n' 字符；不包含aformat\_regexp\_match时，. 能够匹配'n'字符。
* 返回值类型：text
* 示例：

vastbase=# SELECT regexp\_substr('500 Hello World, Redwood Shores, CA', ',[^,]+,') "REGEXPR\_SUBSTR";

REGEXPR\_SUBSTR

-------------------

, Redwood Shores,

(1 row)

* regexp\_replace(string, pattern, replacement [,flags ])
* 描述：替换匹配POSIX正则表达式的子字符串。 如果没有匹配pattern，那么返回不加修改的string串。 如果有匹配，则返回的string串里面的匹配子串将被replacement串替换掉。
* replacement串可以包含n， 其中n是1 到9， 表明string串里匹配模式里第n个圆括号子表达式的子串应该被插入，并且它可以包含&表示应该插入匹配整个模式的子串。
* 可选的flags参数包含零个或多个改变函数行为的单字母标记。i表示进行大小写无关的匹配，g表示替换每一个匹配的子字符串而不仅仅是第一个。m表示按照多行模式匹配。SQL语法兼容A和B的情况下，n选项在GUC参数behavior\_compat\_options的值包含aformat\_regexp\_match时，表示 . 能够匹配 'n' 字符，flags中没有指定n时，默认.不能匹配 'n' 字符；值不包含aformat\_regexp\_match时，. 默认能匹配'n'字符。n 选项的含义与 m 选项一致。
* 返回值类型：varchar
* 示例：

vastbase=# SELECT regexp\_replace('Thomas', '.[mN]a.', 'M');

regexp\_replace

----------------

ThM

(1 row)

vastbase=# SELECT regexp\_replace('foobarbaz','b(..)', E'X1Y', 'g') AS RESULT;

result

-------------

fooXarYXazY

(1 row)

* concat\_ws(sep text, str"any" [, str"any" [, ...] ])
* 描述：以第一个参数为分隔符，链接第二个以后的所有参数。NULL参数被忽略。
* fig: **须知：**
* 如果第一个参数值是NULL，会导致返回结果为NULL。
* 如果第一个参数值是空字符串（''），且数据库SQL兼容模式设置为A的情况下，会导致返回结果为NULL。这是因为A兼容模式>将''作为NULL处理，避免此类行为，可以将数据库SQL兼容模式改为B、C或者PG。
* 返回值类型：text
* 示例：

vastbase=# SELECT concat\_ws(',', 'ABCDE', 2, NULL, 22);

concat\_ws

------------

ABCDE,2,22

(1 row)

* nlssort(string text, sort\_method text)
* 描述：以sort\_method指定的排序方式返回字符串在该排序模式下的编码值，该编码值可用于排序，其决定了string在这种排序模式下的先后位置。目前支持的sort\_method为'nls\_sort=schinese\_pinyin\_m'和'nls\_sort=generic\_m\_ci'。其中，'nls\_sort=generic\_m\_ci'仅支持纯英文不区分大小写排序。
* string类型：text
* sort\_method类型：text
* 返回值类型：text
* 示例：

vastbase=# SELECT nlssort('A', 'nls\_sort=schinese\_pinyin\_m');

nlssort

----------------

01EA0000020006

(1 row)

vastbase=# SELECT nlssort('A', 'nls\_sort=generic\_m\_ci');

nlssort

----------------

01EA000002

(1 row)

* convert(string bytea, src\_encoding name, dest\_encoding name)
* 描述：以dest\_encoding指定的目标编码方式转化字符串bytea。src\_encoding指定源编码方式，在该编码下，string必须是合法的。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT convert('text\_in\_utf8', 'UTF8', 'GBK');

convert

----------------------------

x746578745f696e5f75746638

(1 row)

fig: **说明：**

如果源编码格式到目标编码格式的转化规则不存在，则字符串不进行任何转换直接返回，如GBK和LATIN1之间的转换规则是不存在的，具体转换规则可以通过查看系统表pg\_conversion获得。示例：

vastbase=# show server\_encoding;

server\_encoding

-----------------

LATIN1

(1 row)

vastbase=# SELECT convert\_from('some text', 'GBK');

convert\_from

--------------

some text

(1 row)

db\_latin1=# SELECT convert\_to('some text', 'GBK');

convert\_to

----------------------

x736f6d652074657874

(1 row)

db\_latin1=# SELECT convert('some text', 'GBK', 'LATIN1');

convert

----------------------

x736f6d652074657874

(1 row)

* convert\_from(string bytea, src\_encoding name)
* 描述：以数据库的编码方式转化字符串bytea。
* src\_encoding指定源编码方式，在该编码下，string必须是合法的。
* 返回值类型：text
* 示例：

vastbase=# SELECT convert\_from('text\_in\_utf8', 'UTF8');

convert\_from

--------------

text\_in\_utf8

(1 row)

* convert\_to(string text, dest\_encoding name)
* 描述：将字符串转化为dest\_encoding的编码格式。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT convert\_to('some text', 'UTF8');

convert\_to

----------------------

x736f6d652074657874

(1 row)

* string [NOT] LIKE pattern [ESCAPE escape-character]
* 描述：模式匹配函数。
* 如果pattern不包含百分号或者下划线，该模式只代表它本身，这时候LIKE的行为就像等号操作符。在pattern里的下划线（\_）匹配任何单个字符；而一个百分号（%）匹配零或多个任何字符。
* 要匹配下划线或者百分号本身，在pattern里相应的字符必须前导逃逸字符。缺省的逃逸字符是反斜杠，但是用户可以用ESCAPE子句指定一个。要匹配逃逸字符本身，写两个逃逸字符。
* 返回值类型：Boolean
* 示例：

vastbase=# SELECT 'AA\_BBCC' LIKE '%A@\_B%' ESCAPE '@' AS RESULT;

result

--------

t

(1 row)

vastbase=# SELECT 'AA\_BBCC' LIKE '%A@\_B%' AS RESULT;

result

--------

f

(1 row)

vastbase=# SELECT 'AA@\_BBCC' LIKE '%A@\_B%' AS RESULT;

result

--------

t

(1 row)

* REGEXP\_LIKE(source\_string, pattern [, match\_parameter])
* 描述：正则表达式的模式匹配函数。
* source\_string为源字符串，pattern为正则表达式匹配模式。 match\_parameter为匹配选项，可取值为：
* 'i'：大小写不敏感。
* 'c'：大小写敏感。
* 'n'：允许正则表达式元字符“.”匹配换行符。
* 'm'：将source\_string视为多行。
* 若忽略match\_parameter选项，默认为大小写敏感，“.”不匹配换行符，source\_string视为单行。
* 返回值类型：Boolean
* 示例：

vastbase=# SELECT regexp\_like('ABC', '[A-Z]');

regexp\_like

-------------

t

(1 row)

vastbase=# SELECT regexp\_like('ABC', '[D-Z]');

regexp\_like

-------------

f

(1 row)

vastbase=# SELECT regexp\_like('ABC', '[a-z]','i');

regexp\_like

-------------

t

(1 row)

* format(formatstr text [, str"any" [, ...] ])
* 描述：格式化字符串。
* 返回值类型：text
* 示例：

vastbase=# SELECT format('Hello %s, %1$s', 'World');

format

--------------------

Hello World, World

(1 row)

* md5(string)
* 描述：将string使用MD5加密，并以16进制数作为返回值。
* fig: **说明：**
* MD5加密算法安全性低，存在安全风险，不建议使用。
* 返回值类型：text
* 示例：

vastbase=# SELECT md5('ABC');

md5

----------------------------------

902fbdd2b1df0c4f70b4a5d23525e932

(1 row)

* decode(string text, format text)
* 描述：将二进制数据从文本数据中解码。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT decode('MTIzAAE=', 'base64');

decode

--------------

x3132330001

(1 row)

* similar\_escape(pat text, esc text)
* 描述：将一个 SQL:2008风格的正则表达式转换为POSIX风格。
* 返回值类型：text
* 示例：

vastbase=# select similar\_escape('s+ab','2');

similar\_escape

----------------

^(?:s+ab)$

(1 row)

* encode(data bytea, format text)
* 描述：将二进制数据编码为文本数据。
* 返回值类型：text
* 示例：

vastbase=# SELECT encode(E'123000001', 'base64');

encode

----------

MTIzAAE=

(1 row)

fig: **说明：**

* 若字符串中存在换行符，如字符串由一个换行符和一个空格组成，在vastbase中LENGTH和LENGTHB的值为2。
* 对于CHAR(n) 类型，vastbase中n是指字符个数。因此，对于多字节编码的字符集， LENGTHB函数返回的长度可能大于n。
* vastbase支持多种类型的数据库，目前有4种，分别是A类型、B类型、C类型以及PG类型。在不指定数据库类型时，我们的数据库默认是A类型，A的词法分析器与另外三种不一样，在A中空字符串会被当作是NULL。所以，当使用A类型的数据库时，假如上述字符操作函数中有空字符串作为参数，会出现没有输出的情况。例如：

vastbase=# SELECT translate('12345','123','');

translate

-----------

(1 row)

这是因为内核在调用相应的函数进行处理前，会判断所输入的参数中是否含有NULL，假如有，则不会调用相应的函数，因此会没有输出。而在PG模式下，字符串的处理方式与postgresql保持一致，因此不会有上述问题产生。

#### 数组函数和操作符

**数组操作符**

数组比较是使用默认的B-tree比较函数对所有元素逐一进行比较的。多维数组的元素按照行顺序进行访问。如果两个数组的内容相同但维数不等，决定排序顺序的首要因素是维数。

**数组函数**

| **函数名** | **描述** | **返回类型** |
| --- | --- | --- |
| array\_append(anyarray, anyelement) | 向数组末尾添加元素，只支持一维数组。 | anyarray |
| array\_prepend(anyelement, anyarray) | 向数组开头添加元素，只支持一维数组。 | anyarray |
| array\_cat(anyarray, anyarray) | 连接两个数组，支持多维数组。 | anyarray |
| array\_ndims(anyarray) | 返回数组的维数。 | int |
| array\_dims(anyarray) | 返回数组各个维度中的低位下标值和高位下标值。 | text |
| array\_length(anyarray, int) | 返回指定数组维度的长度。int为指定数组维度。 | int |
| array\_lower(anyarray, int | 返回指定数组维数的下界。int为指定数组维度。 | int |
| array\_upper(anyarray, int) | 返回指定数组维数的上界。int为指定数组维度。 | int |
| array\_to\_string(anyarray, text [, text] | 使用第一个text作为数组的新分隔符，使用第二个text替换数组值为null的值。 | text |
| string\_to\_array(text, text [, text] | 使用第二个text指定分隔符，使用第三个可选的text作为NULL值替换模板，如果分隔后的子串与第三个可选的text完全匹配，则将其替换为NULL。 | text[] |
| unnest(anyarray) | 扩大一个数组为一组行。 | setof anyelement |
| array\_position(anyarray,anyelement,[,integer]) | 实现在数组搜索元素，返回该元素在数组中第一次出现的下标 | int |

**示例**

* array\_append(anyarray, anyelement)

select array\_append(ARRAY[1,2], 3) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

---------

{1,2,3}

(1 row)

* array\_prepend(anyelement, anyarray)

select array\_prepend(1, ARRAY[2,3]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

---------

{1,2,3}

(1 row)

* array\_cat(anyarray, anyarray)

示例1：

select array\_cat(ARRAY[1,2,3], ARRAY[4,5]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

-------------

{1,2,3,4,5}

(1 row)

示例2：

select array\_cat(ARRAY[[1,2],[4,5]], ARRAY[6,7]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

---------------------

{{1,2},{4,5},{6,7}}

(1 row)

* array\_ndims(anyarray)

select array\_ndims(ARRAY[[1,2,3], [4,5,6]]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

2

(1 row)

* array\_dims(anyarray)

select array\_dims(ARRAY[[1,2,3], [4,5,6]]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

------------

[1:2][1:3]

(1 row)

* array\_length(anyarray, int)

示例1：

select array\_length(array[1,2,3], 1) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

3

(1 row)

示例2：

select array\_length(array[[1,2,3],[4,5,6]], 2) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

3

(1 row)

* array\_lower(anyarray, int)

select array\_lower('[0:2]={1,2,3}'::int[], 1) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

0

(1 row)

* array\_upper(anyarray, int)

select array\_upper(ARRAY[1,8,3,7], 1) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

4

(1 row)

* array\_to\_string(anyarray, text [, text])

描述：

返回类型：

select array\_to\_string(ARRAY[1, 2, 3, NULL, 5], ',', '\*') AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

-----------

1,2,3,\*,5

(1 row)

在array\_to\_string中，如果省略null字符串参数或为NULL，运算中将跳过在数组中的任何null元素，并且不会在输出字符串中出现。

* string\_to\_array(text, text [, text])

示例1：

select string\_to\_array('xx~^~yy~^~zz', '~^~', 'yy') AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------------

{xx,NULL,zz}

(1 row)

示例2：

select string\_to\_array('xx~^~yy~^~zz', '~^~', 'y') AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

------------

{xx,yy,zz}

(1 row)

在string\_to\_array中，如果分隔符参数是NULL，输入字符串中的每个字符将在结果数组中变成一个独立的元素。如果分隔符是一个空白字符串，则整个输入的字符串将变为一个元素的数组。否则输入字符串将在每个分隔字符串处分开。如果省略null字符串参数或为NULL，将字符串中没有输入内容的子串替换为NULL。

* unnest(anyarray)

select unnest(ARRAY[1,2]) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

1

2

(2 rows)

* array\_position(anyarray,anyelement,[,integer])

SELECT array\_position(array[1,4,2,5,7],2) AS RESULT;

当结果显示如下信息，则表示函数调用成功。

result

--------

3

(1 row)

#### 聚集函数

**聚集函数**

* sum(expression)
* 描述：所有输入行的expression总和。
* 返回类型：
* 通常情况下输入数据类型和输出数据类型是相同的，但以下情况会发生类型转换：
* 对于SMALLINT或INT输入，输出类型为BIGINT。
* 对于BIGINT输入，输出类型为NUMBER 。
* 对于浮点数输入，输出类型为DOUBLE PRECISION。
* 示例：

vastbase=# SELECT SUM(ss\_ext\_tax) FROM tpcds.STORE\_SALES;

sum

---------------

213267594.69

(1 row)

* max(expression)
* 描述：所有输入行中expression的最大值。
* 参数类型：任意数组、数值、字符串、日期/时间类型、IPV4和IPV6地址（INET型和CIDR型）。
* 返回类型：与参数数据类型相同
* 示例：

vastbase=# SELECT MAX(inv\_quantity\_on\_hand) FROM tpcds.inventory;

* min(expression)
* 描述：所有输入行中expression的最小值。
* 参数类型：任意数组、数值、字符串、日期/时间类型、IPV4和IPV6地址（INET型和CIDR型）。
* 返回类型：与参数数据类型相同
* 示例：

vastbase=# SELECT MIN(inv\_quantity\_on\_hand) FROM tpcds.inventory;

min

-----

0

(1 row)

* avg(expression)
* 描述：所有输入值的均值（算术平均）。
* 返回类型：
* 对于任何整数类型输入，结果都是NUMBER类型。
* 对于任何浮点输入，结果都是DOUBLE PRECISION类型。
* 否则和输入数据类型相同。
* 示例：

vastbase=# SELECT AVG(inv\_quantity\_on\_hand) FROM tpcds.inventory;

avg

-----------------------

500.0387129084044604

(1 row)

* count(expression)
* 描述：返回表中满足expression不为NULL的行数。
* 返回类型：BIGINT
* 示例：

vastbase=# SELECT COUNT(inv\_quantity\_on\_hand) FROM tpcds.inventory;

count

-----------

11158087

(1 row)

* count(\*)
* 描述：返回表中的记录行数。
* 返回类型：BIGINT
* 示例：

vastbase=# SELECT COUNT(\*) FROM tpcds.inventory;

count

-----------

11745000

(1 row)

* median(expression) [over (query partition clause)]
* 描述：返回表达式的中位数，计算时NULL将会被median函数忽略。可以使用distinct关键字排除表达式中的重复记录。输入expression的数据类型可以是数值类型（包括integer、 double、bigint等），也可以是interval类型。其他数据类型不支持求取中位数。
* 返回类型：double或interval类型
* 示例：

SELECT MEDIAN(id) FROM (values(1), (2), (3), (4), (null)) test(id);

median

---------

2.5

(1 row)

* array\_agg(expression)
* 描述：将所有输入值（包括空）连接成一个数组。
* 返回类型：参数类型的数组。
* 示例：

vastbase=# SELECT ARRAY\_AGG(sr\_fee) FROM tpcds.store\_returns WHERE sr\_customer\_sk = 2;

array\_agg

----------------

{22.18,63.21}

(1 row)

* string\_agg(expression, delimiter)
* 描述：将输入值连接成为一个字符串，用分隔符分开。
* 返回类型：和参数数据类型相同。
* 示例：

vastbase=# SELECT string\_agg(sr\_item\_sk, ',') FROM tpcds.store\_returns where sr\_item\_sk < 3;

string\_agg

----------------------------------------------------------------------------------

-------------------------------

1,2,1,2,2,1,1,2,2,1,2,1,2,1,1,1,2,1,1,1,1,1,2,1,1,1,1,1,2,2,1,1,1,1,1,1,1,1,1,2,

2,1,1,1,1,1,1,2,2,1,1,2,1,1,1

(1 row)

* listagg(expression [, delimiter]) WITHIN GROUP(ORDER BY order-list)
* 描述：将聚集列数据按WITHIN GROUP指定的排序方式排列，并用delimiter指定的分隔符拼接成一个字符串。
* expression：必选。指定聚集列名或基于列的有效表达式，不支持DISTINCT关键字和VARIADIC参数。
* delimiter：可选。指定分隔符，可以是字符串常数或基于分组列的确定性表达式，缺省时表示分隔符为空。
* order-list：必选。指定分组内的排序方式。
* 返回类型：text
* 示例：
* 聚集列是文本字符集类型。

vastbase=# SELECT deptno, listagg(ename, ',') WITHIN GROUP(ORDER BY ename) AS employees FROM emp GROUP BY deptno;

deptno | employees

---------+--------------------------------------

10 | CLARK,KING,MILLER

20 | ADAMS,FORD,JONES,SCOTT,SMITH

30 | ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD

(3 rows)

聚集列是整型。

vastbase=# SELECT deptno, listagg(mgrno, ',') WITHIN GROUP(ORDER BY mgrno NULLS FIRST) AS mgrnos FROM emp GROUP BY deptno;

deptno | mgrnos

---------+-------------------------------

10 | 7782,7839

20 | 7566,7566,7788,7839,7902

30 | 7698,7698,7698,7698,7698,7839

(3 rows)

聚集列是浮点类型。

vastbase=# SELECT job, listagg(bonus, '($); ') WITHIN GROUP(ORDER BY bonus DESC) || '($)' AS bonus FROM emp GROUP BY job;

job | bonus

-------------+-------------------------------------------------

CLERK | 10234.21($); 2000.80($); 1100.00($); 1000.22($)

PRESIDENT | 23011.88($)

ANALYST | 2002.12($); 1001.01($)

MANAGER | 10000.01($); 2399.50($); 999.10($)

SALESMAN | 1000.01($); 899.00($); 99.99($); 9.00($)

(5 rows)

聚集列是时间类型。

vastbase=# SELECT deptno, listagg(hiredate, ', ') WITHIN GROUP(ORDER BY hiredate DESC) AS hiredates FROM emp GROUP BY deptno;

deptno | hiredates

---------+------------------------------------------------------------------------------------------------------------------------------

10 | 1982-01-23 00:00:00, 1981-11-17 00:00:00, 1981-06-09 00:00:00

20 | 2001-04-02 00:00:00, 1999-12-17 00:00:00, 1987-05-23 00:00:00, 1987-04-19 00:00:00, 1981-12-03 00:00:00

30 | 2015-02-20 00:00:00, 2010-02-22 00:00:00, 1997-09-28 00:00:00, 1981-12-03 00:00:00, 1981-09-08 00:00:00, 1981-05-01 00:00:00

(3 rows)

聚集列是时间间隔类型。

vastbase=# SELECT deptno, listagg(vacationTime, '; ') WITHIN GROUP(ORDER BY vacationTime DESC) AS vacationTime FROM emp GROUP BY deptno;

deptno | vacationtime

---------+------------------------------------------------------------------------------------

10 | 1 year 30 days; 40 days; 10 days

20 | 70 days; 36 days; 9 days; 5 days

30 | 1 year 1 mon; 2 mons 10 days; 30 days; 12 days 12:00:00; 4 days 06:00:00; 24:00:00

(3 rows)

分隔符缺省时，默认为空。

vastbase=# SELECT deptno, listagg(job) WITHIN GROUP(ORDER BY job) AS jobs FROM emp GROUP BY deptno;

deptno | jobs

---------+----------------------------------------------

10 | CLERKMANAGERPRESIDENT

20 | ANALYSTANALYSTCLERKCLERKMANAGER

30 | CLERKMANAGERSALESMANSALESMANSALESMANSALESMAN

(3 rows)

listagg作为窗口函数时，OVER子句不支持ORDER BY的窗口排序，listagg列为对应分组的有序聚集。

vastbase=# SELECT deptno, mgrno, bonus, listagg(ename,'; ') WITHIN GROUP(ORDER BY hiredate) OVER(PARTITION BY deptno) AS employees FROM emp;

deptno | mgrno | bonus | employees

---------+-------+----------+-------------------------------------------

10 | 7839 | 10000.01 | CLARK; KING; MILLER

10 | | 23011.88 | CLARK; KING; MILLER

10 | 7782 | 10234.21 | CLARK; KING; MILLER

20 | 7566 | 2002.12 | FORD; SCOTT; ADAMS; SMITH; JONES

20 | 7566 | 1001.01 | FORD; SCOTT; ADAMS; SMITH; JONES

20 | 7788 | 1100.00 | FORD; SCOTT; ADAMS; SMITH; JONES

20 | 7902 | 2000.80 | FORD; SCOTT; ADAMS; SMITH; JONES

20 | 7839 | 999.10 | FORD; SCOTT; ADAMS; SMITH; JONES

30 | 7839 | 2399.50 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

30 | 7698 | 9.00 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

30 | 7698 | 1000.22 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

30 | 7698 | 99.99 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

30 | 7698 | 1000.01 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

30 | 7698 | 899.00 | BLAKE; TURNER; JAMES; MARTIN; WARD; ALLEN

(14 rows)

* covar\_pop(Y, X)
* 描述：总体协方差。
* 返回类型：double precision
* 示例：

vastbase=# SELECT COVAR\_POP(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

covar\_pop

-------------------

829.749627587403

(1 row)

* covar\_samp(Y, X)
* 描述：样本协方差。
* 返回类型：double precision
* 示例：

vastbase=# SELECT COVAR\_SAMP(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

covar\_samp

-------------------

830.052235037289

(1 row)

* stddev\_pop(expression)
* 描述：总体标准差。
* 返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。
* 示例：

vastbase=# SELECT STDDEV\_POP(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

stddev\_pop

-------------------

289.224294957556

(1 row)

* stddev\_samp(expression)
* 描述：样本标准差。
* 返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。
* 示例：

vastbase=# SELECT STDDEV\_SAMP(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

stddev\_samp

-------------------

289.224359757315

(1 row)

* var\_pop(expression)
* 描述：总体方差（总体标准差的平方）
* 返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。
* 示例：

vastbase=# SELECT VAR\_POP(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

var\_pop

---------------------

83650.692793695475

(1 row)

* var\_samp(expression)
* 描述：样本方差（样本标准差的平方）
* 返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。
* 示例：

vastbase=# SELECT VAR\_SAMP(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

var\_samp

---------------------

83650.730277028768

(1 row)

* bit\_and(expression)
* 描述：所有非NULL输入值的按位与(AND)，如果全部输入值皆为NULL，那么结果也为NULL 。
* 返回类型：和参数数据类型相同。
* 示例：

vastbase=# SELECT BIT\_AND(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

bit\_and

----------

0

(1 row)

* bit\_or(expression)
* 描述：所有非NULL输入值的按位或(OR)，如果全部输入值皆为NULL，那么结果也为NULL。
* 返回类型：和参数数据类型相同
* 示例：

vastbase=# SELECT BIT\_OR(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

bit\_or

---------

1023

(1 row)

* bool\_and(expression)
* 描述：如果所有输入值都是真，则为真，否则为假。
* 返回类型：bool
* 示例：

vastbase=# SELECT bool\_and(100 <2500);

bool\_and

-----------

t

(1 row)

* bool\_or(expression)
* 描述：如果所有输入值只要有一个为真，则为真，否则为假。
* 返回类型：bool
* 示例：

vastbase=# SELECT bool\_or(100 <2500);

bool\_or

-----------

t

(1 row)

* corr(Y, X)
* 描述：相关系数
* 返回类型：double precision
* 示例：

vastbase=# SELECT CORR(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

corr

--------------------

.0381383624904186

(1 row)

* every(expression)
* 描述：等效于bool\_and。
* 返回类型：bool
* 示例：

vastbase=# SELECT every(100 <2500);

every

--------

t

(1 row)

* regr\_avgx(Y, X)
* 描述：自变量的平均值 (sum(X)/N)
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_AVGX(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_avgx

-------------------

578.606576740795

(1 row)

* regr\_avgy(Y, X)
* 描述：因变量的平均值 (sum(Y)/N)
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_AVGY(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_avgy

-------------------

50.0136711629602

(1 row)

* regr\_count(Y, X)
* 描述：两个表达式都不为NULL的输入行数。
* 返回类型：bigint
* 示例：

vastbase=# SELECT REGR\_COUNT(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_count

-------------

2743

(1 row)

* regr\_intercept(Y, X)
* 描述：根据所有输入的点(X, Y)按照最小二乘法拟合成一个线性方程，然后返回该直线的Y轴截距。
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_INTERCEPT(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_intercept

-------------------

49.2040847848607

(1 row)

* regr\_r2(Y, X)
* 描述：相关系数的平方。
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_R2(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_r2

---------------------

.00145453469345058

(1 row)

* regr\_slope(Y, X)
* 描述：根据所有输入的点(X, Y)按照最小二乘法拟合成一个线性方程， 然后返回该直线的斜率。
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_SLOPE(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_slope

---------------------

.00139920009665259

(1 row)

* regr\_sxx(Y, X)
* 描述：sum(X^2) - sum(X)^2/N （自变量的“平方和”）
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_SXX(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_sxx

-------------------

1626645991.46135

(1 row)

* regr\_sxy(Y, X)
* 描述：sum(X\*Y) - sum(X) \* sum(Y)/N （自变量和因变量的“乘方积”）
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_SXY(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_sxy

-------------------

2276003.22847225

(1 row)

* regr\_syy(Y, X)
* 描述：sum(Y^2) - sum(Y)^2/N（因变量的"平方和"）
* 返回类型：double precision
* 示例：

vastbase=# SELECT REGR\_SYY(sr\_fee, sr\_net\_loss) FROM tpcds.store\_returns WHERE sr\_customer\_sk < 1000;

regr\_syy

------------------

2189417.6547314

(1 row)

* stddev(expression)
* 描述：stddev\_samp的别名。
* 返回类型：对于浮点类型的输入返回double precision，其他输入返回numeric。
* 示例：

vastbase=# SELECT STDDEV(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

stddev

-------------------

289.224359757315

(1 row)

* variance(expexpression,ression)
* 描述：var\_samp的别名。
* 返回类型：对于浮点类型的输入返回double precision类型，其他输入返回numeric类型。
* 示例：

vastbase=# SELECT VARIANCE(inv\_quantity\_on\_hand) FROM tpcds.inventory WHERE inv\_warehouse\_sk = 1;

variance

---------------------

83650.730277028768

(1 row)

* delta
* 描述：返回当前行和前一行的差值。
* 参数：numeric
* 返回值类型：numeric
* checksum(expression)
* 描述：返回所有输入值的CHECKSUM值。使用该函数可以用来验证Vastabse数据库（不支持Vastbase之外的其他数据库）的备份恢复或者数据迁移操作前后表中的数据是否相同。在备份恢复或者数据迁移操作前后都需要用户通过手工执行SQL命令的方式获取执行结果，通过对比获取的执行结果判断操作前后表中的数据是否相同。
* image1
* 对于大表，CHECKSUM函数可能会需要很长时间。
* 如果某两表的CHECKSUM值不同，则表明两表的内容是不同的。由于CHECKSUM函数中使用散列函数不能保证无冲突，因此两个不同内容的表可能会得到相同的CHECKSUM值，存在这种情况的可能性较小。对于列进行的CHECKSUM也存在相同的情况。
* 对于时间类型timestamp, timestamptz和smalldatetime，计算CHECKSUM值时请确保时区设置一致。
* 若计算某列的CHECKSUM值，且该列类型可以默认转为TEXT类型，则expression为列名。
* 若计算某列的CHECKSUM值，且该列类型不能默认转为TEXT类型，则expression为列名::TEXT。
* 若计算所有列的CHECKSUM值，则expression为表名::TEXT。

可以默认转换为TEXT类型的类型包括：char、name、 int8、 int2、 int1、 int4、 raw、 pg\_node\_tree、 float4、 float8、 bpchar、 varchar、 nvarchar、 nvarchar2、 date、 timestamp、 timestamptz、 numeric、 smalldatetime，其他类型需要强制转换为TEXT。

* 返回类型：numeric。
* 示例：
* 表中可以默认转为TEXT类型的某列的CHECKSUM值。

vastbase=# SELECT CHECKSUM(inv\_quantity\_on\_hand) FROM tpcds.inventory;

checksum

-------------------

24417258945265247

(1 row)

表中不能默认转为TEXT类型的某列的CHECKSUM值。注意此时CHECKSUM参数是列名::TEXT。

vastbase=# SELECT CHECKSUM(inv\_quantity\_on\_hand::TEXT) FROM tpcds.inventory;

checksum

-------------------

24417258945265247

(1 row)

表中所有列的CHECKSUM值。注意此时CHECKSUM参数是表名::TEXT，且表名前不加Schema。

vastbase=# SELECT CHECKSUM(inventory::TEXT) FROM tpcds.inventory;

checksum

------------------

25223696246875800

(1 row)

* first(anyelement)
* 描述：返回第一个非NULL输入。
* 返回类型：anyelement

vastbase=# select \* from tba;

name

------

A

A

D

(4 rows)

vastbase=# select first(name) from tba;

first

------

A

(1 rows)

* last(anyelement)
* 描述：返回最后一个非NULL输入。
* 返回类型：anyelement

vastbase=# select \* from tba;

name

------

A

A

D

(4 rows)

vastbase=# select last(name) from tba;

last

------

D

(1 rows)

* mode() within group (order by value anyelement)
* 描述：返回某列中出现频率最高的值，如果多个值频率相同，则返回最小的那个值。排序方式和该列类型的默认排序方式相同。其中value为输入参数，可以为任意类型。
* 返回类型：与输入参数类型相同。
* 示例：

vastbase=# select mode() within group (order by value) from (values(1, 'a'), (2, 'b'), (2, 'c')) v(value, tag);

mode

-------

2

(1 row)

vastbase=# select mode() within group (order by tag) from (values(1, 'a'), (2, 'b'), (2, 'c')) v(value, tag);

mode

-------

a

(1 row)

* json\_agg(any)
* 描述：将值聚集为json数组。
* 返回类型：array-json
* 示例：

vastbase=#create table t2(c1 int,c2 int);

vastbase=#insert into t2 values(1,1),(2,2),(3,3),(4,4),(4,4),(5,5);

vastbase=# select \* from t2;

c1 | c2

----+----

1 | 1

2 | 2

3 | 3

4 | 4

4 | 4

5 | 5

(6 rows)

vastbase=# select c1, json\_agg(c2) c2 from t2 group by c1;

c1 | c2

----+--------

1 | [1]

4 | [4, 4]

3 | [3]

5 | [5]

2 | [2]

(5 rows)

* json\_object\_agg(any, any)
* 描述：将值聚集为json对象。
* 返回类型：object-json
* 示例：

vastbase=# select \* from t2;

c1 | c2

-----+----

1 | 1

2 | 2

3 | 3

4 | 4

4 | 4

5 | 5

(6 rows)

vastbase=# select json\_object\_agg(c1, c2) from t1 group by c1 order by c1;

json\_object\_agg

------------------

{ "1" : 1 }

{ "2" : 2 }

{ "3" : 3 }

(3 rows)

#### 系统管理函数

##### 配置设置函数

配置设置函数是可以用于查询以及修改运行时配置参数的函数。

* current\_setting(setting\_name)

描述：当前的设置值。

返回值类型：text

备注：current\_setting用于以查询形式获取setting\_name的当前值。和SQL语句SHOW是等效的。比如：

vastbase=# SELECT current\_setting('datestyle');

current\_setting

-----------------

ISO, MDY

(1 row)

* set\_working\_grand\_version\_num\_manually(tmp\_version)

描述：通过切换授权版本号来更新和升级数据库的新特性。

返回值类型：void

* shell\_in(type)

描述： 为shell类型输入路由（那些尚未填充的类型）。

返回值类型：void

* shell\_out(type)

描述：为shell 类型输出路由（那些尚未填充的类型）。

返回值类型：void

* set\_config(setting\_name, new\_value, is\_local)

描述：设置参数并返回新值。

返回值类型：text

备注：set\_config将参数setting\_name设置为new\_value。如果is\_local为true，则new\_value将只应用于当前事务。如果希望new\_value应用于当前会话，可以使用false，和SQL语句SET是等效的。例如：

vastbase=# SELECT set\_config('log\_statement\_stats', 'off', false);

set\_config

------------

off

(1 row)

##### 通用文件访问函数

通用文件访问函数提供了对数据库服务器上的文件的本地访问接口。只有Vastbase目录和log\_directory目录里面的文件可以访问。使用相对路径访问Vastbase目录里面的文件，以及匹配log\_directory配置而设置的路径访问日志文件。只有数据库初始化用户才能使用这些函数。

* pg\_ls\_dir(dirname text)

描述：列出目录中的文件。

返回值类型：setof text

备注：pg\_ls\_dir返回指定目录里面的除了特殊项“.”和“..”之外所有名称。

示例：

vastbase=# SELECT pg\_ls\_dir('./');

pg\_ls\_dir

----------------------

.postgresql.conf.swp

postgresql.conf

pg\_tblspc

PG\_VERSION

pg\_ident.conf

core

server.crt

pg\_serial

pg\_twophase

postgresql.conf.lock

pg\_stat\_tmp

pg\_notify

pg\_subtrans

pg\_ctl.lock

pg\_xlog

pg\_clog

base

pg\_snapshots

postmaster.opts

postmaster.pid

server.key.rand

server.key.cipher

pg\_multixact

pg\_errorinfo

server.key

pg\_hba.conf

pg\_replslot

.pg\_hba.conf.swp

cacert.pem

pg\_hba.conf.lock

global

gaussdb.state

(32 rows)

* pg\_read\_file(filename text, offset bigint, length bigint)

描述：返回一个文本文件的内容。

返回值类型：text

备注：pg\_read\_file返回一个文本文件的一部分，从offset开始，最多返回length字节（如果先达到文件结尾，则小于这个数值）。如果offset是负数，则它是相对于文件结尾回退的长度。如果省略了offset和length，则返回整个文件。

示例：

vastbase=# SELECT pg\_read\_file('postmaster.pid',0,100);

pg\_read\_file

---------------------------------------

53078 +

/srv/BigData/hadoop/data1/dbnode+

1500022474 +

8000 +

/var/run/FusionInsight +

localhost +

2

(1 row)

* pg\_read\_binary\_file(filename text [, offset bigint, length bigint,missing\_ok boolean])

描述：返回一个二进制文件的内容。

返回值类型：bytea

备注：pg\_read\_binary\_file的功能与pg\_read\_file类似，除了结果的返回值为bytea类型不一致，相应地不会执行编码检查。与convert\_from函数结合，这个函数可以用来读取用指定编码的一个文件。

vastbase=# SELECT convert\_from(pg\_read\_binary\_file('filename'), 'UTF8');

* pg\_stat\_file(filename text)

描述：返回一个文本文件的状态信息。

返回值类型：record

备注：pg\_stat\_file返回一条记录，其中包含：文件大小、最后访问时间戳、最后更改时间戳、最后文件状态修改时间戳以及标识传入参数是否为目录的Boolean值。典型的用法：

vastbase=# SELECT \* FROM pg\_stat\_file('filename');

vastbase=# SELECT (pg\_stat\_file('filename')).modification;

示例：

vastbase=# SELECT convert\_from(pg\_read\_binary\_file('postmaster.pid'), 'UTF8');

convert\_from

--------------------------------------

4881 +

/srv/BigData/gaussdb/data1/dbnode+

1496308688 +

25108 +

/opt/user/Bigdata/gaussdb/gaussdb\_tmp +

\* +

25108001 43352069 +

(1 row)

vastbase=# SELECT \* FROM pg\_stat\_file('postmaster.pid');

size | access | modification | change

| creation | isdir

------+------------------------+------------------------+------------------------

+----------+-------

117 | 2017-06-05 11:06:34+08 | 2017-06-01 17:18:08+08 | 2017-06-01 17:18:08+08

| | f

(1 row)

vastbase=# SELECT (pg\_stat\_file('postmaster.pid')).modification;

modification

------------------------

2017-06-01 17:18:08+08

(1 row)

##### 服务器信号函数

服务器信号函数向其他服务器进程发送控制信号。只有系统管理员才能使用这些函数。

* pg\_cancel\_backend(pid int)
* 描述：取消一个后端的当前查询。
* 返回值类型：Boolean
* 备注：pg\_cancel\_backend向由pid标识的后端进程发送一个查询取消（SIGINT）信号。一个活动的后端进程的PID可以从pg\_stat\_activity视图的pid字段找到，或者在服务器上用ps列出数据库进程。具有SYSADMIN权限的用户，后端进程所连接的数据库的属主，后端进程的属主或者继承了内置角色gs\_role\_signal\_backend权限的用户有权使用该函数。
* pg\_reload\_conf()
* 描述：导致所有服务器进程重新装载它们的配置文件（需要系统管理员角色）。
* 返回值类型：Boolean
* 备注：pg\_reload\_conf给服务器发送一个SIGHUP信号，导致所有服务器进程重新装载配置文件。
* pg\_rotate\_logfile()
* 描述：滚动服务器的日志文件（需要系统管理员角色）。
* 返回值类型：Boolean
* 备注：pg\_rotate\_logfile给日志文件管理器发送信号，告诉它立即切换到一个新的输出文件。这个函数只有在redirect\_stderr用于日志输出的时候才有用，否则根本不存在日志文件管理器子进程。
* pg\_terminate\_backend(pid int)
* 描述：终止一个后台线程。
* 返回值类型：Boolean
* 备注：如果成功，函数返回true，否则返回false。具有SYSADMIN权限的用户，后端进程所连接的数据库的属主，后端进程的属主或者继承了内置角色gs\_role\_signal\_backend权限的用户有权使用该函数。
* 示例：

vastbase=# SELECT pid from pg\_stat\_activity;

pid

-----------------

140657876268816

(1 rows)

vastbase=# SELECT pg\_terminate\_backend(140657876268816);

pg\_terminate\_backend

----------------------

t

(1 row)

* pg\_terminate\_session(pid int64, sessionid int64)
* 描述：终止一个后台session。
* 返回值类型：Boolean
* 备注：如果成功，函数返回true，否则返回false。具有SYSADMIN权限的用户，会话所连接的数据库的属主，会话的属主或者继承了内置角色gs\_role\_signal\_backend权限的用户有权使用该函数。

##### 备份恢复控制函数

**备份控制函数**

备份控制函数可帮助进行在线备份。

* pg\_create\_restore\_point(name text)
* 描述：为执行恢复创建一个命名点。（需要管理员角色）
* 返回值类型：text
* 备注：pg\_create\_restore\_point创建了一个可以用作恢复目的、有命名的事务日志记录，并返回相应的事务日志位置。在恢复过程中，recovery\_target\_name可以通过这个名称定位对应的日志恢复点，并从此处开始执行恢复操作。避免使用相同的名称创建多个恢复点，因为恢复操作将在第一个匹配（恢复目标）的名称上停止。
* pg\_current\_xlog\_location()
* 描述：获取当前事务日志的写入位置。
* 返回值类型：text
* 备注：pg\_current\_xlog\_location使用与前面那些函数相同的格式显示当前事务日志的写入位置。如果是只读操作，不需要系统管理员权限。
* pg\_current\_xlog\_insert\_location()
* 描述：获取当前事务日志的插入位置。
* 返回值类型：text
* 备注：pg\_current\_xlog\_insert\_location显示当前事务日志的插入位置。插入点是事务日志在某个瞬间的“逻辑终点”，而实际的写入位置则是从服务器内部缓冲区写出时的终点。写入位置是可以从服务器外部检测到的终点，如果要归档部分完成事务日志文件，则该操作即可实现。插入点主要用于服务器调试目的。如果是只读操作，不需要系统管理员权限。
* gs\_current\_xlog\_insert\_end\_location()
* 描述：获取当前事务日志的插入位置。
* 返回值类型：text
* 备注：gs\_current\_xlog\_insert\_end\_location显示当前事务日志的实际插入位置。
* pg\_start\_backup(label text [, fast boolean ])
* 描述：开始执行在线备份。（需要管理员角色或复制的角色）
* 返回值类型：text
* 备注：pg\_start\_backup接受一个用户定义的备份标签（通常这是备份转储文件存放地点的名称）。这个函数向Vastbase的数据目录写入一个备份标签文件，然后以文本方式返回备份的事务日志起始位置。

vastbase=# SELECT pg\_start\_backup('label\_goes\_here');

pg\_start\_backup

-----------------

0/3000020

(1 row)

* pg\_stop\_backup()
* 描述：完成执行在线备份。（需要管理员角色或复制的角色）
* 返回值类型：text
* 备注：pg\_stop\_backup删除pg\_start\_backup创建的标签文件，并且在事务日志归档区里创建一个备份历史文件。这个历史文件包含给予pg\_start\_backup的标签、备份的事务日志起始与终止位置、备份的起始和终止时间。返回值是备份的事务日志终止位置。计算出中止位置后，当前事务日志的插入点将自动前进到下一个事务日志文件，这样，结束的事务日志文件可以被立即归档从而完成备份。
* pg\_switch\_xlog()
* 描述：切换到一个新的事务日志文件。（需要管理员角色）
* 返回值类型：text
* 备注：pg\_switch\_xlog移动到下一个事务日志文件，以允许将当前日志文件归档（假定使用连续归档）。返回值是刚完成的事务日志文件的事务日志结束位置+1。如果从最后一次事务日志切换以来没有活动的事务日志，则pg\_switch\_xlog什么事也不做，直接返回当前事务日志文件的开始位置。
* pg\_xlogfile\_name(location text)
* 描述：将事务日志的位置字符串转换为文件名。
* 返回值类型：text
* 备注：pg\_xlogfile\_name仅抽取事务日志文件名称。如果给定的事务日志位置恰好位于事务日志文件的交界上，这两个函数都返回前一个事务日志文件的名称。这对于管理事务日志归档来说是非常有利的，因为前一个文件是当前最后一个需要归档的文件。
* pg\_xlogfile\_name\_offset(location text)
* 描述：将事务日志的位置字符串转换为文件名并返回在文件中的字节偏移量。
* 返回值类型：text,integer
* 备注：可以使用pg\_xlogfile\_name\_offset从前述函数的返回结果中抽取相应的事务日志文件名称和字节偏移量。例如：

vastbase=# SELECT \* FROM pg\_xlogfile\_name\_offset(pg\_stop\_backup());

NOTICE: pg\_stop\_backup cleanup done, waiting for required WAL segments to be archived

NOTICE: pg\_stop\_backup complete, all required WAL segments have been archived

file\_name | file\_offset

--------------------------+-------------

000000010000000000000003 | 272

(1 row)

* pg\_xlog\_location\_diff(location text, location text)
* 描述：计算两个事务日志位置之间在字节上的区别。
* 返回值类型：numeric
* pg\_cbm\_tracked\_location()
* 描述：用于查询cbm解析到的lsn位置。
* 返回值类型：text
* pg\_cbm\_get\_merged\_file(startLSNArg text, endLSNArg text)
* 描述：用于将指定lsn范围之内的cbm文件合并成一个cbm文件，并返回合并完的cbm文件名。
* 返回值类型：text
* 备注：必须是系统管理员或运维管理员才能获取cbm合并文件。
* pg\_cbm\_get\_changed\_block(startLSNArg text, endLSNArg text)
* 描述：用于将指定lsn范围之内的cbm文件合并成一个表，并返回表的各行记录。
* 返回值类型：records
* 备注：pg\_cbm\_get\_changed\_block返回的表字段包含：合并起始的lsn、合并截止的lsn、表空间oid、库oid、表的relfilenode、表的fork number、表是否被删除、表是否被创建、表是否被截断、表被截断后的页面数、有多少页被修改以及被修改的页号的列表。
* pg\_cbm\_recycle\_file(targetLSNArg text)
* 描述：删除不再使用的cbm文件，并返回删除后的第一条lsn。
* 返回值类型：text
* pg\_cbm\_force\_track(targetLSNArg text,timeOut int)
* 描述：强制执行一次cbm追踪到指定的xlog位置，并返回实际追踪结束点的xlog位置。
* 返回值类型：text
* pg\_enable\_delay\_ddl\_recycle()
* 描述：开启延迟DDL功能，并返回开启点的xlog位置。需要管理员角色或运维管理员角色打开operate\_mode。
* 返回值类型：text
* pg\_disable\_delay\_ddl\_recycle(barrierLSNArg text, isForce bool)
* 描述：关闭延迟DDL功能，并返回本次延迟DDL生效的xlog范围。需要管理员角色或运维管理员角色打开operate\_mode。
* 返回值类型：records
* pg\_enable\_delay\_xlog\_recycle()
* 描述：开启延迟xlog回收功能，数据库主节点修复使用。
* 返回值类型：void
* pg\_disable\_delay\_xlog\_recycle()
* 描述：关闭延迟xlog回收功能，数据库主节点修复使用。
* 返回值类型：void
* pg\_cbm\_rotate\_file(rotate\_lsn text)
* 描述：等待cbm解析到rotate\_lsn之后，强制切换文件，在build期间调用。
* 返回值类型：void。
* gs\_roach\_stop\_backup(backupid text)
* 描述：停止一个内部备份工具GaussRoach开启的备份。与pg\_stop\_backup系统函数类似，但更轻量。
* 返回值类型：text，内容为当前日志的插入位置。
* 备注：目前Vastbase不支持。
* gs\_roach\_enable\_delay\_ddl\_recycle(backupid name)
* 描述：开启延迟DDL功能，并返回开启点的日志位置。与pg\_enable\_delay\_ddl\_recycle系统函数类似，但更轻量。并且，通过传入不同的backupid，可以支持并发打开延迟DDL。
* 返回值类型：text，内容为返回开启点的日志位置。
* 备注：目前Vastbase不支持。
* gs\_roach\_disable\_delay\_ddl\_recycle(backupid text)
* 描述：关闭延迟DDL功能，并返回本次延迟DDL生效的日志范围，并删除该范围内被用户删除的列存表物理文件。与pg\_enable\_delay\_ddl\_recycle系统函数类似，但更轻量。并且，通过传入不同的backupid，可以支持并发关闭延迟DDL功能。
* 返回值类型：records，内容为本次延迟DDL生效的日志范围。
* 备注：目前Vastbase不支持。
* gs\_roach\_switch\_xlog(request\_ckpt bool)
* 描述：切换当前使用的日志段文件，并且，如果request\_ckpt为true，则触发一个全量检查点。
* 返回值类型：text，内容为切段日志的位置。
* 备注：目前Vastbase不支持。

**恢复控制函数**

恢复信息函数提供了当前备机状态的信息。这些函数可能在恢复期间或正常运行中执行。

* pg\_is\_in\_recovery()
* 描述：如果恢复仍然在进行中则返回true。
* 返回值类型：bool
* pg\_last\_xlog\_receive\_location()
* 描述：获取最后接收事务日志的位置并通过流复制将其同步到磁盘。当流复制正在进行时，事务日志将持续递增。如果恢复已完成，则最后一次获取的WAL记录会被静态保持并在恢复过程中同步到磁盘。如果流复制不可用，或还没有开始，这个函数返回NULL。
* 返回值类型：text
* pg\_last\_xlog\_replay\_location()
* 描述：获取最后一个事务日志在恢复时重放的位置。如果恢复仍在进行，事务日志将持续递增。如果已经完成恢复，则将保持在恢复期间最后接收WAL记录的值。如果未进行恢复但服务器正常启动时，则这个函数返回NULL。
* 返回值类型：text
* pg\_last\_xact\_replay\_timestamp()
* 描述：获取最后一个事务在恢复时重放的时间戳。这是为在主节点上生成事务提交或终止WAL记录的时间。如果在恢复时没有事务重放，则这个函数返回NULL。如果恢复仍在进行，则事务日志将持续递增。如果恢复已经完成，则将保持在恢复期间最后接收WAL记录的值。如果服务器无需恢复就已正常启动，则这个函数返回NULL。
* 返回值类型：timestamp with time zone

恢复控制函数控制恢复的进程。这些函数可能只在恢复时被执行。

* pg\_is\_xlog\_replay\_paused()
* 描述：如果恢复暂停则返回true。
* 返回值类型：bool
* pg\_xlog\_replay\_pause()
* 描述：立即暂停恢复。
* 返回值类型：void
* pg\_xlog\_replay\_resume()
* 描述：如果恢复处于暂停状态，则重新启动。
* 返回值类型：void

当恢复暂停时，没有发生数据库更改。如果是在热备里，所有新的查询将看到一致的数据库快照，并且不会有进一步的查询冲突产生，直到恢复继续。

如果不能使用流复制，则暂停状态将无限的延续。当流复制正在进行时，将连续接收WAL记录，最终将填满可用磁盘空间，这个进度取决于暂停的持续时间，WAL生成的速度和可用的磁盘空间。

##### 快照同步函数

快照同步函数是导出当前快照的标识符。

* pg\_export\_snapshot()
* 描述：保存当前的快照并返回它的标识符。
* 返回值类型：text
* 备注：函数pg\_export\_snapshot保存当前的快照并返回一个文本字符串标识此快照。这个字符串必须传递给想要导入快照的客户端。可用在set transaction snapshot snapshot\_id时导入snapshot，但是应用的前提是该事务设置了SERIALIZABLE或REPEATABLE READ隔离级别。而Vastbase目前是不支持这两种隔离级别的。该函数的输出不可用做set transaction snapshot的输入。
* pg\_export\_snapshot\_and\_csn()
* 描述：保存当前的快照并返回它的标识符。比pg\_export\_snapshot()多返回一列CSN，表示当前快照的CSN。
* 返回值类型：text

##### 数据库对象函数

**数据库对象尺寸函数**

数据库对象尺寸函数计算数据库对象使用的实际磁盘空间。

* pg\_column\_size(any)
* 描述：存储一个指定的数值需要的字节数（可能压缩过）。
* 返回值类型：int
* 备注：pg\_column\_size显示用于存储某个独立数据值的空间。

vastbase=# SELECT pg\_column\_size(1);

pg\_column\_size

----------------

4

(1 row)

* pg\_database\_size(oid)
* 描述：指定OID代表的数据库使用的磁盘空间。
* 返回值类型：bigint
* pg\_database\_size(name)
* 描述：指定名称的数据库使用的磁盘空间。
* 返回值类型：bigint
* 备注：pg\_database\_size接受一个数据库的OID或者名称，然后返回该对象使用的全部磁盘空间。
* 示例：

vastbase=# SELECT pg\_database\_size('postgres');

pg\_database\_size

------------------

51590112

(1 row)

* pg\_relation\_size(oid)
* 描述：指定OID代表的表或者索引所使用的磁盘空间。
* 返回值类型：bigint
* get\_db\_source\_datasize()
* 描述：估算当前数据库非压缩态的数据总容量。
* 返回值类型：bigint
* 备注：（1）调用该函数前需要做analyze；（2）通过估算列存的压缩率计算非压缩态的数据总容量。
* 示例：

vastbase=# analyze;

ANALYZE

vastbase=# select get\_db\_source\_datasize();

get\_db\_source\_datasize

------------------------

35384925667

(1 row)

* pg\_relation\_size(text)
* 描述：指定名称的表或者索引使用的磁盘空间。表名称可以用模式名修饰。
* 返回值类型：bigint
* pg\_relation\_size(relation regclass, fork text)
* 描述：指定表或索引的指定分叉树（'main'，'fsm'或'vm'）使用的磁盘空间。
* 返回值类型：bigint
* pg\_relation\_size(relation regclass)
* 描述：pg\_relation\_size(..., 'main')的简写。
* 返回值类型：bigint
* 备注：pg\_relation\_size接受一个表、索引、压缩表的OID或者名称，然后返回它们的字节大小。
* pg\_partition\_size(oid,oid)
* 描述：指定OID代表的分区使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。
* 返回值类型：bigint
* pg\_partition\_size(text, text)
* 描述：指定名称的分区使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。
* 返回值类型：bigint
* pg\_partition\_indexes\_size(oid,oid)
* 描述：指定OID代表的分区的索引使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。
* 返回值类型：bigint
* pg\_partition\_indexes\_size(text,text)
* 描述：指定名称的分区的索引使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。
* 返回值类型：bigint
* pg\_indexes\_size(regclass)
* 描述：附加到指定表的索引使用的总磁盘空间。
* 返回值类型：bigint
* pg\_size\_pretty(bigint)
* 描述：将以64位整数表示的字节值转换为具有单位的易读格式。
* 返回值类型：text
* pg\_size\_pretty(numeric)
* 描述：将以数值表示的字节值转换为具有单位的易读格式。
* 返回值类型：text
* 备注：pg\_size\_pretty用于把其他函数的结果格式化成一种易读的格式，可以根据情况使用KB 、MB 、GB 、TB。
* pg\_table\_size(regclass)
* 描述：指定的表使用的磁盘空间，不计索引（但是包含TOAST，自由空间映射和可见性映射）。
* 返回值类型：bigint
* pg\_tablespace\_size(oid)
* 描述：指定OID代表的表空间使用的磁盘空间。
* 返回值类型：bigint
* pg\_tablespace\_size(name)
* 描述：指定名称的表空间使用的磁盘空间。
* 返回值类型：bigint
* 备注：
* pg\_tablespace\_size接受一个数据库的OID或者名称，然后返回该对象使用的全部磁盘空间。
* pg\_total\_relation\_size(oid)
* 描述：指定OID代表的表使用的磁盘空间，包括索引和压缩数据。
* 返回值类型：bigint
* pg\_total\_relation\_size(regclass)
* 描述：指定的表使用的总磁盘空间，包括所有的索引和TOAST数据。
* 返回值类型：bigint
* pg\_total\_relation\_size(text)
* 描述：指定名称的表所使用的全部磁盘空间，包括索引和压缩数据。表名称可以用模式名修饰。
* 返回值类型：bigint
* 备注：pg\_total\_relation\_size接受一个表或者一个压缩表的OID或者名称，然后返回以字节计的数据和所有相关的索引和压缩表的尺寸。
* datalength(any)
* 描述：计算一个指定的数据需要的字节数（不考虑数据的管理空间和数据压缩、数据类型转换等情况）。
* 返回值类型：int
* 备注：datalength用于计算某个独立数据值的空间。
* 示例：

vastbase=# SELECT datalength(1);

datalength

------------

4

(1 row)

目前支持的数据类型及计算方式见下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **数据类型** | | | **存储空间** |
| 数值类型 | 整数类型 | TINYINT | 1 |
| SMALLINT | 2 |
| INTEGER | 4 |
| BINARY\_INTEGER | 4 |
| BIGINT | 8 |
| 任意精度型 | DECIMAL | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| NUMERIC | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| NUMBER | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| 序列整型 | SMALLSERIAL | 2 |
| SERIAL | 4 |
| LARGESERIAL | 8 |
| BIGSERIAL | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| 浮点类型 | FLOAT4 | 4 |
| DOUBLE PRECISION | 8 |
| FLOAT8 | 8 |
| BINARY\_DOUBLE | 8 |
| FLOAT[(p)] | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| DEC[(p[,s])] | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| INTEGER[(p[,s])] | 每4位十进制数占两个字节，小数点前后数字分别计算 |
| 布尔类型 | 布尔类型 | BOOLEAN | 1 |
| 字符类型 | 字符类型 | CHAR | n |
| CHAR(n) | n |
| CHARACTER(n) | n |
| NCHAR(n) | n |
| VARCHAR(n) | n |
| CHARACTER | 字符实际字节数 |
| VARYING(n) | 字符实际字节数 |
| VARCHAR2(n) | 字符实际字节数 |
| NVARCHAR(n) | 字符实 际字节数 |
| NVARCHAR2(n) | 字符实际字节数 |
| TEXT | 字符实际字节数 |
| CLOB | 字符实际字节数 |
| 时间类型 | 时间类型 | DATE | 8 |
| TIME | 8 |
| TIMEZ | 12 |
| TIMESTAMP | 8 |
| TIMESTAMPZ | 8 |
| SMALLDATETIME | 8 |
| INTERVAL DAY TO SECOND | 16 |
| INTERVAL | 16 |
| RELTIME | 4 |
| ABSTIME | 4 |
| TINTERVAL | 12 |

**数据库对象位置函数**

* pg\_relation\_filenode(relation regclass)
* 描述：指定关系的文件节点数。
* 返回值类型：oid
* 备注：pg\_relation\_filenode接受一个表、索引、序列或压缩表的OID或者名称，并且返回当前分配给它的“filenode”数。文件节点是关系使用的文件名称的基本组件。对大多数表来说，结果和pg\_class.relfilenode相同，但对确定的系统目录来说，relfilenode为0而且这个函数必须用来获取正确的值。如果传递一个没有存储的关系，比如一个视图，那么这个函数返回NULL。
* pg\_relation\_filepath(relation regclass)
* 描述：指定关系的文件路径名。
* 返回值类型：text
* 备注：pg\_relation\_filepath类似于pg\_relation\_filenode，但是它返回关系的整个文件路径名（相对于vastbase的数据目录PGDATA）。
* pg\_filenode\_relation(tablespace oid, filenode oid)
* 描述：获取对应的tablespace和relfilenode所对应的表名。
* 返回类型：regclass
* pg\_partition\_filenode(partition\_oid)
* 描述：获取到指定分区表的oid锁对应的filenode。
* 返回类型：oid
* pg\_partition\_filepath(partition\_oid)
* 描述：指定分区的文件路径名。
* 返回值类型：text

**回收站对象函数**

* gs\_is\_recycle\_object(classid, objid, objname)
* 描述：判断是否为回收站对象。
* 返回值类型：bool

##### 咨询锁函数

咨询锁函数用于管理咨询锁（Advisory Lock）。

* pg\_advisory\_lock(key bigint)
* 描述：获取会话级别的排它咨询锁。
* 返回值类型：void
* 备注：pg\_advisory\_lock锁定应用程序定义的资源，该资源可以用一个64位或两个不重叠的32位键值标识。如果已经有另外的会话锁定了该资源，则该函数将阻塞到该资源可用为止。这个锁是排它的。多个锁定请求将会被压入栈中，因此，如果同一个资源被锁定了三次，它必须被解锁三次以将资源释放给其他会话使用。
* pg\_advisory\_lock(key1 int, key2 int)
* 描述：获取会话级别的排它咨询锁。
* 返回值类型：void
* 备注：只允许sysadmin对键值对(65535, 65535)加会话级别的排它咨询锁，普通用户无权限。
* pg\_advisory\_lock(int4, int4, Name)
* 描述：获取指定数据库的排它咨询锁。
* 返回值类型：void
* pg\_advisory\_lock\_shared(key bigint)
* 描述：获取会话级别的共享咨询锁。
* 返回值类型：void
* pg\_advisory\_lock\_shared(key1 int, key2 int)
* 描述：获取会话级别的共享咨询锁。
* 返回值类型：void
* 备注：pg\_advisory\_lock\_shared类似于pg\_advisory\_lock，不同之处仅在于共享锁会话可以和其他请求共享锁的会话共享资源，但排它锁除外。
* pg\_advisory\_unlock(key bigint)
* 描述：释放会话级别的排它咨询锁。
* 返回值类型：Boolean
* pg\_advisory\_unlock(key1 int, key2 int)
* 描述：释放会话级别的排它咨询锁。
* 返回值类型：Boolean
* 备注：pg\_advisory\_unlock释放先前取得的排它咨询锁。如果释放成功则返回true。如果实际上并未持有指定的锁，将返回false并在服务器中产生一条SQL警告信息。
* pg\_advisory\_unlock(int4, int4, Name)
* 描述：释放指定数据库上的排它咨询锁。
* 返回值类型：Boolean
* 备注：如果释放成功则返回true；如果未持有锁，则返回false。
* pg\_advisory\_unlock\_shared(key bigint)
* 描述：释放会话级别的共享咨询锁。
* 返回值类型：Boolean
* pg\_advisory\_unlock\_shared(key1 int, key2 int)
* 描述：释放会话级别的共享咨询锁。
* 返回值类型：Boolean
* 备注：pg\_advisory\_unlock\_shared类似于pg\_advisory\_unlock，不同之处在于该函数释放的是共享咨询锁。
* pg\_advisory\_unlock\_all()
* 描述：释放当前会话持有的所有咨询锁。
* 返回值类型：void
* 备注：pg\_advisory\_unlock\_all将会释放当前会话持有的所有咨询锁，该函数在会话结束的时候被隐含调用，即使客户端异常地断开连接也是一样。
* pg\_advisory\_xact\_lock(key bigint)
* 描述：获取事务级别的排它咨询锁。
* 返回值类型：void
* pg\_advisory\_xact\_lock(key1 int, key2 int)
* 描述：获取事务级别的排它咨询锁。
* 返回值类型：void
* 备注：pg\_advisory\_xact\_lock类似于pg\_advisory\_lock，不同之处在于锁是自动在当前事务结束时释放，而且不能被显式的释放。只允许sysadmin对键值对(65535, 65535)加事务级别的排它咨询锁，普通用户无权限。
* pg\_advisory\_xact\_lock\_shared(key bigint)
* 描述：获取事务级别的共享咨询锁。
* 返回值类型：void
* pg\_advisory\_xact\_lock\_shared(key1 int, key2 int)
* 描述：获取事务级别的共享咨询锁。
* 返回值类型：void
* 备注：pg\_advisory\_xact\_lock\_shared类似于pg\_advisory\_lock\_shared，不同之处在于锁是在当前事务结束时自动释放，而且不能被显式的释放。
* pg\_try\_advisory\_lock(key bigint)
* 描述：尝试获取会话级排它咨询锁。
* 返回值类型：Boolean
* 备注：pg\_try\_advisory\_lock类似于pg\_advisory\_lock，不同之处在于该函数不会阻塞以等待资源的释放。它要么立即获得锁并返回true，要么返回false表示目前不能锁定。
* pg\_try\_advisory\_lock(key1 int, key2 int)
* 描述：尝试获取会话级排它咨询锁。
* 返回值类型：Boolean
* 备注：只允许sysadmin对键值对(65535, 65535)加会话级别的排它咨询锁，普通用户无权限。
* pg\_try\_advisory\_lock\_shared(key bigint)
* 描述：尝试获取会话级共享咨询锁。
* 返回值类型：Boolean
* pg\_try\_advisory\_lock\_shared(key1 int, key2 int)
* 描述：尝试获取会话级共享咨询锁。
* 返回值类型：Boolean
* 备注：pg\_try\_advisory\_lock\_shared类似于pg\_try\_advisory\_lock，不同之处在于该函数尝试获得共享锁而不是排它锁。
* pg\_try\_advisory\_xact\_lock(key bigint)
* 描述：尝试获取事务级别的排它咨询锁。
* 返回值类型：Boolean
* pg\_try\_advisory\_xact\_lock(key1 int, key2 int)
* 描述：尝试获取事务级别的排它咨询锁。
* 返回值类型：Boolean
* 备注：pg\_try\_advisory\_xact\_lock类似于pg\_try\_advisory\_lock，不同之处在于如果得到锁，在当前事务的结束时自动释放，而且不能被显式的释放。只允许sysadmin对键值对(65535, 65535)加事务级别的排它咨询锁，普通用户无权限。
* pg\_try\_advisory\_xact\_lock\_shared(key bigint)
* 描述：尝试获取事务级别的共享咨询锁。
* 返回值类型：Boolean
* pg\_try\_advisory\_xact\_lock\_shared(key1 int, key2 int)
* 描述：尝试获取事务级别的共享咨询锁。
* 返回值类型：Boolean
* 备注：pg\_try\_advisory\_xact\_lock\_shared类似于pg\_try\_advisory\_lock\_shared，不同之处在于如果得到锁，在当前事务结束时自动释放，而且不能被显式的释放。
* lock\_cluster\_ddl()
* 描述：尝试对Vastbase内所有存活的数据库主节点获取会话级别的排他咨询锁。
* 返回值类型：Boolean
* 备注：只允许sysadmin调用，普通用户无权限。
* unlock\_cluster\_ddl()
* 描述：尝试对数据库主节点会话级别的排他咨询锁。
* 返回值类型：Boolean

##### 逻辑复制函数

* pg\_create\_logical\_replication\_slot('slot\_name', 'plugin\_name')
* 描述：创建逻辑复制槽。
* 参数说明：
* slot\_name
  + 流复制槽名称。
  + 取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字符。
* plugin\_name
  + 插件名称。
  + 取值范围：字符串，当前支持mppdb\_decoding。
* 返回值类型：name, text
* 备注：第一个返回值表示slot\_name，第二个返回值表示该逻辑复制槽解码的起始LSN位置。调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。此函数目前只支持在主机调用。
* pg\_create\_physical\_replication\_slot('slot\_name', 'isDummyStandby')
* 描述：创建新的物理复制槽。
* 参数说明：
* slot\_name

流复制槽名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* isDummyStandby

是否是从从备连接主机创建的复制槽。

类型：bool。

返回值类型：name, text

* 备注：调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。目前默认不支持主备从部署模式。
* pg\_drop\_replication\_slot('slot\_name')
* 描述：删除流复制槽。
* 参数说明：
* slot\_name

流复制槽名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

返回值类型：void

* 备注：调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。此函数目前只支持在主机调用。
* pg\_logical\_slot\_peek\_changes('slot\_name', 'LSN', upto\_nchanges, 'options\_name', 'options\_value')
* 描述：解码并不推进流复制槽（下次解码可以再次获取本次解出的数据）。
* 参数说明：
* slot\_name

流复制槽名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* LSN

日志的LSN，表示只解码小于等于此LSN的日志。

取值范围：字符串（LSN，格式为xlogid/xrecoff），如 '1/2AAFC60'。为NULL时表示不对解码截止的日志位置做限制。

* upto\_nchanges

解码条数（包含begin和commit）。假设一共有三条事务，分别 包含3、5、7条记录，如果upto\_nchanges为4，那么会解码出 前两个事务共8条记录。解码完第二条事务时发现解码条数记录 大于等于upto\_nchanges，会停止解码。

取值范围：非负整数。

* fig: **说明：**
* LSN和upto\_nchanges中任一参数达到限制，解码都会结束。
* options：此项为可选参数，由一系列options\_name和options\_value一一对应组成。
* include-xids

解码出的data列是否包含xid信息。

取值范围：0或1，默认值为1。

0：设为0时，解码出的data列不包含xid信息。

1：设为1时，解码出的data列包含xid信息。

* skip-empty-xacts

解码时是否忽略空事务信息。

取值范围：0或1，默认值为0。

0：设为0时，解码时不忽略空事务信息。

1：设为1时，解码时会忽略空事务信息。

* include-timestamp

解码信息是否包含commit时间戳。

取值范围：0或1，默认值为0。

0：设为0时，解码信息不包含commit时间戳。

1：设为1时，解码信息包含commit时间戳。

* only-local

是否仅解码本地日志。

取值范围：0或1，默认值为1。

0：设为0时，解码非本地日志和本地日志。

1：设为1时，仅解码本地日志。

* force-binary

是否以二进制格式输出解码结果。

取值范围：0，默认值为0。设为0时，以文本格式输出解码结果。

* white-table-list

白名单参数，包含需要进行解码的schema和表名。

取值范围：包含白名单中表名的字符串，不同的表以','为分隔符 进行隔离；使用'\*'来模糊匹配所有情况；schema名和表名间以'.' 分割，不允许存在任意空白符。例：select \* from pg\_logical\_slot\_peek\_changes('slot1', NULL, 4096, 'white-table-list', 'public.t1,public.t2');

返回值类型：text, xid, text

备注：函数返回解码结果，每一条解码结果包含三列，对应上述返回值类型，分别表示LSN位置、xid和解码内容。

调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。

* pg\_logical\_slot\_get\_changes('slot\_name', 'LSN', upto\_nchanges, 'options\_name', 'options\_value')
* 描述：解码并推进流复制槽。
* 参数说明：与pg\_logical\_slot\_peek\_changes一致，详细内容请参见pg\_logical\_slot\_peek\_ch...。
* 备注：调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。此函数目前只支持在主机调用。
* pg\_logical\_slot\_peek\_binary\_changes('slot\_name', 'LSN', upto\_nchanges, 'options\_name', 'options\_value')
* 描述：以二进制格解码且不推进流复制槽（下次解码可以再次获取本次解出的数据）。
* 参数说明：
* slot\_name

流复制槽名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* LSN

日志的LSN，表示只解码小于等于此LSN的日志。

取值范围：字符串（LSN，格式为xlogid/xrecoff），如 '1/2AAFC60'。为NULL时表示不对解码截止的日志位置做限制。

* upto\_nchanges

解码条数（包含begin和commit）。假设一共有三条事务，分别 包含3、5、7条记录，如果upto\_nchanges为4，那么会解码出 前两个事务共8条记录。解码完第二条事务时发现解码条数记录 大于等于upto\_nchanges，会停止解码。

* + 取值范围：非负整数。

fig: **说明：**

* + LSN和upto\_nchanges中任一参数达到限制，解码都会结束。
    - options：此项为可选参数，由一系列options\_name和options\_value一一对应组成。
    - include-xids
    - 解码出的data列是否包含xid信息。
    - 取值范围：0或1，默认值为1。

0：设为0时，解码出的data列不包含xid信息。

1：设为1时，解码出的data列包含xid信息。

* + - skip-empty-xacts
    - 解码时是否忽略空事务信息。
    - 取值范围：0或1，默认值为0。

0：设为0时，解码时不忽略空事务信息。

1：设为1时，解码时会忽略空事务信息。

* + - include-timestamp
    - 解码信息是否包含commit时间戳。
    - 取值范围：0或1，默认值为0。

0：设为0时，解码信息不包含commit时间戳。

1：设为1时，解码信息包含commit时间戳。

* + - only-local
    - 是否仅解码本地日志。
    - 取值范围：0或1，默认值为1。

0：设为0时，解码非本地日志和本地日志。

1：设为1时，仅解码本地日志。

* + - force-binary
    - 是否以二进制格式输出解码结果。
    - 取值范围：0或1，默认值为0，均以二进制格式输出结果。
    - white-table-list
    - 白名单参数，包含需要进行解码的schema和表名。
    - 取值范围：包含白名单中表名的字符串，不同的表以','为分隔符进行隔离；使用'\*'来模糊匹配所有情况；schema名和表名间以'.'分割，不允许存在任意空白符。例：select \* from pg\_logical\_slot\_peek\_binary\_changes('slot1', NULL, 4096, 'white-table-list', 'public.t1,public.t2');

返回值类型：text, xid, bytea

备注：函数返回解码结果，每一条解码结果包含三列，对应上述返回值类型，分别表示LSN位置、xid和二进制格式的解码内容。调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。

* pg\_logical\_slot\_get\_binary\_changes('slot\_name', 'LSN', upto\_nchanges, 'options\_name', 'options\_value')
* 描述：以二进制格式解码并推进流复制槽。
* 参数说明：与pg\_logical\_slot\_peek\_binary\_changes一致，详细内容请参见pg\_logical\_slot\_peek\_bi...。
* 备注：调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。
* pg\_replication\_slot\_advance ('slot\_name', 'LSN')
* 描述：直接推进流复制槽到指定LSN，不输出解码结果。
* 参数说明：
* slot\_name

流复制槽名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* LSN

推进到的日志LSN位置，下次解码时只会输出提交位置比该LSN 大的事务结果。如果输入的LSN比当前流复制槽记录的推进位置 还要小，则直接返回；如果输入的LSN比当前最新物理日志LSN 还要大，则推进到当前最新物理日志LSN。

取值范围：字符串（LSN，格式为xlogid/xrecoff）。

返回值类型：name, text

* 备注：返回值分别对应slot\_name和实际推进至的LSN。调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。此函数目前只支持在主机调用。
* pg\_logical\_get\_area\_changes('LSN\_start', 'LSN\_end', upto\_nchanges, 'decoding\_plugin', 'xlog\_path', 'options\_name', 'options\_value')
* 描述：没有ddl的前提下，指定lsn区间进行解码，或者指定xlog文件进行解码。
* 约束条件如下：
  1. 调用接口时，日志级别wal\_level=logical，且只有在wal\_level=logical期间产生的日志文件才能被解析，如果使用的xlog文件为非logical级别，则解码内容没有对应的值和类型，无其他影响。
  2. xlog文件只能被完全同构的dn的某个副本解析，确保可以找到数据对应的元信息，且没有DDL操作和VACUUM FULL。
  3. 用户可以找到需要解析的xlog。
  4. 用户需要注意一次不要读入过多xlog文件，推荐一次一个，一个xlog文件估测占用内存为xlog文件大小的2~3倍。
  5. 无法解码扩容前的xlog文件。
* 参数说明：
* LSN\_start

指定开始解码的lsn。

取值范围：字符串（LSN，格式为xlogid/xrecoff），如 '1/2AAFC60'。为NULL时表示不对解码截止的日志位置做限制。

* LSN\_end

指定解码结束的lsn。

取值范围：字符串（LSN，格式为xlogid/xrecoff），如 '1/2AAFC60'。为NULL时表示不对解码截止的日志位置做限制。

* upto\_nchanges

解码条数（包含begin和commit）。假设一共有三条事务，分别 包含3、5、7条记录，如果upto\_nchanges为4，那么会解码出 前两个事务共8条记录。解码完第二条事务时发现解码条数记录 大于等于upto\_nchanges，会停止解码。

取值范围：非负整数。

fig: **说明：**

SN和upto\_nchanges中任一参数达到限制，解码都会结束。

* decoding\_plugin

解码插件，指定解码内容输出格式的so插件。

取值范围：提供mppdb\_decoding和sql\_decoding两个解码插 件。

* xlog\_path

解码插件，指定解码文件的xlog绝对路径，文件级别

取值范围：NULL 或者 xlog文件绝对路径的字符串。

* options：此项为可选参数，由一系列options\_name和options\_value一一对应组成，可以缺省，详见pg\_logical\_slot\_peek\_changes。
* 示例：

vastbase=# SELECT pg\_current\_xlog\_location();

pg\_current\_xlog\_location

--------------------------

0/E62E238

(1 row)

vastbase=# create table t1 (a int primary key,b int,c int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "t1\_pkey" for table "t1"

CREATE TABLE

vastbase=# insert into t1 values(1,1,1);

INSERT 0 1

vastbase=# insert into t1 values(2,2,2);

INSERT 0 1

vastbase=# select data from pg\_logical\_get\_area\_changes('0/E62E238',NULL,NULL,'sql\_decoding',NULL);

location | xid | data

-----------+-------+---------------------------------------------------

0/E62E8D0 | 27213 | COMMIT (at 2022-01-26 15:08:03.349057+08) 3020226

0/E6325F0 | 27214 | COMMIT (at 2022-01-26 15:08:07.309869+08) 3020234

……

* pg\_get\_replication\_slots()
* 描述：获取复制槽列表。
* 返回值类型：text，text，text，oid，boolean，xid，xid，text，boolean
* 示例：

vastbase=# select \* from pg\_get\_replication\_slots();

slot\_name | plugin | slot\_type | datoid | active | xmin | catalog\_xmin | restart\_lsn | dummy\_standby

-----------+----------------+-----------+--------+--------+------+--------------+-------------+---------------

wkl001 | mppdb\_decoding | logical | 15914 | f | | 2079556 | 4/1B81D920 | f

dn\_6002 | | physical | 0 | t | | | 8/7CB63BD8 | f

dn\_6004 | | physical | 0 | t | | | 8/7CB63BD8 | f

dn\_6003 | | physical | 0 | t | | | 8/7CB63BD8 | f

gfslot001 | mppdb\_decoding | logical | 15914 | f | | 2412553 | 4/A54B2428 | f

(5 rows)

* gs\_get\_parallel\_decode\_status()
* 描述：监控各个解码线程的读取日志队列和解码结果队列的长度，以便定位并行解码性能瓶颈。
* 返回值类型：text, int, text, text
* 示例：

vastbase=# select \* from gs\_get\_parallel\_decode\_status();

slot\_name | parallel\_decode\_num | read\_change\_queue\_length | decode\_change\_queue\_length

-----------+---------------------+------------------------------------------------------------------+------------------------------------------------------------------

slot1 | 3 | queue0: 33, queue1: 36, queue2: 1017 | queue0: 1011, queue1: 1008, queue2: 27

slot2 | 5 | queue0: 452, queue1: 1017, queue2: 233, queue3: 585, queue4: 183 | queue0: 754, queue1: 188, queue2: 972, queue3: 620, queue4: 1022

(2 rows)

备注：返回值的slot\_name代表复制槽名，parallel\_decode\_num代表该复制槽的并行解码线程数，read\_change\_queue\_length列出了每个解码线程读取日志队列的当前长度，decode\_change\_queue\_length列出了每个解码线程解码结果队列的当前长度。

* pg\_replication\_origin\_create (node\_name)
* 描述：用给定的外部名称创建一个复制源，并且返回分配给它的内部ID。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

待创建的复制源的名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

返回值类型：oid

* pg\_replication\_origin\_drop (node\_name)
* 描述：删除一个以前创建的复制源，包括任何相关的重放进度。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

待删除的复制源的名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* pg\_replication\_origin\_oid (node\_name)
* 描述：根据名称查找复制源并返回内部ID。如果没有发现这样的复制源，则抛出错误。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

要查找的复制源的名称

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

返回值类型：oid

* pg\_replication\_origin\_session\_setup (node\_name)
* 描述：将当前会话标记为从给定的原点回放，从而允许跟踪回放进度。只能在当前没有选择原点时使用。使用pg\_replication\_origin\_session\_reset 命令来撤销。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

复制源名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* pg\_replication\_origin\_session\_reset ()
* 描述：取消pg\_replication\_origin\_session\_setup()的效果。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* pg\_replication\_origin\_session\_is\_setup ()
* 描述：如果在当前会话中选择了复制源则返回真。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 返回值类型：boolean
* pg\_replication\_origin\_session\_progress (flush)
* 描述：返回当前会话中选择的复制源的重放位置。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* flush

决定对应的本地事务是否被确保已经刷入磁盘。

取值范围：boolean

返回值类型：LSN

* pg\_replication\_origin\_xact\_setup (origin\_lsn, origin\_timestamp)
* 描述：将当前事务标记为重放在给定LSN和时间戳上提交的事务。只能在使用pg\_replication\_origin\_session\_setup选择复制源时调用。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* origin\_lsn

复制源回放位置。

取值范围：LSN

* origin\_timestamp

事务提交时间。

取值范围：timestamp with time zone

* pg\_replication\_origin\_xact\_reset ()
* 描述：取消pg\_replication\_origin\_xact\_setup()的效果。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* pg\_replication\_origin\_advance (node\_name, lsn)
* 描述：
* 将给定节点的复制进度设置为给定的位置。这主要用于设置初始位置，或在配置更改或类似的变更后设置新位置。
* 注意：这个函数的使用不当可能会导致不一致的复制数据。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

已有复制源名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* lsn

复制源回放位置。

取值范围：LSN

* pg\_replication\_origin\_progress (node\_name, flush)
* 描述：返回给定复制源的重放位置。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 参数说明：
* node\_name

复制源名称。

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

* flush

决定对应的本地事务是否被确保已经刷入磁盘。

取值范围：boolean

* pg\_show\_replication\_origin\_status()
* 描述：获取复制源的复制状态。
* 备注：调用该函数的用户需要具有SYSADMIN权限。
* 返回值类型：
* local\_id：oid，复制源id。
* external\_id：text，复制源名称。
* remote\_lsn：LSN，复制源的lsn位置。
* local\_lsn：LSN，本地的lsn位置。
* pg\_get\_publication\_tables(pub\_name)
* 描述：根据发布的名称，返回对应发布要发布的表的relid列表
* 参数说明：
* pub\_name

已存在的发布名称

取值范围：字符串，不支持除字母，数字，以及（\_?-.）以外的字 符。

返回值类型：relid列表

* pg\_stat\_get\_subscription(sub\_oid oid) → record
* 描述：
* 输入订阅的oid，返回订阅的状态信息。
* 参数说明：
* subid
* 订阅的oid。
* 取值范围：oid
* 返回值类型：
* relid：oid，表的oid。
* pid：thread\_id，后台apply/sync线程的thread id。
* received\_lsn：pg\_lsn，从发布端接收到的最近的lsn。
* last\_msg\_send\_time：timestamp，最近发布端发送消息的时间。
* last\_msg\_receipt\_time：timestamp，最新订阅端收到消息的时间。
* latest\_end\_lsn：pg\_lsn，最近一次收到保活消息时发布端的lsn。
* latest\_end\_time：timstamp，最近一次收到保活消息的时间。

##### 段页式存储函数

* local\_segment\_space\_info(tablespacename TEXT, databasename TEXT)
* 描述：输出为该表空间下所有ExtentGroup的使用信息。
* 返回值类型：

|  |  |
| --- | --- |
| **node\_name** | **节点名称。** |
| extent\_size | 该ExtentGroup的extent规格，单位是block数。 |
| forknum | Fork号。 |
| total\_blocks | 物理文件总extent数目。 |
| meta\_data\_blocks | 表空间管理的metadata占用的block数，只包括space header、map page等，不包括segment head。 |
| used\_data\_blocks | 存数据占用的extent数目。包括segment head。 |
| utilization | 使用的block数占总block数的百分比。即(used\_data\_blocks+meta\_data\_block)/total\_blocks。 |
| high\_water\_mark | 高水位线，被分配出去的extent，最大的物理页号。超过高水位线的block都没有被使用，可以被直接回收。 |

例如：

select \* from local\_segment\_space\_info('pg\_default', 'postgres');

node\_name | extent\_size | forknum | total\_blocks | meta\_data\_blocks | used\_data\_blocks | utilization | high\_water\_mark

-------------------+-------------+---------+--------------+------------------+------------------+-------------+-----------------

dn\_6001\_6002\_6003 | 1 | 0 | 16384 | 4157 | 1 | .253784 | 4158

dn\_6001\_6002\_6003 | 8 | 0 | 16384 | 4157 | 8 | .254211 | 4165

(2 rows)

* pg\_stat\_segment\_extent\_usage(int4 tablespace oid, int4 database oid, int4 extent\_type, int4 forknum)
* 描述：每次返回一个ExtentGroup中，每个被分配出去的extent的使用情况。extent\_type表示ExtentGroup的类型，合理取值为[1,5]的int值。在此范围外的会报error。forknum 表示fork号，合法取值为[0,4]的int值，目前只有三种值有效，数据文件为0，FSM文件为1，visibility map文件为2。
* 返回值类型：

| **名称** | **描述** |
| --- | --- |
| start\_block | Extent的起始物理页号。 |
| extent\_size | Extent的大小。 |
| usage\_type | Extent的使用类型，比如segment head、data extent等。 |
| ower\_location | 有指针指向该extent的对象的位置。比如data extent的owner就是它所属的segment的head位置。 |
| special\_data | 该extent在它owner中的位置。该字段的数据跟使用类型有关。比如data extent的special data就是它在所属segment中的extent id。 |

其中，usage\_type为枚举类型，每一项的含义为：

* Non-bucket table segment head ：非hashbucket表的数据段头。
* Non-bucket table fork head：非段页式表的fork段头。
* Data extent：数据块。

例如：

select \* from pg\_stat\_segment\_extent\_usage((select oid::int4 from pg\_tablespace where spcname='pg\_default'), (select oid::int4 from pg\_database where datname='postgres'), 1, 0);

start\_block | extent\_size | usage\_type | ower\_location | special\_data

-------------+-------------+------------------------+---------------+--------------

4157 | 1 | Data extent | 4294967295 | 0

4158 | 1 | Data extent | 4157 | 0

* local\_space\_shrink(tablespacename TEXT, databasename TEXT)
* 描述：当前节点上对指定段页式空间做物理空间收缩。注意，目前只支持对当前连接的database做shrink。
* 返回值：空
* gs\_space\_shrink(int4 tablespace, int4 database, int4 extent\_type, int4 forknum)
* 描述：效果跟local\_space\_shrink类似，对指定段页式空间做物理空间收缩,但参数不同，传入的是tablespace和database的oid，extent\_type为[2,5]的int值。注意：extent\_type = 1表示段页式元数据，目前不支持对元数据所在的物理文件做收缩。该函数仅限工具使用，不建议用户直接使用。
* 返回值：空
* pg\_stat\_remain\_segment\_info()
* 描述：展示在当前节点上，因为故障等原因而残留的extent。残留extent主要分为两类：分配而未被利用的segment和分配出去而未被利用的extent。两者主要区别在于segment会包含多个extent，回收时，要将segment上的extent一并全部回收。
* 返回值类型：

|  |  |
| --- | --- |
| **名称** | **描述** |
| space\_id | 表空间ID |
| db\_id | 数据库ID |
| block\_id | Extent的ID |
| type | Extent的类型，当前有三种：ALLOC\_SEGMENT、DROP\_SEGMENT、SHRINK\_EXTENT |

其中type的三种类型分别表示：

* ALLOC\_SEGMENT:用户创建一张段页式表，当segment刚被分配，但是建表语句所在事务仍未提交时，节点故障，导致该segment被分配后，没有被使用。
* DROP\_SEGMENT:用户删除段页式表，当该事务成功提交，但是此表的segment页面对应的bit位未被重置，就发生掉电等故障，造成该segment未被使用，也未被释放。
* SHRINK\_EXTENT:用户对段页式表执行shrink操作，在未对空置出的extent进行释放时，发生掉电等故障，造成该extent残留，无法被重新利用。
* 例如：

select \* from pg\_stat\_remain\_segment\_info();

space\_id | db\_id | block\_id | type

----------+-------+----------+------

1663 | 16385| 4156| ALLOC\_SEGMENT

* pg\_free\_remain\_segment(int4 spaceId, int4 dbId, int4 segmentId)
* 描述：释放指定的残留extent。参数取值必须为从函数pg\_stat\_remain\_segment\_info中查询获取。函数会对传入值校验，如果指定extent不在记录的残留extent中，将返回错误信息。指定的extent如果为单个extent，则只将其独自释放；如果为一个segment，则会将此segment以及此segment上记录的所有extent释放。
* 返回值：空

##### 其他函数

* plan\_seed()

描述：获取前一次查询语句的seed值（内部使用）。

返回值类型：int

* pg\_stat\_get\_env()

描述：获取当前节点的环境变量信息，仅sysadmin和monitor admin可 以访问。

返回值类型：record

示例：

vastbase=# select pg\_stat\_get\_env();

pg\_stat\_get\_env

------------------------------------------------------------------------------------------------

(node1,localhost,11677,5432,/home/vastbase/local/vastbase,/home/vastbase/data/vastbase,pg\_log)

(1 row)

* pg\_catalog.plancache\_clean()

描述：清理节点上无人使用的全局计划缓存。

返回值类型：bool

* pg\_catalog.plancache\_status()

描述：显示节点上全局计划缓存的信息，函数返回信息和 [GLOBAL\_PLANCACHE\_STATUS](https://docs.vastdata.com.cn/zh/docs/VastbaseG100Ver2.2.10/doc/%E5%BC%80%E5%8F%91%E8%80%85%E6%8C%87%E5%8D%97/GLOBAL_PLANCACHE_STATUS.html)一致。

返回值类型：record

* textlen(text)

描述：提供查询text的逻辑长度的方法。

返回值类型：int

* threadpool\_status()

描述：显示线程池中工作线程及会话的状态信息。

返回值类型：record

* get\_local\_active\_session()

描述：提供当前节点保存在内存中的历史活跃session状态的采样记 录。

返回值类型：record

* pg\_stat\_get\_thread()

描述：提供当前节点下所有线程的状态信息，sysadmin和monitor admin用户可以查看所有线程信息，普通用户查看本用户的线程信息。

返回值类型：record

* pg\_stat\_get\_sql\_count()

描述：提供当前节点中用户执行的 SELECT/UPDATE/INSERT/DELETE/MERGE INTO语句的计数结果， sysadmin和monitor admin用户可以查看所有用户的信息，普通用户查 看本用户的统计信息。

返回值类型：record

* pg\_stat\_get\_data\_senders()

描述：提供当前活跃的数据复制发送线程的详细信息。

返回值类型：record

* get\_wait\_event\_info()

描述：提供wait event事件的具体信息。

返回值类型：record

* generate\_wdr\_report(begin\_snap\_id bigint, end\_snap\_id bigint, report\_type cstring, report\_scope cstring, node\_name cstring)

描述：基于两个snapshot生成系统诊断报告。需要在postgres库下执 行，默认初始化用户或monadmin用户可以访问， V500R001C20SPC002及其之前的版本初始化用户或sysadmin用户可以 访问。只可在系统库中查询到结果，用户库中无法查询。

返回值类型：record

**表 1** generate\_wdr\_report 参数说明

| **参数** | **说明** | **取值范围** |
| --- | --- | --- |
| begin\_snap\_id | 生成某段时间内性能诊断报告的开始snapshotid。 | - |
| end\_snap\_id | 结束snapshot的id，默认end\_snap\_id大于begin\_snap\_id。 | - |
| report\_type | 指定生成report的类型。 | summarydetailall，即同时包含summary 和 detail。 |
| report\_scope | 指定生成report的范围。 | cluster：数据库级别的信息node：节点级别的信息 |
| node\_name | 在report\_scope指定为node时，需要把该参数指定为对应节点的名称。（节点名称可以执行select \* from pg\_node\_env;查询）。在report\_scope为cluster时，该值可以省略或者指定为空或NULL。 | cluster：省略/空/NULLnode： vastbase中的节点名称 |

* create\_wdr\_snapshot()

描述：手工生成系统诊断快照，该函数需要sysadmin权限。

返回值类型：text

* kill\_snapshot()

描述：kill后台的WDR snapshot线程，调用该函数的用户需要具有SYSADMIN权限或具有REPLICATION权限或继承了内置角色gs\_role\_replication的权限。

返回值类型：void

* capture\_view\_to\_json(text,integer)

描述：将视图的结果存入GUC: perf\_directory所指定的目录，如果is\_crossdb为1，则表示对于所有的database都会访问一次view；如果is\_crossdb为0，则表示仅对当前database进行一次视图访问。该函数只有sysadmin和monitor admin用户可以执行。

返回值类型：int

* reset\_unique\_sql

描述：用来清理数据库节点内存中的Unique SQL（需要sysadmin权限）。

返回值类型：bool

表 2 reset\_unique\_sql参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 描述 |
| scope | text | 清理范围类型：   * 'GLOBAL'：清理所有的节点，如果是'GLOBAL'，则只可以为主节点执行此函数。 * 'LOCAL'：清理本节点。 |
| clean\_type | text | * 'BY\_USERID'：按用户ID来进行清理Unique SQL。 * 'BY\_CNID'：按主节点的ID来进行清理Unique SQL。 * 'ALL'：全部清理。 |
| clean\_value | int8 | 具体清理type对应的清理值。 |

* image1scope的取值GLOBAL和LOCAL针对分布式，对于vastbase而言两者意义相同，均表示清理本节点。
* clean\_type的值BY\_CNID仅针对分布式，对于vastbase无效。
* wdr\_xdb\_query(db\_name\_str text, query text)

描述：提供本地跨数据库执行query的能力。例如: 在连接到postgres 库时, 访问test库下的表。

select col1 from wdr\_xdb\_query('dbname=test','select col1 from t1') as dd(col1 int);

返回值类型：record

* pg\_wlm\_jump\_queue(pid int)

描述：调整任务到数据库主节点队列的最前端。

返回值类型：boolean

true：成功。

false：失败。

* gs\_wlm\_switch\_cgroup(pid int, cgroup text)

描述：调整作业的优先级到新控制组。

返回值类型：boolean

true：成功。

false：失败。

* pv\_session\_memctx\_detail(threadid tid, MemoryContextName text)

描述：将线程tid的MemoryContextName内存上下文信息记录到“$GAUSSLOG/pg\_log/${node\_name}/dumpmem”目录下的“threadid\_timestamp.log”文件中。其中threadid可通过视图GS\_SESSION\_MEMORY\_DETAIL中的sessid后获得。在正式发布的版本中仅接受MemoryContextName为空串（两个单引号表示输入为空串，即”）的输入，此时会记录所有的内存上下文信息，否则不会有任何操作。对供内部开发人员和测试人员调试用的DEBUG版本，可以指定需要统计的MemoryContextName，此时会将该Context所有的内存使用情况记录到指定文件。该函数需要管理员权限的用户才能执行。

返回值类型：boolean

true：成功。

false：失败。

* pg\_shared\_memctx\_detail(MemoryContextName text)

描述：将MemoryContextName内存上下文信息记录到“$GAUSSLOG/pg\_log/${node\_name}/dumpmem”目录下的“threadid\_timestamp.log”文件中。该函数功能仅在DEBUG版本中供内部开发人员和测试人员调试使用，在正式发布版本中调用该函数不会有任何操作。该函数需要管理员权限的用户才能执行。

返回值类型：boolean

true：成功。

false：失败。

* local\_bgwriter\_stat()

描述：显示本实例的bgwriter线程刷页信息，候选buffer链中页面个数，buffer淘汰信息。

返回值类型：record

* local\_candidate\_stat()

描述：显示本实例的候选buffer链中页面个数，buffer淘汰信息，包含normal buffer pool和segment buffer pool。

返回值类型：record

* local\_ckpt\_stat()

描述：显示本实例的检查点信息和各类日志刷页情况。

返回值类型：record

* local\_double\_write\_stat()

描述：显示本实例的双写文件的情况。

返回值类型：record

表 3 local\_double\_write\_stat参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 实例名称。 |
| curr\_dwn | int8 | 当前双写文件的序列号。 |
| curr\_start\_page | int8 | 当前双写文件恢复起始页面。 |
| file\_trunc\_num | int8 | 当前双写文件复用的次数。 |
| file\_reset\_num | int8 | 当前双写文件写满后发生重置的次数。 |
| total\_writes | int8 | 当前双写文件总的I/O次数。 |
| low\_threshold\_writes | int8 | 低效率写双写文件的I/O次数（一次I/O刷页数量少于16页面）。 |
| high\_threshold\_writes | int8 | 高效率写双写文件的I/O次数（一次I/O刷页数量多于一批，421个页面）。 |
| total\_pages | int8 | 当前刷页到双写文件区的总的页面个数。 |
| low\_threshold\_pages | int8 | 低效率刷页的页面个数。 |
| high\_threshold\_pages | int8 | 高效率刷页的页面个数。 |
| file\_id | int8 | 当前双写文件的id号。 |

* local\_single\_flush\_dw\_stat()

描述：显示本实例的单页面淘汰双写文件的情况。

返回值类型：record

* local\_pagewriter\_stat()

描述：显示本实例的刷页信息和检查点信息。

返回值类型：record

* local\_redo\_stat()

描述：显示本实例的备机的当前回放状态。

返回值类型：record

备注：返回的回放状态主要包括当前回放位置、回放最小恢复点位置等信息。

* local\_recovery\_status()

描述：显示本实例的主机和备机的日志流控信息。

返回值类型：record

* gs\_wlm\_node\_recover(boolean isForce)

描述：获取当前内存中记录的TopSQL查询语句级别相关统计信息，当传入的参数不为0时，会将这部分信息从内存中清理掉。

返回值类型：record

* gs\_wlm\_node\_clean(cstring nodename)

描述：动态负载管理节点故障后做数据清理操作。该函数只有管理员用户可以执行，属于数据库实例管理模块调用的，不建议用户直接调用。该视图在集中式和单机环境上不支持。

返回值类型：bool

* gs\_cgroup\_map\_ng\_conf(group name)

描述：读取指定逻辑数据库的cgroup配置文件。

返回值类型：record

* gs\_wlm\_switch\_cgroup(sess\_id int8, cgroup name)

描述：切换指定会话的控制组。

返回值类型：record

* hdfs\_fdw\_handler()

描述：用于外表重写功能，定义外表时需要定义的函数。

返回值类型：record

* hdfs\_fdw\_validator(text[], oid)

描述：用于外表重写功能，定义外表时需要定义的函数。

返回值类型：record

* comm\_client\_info()

描述：用于查询单个节点活跃的客户端连接信息。

返回值类型：setof record

* pg\_sync\_cstore\_delta(text)

描述：同步指定列存表的delta表表结构，使其与列存表主表一致。

返回值类型：bigint

* pg\_sync\_cstore\_delta()

描述：同步所有列存表的delta表表结构，使其与列存表主表一致。

返回值类型：bigint

* pg\_get\_flush\_lsn()

描述：返回当前节点flush的xlog位置。

返回值类型：text

* pg\_get\_sync\_flush\_lsn()

描述：返回当前节点多数派flush的xlog位置。

返回值类型：text

* dbe\_perf.get\_global\_full\_sql\_by\_timestamp(start\_timestamp timestamp with time zone, end\_timestamp timestamp with time zone)

描述：获取数据库级的全量SQL(Full SQL)信息。只可在系统库中查询到结果，用户库中无法查询。

返回值类型：record

表 4 dbe\_perf.get\_global\_full\_sql\_by\_timestamp参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| start\_timestamp | timestamp with time zone | SQL启动时间范围的开始时间点。 |
| end\_timestamp | timestamp with time zone | SQL启动时间范围的结束时间点。 |

* dbe\_perf.get\_global\_slow\_sql\_by\_timestamp(start\_timestamp timestamp with time zone, end\_timestamp timestamp with time zone)

描述：获取数据库级的慢SQL(Slow SQL)信息。只可在系统库中查询到结果，用户库中无法查询。

返回值类型：record

表 5 dbe\_perf.get\_global\_slow\_sql\_by\_timestamp参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| start\_timestamp | timestamp with time zone | SQL启动时间范围的开始时间点。 |
| end\_timestamp | timestamp with time zone | SQL启动时间范围的结束时间点。 |

* statement\_detail\_decode(detail text, format text, pretty boolean)

描述：解析全量/慢SQL语句中的details字段的信息。只可在系统库中查询到结果，用户库中无法查询。

返回值类型：text

表 6 statement\_detail\_decode参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| detail | text | SQL语句产生的事件的集合（不可读）。 |
| format | text | 解析输出格式，取值为 plaintext。 |
| pretty | boolean | 当format为plaintext时，是否以优雅的格式展示：true表示通过“\n”分隔事。false表示通过“，”分隔事件。 |

* get\_prepared\_pending\_xid

描述：当恢复完成时，返回nextxid。

参数：nan

返回值类型：text

* pg\_clean\_region\_info

描述：清理regionmap。

参数：nan

返回值类型：character varying

* pg\_get\_delta\_info

描述：从单个dn获取delta info。

参数：rel text、 schema\_name text

返回值类型：part\_name text、 live\_tuple bigint、 data\_size bigint、 blocknum bigint

* pg\_get\_replication\_slot\_name

描述：获取slot name。

参数：nan

返回值类型：text

* pg\_get\_running\_xacts

描述：获取运行中的xact。

参数：nan

返回值类型：handle integer、 gxid xid、 state tinyint、 node text、 xmin xid、 vacuum boolean、 timeline bigint、 prepare\_xid xid、 pid bigint、 next\_xid xid

* pg\_get\_variable\_info

描述：获取共享内存变量cache。

参数：nan

返回值类型：node\_name text、 nextOid oid、 nextXid xid、 oldestXid xid、 xidVacLimit xid、 oldestXidDB oid、 lastExtendCSNLogpage xid、 startExtendCSNLogpage xid、 nextCommitSeqNo xid、 latestCompletedXid xid、 startupMaxXid xid

* pg\_get\_xidlimit

描述：从共享内存获取事物id信息。

参数：nan

返回值类型：nextXid xid、 oldestXid xid、 xidVacLimit xid、 xidWarnLimit xid、 xidStopLimit xid、 xidWrapLimit xid、 oldestXidDB oid

* get\_global\_user\_transaction()

描述：返回所有节点上各用户的事务相关信息。

返回值类型：node\_name name、 usename name、 commit\_counter bigint、 rollback\_counter bigint、 resp\_min bigint、 resp\_max bigint、 resp\_avg bigint、 resp\_total bigint、 bg\_commit\_counter bigint、 bg\_rollback\_counter bigint、 bg\_resp\_min bigint、 bg\_resp\_max bigint、 bg\_resp\_avg bigint、 bg\_resp\_total bigint

* pg\_collation\_for

描述：返回入参字符串对应的排序规则。

参数：any（如果是常量必须进行显式类型转换）

返回值类型：text

* pgxc\_unlock\_for\_sp\_database(name Name)

描述：释放指定数据库锁。

参数：数据库名

返回值类型：布尔

* pgxc\_lock\_for\_sp\_database(name Name)

描述：对指定的数据库加锁。

参数：数据库名

返回值类型：布尔

* copy\_error\_log\_create()

描述：创建COPY FROM容错机制所需要的错误表 （public.pgxc\_copy\_error\_log）。

image1返回值类型：Boolean

* 此函数会尝试创建public.pgxc\_copy\_error\_log表，表的详细信息请参见 [表7](https://docs.vastdata.com.cn/zh/docs/VastbaseG100Ver2.2.10/doc/%E5%BC%80%E5%8F%91%E8%80%85%E6%8C%87%E5%8D%97/%E5%85%B6%E5%AE%83%E5%87%BD%E6%95%B0.html" \l "public.pgxc_copy_error_log)。
* 在relname列上创建B-tree索引，并REVOKE ALL on public.pgxc\_copy\_error\_log FROM public对错误表进行权限控制（与 COPY语句权限一致）。
* 由于尝试创建的public.pgxc\_copy\_error\_log定义是一张行存表，因此数据库实例上必须支持行存表的创建才能够正常运行此函数，并使用后续的COPY容错功能。需要特别注意的是，enable\_hadoop\_env这个GUC参数开启后会禁止在数据库实例内创建行存表（GaussDB Kernel默认为off）。
* 此函数自身权限为Sysadmin及以上（与错误表、COPY权限一致）。
* 若创建前public.pgxc\_copy\_error\_log表已存在或者copy\_error\_log\_relname\_idx索引已存在，则此函数会报错回滚。

表 7 错误表public.pgxc\_copy\_error\_log信息

| **列名称** | **类型** | **描述** |
| --- | --- | --- |
| relname | character varying | 表名称。以模式名.表名形式显示。 |
| begintime | timestamp with time zone | 出现数据格式错误的时间。 |
| filename | character varying | 出现数据格式错误的数据源文件名。 |
| lineno | bigint | 在数据源文件中，出现数据格式错误的行号。 |
| rawrecord | text | 在数据源文件中，出现数据格式错误的原始记录。 |
| detail | text | 详细错误信息。 |

* dynamic\_func\_control(scope text, function\_name text, action text, “{params}” text[])

描述：动态开启内置的功能，当前仅支持动态开启全量SQL。

返回值类型：record

表 8 dynamic\_func\_control参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| scope | text | 动态开启功能的范围，当前仅支持'LOCAL'。 |
| function\_name | text | 功能的名称，当前仅支持'STMT'。 |
| action | text | 当function\_name为'STMT'时，action仅支持TRACK/UNTRACK/LIST/CLEAN：TRACK - 开始记录归一化SQL的全量SQL信息。UNTRACK - 取消记录归一化SQL的全量SQL信息。LIST - 列取当前TRACK的归一化SQL的信息。CLEAN - 清理记录当前归一化SQL的信息。 |
| params | text[] | 当function\_name为'STMT'时，对应不同的action时，对应的params设置如下：TRACK - '{“归一化SQLID”, “L0/L1/L2”}'UNTRACK - '{“归一化SQLID”}'LIST - '{}'CLEAN - '{}' |

* gs\_parse\_page\_bypath(path text, blocknum bigint, relation\_type text, read\_memory boolean)

描述：用于解析指定表页面，并返回存放解析内容的路径。

返回值类型：text

备注：必须是系统管理员或运维管理员才能执行此函数。

表 9gs\_parse\_page\_bypath参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| path | text | 对于普通表或段页式表，相对路径为：tablespace name/database oid/表的relfilenode(物理文件名)。例如：base/16603/16394。表文件的相对路径可以通过pg\_relation\_filepath(table\_name text)查找。合法的path格式列举：global/relNodebase/dbNode/relNodepg\_tblspc/spcNode/version\_dir/dbNode/relNode |
| blocknum | bigint | -1 所有block的信息0- MaxBlockNumber 对应block的信息 |
| relation\_type | text | heap(astore 表)uheap(ustore 表)btree\_index(BTree 索引)ubtree\_index(UBTree 索引)segment(段页式) |
| read\_memory | boolean | false，从磁盘文件解析。true，首先尝试从共享缓冲区中解析该页面；如果共享缓冲区中不存在，则从磁盘文件解析。 |

* gs\_xlogdump\_lsn(start\_lsn text, end\_lsn text)

描述：用于解析指定lsn范围之内的XLOG日志，并返回存放解析内容 的路径。可以通过pg\_current\_xlog\_location()获取当前XLOG位置。

返回值类型：text

参数：LSN起始位置，LSN结束位置

备注：必须是系统管理员或运维管理员才能执行此函数。

* gs\_xlogdump\_xid(c\_xid xid)

描述：用于解析指定xid的XLOG日志，并返回存放解析内容的路径。 可以通过txid\_current()获取当前事务ID。

参数：事务ID

返回值类型：text

备注：必须是系统管理员或运维管理员才能执行此函数。

* gs\_xlogdump\_tablepath(path text, blocknum bigint, relation\_type text)

描述：用于解析指定表页面对应的日志，并返回存放解析内容的路径。

返回值类型：text

备注：必须是系统管理员或运维管理员才能执行此函数。

表 10 gs\_xlogdump\_tablepath参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| path | text | 对于普通表或段页式表，相对路径为：tablespace name/database oid/表的relfilenode(物理文件名)。例如：base/16603/16394。表文件的相对路径可以通过pg\_relation\_filepath(table\_name text)查找。合法的path格式列举：global/relNodebase/dbNode/relNodepg\_tblspc/spcNode/version\_dir/dbNode/relNode |
| blocknum | bigint | -1 所有block的信息0- MaxBlockNumber 对应block的信息 |
| relation\_type | text | heap(astore 表)uheap(ustore 表)btree\_index(BTree 索引)ubtree\_index(UBTree 索引)segment(段页式) |

* gs\_xlogdump\_parsepage\_tablepath(path text, blocknum bigint, relation\_type text, read\_memory boolean)

描述：用于解析指定表页面和表页面对应的日志，并返回存放解析内容 的路径。可以看做一次执行gs\_parse\_page\_bypath和 gs\_xlogdump\_tablepath。该函数执行的前置条件是表文件存在。如果想 查看已删除的表的相关日志，请直接调用gs\_xlogdump\_tablepath。

返回值类型：text

备注：必须是系统管理员或运维管理员才能执行此函数。

表 11 gs\_xlogdump\_parsepage\_tablepath参数说明

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| path | text | 对于普通表或段页式表，相对路径为：tablespace name/database oid/表的relfilenode(物理文件名)；例如：base/16603/16394表文件的相对路径可以通过pg\_relation\_filepath(table\_name text)查找。合法的path格式列举：global/relNodebase/dbNode/relNodepg\_tblspc/spcNode/version\_dir/dbNode/relNode |
| blocknum | bigint | -1 所有block的信息0- MaxBlockNumber 对应block的信息 |
| relation\_type | text | heap(astore 表)uheap(ustore 表)btree\_index(BTree 索引)ubtree\_index(UBTree 索引)segment(段页式) |
| read\_memory | boolean | false，从磁盘文件解析true，首先尝试从共享缓冲区中解析该页面；如果共享缓冲区中不存在，则从磁盘文件解析 |

* gs\_index\_verify(Oid oid, uint32:wq blkno)

描述：用于校验UBtree索引页面或者索引树上key的顺序是否正确。

返回值类型：record

表 12 gs\_index\_verify参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| oid | Oid | 索引文件relfilenode,可以通过select relfilenode from pg\_class where relname='name'查询，其中name表示对应的索引文件名字。 |
| blkno | uint32 | 0 ，表示检验整个索引树上所有页面。大于0，表示校验页面编码等于blkno的索引页面。 |

* gs\_index\_recycle\_queue(Oid oid, int type, uint32 blkno)

描述：用于解析UBtree索引回收队列信息。

返回值类型：record

表 13 gs\_index\_recycle\_queue参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| oid | Oid | 索引文件relfilenode,可以通过select relfilenode from pg\_class where relname='name'查询，其中name表示对应的索引文件名字。 |
| type | int | 1. 表示解析整个待回收队列。 2. 表示解析整个空页队列。 3. 表示解析单个页面。 |
| blkno | uint32 | 回收队列页面编号，该参数只有在type=2的时候有效，blkno有效取值范围为1~4294967294。 |

* gs\_stat\_wal\_entrytable(int64 idx)

描述：用于输出xlog中预写日志插入状态表的内容。

返回值类型：record

表 14 gs\_stat\_wal\_entrytable参数说明

| **参数类型** | **参数名** | **类型** | **描述** |
| --- | --- | --- | --- |
| 输入参数 | idx | int64 | -1：查询数组所有元素。0-最大值：具体某个数组元素内容。 |
| 输出参数 | idx | uint64 | 记录对应数组中的下标。 |
| 输出参数 | endlsn | uint64 | 记录的LSN标签。 |
| 输出参数 | lrc | int32 | 记录对应的LRC。 |
| 输出参数 | status | uint32 | 标识当前entry对应的xlog是否已经完全拷贝到wal buffer中：0：非COPIED1：COPIED |

* gs\_walwriter\_flush\_position()

描述：输出预写日志的刷新位置。

返回值类型：record

表 15 gs\_walwriter\_flush\_position参数说明

| **参数类型** | **参数名** | **类型** | **描述** |
| --- | --- | --- | --- |
| 输出参数 | last\_flush\_status\_entry | int32 | Xlog flush上一个刷盘的tblEntry下标索引。 |
| 输出参数 | last\_scanned\_lrc | int32 | Xlog flush上一次扫描到的最后一个tblEntry记录的LRC。 |
| 输出参数 | curr\_lrc | int32 | WALInsertStatusEntry状态表中LRC最新的使用情况，该LRC表示下一个Xlog记录写入时在WALInsertStatusEntry对应的LRC值。 |
| 输出参数 | curr\_byte\_pos | uint64 | Xlog记录写入WAL文件，最新分配的位置，下一个xlog记录插入点。 |
| 输出参数 | prev\_byte\_size | uint32 | 上一个xlog记录的长度。 |
| 输出参数 | flush\_result | uint64 | 当前全局xlog刷盘的位置。 |
| 输出参数 | send\_result | uint64 | 当前主机上xlog发送位置。 |
| 输出参数 | shm\_rqst\_write\_pos | uint64 | 共享内存中记录的XLogCtl中LogwrtRqst请求的write位置。 |
| 输出参数 | shm\_rqst\_flush\_pos | uint64 | 共享内存中记录的XLogCtl中LogwrtRqst请求的flush位置。 |
| 输出参数 | shm\_result\_write\_pos | uint64 | 共享内存中记录的XLogCtl中LogwrtResult的write位置。 |
| 输出参数 | shm\_result\_flush\_pos | uint64 | 共享内存中记录的XLogCtl中LogwrtResult的flush位置。 |
| 输出参数 | curr\_time | text | 当前时间。 |

* gs\_walwriter\_flush\_stat(int operation)

描述：用于统计预写日志write与sync的次数频率与数据量，以及xlog 文件的信息。

返回值类型：record

表 16 gs\_walwriter\_flush\_stat参数说明

| **参数类型** | **参数名** | **类型** | **描述** |
| --- | --- | --- | --- |
| 输入参数 | operation | int | -1：关闭统计开关(默认状态为关闭)。0：打开统计开关。1：查询统计信息。2：重置统计信息。 |
| 输出参数 | write\_times | uint64 | Xlog调用write接口的次数。 |
| 输出参数 | sync\_times | uint64 | Xlog调用sync接口次数。 |
| 输出参数 | total\_xlog\_sync\_bytes | uint64 | Backend线程请求写入xlog总量统计值。 |
| 输出参数 | total\_actual\_xlog\_sync\_bytes | uint64 | 调用sync接口实际刷盘的xlog总量统计值。 |
| 输出参数 | avg\_write\_bytes | uint32 | 每次调用XLogWrite接口请求写的xlog量。 |
| 输出参数 | avg\_actual\_write\_bytes | uint32 | 实际每次调用write接口写的xlog量。 |
| 输出参数 | avg\_sync\_bytes | uint32 | 平均每次请求sync的xlog量。 |
| 输出参数 | avg\_actual\_sync\_bytes | uint32 | 实际每次调用sync刷盘xlog量。 |
| 输出参数 | total\_write\_time | uint64 | 调用write操作总时间统计(单位：us)。 |
| 输出参数 | total\_sync\_time | uint64 | 调用sync操作总时间统计(单位：us)。 |
| 输出参数 | avg\_write\_time | uint32 | 每次调用write接口平均时间(单位：us)。 |
| 输出参数 | avg\_sync\_time | uint32 | 每次调用sync接口平均时间(单位：us)。 |
| 输出参数 | curr\_init\_xlog\_segno | uint64 | 当前最新创建的xlog段文件编号。 |
| 输出参数 | curr\_open\_xlog\_segno | uint64 | 当前正在写的xlog段文件编号。 |
| 输出参数 | last\_reset\_time | text | 上一次重置统计信息的时间。 |
| 输出参数 | curr\_time | text | 当前时间。 |

* gs\_comm\_proxy\_thread\_status()

描述：用于在数据库实例配置用户态网络的场景下，代理通信库 comm\_proxy收发数据包统计。

参数：nan

返回值类型：record

image1此函数的查询仅在集中式环境开始部署用户态网络，且 comm\_proxy\_attr参数中enable\_dfx配置为true的条件下显示具体信 息。其他场景报错不支持查询。

##### Undo系统函数

* gs\_undo\_meta(type, zoneId, location)
* 描述：Undo各模块元信息。
* 参数说明：
* type(元信息类型)
* 0表示Undo Zone(Record) 对应的元信息。
* 1表示Undo Zone(Transaction Slot) 对应的元信息。
* 2表示Undo Space(Record) 对应的元信息。
* 3表示Undo Space(Transaction Slot) 对应的元信息。
* zoneId(undo zone编号)
* -1表示所有undo zone的元信息。
* 0-1024\*1024表示对应zoneid的元信息。
* location(读取位置）
* 0表示从当前内存中读取。
* 1表示从物理文件中读取。
* 返回值类型：record
* gs\_undo\_translot(location, zoneId)
* 描述：Undo事务槽信息。
* 参数说明：
* location(读取位置)
* 0表示从当前内存中读取。
* 1表示从物理文件中读取。
* zoneId(undo zone编号)
* -1表示所有undo zone的元信息。
* 0-1024\*1024表示对应zoneId的元信息。
* 返回值类型：record
* gs\_stat\_undo()
* 描述：Undo统计信息。
* 返回值类型：record
* **表 1** gs\_stat\_undo参数说明

| **参数类型** | **参数名** | **类型** | **描述** |
| --- | --- | --- | --- |
| 输出参数 | curr\_used\_zone\_count | uint32 | 当前使用的Undo zone数量。 |
| 输出参数 | top\_used\_zones | text | 前三个使用量最大的Undo zone信息，格式输出为：  (zoneId1:使用大小，zoneId2:使用大小，zoneId3:使用大小)。 |
| 输出参数 | curr\_used\_undo\_size | uint32 | 当前使用的Undo总空间大小，单位为MB。 |
| 输出参数 | undo\_threshold | uint32 | 为guc参数undo\_space\_limit\_size \* 80%计算的结果,单位为MB。 |
| 输出参数 | oldest\_xid\_in\_undo | uint64 | 当前Undo空间回收到的事务xid(小于该xid事务产生的Undo记录都已经被回收)。 |
| 输出参数 | oldest\_xmin | uint64 | 最老的活跃事务。 |
| 输出参数 | total\_undo\_chain\_len | int64 | 所有访问过的Undo链总长度。 |
| 输出参数 | max\_undo\_chain\_len | int64 | 最大访问过的Undo链长度。 |
| 输出参数 | create\_undo\_file\_count | uint32 | 创建的Undo文件数量统计。 |
| 输出参数 | discard\_undo\_file\_count | uint32 | 删除的Undo文件数量统计。 |

* gs\_undo\_record(undoptr)
* 描述：Undo记录解析。
* 参数说明：
* undoptr(undo记录指针)
* 返回值类型：record

#### 逻辑操作符

常用的逻辑操作符有AND、OR和NOT，他们的运算结果有三个值，分别为TRUE、FALSE和NULL，其中NULL代表未知。他们运算优先级顺序为：NOT>AND>OR。

运算规则请参见表1，表中的a和b代表逻辑表达式。

**表 1** 运算规则表

| a | b | a **AND** b的结果 | a**OR**b的结果 | **NOT** a的结果 |
| --- | --- | --- | --- | --- |
| TRUE | TRUE | TRUE | TRUE | FALSE |
| TRUE | FALSE | FALSE | TRUE | FALSE |
| TRUE | NULL | NULL | TRUE | FALSE |
| FALSE | FALSE | FALSE | FALSE | TRUE |
| FALSE | NULL | FALSE | NULL | TRUE |
| NULL | NULL | NULL | NULL | NOT NULL |

fig: **说明：**

操作符AND和OR具有交换性，即交换左右两个操作数，不影响其结果。

#### 比较操作符

大部分数据类型都可用比较操作符进行比较，并返回一个布尔类型的值。

比较操作符均为双目操作符，被比较的两个数据类型必须是相同的数据类型或者是可以进行隐式转换的类型。

Vastbase提供的比较操作符请参见表。

**表 1** 比较操作符

| **操作符** | **描述** |
| --- | --- |
| < | 小于 |
| > | 大于 |
| <= | 小于或等于 |
| >= | 大于或等于 |
| = | 等于 |
| <> 或 !=或^= | 不等于 |

比较操作符可以用于所有相关的数据类型。所有比较操作符都是双目操作符，返回布尔类型数值。不等号的计算优先级高于等号。当输入的数据不同且无法隐式转换时, 比较操作将会失败。例如像1<2<3这样的表达式是非法的，因为布尔值和3之间无法用小于号（<）比较。

#### 二进制字符串函数和操作符

**字符串操作符**

SQL定义了一些字符串函数，在这些函数里使用关键字而不是逗号来分隔参数。

* octet\_length(string)
* 描述：二进制字符串中的字节数。
* 返回值类型：int
* 示例：

vastbase=# SELECT octet\_length(E'jo\\000se'::bytea) AS RESULT;

result

--------

5

(1 row)

* overlay(string placing string from int [for int])
* 描述：替换子串。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT overlay(E'Th\\000omas'::bytea placing E'\\002\\003'::bytea from 2 for 3) AS RESULT;

result

----------------

\x5402036d6173

(1 row)

* position(substring in string)
* 描述：特定子字符串的位置。
* 返回值类型：int
* 示例：

vastbase=# SELECT position(E'\\000om'::bytea in E'Th\\000omas'::bytea) AS RESULT;

result

--------

3

(1 row)

* substring(string [from int] [for int])
* 描述：截取子串。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT substring(E'Th\\000omas'::bytea from 2 for 3) AS RESULT;

result

----------

\x68006f

(1 row)

* substr(string, from int [, for int])
* 描述：截取子串。
* 返回值类型：bytea
* 示例：

vastbase=# select substr(E'Th\\000omas'::bytea,2, 3) as result;

result

----------

\x68006f

(1 row)

* trim([both] bytes from string)
* 描述：从string的开头和结尾删除只包含bytes中字节的最长字符串。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT trim(E'\\000'::bytea from E'\\000Tom\\000'::bytea) AS RESULT;

result

----------

\x546f6d

(1 row)

**二进制字符串函数**

Vastbase也提供了函数调用所使用的常用语法。

* btrim(string bytea,bytes bytea)
* 描述：从string的开头和结尾删除只包含bytes中字节的最长的字符串。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT btrim(E'\\000trim\\000'::bytea, E'\\000'::bytea) AS RESULT;

result

------------

\x7472696d

(1 row)

* get\_bit(string, offset)
* 描述：从字符串中抽取位。
* 返回值类型：int
* 示例：

vastbase=# SELECT get\_bit(E'Th\\000omas'::bytea, 45) AS RESULT;

result

--------

1

(1 row)

* get\_byte(string, offset)
* 描述：从字符串中抽取字节。
* 返回值类型：int
* 示例：

vastbase=# SELECT get\_byte(E'Th\\000omas'::bytea, 4) AS RESULT;

result

--------

109

(1 row)

* set\_bit(string,offset, newvalue)
* 描述：设置字符串中的位。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT set\_bit(E'Th\\000omas'::bytea, 45, 0) AS RESULT;

result

------------------

\x5468006f6d4173

(1 row)

* set\_byte(string,offset, newvalue)
* 描述：设置字符串中的字节。
* 返回值类型：bytea
* 示例：

vastbase=# SELECT set\_byte(E'Th\\000omas'::bytea, 4, 64) AS RESULT;

result

------------------

\x5468006f406173

(1 row)

#### 位串函数和操作符

**位串操作符**

除了常用的比较操作符之外，还可以使用以下的操作符。&、|和#的位串操作数必须等长。在位移的时候，保留原始的位串长度（并以0填充）。

* ||
* 描述：位串之间进行连接。
* 示例：

vastbase=# SELECT B'10001' || B'011' AS RESULT;

result

----------

10001011

(1 row)

fig: **说明：**

单字段内部连续连接操作不建议超过180次。如果超过180次，需拆分为多个连续连接的字符串，在它们之间再执行连接操作。例如：str1||str2||str3||str4 拆分为 (str1||str2)||(str3||str4)。

* &
* 描述：位串之间进行“与”操作。
* 示例：

vastbase=# SELECT B'10001' & B'01101' AS RESULT;

result

--------

00001

(1 row)

* |
* 描述：位串之间进行“或”操作。
* 示例：

vastbase=# SELECT B'10001' | B'01101' AS RESULT;

result

--------

11101

(1 row)

* #
* 描述：位串之间如果不一致进行“或”操作。如果两个位串中对应位置都为1或者0，则该位置返回为0。
* 示例：

vastbase=# SELECT B'10001' # B'01101' AS RESULT;

result

--------

11100

(1 row)

* ~
* 描述：位串之间进行“非”操作。
* 示例：

vastbase=# SELECT ~B'10001'AS RESULT;

result

----------

01110

(1 row)

* <<
* 描述：位串进行左移操作。
* 示例：

vastbase=# SELECT B'10001' << 3 AS RESULT;

result

----------

01000

(1 row)

* >>
* 描述：位串进行右移操作。
* 示例：

vastbase=# SELECT B'10001' >> 2 AS RESULT;

result

----------

00100

(1 row)

下面的SQL标准函数除了可以用于字符串之外，也可以用于位串：lengthbit\_length、octet\_length、position、substring、overlay。

下面的函数用于位串和二进制字符串：get\_bit、set\_bit。当用于位串时，这些函数位数从字符串的第一位（最左边）作为0位。

另外，可以在整数和bit之间来回转换。示例：

vastbase=# SELECT 44::bit(10) AS RESULT;

result

------------

0000101100

(1 row)

vastbase=# SELECT 44::bit(3) AS RESULT;

result

--------

100

(1 row)

vastbase=# SELECT cast(-44 as bit(12)) AS RESULT;

result

--------------

111111010100

(1 row)

vastbase=# SELECT '1110'::bit(4)::integer AS RESULT;

result

--------

14

(1 row)

vastbase=# select substring('10101111'::bit(8), 2);

substring

-----------

0101111

(1 row)

fig: **说明：**

只是转换为“bit”的意思是转换成bit(1)，因此只会转换成整数的最低位。

#### 模式匹配操作符

数据库提供了三种独立的实现模式匹配的方法：SQL LIKE操作符、SIMILAR TO操作符和POSIX-风格的正则表达式。除了这些基本的操作符外，还有一些函数可用于提取或替换匹配子串并在匹配位置分离一个串。

* LIKE
* 描述：判断字符串是否能匹配上LIKE后的模式字符串。如果字符串与提供的模式匹配，则LIKE表达式返回为真（NOT LIKE表达式返回假），否则返回为假（NOT LIKE表达式返回真）。
* 匹配规则：
* 1、此操作符只有在它的模式匹配整个串的时候才能成功。如果要匹配在串内任何位置的序列，该模式必须以百分号开头和结尾。
* 2、下划线（\_）代表（匹配）任何单个字符；百分号（%）代表任意串的通配符。
* 3、要匹配文本里的下划线或者百分号，在提供的模式里相应字符必须前导逃逸字符。逃逸字符的作用是禁用元字符的特殊含义，缺省的逃逸字符是反斜线，也可以用ESCAPE子句指定一个不同的逃逸字符。
* 4、要匹配逃逸字符本身，写两个逃逸字符。例如要写一个包含反斜线的模式常量，那你就要在SQL语句里写两个反斜线。
* fig: **说明：**
* 参数standard\_conforming\_strings设置为off时，在文串常量中写的任何反斜线都需要被双写。因此，写一个匹配单个反斜线的模式实际上要在语句里写四个反斜线（你可以通过用ESCAPE选择一个不同的逃逸字符来避免这种情况，这样反斜线就不再是LIKE的特殊字符了。但仍然是字符文本分析器的特殊字符，所以你还是需要两个反斜线）。在兼容MYSQL数据模式时，您也可以通过写ESCAPE ''的方式不选择逃逸字符，这样可以有效地禁用逃逸机制，但是没有办法关闭下划线和百分号在模式中的特殊含义。
* 5、关键字ILIKE可以用于替换LIKE，区别是LIKE大小写敏感，ILIKE大小写不敏感。
* 6、操作符~~等效于LIKE，操作符~~\*等效于ILIKE。
* 示例：

vastbase=# SELECT 'abc' LIKE 'abc' AS RESULT;

result

-----------

t

(1 row)

vastbase=# SELECT 'abc' LIKE 'a%' AS RESULT;

result

-----------

t

(1 row)

vastbase=# SELECT 'abc' LIKE '\_b\_' AS RESULT;

result

-----------

t

(1 row)

vastbase=# SELECT 'abc' LIKE 'c' AS RESULT;

result

-----------

f

(1 row)

* SIMILAR TO
* 描述：SIMILAR TO操作符根据自己的模式是否匹配给定串而返回真或者假。它和LIKE非常类似，只不过它使用SQL标准定义的正则表达式理解模式。
* 匹配规则：
* 1、和LIKE一样，此操作符只有在它的模式匹配整个串的时候才能成功。如果要匹配在串内任何位置的序列，该模式必须以百分号开头和结尾。
* 2、下划线（\_）代表（匹配）任何单个字符；百分号（%）代表任意串的通配符。
* 3、SIMILAR TO也支持下面这些从POSIX正则表达式借用的模式匹配元字符。

| **元字符** | **含义** |
| --- | --- |
| | | 表示选择（两个候选之一） |
| \* | 表示重复前面的项零次或更多次 |
| + | 表示重复前面的项一次或更多次 |
| ? | 表示重复前面的项零次或一次 |
| {m} | 表示重复前面的项刚好m次 |
| {m,} | 表示重复前面的项m次或更多次 |
| {m,n} | 表示重复前面的项至少m次并且不超过n次 |
| () | 把多个项组合成一个逻辑项 |
| […] | 声明一个字符类，就像POSIX正则表达式一样 |

4、前导逃逸字符可以禁止所有这些元字符的特殊含义。逃逸字符的使用规则和LIKE一样。

正则表达式函数：

支持使用函数substring(string from pa...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的substring(string from pa...)截取匹配SQL正则表达式的子字符串。

示例：

vastbase=# SELECT 'abc' SIMILAR TO 'abc' AS RESULT;

result

-----------

t

(1 row)

vastbase=# SELECT 'abc' SIMILAR TO 'a' AS RESULT;

result

-----------

f

(1 row)

vastbase=# SELECT 'abc' SIMILAR TO '%(b|d)%' AS RESULT;

result

-----------

t

(1 row)

vastbase=# SELECT 'abc' SIMILAR TO '(b|c)%' AS RESULT;

result

-----------

f

(1 row)

* POSIX正则表达式
* 描述：正则表达式是一个字符序列，它是定义一个串集合（一个正则集）的缩写。如果一个串是正则表达式描述的正则集中的一员时，我们就说这个串匹配该正则表达式。 POSIX正则表达式提供了比LIKE和SIMILAR TO操作符更强大的含义。[表1](#X93838b3daec8cffb83749746c3dfd39d3a50269)列出了所有可用于POSIX正则表达式模式匹配的操作符。
* **表 1** 正则表达式匹配操作符

| **操作符** | **描述** | **例子** |
| --- | --- | --- |
| ~ | 匹配正则表达式，大小写敏感 | 'thomas' ~ '.thomas.' |
| ~ | 匹配正则表达式，大小写不敏感 | 'thomas' ~ '.Thomas.' |
| !~ | 不匹配正则表达式，大小写敏感 | 'thomas' !~ '.Thomas.' |
| !~ | 不匹配正则表达式，大小写不敏感 | 'thomas' !~ '.vadim.' |

匹配规则：

1、与LIKE不同，正则表达式允许匹配串里的任何位置，除非该正则表达式显式地挂接在串的开头或者结尾。

2、除了上文提到的元字符外， POSIX正则表达式还支持下列模式匹配元字符。

正则表达式函数：

POSIX正则表达式支持下面函数。

* substring(string from pa...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的substring(string from pa...)函数提供了抽取一个匹配POSIX正则表达式模式的子串的方法。
* regexp\_count(string tex...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_count(string tex...)函数提供了获取匹配POSIX正则表达式模式的子串数量的功能。
* regexp\_instr(string tex...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_instr(string tex...)函数提供了获取匹配POSIX正则表达式模式子串位置的功能。
* regexp\_substr(string te...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_substr(string te...)函数提供了抽取一个匹配POSIX正则表达式模式的子串的方法。
* regexp\_replace(string, p...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_replace(string, p...)函数提供了将匹配POSIX正则表达式模式的子串替换为新文本的功能。
* regexp\_matches(string te...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_matches(string te...)函数返回一个文本数组，该数组由匹配一个POSIX正则表达式模式得到的所有被捕获子串构成。
* regexp\_split\_to\_table(st...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_split\_to\_table(st...)函数把一个POSIX正则表达式模式当作一个定界符来分离一个串。
* regexp\_split\_to\_array(st...(参考：开发者指南->SQL语法参考->函数与操作符->字符处理函数和操作符章节的regexp\_split\_to\_array(st...)和regexp\_split\_to\_table类似，是一个正则表达式分离函数，不过它的结果以一个text数组的形式返回。
* fig: **说明：**
* 正则表达式分离函数会忽略零长度的匹配，这种匹配发生在串的开头或结尾或者正好发生在前一个匹配之后。这和正则表达式匹配的严格定义是相悖的，后者由regexp\_matches实现，但是通常前者是实际中最常用的行为。

示例：

vastbase=# SELECT 'abc' ~ 'Abc' AS RESULT;

result

--------

f

(1 row)

vastbase=# SELECT 'abc' ~\* 'Abc' AS RESULT;

result

--------

t

(1 row)

vastbase=# SELECT 'abc' !~ 'Abc' AS RESULT;

result

--------

t

(1 row)

vastbase=# SELECT 'abc'!~\* 'Abc' AS RESULT;

result

--------

f

(1 row)

vastbase=# SELECT 'abc' ~ '^a' AS RESULT;

result

--------

t

(1 row)

vastbase=# SELECT 'abc' ~ '(b|d)'AS RESULT;

result

--------

t

(1 row)

vastbase=# SELECT 'abc' ~ '^(b|c)'AS RESULT;

result

--------

f

(1 row)

虽然大部分的正则表达式搜索都能很快地执行，但是正则表达式仍可能被人为地弄成需要任意长的时间和任意量的内存进行处理。不建议从非安全模式来源接受正则表达式搜索模式，如果必须这样做，建议加上语句超时限制。使用SIMILAR TO模式的搜索具有同样的安全性危险，因为SIMILAR TO提供了很多和POSIX-风格正则表达式相同的能力。LIKE搜索比其他两种选项简单得多，因此在接受非安全模式来源搜索时要更安全些。

#### 几何函数和操作符

**几何操作符**

* +
* 描述：平移。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' + point '(2.0,0)' AS RESULT;

result

-------------

(3,1),(2,0)

(1 row)

* -
* 描述：平移。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' - point '(2.0,0)' AS RESULT;

result

---------------

(-1,1),(-2,0)

(1 row)

* \*
* 描述：伸展/旋转。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' \* point '(2.0,0)' AS RESULT;

result

-------------

(2,2),(0,0)

(1 row)

* /
* 描述：收缩/旋转。
* 示例：

vastbase=# SELECT box '((0,0),(2,2))' / point '(2.0,0)' AS RESULT;

result

-------------

(1,1),(0,0)

(1 row)

* #
* 描述：两个图形交面。
* 示例：

vastbase=# SELECT box '((1,-1),(-1,1))' # box '((1,1),(-2,-2))' AS RESULT;

result

---------------

(1,1),(-1,-1)

(1 row)

* #
* 描述：图形的路径数目或多边形顶点数。
* 示例：

vastbase=# SELECT # path'((1,0),(0,1),(-1,0))' AS RESULT;

result

--------

3

(1 row)

* @-@
* 描述：图形的长度或者周长。
* 示例：

vastbase=# SELECT @-@ path '((0,0),(1,0))' AS RESULT;

result

--------

2

(1 row)

* @@
* 描述：图形的中心。
* 示例：

vastbase=# SELECT @@ circle '((0,0),10)' AS RESULT;

result

--------

(0,0)

(1 row)

* <->
* 描述：两个图形之间的距离。
* 示例：

vastbase=# SELECT circle '((0,0),1)' <-> circle '((5,0),1)' AS RESULT;

result

--------

3

(1 row)

* &&
* 描述：两个图形是否重叠（有一个共同点就为真）。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' && box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* <<
* 描述：图形是否全部在另一个图形的左边（没有相同的横坐标）。
* 示例：

vastbase=# SELECT circle '((0,0),1)' << circle '((5,0),1)' AS RESULT;

result

--------

t

(1 row)

* >>
* 描述：图形是否全部在另一个图形的右边（没有相同的横坐标）。
* 示例：

vastbase=# SELECT circle '((5,0),1)' >> circle '((0,0),1)' AS RESULT;

result

--------

t

(1 row)

* &<
* 描述：图形的最右边是否不超过在另一个图形的最右边。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' &< box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* &>
* 描述：图形的最左边是否不超过在另一个图形的最左边。
* 示例：

vastbase=# SELECT box '((0,0),(3,3))' &> box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* <<|
* 描述：图形是否全部在另一个图形的下边（没有相同的纵坐标）。
* 示例：

vastbase=# SELECT box '((0,0),(3,3))' <<| box '((3,4),(5,5))' AS RESULT;

result

--------

t

(1 row)

* |>>
* 描述：图形是否全部在另一个图形的上边（没有相同的纵坐标）。
* 示例：

vastbase=# SELECT box '((3,4),(5,5))' |>> box '((0,0),(3,3))' AS RESULT;

result

--------

t

(1 row)

* &<|
* 描述：图形的最上边是否不超过另一个图形的最上边。
* 示例：

vastbase=# SELECT box '((0,0),(1,1))' &<| box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* |&>
* 描述：图形的最下边是否不超过另一个图形的最下边。
* 示例：

vastbase=# SELECT box '((0,0),(3,3))' |&> box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* <^
* 描述：图形是否低于另一个图形（允许两个图形有接触）。
* 示例：

vastbase=# SELECT box '((0,0),(-3,-3))' <^ box '((0,0),(2,2))' AS RESULT;

result

--------

t

(1 row)

* >^
* 描述：图形是否高于另一个图形（允许两个图形有接触）。
* 示例：

vastbase=# SELECT box '((0,0),(2,2))' >^ box '((0,0),(-3,-3))' AS RESULT;

result

--------

t

(1 row)

* ?#
* 描述：两个图形是否相交。
* 示例：

vastbase=# SELECT lseg '((-1,0),(1,0))' ?# box '((-2,-2),(2,2))' AS RESULT;

result

--------

t

(1 row)

* ?-
* 描述：图形是否处于水平位置。
* 示例：

vastbase=# SELECT ?- lseg '((-1,0),(1,0))' AS RESULT;

result

--------

t

(1 row)

* ?-
* 描述：图形是否水平对齐。
* 示例：

vastbase=# SELECT point '(1,0)' ?- point '(0,0)' AS RESULT;

result

--------

t

(1 row)

* ?|
* 描述：图形是否处于竖直位置。
* 示例：

vastbase=# SELECT ?| lseg '((-1,0),(1,0))' AS RESULT;

result

--------

f

(1 row)

* ?|
* 描述：图形是否竖直对齐。
* 示例：

vastbase=# SELECT point '(0,1)' ?| point '(0,0)' AS RESULT;

result

--------

t

(1 row)

* ?-|
* 描述：两条线是否垂直。
* 示例：

vastbase=# SELECT lseg '((0,0),(0,1))' ?-| lseg '((0,0),(1,0))' AS RESULT;

result

--------

t

(1 row)

* ?||
* 描述：两条线是否平行。
* 示例：

vastbase=# SELECT lseg '((-1,0),(1,0))' ?|| lseg '((-1,2),(1,2))' AS RESULT;

result

--------

t

(1 row)

* @>
* 描述：图形是否包含另一个图形。
* 示例：

vastbase=# SELECT circle '((0,0),2)' @> point '(1,1)' AS RESULT;

result

--------

t

(1 row)

* <@
* 描述：图形是否被包含于另一个图形。
* 示例：

vastbase=# SELECT point '(1,1)' <@ circle '((0,0),2)' AS RESULT;

result

--------

t

(1 row)

* ~=
* 描述：两个图形是否相同。
* 示例：

vastbase=# SELECT polygon '((0,0),(1,1))' ~= polygon '((1,1),(0,0))' AS RESULT;

result

--------

t

(1 row)

**几何函数**

* area(object)
* 描述：计算图形的面积。
* 返回类型：double precision
* 示例：

vastbase=# SELECT area(box '((0,0),(1,1))') AS RESULT;

result

--------

1

(1 row)

* center(object)
* 描述：计算图形的中心。
* 返回类型：point
* 示例：

vastbase=# SELECT center(box '((0,0),(1,2))') AS RESULT;

result

---------

(0.5,1)

(1 row)

* diameter(circle)
* 描述：计算圆的直径。
* 返回类型：double precision
* 示例：

vastbase=# SELECT diameter(circle '((0,0),2.0)') AS RESULT;

result

--------

4

(1 row)

* height(box)
* 描述：矩形的竖直高度。
* 返回类型：double precision
* 示例：

vastbase=# SELECT height(box '((0,0),(1,1))') AS RESULT;

result

--------

1

(1 row)

* isclosed(path)
* 描述：图形是否为闭合路径。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT isclosed(path '((0,0),(1,1),(2,0))') AS RESULT;

result

--------

t

(1 row)

* isopen(path)
* 描述：图形是否为开放路径。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT isopen(path '[(0,0),(1,1),(2,0)]') AS RESULT;

result

--------

t

(1 row)

* length(object)
* 描述：计算图形的长度。
* 返回类型：double precision
* 示例：

vastbase=# SELECT length(path '((-1,0),(1,0))') AS RESULT;

result

--------

4

(1 row)

* npoints(path)
* 描述：计算路径的顶点数。
* 返回类型：int
* 示例：

vastbase=# SELECT npoints(path '[(0,0),(1,1),(2,0)]') AS RESULT;

result

--------

3

(1 row)

* npoints(polygon)
* 描述：计算多边形的顶点数。
* 返回类型：int
* 示例：

vastbase=# SELECT npoints(polygon '((1,1),(0,0))') AS RESULT;

result

--------

2

(1 row)

* pclose(path)
* 描述：把路径转换为闭合路径。
* 返回类型：path
* 示例：

vastbase=# SELECT pclose(path '[(0,0),(1,1),(2,0)]') AS RESULT;

result

---------------------

((0,0),(1,1),(2,0))

(1 row)

* popen(path)
* 描述：把路径转换为开放路径。
* 返回类型：path
* 示例：

vastbase=# SELECT popen(path '((0,0),(1,1),(2,0))') AS RESULT;

result

---------------------

[(0,0),(1,1),(2,0)]

(1 row)

* radius(circle)
* 描述：计算圆的半径。
* 返回类型：double precision
* 示例：

vastbase=# SELECT radius(circle '((0,0),2.0)') AS RESULT;

result

--------

2

(1 row)

* width(box)
* 描述：计算矩形的水平尺寸。
* 返回类型：double precision
* 示例：

vastbase=# SELECT width(box '((0,0),(1,1))') AS RESULT;

result

--------

1

(1 row)

**几何类型转换函数**

* box(circle)
* 描述：将圆转换成矩形
* 返回类型：box
* 示例：

vastbase=# SELECT box(circle '((0,0),2.0)') AS RESULT;

result

---------------------------------------------------------------------------

(1.41421356237309,1.41421356237309),(-1.41421356237309,-1.41421356237309)

(1 row)

* box(point, point)
* 描述：将点转换成矩形
* 返回类型：box
* 示例：

vastbase=# SELECT box(point '(0,0)', point '(1,1)') AS RESULT;

result

-------------

(1,1),(0,0)

(1 row)

* box(polygon)
* 描述：将多边形转换成矩形
* 返回类型：box
* 示例：

vastbase=# SELECT box(polygon '((0,0),(1,1),(2,0))') AS RESULT;

result

-------------

(2,1),(0,0)

(1 row)

* circle(box)
* 描述：矩形转换成圆
* 返回类型：circle
* 示例：

vastbase=# SELECT circle(box '((0,0),(1,1))') AS RESULT;

result

-------------------------------

<(0.5,0.5),0.707106781186548>

(1 row)

* circle(point, double precision)
* 描述：将圆心和半径转换成圆
* 返回类型：circle
* 示例：

vastbase=# SELECT circle(point '(0,0)', 2.0) AS RESULT;

result

-----------

<(0,0),2>

(1 row)

* circle(polygon)
* 描述：将多边形转换成圆
* 返回类型：circle
* 示例：

vastbase=# SELECT circle(polygon '((0,0),(1,1),(2,0))') AS RESULT;

result

-------------------------------------------

<(1,0.333333333333333),0.924950591148529>

(1 row)

* lseg(box)
* 描述：矩形对角线转化成线段
* 返回类型：lseg
* 示例：

vastbase=# SELECT lseg(box '((-1,0),(1,0))') AS RESULT;

result

----------------

[(1,0),(-1,0)]

(1 row)

* lseg(point, point)
* 描述：点转换成线段
* 返回类型：lseg
* 示例：

vastbase=# SELECT lseg(point '(-1,0)', point '(1,0)') AS RESULT;

result

----------------

[(-1,0),(1,0)]

(1 row)

* slope(point, point)
* 描述：计算两个点构成直线的斜率
* 返回类型: double
* 示例：

vastbase=# SELECT slope(point '(1,1)', point '(0,0)') AS RESULT;

result

--------

1

(1 row)

* path(polygon)
* 描述：多边形转换成路径
* 返回类型：path
* 示例：

vastbase=# SELECT path(polygon '((0,0),(1,1),(2,0))') AS RESULT;

result

---------------------

((0,0),(1,1),(2,0))

(1 row)

* point(double precision, double precision)
* 描述：节点
* 返回类型：point
* 示例：

vastbase=# SELECT point(23.4, -44.5) AS RESULT;

result

--------------

(23.4,-44.5)

(1 row)

* point(box)
* 描述：矩形的中心
* 返回类型：point
* 示例：

vastbase=# SELECT point(box '((-1,0),(1,0))') AS RESULT;

result

--------

(0,0)

(1 row)

* point(circle)
* 描述：圆心
* 返回类型：point
* 示例：

vastbase=# SELECT point(circle '((0,0),2.0)') AS RESULT;

result

--------

(0,0)

(1 row)

* point(lseg)
* 描述：线段的中心
* 返回类型：point
* 示例：

vastbase=# SELECT point(lseg '((-1,0),(1,0))') AS RESULT;

result

--------

(0,0)

(1 row)

* point(polygon)
* 描述：多边形的中心
* 返回类型：point
* 示例：

vastbase=# SELECT point(polygon '((0,0),(1,1),(2,0))') AS RESULT;

result

-----------------------

(1,0.333333333333333)

(1 row)

* polygon(box)
* 描述：矩形转换成4点多边形
* 返回类型：polygon
* 示例：

vastbase=# SELECT polygon(box '((0,0),(1,1))') AS RESULT;

result

---------------------------

((0,0),(0,1),(1,1),(1,0))

(1 row)

* polygon(circle)
* 描述：圆转换成12点多边形
* 返回类型：polygon
* 示例：

vastbase=# SELECT polygon(circle '((0,0),2.0)') AS RESULT;

result

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

((-2,0),(-1.73205080756888,1),(-1,1.73205080756888),(-1.22464679914735e-16,2),(1,1.73205080756888),(1.73205080756888,1),(2,2.44929359829471e-16),(1.73205080756888,-0.999999999999999),(1,-1.73205080756888),(3.67394039744206e-16,-2),(-0.999999999999999,-1.73205080756888),(-1.73205080756888,-1))

(1 row)

* polygon(npts, circle)
* 描述：圆转换成npts点多边形
* 返回类型：polygon
* 示例：

vastbase=# SELECT polygon(12, circle '((0,0),2.0)') AS RESULT;

result

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

((-2,0),(-1.73205080756888,1),(-1,1.73205080756888),(-1.22464679914735e-16,2),(1,1.73205080756888),(1.73205080756888,1),(2,2.44929359829471e-16),(1.73205080756888,-0.999999999999999),(1,-1.73205080756888),(3.67394039744206e-16,-2),(-0.999999999999999,-1.73205080756888),(-1.73205080756888,-1))

(1 row)

* polygon(path)
* 描述：路径转换成多边形
* 返回类型：polygon
* 示例：

vastbase=# SELECT polygon(path '((0,0),(1,1),(2,0))') AS RESULT;

result

---------------------

((0,0),(1,1),(2,0))

(1 row)

#### 网络地址函数和操作符

**cidr和inet操作符**

操作符<<、<<=、>>、>>=对子网进行测试。它们只考虑两个地址的网络部分（忽略任何主机部分），然后判断其中一个网络是等于另外一个网络，还是另外一个网络的子网。

* <
* 描述：小于
* 示例：

vastbase=# SELECT inet '192.168.1.5' < inet '192.168.1.6' AS RESULT;

result

--------

t

(1 row)

* <=
* 描述：小于或等于
* 示例：

vastbase=# SELECT inet '192.168.1.5' <= inet '192.168.1.5' AS RESULT;

result

--------

t

(1 row)

* =
* 描述：等于
* 示例：

vastbase=# SELECT inet '192.168.1.5' = inet '192.168.1.5' AS RESULT;

result

--------

t

(1 row)

* >=
* 描述：大于或等于
* 示例：

vastbase=# SELECT inet '192.168.1.5' >= inet '192.168.1.5' AS RESULT;

result

--------

t

(1 row)

* >
* 描述：大于
* 示例：

vastbase=# SELECT inet '192.168.1.5' > inet '192.168.1.4' AS RESULT;

result

--------

t

(1 row)

* <>
* 描述：不等于
* 示例：

vastbase=# SELECT inet '192.168.1.5' <> inet '192.168.1.4' AS RESULT;

result

--------

t

(1 row)

* <<
* 描述：包含于
* 示例：

vastbase=# SELECT inet '192.168.1.5' << inet '192.168.1/24' AS RESULT;

result

--------

t

(1 row)

* <<=
* 描述：包含于或等于
* 示例：

vastbase=# SELECT inet '192.168.1/24' <<= inet '192.168.1/24' AS RESULT;

result

--------

t

(1 row)

* >>
* 描述：包含
* 示例：

vastbase=# SELECT inet '192.168.1/24' >> inet '192.168.1.5' AS RESULT;

result

--------

t

(1 row)

* >>=
* 描述：包含或等于
* 示例：

vastbase=# SELECT inet '192.168.1/24' >>= inet '192.168.1/24' AS RESULT;

result

--------

t

(1 row)

* ~
* 描述：位非
* 示例：

vastbase=# SELECT ~ inet '192.168.1.6' AS RESULT;

result

---------------

63.87.254.249

(1 row)

* &
* 描述：两个网络地址的每一位都进行“与”操作
* 示例：

vastbase=# SELECT inet '192.168.1.6' & inet '10.0.0.0' AS RESULT;

result

---------

0.0.0.0

(1 row)

* |
* 描述：两个网络地址的每一位都进行“或”操作
* 示例：

vastbase=# SELECT inet '192.168.1.6' | inet '10.0.0.0' AS RESULT;

result

-------------

202.168.1.6

(1 row)

* +
* 描述：加
* 示例：

vastbase=# SELECT inet '192.168.1.6' + 25 AS RESULT;

result

--------------

192.168.1.31

(1 row)

* -
* 描述：减
* 示例：

vastbase=# SELECT inet '192.168.1.43' - 36 AS RESULT;

result

-------------

192.168.1.7

(1 row)

* -
* 描述：减
* 示例：

vastbase=# SELECT inet '192.168.1.43' - inet '192.168.1.19' AS RESULT;

result

--------

24

(1 row)

**cidr和inet函数**

函数abbrev、host、text主要是为了提供可选的显示格式。

* abbrev(inet)
* 描述：缩写显示格式文本。
* 返回类型：text
* 示例：

vastbase=# SELECT abbrev(inet '10.1.0.0/16') AS RESULT;

result

-------------

10.1.0.0/16

(1 row)

* abbrev(cidr)
* 描述：缩写显示格式文本。
* 返回类型：text
* 示例：

vastbase=# SELECT abbrev(cidr '10.1.0.0/16') AS RESULT;

result

---------

10.1/16

(1 row)

* broadcast(inet)
* 描述：网络广播地址。
* 返回类型：inet
* 示例：

vastbase=# SELECT broadcast('192.168.1.5/24') AS RESULT;

result

------------------

192.168.1.255/24

(1 row)

* family(inet)
* 描述：抽取地址族，4为IPv4，6为IPv6。
* 返回类型：int
* 示例：

vastbase=# SELECT family('127.0.0.1') AS RESULT;

result

--------

4

(1 row)

* host(inet)
* 描述：将主机地址类型抽出为文本。
* 返回类型：text
* 示例：

vastbase=# SELECT host('192.168.1.5/24') AS RESULT;

result

-------------

192.168.1.5

(1 row)

* hostmask(inet)
* 描述：为网络构造主机掩码。
* 返回类型：inet
* 示例：

vastbase=# SELECT hostmask('192.168.23.20/30') AS RESULT;

result

---------

0.0.0.3

(1 row)

* masklen(inet)
* 描述：抽取子网掩码长度。
* 返回类型：int
* 示例：

vastbase=# SELECT masklen('192.168.1.5/24') AS RESULT;

result

--------

24

(1 row)

* netmask(inet)
* 描述：为网络构造子网掩码。
* 返回类型：inet

示例：

vastbase=# SELECT netmask('192.168.1.5/24') AS RESULT;

result

---------------

255.255.255.0

(1 row)

* network(inet)
* 描述：抽取地址的网络部分。
* 返回类型：cidr
* 示例：

vastbase=# SELECT network('192.168.1.5/24') AS RESULT;

result

----------------

192.168.1.0/24

(1 row)

* set\_masklen(inet, int)
* 描述：为inet数值设置子网掩码长度。
* 返回类型：inet
* 示例：

vastbase=# SELECT set\_masklen('192.168.1.5/24', 16) AS RESULT;

result

----------------

192.168.1.5/16

(1 row)

* set\_masklen(cidr, int)
* 描述：为cidr数值设置子网掩码长度。
* 返回类型：cidr
* 示例：

vastbase=# SELECT set\_masklen('192.168.1.0/24'::cidr, 16) AS RESULT;

result

----------------

192.168.0.0/16

(1 row)

* text(inet)
* 描述：把IP地址和掩码长度抽取为文本。
* 返回类型：text
* 示例：

vastbase=# SELECT text(inet '192.168.1.5') AS RESULT;

result

----------------

192.168.1.5/32

(1 row)

任何cidr值都能以显式或者隐式的方式转换为inet值，因此上述能够操作inet值的函数也同样能够操作cidr值。inet值也可以转换为cidr值，此时inet子网掩码右侧的所有位都将转换为零，以创建一个有效的cidr值。另外，用户还可以使用常规的类型转换语法将一个文本字符串转换为inet或cidr值。例如：inet(expression)或colname::cidr。

**macaddr函数**

函数trunc(macaddr)返回一个MAC地址，该地址的最后三个字节设置为零。

* trunc(macaddr)
* 描述：把后三个字节置为零。
* 返回类型：macaddr
* 示例：

vastbase=# SELECT trunc(macaddr '12:34:56:78:90:ab') AS RESULT;

result

-------------------

12:34:56:00:00:00

(1 row)

macaddr类型还支持标准关系操作符（>、<=等）用于词法排序，和按位运算符（~、&和|）非、与和或。

#### HLL函数和操作符

**哈希函数**

* hll\_hash\_boolean(bool)
* 描述：对bool类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_boolean(FALSE);

hll\_hash\_boolean

---------------------

-5451962507482445012

(1 row)

* hll\_hash\_boolean(bool, int32)
* 描述：设置hash seed（即改变哈希策略）并对bool类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_boolean(FALSE, 10);

hll\_hash\_boolean

--------------------

-1169037589280886076

(1 row)

* hll\_hash\_smallint(smallint)
* 描述：对smallint类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_smallint(100::smallint);

hll\_hash\_smallint

---------------------

962727970174027904

(1 row)

fig: **说明：**

数值大小相同的参数使用不同数据类型的哈希函数计算，最后结果会不一样，因为不同类型哈希函数会选取不同的哈希计算策略。

* hll\_hash\_smallint(smallint, int32)
* 描述：设置hash seed（即改变哈希策略）同时对smallint类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vatbase=# SELECT hll\_hash\_smallint(100::smallint, 10);

hll\_hash\_smallint

---------------------

-9056177146160443041

(1 row)

* hll\_hash\_integer(integer)
* 描述：对integer类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_integer(0);

hll\_hash\_integer

----------------------

5156626420896634997

(1 row)

* hll\_hash\_integer(integer, int32)
* 描述：对integer类型数据计算哈希值，并设置hashseed（即改变哈希策略）。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_integer(0, 10);

hll\_hash\_integer

--------------------

-5035020264353794276

(1 row)

* hll\_hash\_bigint(bigint)
* 描述：对bigint类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_bigint(100::bigint);

hll\_hash\_bigint

---------------------

-2401963681423227794

(1 row)

* hll\_hash\_bigint(bigint, int32)
* 描述：对bigint类型数据计算哈希值，并设置hashseed（即改变哈希策略）。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_bigint(100::bigint, 10);

hll\_hash\_bigint

---------------------

-2305749404374433531

(1 row)

* hll\_hash\_bytea(bytea)
* 描述：对bytea类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_bytea(E'\x');

hll\_hash\_bytea

----------------

0

(1 row)

* hll\_hash\_bytea(bytea, int32)
* 描述：对bytea类型数据计算哈希值，并设置hashseed（即改变哈希策略）。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_bytea(E'\x', 10);

hll\_hash\_bytea

---------------------

7233188113542599437

(1 row)

* hll\_hash\_text(text)
* 描述：对text类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_text('AB');

hll\_hash\_text

---------------------

-5666002586880275174

(1 row)

* hll\_hash\_text(text, int32)
* 描述：对text类型数据计算哈希值，并设置hashseed（即改变哈希策略）。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# SELECT hll\_hash\_text('AB', 10);

hll\_hash\_text

--------------------

-2215507121143724132

(1 row)

* hll\_hash\_any(anytype)
* 描述：对任意类型数据计算哈希值。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# select hll\_hash\_any(1);

hll\_hash\_any

----------------------

-1316670585935156930

(1 row)

vastbase=# select hll\_hash\_any('08:00:2b:01:02:03'::macaddr);

hll\_hash\_any

----------------------

-3719950434455589360

(1 row)

* hll\_hash\_any(anytype, int32)
* 描述：对任意类型数据计算哈希值，并设置hashseed（即改变哈希策略）。
* 返回值类型：hll\_hashval
* 示例：

vastbase=# select hll\_hash\_any(1, 10);

hll\_hash\_any

----------------------

7048553517657992351

(1 row)

* hll\_hashval\_eq(hll\_hashval, hll\_hashval)
* 描述：比较两个hll\_hashval类型数据是否相等。
* 返回值类型：bool
* 示例：

vastbase=# select hll\_hashval\_eq(hll\_hash\_integer(1), hll\_hash\_integer(1));

hll\_hashval\_eq

----------------

t

(1 row)

* hll\_hashval\_ne(hll\_hashval, hll\_hashval)
* 描述：比较两个hll\_hashval类型数据是否不相等。
* 返回值类型：bool
* 示例：

vastbase=# select hll\_hashval\_ne(hll\_hash\_integer(1), hll\_hash\_integer(1));

hll\_hashval\_ne

----------------

f

(1 row)

**日志函数**

hll主要存在三种模式Explicit、Sparse、Full。当数据规模比较小的时候会使用Explicit模式，这种模式下distinct值的计算是没有误差的；随着distinct值越来越多，hll会先后转换为Sparse模式和Full模式，这两种模式在计算结果上没有任何区别，只影响hll函数的计算效率和hll对象的存储空间。下面的函数可以用于查看hll的一些参数。

* hll\_print(hll)
* 描述：打印hll的一些debug参数信息。
* 示例：

vastbase=# select hll\_print(hll\_empty());

hll\_print

-------------------------------------------------------------------------------

type=1(HLL\_EMPTY), log2m=14, log2explicit=10, log2sparse=12, duplicatecheck=0

(1 row)

* hll\_type(hll)
* 描述：查看当前hll的类型。返回值具体含义如下：返回值0，表示HLL\_UNINIT，未初始化的hll对象；返回值1，表示HLL\_EMPTY，hll空对象；返回值2，表示HLL\_EXPLICIT，Explicit模式的hll对象；返回值3，表示HLL\_SPARSE，Sparse模式的hll对象；返回值4，表示HLL\_FULL，Full模式的hll对象；返回值5，表示HLL\_UNDEFINED，不合法的hll对象。
* 示例：

vastbase=# select hll\_type(hll\_empty());

hll\_type

----------

1

(1 row)

* hll\_log2m(hll)
* 描述：查看当前hll数据结构中的log2m数值，log2m是分桶数的对数值，此值会影响最后hll计算distinct误差率，误差率计算公式为±1.04/√(2 ^ log2m)。当显式指定log2m的取值为10-16之间时，hll会设置分桶数为2log2m。当显示指定log2explicit为-1时，会采用内置默认值。
* 示例：

vastbase=# select hll\_log2m(hll\_empty());

hll\_log2m

-----------

14

(1 row)

vastbase=# select hll\_log2m(hll\_empty(10));

hll\_log2m

-----------

10

(1 row)

vastbase=# select hll\_log2m(hll\_empty(-1));

hll\_log2m

-----------

14

(1 row)

* hll\_log2explicit(hll)
* 描述：查看当前hll数据结构中的log2explicit数值。hll通常会由Explicit模式到Sparse模式再到Full模式，这个过程称为promotion hierarchy策略。可以通过调整log2explicit值的大小改变策略，比如log2explicit为0的时候就会跳过Explicit模式而直接进入Sparse模式。当显式指定log2explicit的取值为1-12之间时，hll会在数据段长度超过2log2explicit时转为Sparse模式。当显示指定log2explicit为-1时，会采用内置默认值。
* 示例：

vastbase=# select hll\_log2explicit(hll\_empty());

hll\_log2explicit

------------------

10

(1 row)

vastbase=# select hll\_log2explicit(hll\_empty(12, 8));

hll\_log2explicit

------------------

8

(1 row)

vastbase=# select hll\_log2explicit(hll\_empty(12, -1));

hll\_log2explicit

------------------

10

(1 row)

* hll\_log2sparse(hll)
* 描述：查看当前hll数据结构中的log2sparse数值。hll通常会由Explicit模式到Sparse模式再到Full模式，这个过程称为promotion hierarchy策略。可以通过调整log2sparse值的大小改变策略，比如log2sparse为0的时候就会跳过Sparse模式而直接进入Full模式。当显式指定Sparse的取值为1-14之间时，hll会在数据段长度超过2log2sparse时转为Full模式。当显示指定log2sparse为-1时，会采用内置默认值。
* 示例：

vastbase=# select hll\_log2sparse(hll\_empty());

hll\_log2sparse

---------------

12

(1 row)

vastbase=# select hll\_log2sparse(hll\_empty(12, 8, 10));

hll\_log2sparse

----------------

10

(1 row)

vastbase=# select hll\_log2sparse(hll\_empty(12, 8, -1));

hll\_log2sparse

----------------

12

(1 row)

* hll\_duplicatecheck(hll)
* 描述：是否启用duplicatecheck，0是关闭，1是开启。默认关闭，对于有较多重复值出现的情况，可以开启以提高效率。当显示指定duplicatecheck为-1时，会采用内置默认值。
* 示例：

vastbase=# select hll\_duplicatecheck(hll\_empty());

hll\_duplicatecheck

--------------------

0

(1 row)

vastbase=# select hll\_duplicatecheck(hll\_empty(12, 8, 10, 1));

hll\_duplicatecheck

--------------------

1

(1 row)

vastbase=# select hll\_duplicatecheck(hll\_empty(12, 8, 10, -1));

hll\_duplicatecheck

--------------------

0

(1 row)

**功能函数**

* hll\_empty()
* 描述：创建一个空的hll。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_empty();

hll\_empty

------------------------------------------------------------

x484c4c00000000002b05000000000000000000000000000000000000

(1 row)

* hll\_empty(int32 log2m)
* 描述：创建空的hll并指定参数log2m，取值范围是10到16。若输入-1，则采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_empty(10);

hll\_empty

------------------------------------------------------------

x484c4c00000000002b04000000000000000000000000000000000000

(1 row)

vastbase=# select hll\_empty(-1);

hll\_empty

------------------------------------------------------------

x484c4c00000000002b05000000000000000000000000000000000000

(1 row)

* hll\_empty(int32 log2m, int32 log2explicit)
* 描述：创建空的hll并依次指定参数log2m、log2explicit。log2explicit取值范围是0到12，0表示直接跳过Explicit模式。该参数可以用来设置Explicit模式的阈值大小，在数据段长度达到2log2explicit后切换为Sparse模式或者Full模式。若输入-1，则log2explicit采用内置默认值。
* 返回值类型: hll
* 示例：

vastbase=# select hll\_empty(10, 4);

hll\_empty

------------------------------------------------------------

x484c4c00000000001304000000000000000000000000000000000000

(1 row)

vastbase=# select hll\_empty(10, -1);

hll\_empty

------------------------------------------------------------

x484c4c00000000002b04000000000000000000000000000000000000

(1 row)

* hll\_empty(int32 log2m, int32 log2explicit, int64 log2sparse)
* 描述：创建空的hll并依次指定参数log2m、log2explicit、log2sparse。log2sparse取值范围是0到14，0表示直接跳过Sparse模式。该参数可以用来设置Sparse模式的阈值大小，在数据段长度达到2log2sparse后切换为Full模式。若输入-1，则log2sparse采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_empty(10, 4, 8);

hll\_empty

------------------------------------------------------------

x484c4c00000000001204000000000000000000000000000000000000

(1 row)

vastbase=# select hll\_empty(10, 4, -1);

hll\_empty

------------------------------------------------------------

x484c4c00000000001304000000000000000000000000000000000000

(1 row)

* hll\_empty(int32 log2m, int32 log2explicit, int64 log2sparse, int32 duplicatecheck)
* 描述：创建空的hll并依次指定参数log2m、log2explicit、log2sparse、duplicatecheck。duplicatecheck取0或者1，表示是否开启该模式，默认情况下该模式会关闭。若输入-1，则duplicatecheck采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_empty(10, 4, 8, 0);

hll\_empty

------------------------------------------------------------

x484c4c00000000001204000000000000000000000000000000000000

(1 row)

vastbase=# select hll\_empty(10, 4, 8, -1);

hll\_empty

------------------------------------------------------------

x484c4c00000000001204000000000000000000000000000000000000

(1 row)

* hll\_add(hll, hll\_hashval)
* 描述：把hll\_hashval加入到hll中。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_add(hll\_empty(), hll\_hash\_integer(1));

hll\_add

----------------------------------------------------------------------------

x484c4c08000002002b0900000000000000f03f3e2921ff133fbaed3e2921ff133fbaed00

(1 row)

* hll\_add\_rev(hll\_hashval, hll)
* 描述：把hll\_hashval加入到hll中，和hll\_add功能一样，只是参数位置进行了交换。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_add\_rev(hll\_hash\_integer(1), hll\_empty());

hll\_add\_rev

----------------------------------------------------------------------------

x484c4c08000002002b0900000000000000f03f3e2921ff133fbaed3e2921ff133fbaed00

(1 row)

* hll\_eq(hll, hll)
* 描述：比较两个hll是否相等。
* 返回值类型：bool
* 示例：

vastbase=# select hll\_eq(hll\_add(hll\_empty(), hll\_hash\_integer(1)), hll\_add(hll\_empty(), hll\_hash\_integer(2)));

hll\_eq

--------

f

(1 row)

* hll\_ne(hll, hll)
* 描述：比较两个hll是否不相等。
* 返回值类型：bool
* 示例：

vastbase=# select hll\_ne(hll\_add(hll\_empty(), hll\_hash\_integer(1)), hll\_add(hll\_empty(), hll\_hash\_integer(2)));

hll\_ne

--------

t

(1 row)

* hll\_cardinality(hll)
* 描述：计算hll的distinct值。
* 返回值类型：int
* 示例：

vastbase=# select hll\_cardinality(hll\_empty() || hll\_hash\_integer(1));

hll\_cardinality

-----------------

1

(1 row)

* hll\_union(hll, hll)
* 描述：把两个hll数据结构union成一个。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_union(hll\_add(hll\_empty(), hll\_hash\_integer(1)), hll\_add(hll\_empty(), hll\_hash\_integer(2)));

hll\_union

--------------------------------------------------------------------------------------------

x484c4c10002000002b090000000000000000400000000000000000b3ccc49320cca1ae3e2921ff133fbaed00

(1 row)

**聚合函数**

* hll\_add\_agg(hll\_hashval)
* 描述：把哈希后的数据按照分组放到hll中。
* 返回值类型：hll
* 示例：

--准备数据

vastbase=# create table t\_id(id int);

vastbase=# insert into t\_id values(generate\_series(1,500));

vastbase=# create table t\_data(a int, c text);

vastbase=# insert into t\_data select mod(id,2), id from t\_id;

--创建表并指定列为hll

vastbase=# create table t\_a\_c\_hll(a int, c hll);

--根据a列group by对数据分组，把各组数据加到hll中

vastbase=# insert into t\_a\_c\_hll select a, hll\_add\_agg(hll\_hash\_text(c)) from t\_data group by a;

--得到每组数据中hll的Distinct值

vastbase=# select a, #c as cardinality from t\_a\_c\_hll order by a;

a | cardinality

---+------------------

0 | 247.862354346299

1 | 250.908710610377

(2 rows)

* hll\_add\_agg(hll\_hashval, int32 log2m)
* 描述：把哈希后的数据按照分组放到hll中，并指定参数log2m，取值范围是10到16。若输入-1或者NULL，则采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_cardinality(hll\_add\_agg(hll\_hash\_text(c), 12)) from t\_data;

hll\_cardinality

------------------

497.965240179228

(1 row)

* hll\_add\_agg(hll\_hashval, int32 log2m, int32 log2explicit)
* 描述：把哈希后的数据按照分组放到hll中，依次指定参数log2m、log2explicit。log2explicit取值范围是0到12，0表示直接跳过Explicit模式。该参数可以用来设置Explicit模式的阈值大小，在数据段长度达到2log2explicit后切换为Sparse模式或者Full模式。若输入-1或者NULL，则log2explicit采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_cardinality(hll\_add\_agg(hll\_hash\_text(c), NULL, 1)) from t\_data;

hll\_cardinality

------------------

498.496062953313

(1 row)

* hll\_add\_agg(hll\_hashval, int32 log2m, int32 log2explicit, int64 log2sparse)
* 描述：把哈希后的数据按照分组放到hll中，依次指定参数log2m、log2explicit、log2sparse。log2sparse取值范围是0到14，0表示直接跳过Sparse模式。该参数可以用来设置Sparse模式的阈值大小，在数据段长度达到2log2sparse后切换为Full模式。若输入-1或者NULL，则log2sparse采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_cardinality(hll\_add\_agg(hll\_hash\_text(c), NULL, 6, 10)) from t\_data;

hll\_cardinality

------------------

498.496062953313

(1 row)

* hll\_add\_agg(hll\_hashval, int32 log2m, int32 log2explicit, int64 log2sparse, int32 duplicatecheck)
* 描述：把哈希后的数据按照分组放到hll中, 依次制定参数log2m、log2explicit、log2sparse、duplicatecheck，duplicatecheck取值范围是0或者1，表示是否开启该模式，默认情况下该模式会关闭。若输入-1或者NULL，则duplicatecheck采用内置默认值。
* 返回值类型：hll
* 示例：

vastbase=# select hll\_cardinality(hll\_add\_agg(hll\_hash\_text(c), NULL, 6, 10, -1)) from t\_data;

hll\_cardinality

------------------

498.496062953313

(1 row)

* hll\_union\_agg(hll)
* 描述：将多个hll类型数据union成一个hll。
* 返回值类型：hll
* 示例：

--将各组中的hll数据union成一个hll，并计算distinct值。

vastbase=# select #hll\_union\_agg(c) as cardinality from t\_a\_c\_hll;

cardinality

------------------

498.496062953313

(1 row)

fig: **说明：**

注意：当两个或者多个hll数据结构做union的时候，必须要保证其中每一个hll里面的精度参数一样，否则将不可以进行union。同样的约束也适用于函数hll\_union(hll,hll)。

**废弃函数**

由于版本升级，HLL（HyperLogLog）有一些旧的函数废弃，用户可以用类似的函数进行替代。

* hll\_schema\_version(hll)
* 描述：查看当前hll中的schema version。旧版本schema version是常值1，用来进行hll字段的头部校验，重构后的hll在头部增加字段“HLL”进行校验，schema version不再使用。
* hll\_regwidth(hll)
* 描述：查看hll数据结构中桶的位数大小。旧版本桶的位数regwidth取值1~5，会存在较大的误差，也限制了基数估计上限。 重构后regwidth为固定值6，不再使用regwidth变量。
* hll\_expthresh(hll)
* 描述：得到当前hll中expthresh大小。采用hll\_log2explicit(hll)替代类似功能。
* hll\_sparseon(hll)
* 描述：是否启用Sparse模式。采用hll\_log2sparse(hll)替代类似功能，0表示关闭Sparse模式。

**内置函数**

HLL（HyperLogLog）有一系列内置函数用于内部对数据进行处理，一般情况下用户不需要熟知这些函数的使用。详情见[表1](#table763375795510)。

**表 1** 内置函数

| **函数名称** | **功能描述** |
| --- | --- |
| hll\_in | 以string格式接收hll数据。 |
| hll\_out | 以string格式发送hll数据。 |
| hll\_recv | 以bytea格式接收hll数据。 |
| hll\_send | 以bytea格式发送hll数据。 |
| hll\_trans\_in | 以string格式接收hll\_trans\_type数据。 |
| hll\_trans\_out | 以string格式发送hll\_trans\_type数据。 |
| hll\_trans\_recv | 以bytea形式接收hll\_trans\_type数据。 |
| hll\_trans\_send | 以bytea形式发送hll\_trans\_type数据。 |
| hll\_typmod\_in | 接收typmod类型数据。 |
| hll\_typmod\_out | 发送typmod类型数据。 |
| hll\_hashval\_in | 接收hll\_hashval类型数据。 |
| hll\_hashval\_out | 发送hll\_hashval类型数据。 |
| hll\_add\_trans0 | 类似于hll\_add所提供的功能，初始化时无指定入参，通常在聚合运算的第一阶段DN上使用。 |
| hll\_add\_trans1 | 类似于hll\_add所提供的功能，初始化时指定一个入参，通常在聚合运算的第一阶段DN上使用。 |
| hll\_add\_trans2 | 类似于hll\_add所提供的功能，初始化时指定两个入参，通常在聚合运算的第一阶段DN上使用。 |
| hll\_add\_trans3 | 类似于hll\_add所提供的功能，初始化时指定三个入参，通常在聚合运算的第一阶段DN上使用。 |
| hll\_add\_trans4 | 类似于hll\_add所提供的功能，初始化时指定四个入参，通常在聚合运算的第一阶段DN上使用。 |
| hll\_union\_trans | 类似hll\_union所提供的功能，在聚合运算的第一阶段DN上使用。 |
| hll\_union\_collect | 类似于hll\_union所提供的功能，在聚合运算第二阶段DN上使用，汇总各个DN上的结果。 |
| hll\_pack | 在聚合运算第三阶段DN上使用，把自定义hll\_trans\_type类型最后转换成hll类型。 |
| hll | 用于hll类型转换成hll类型，根据输入参数会设定指定参数。 |
| hll\_hashval | 用于bigint类型转换成hll\_hashval类型。 |
| hll\_hashval\_int4 | 用于int4类型转换成hll\_hashval类型。 |

**操作符**

* =
* 描述：比较hll或hll\_hashval的值是否相等。
* 返回值类型：bool
* 示例：

--hll

vastbase=# select (hll\_empty() || hll\_hash\_integer(1)) = (hll\_empty() || hll\_hash\_integer(1));

column

----------

t

(1 row)

--hll\_hashval

vastbase=# select hll\_hash\_integer(1) = hll\_hash\_integer(1);

?column?

----------

t

(1 row)

* <> or !=
* 描述：比较hll或hll\_hashval是否不相等。
* 返回值类型：bool
* 示例：

--hll

vastbase=# select (hll\_empty() || hll\_hash\_integer(1)) <> (hll\_empty() || hll\_hash\_integer(2));

?column?

----------

t

(1 row)

--hll\_hashval

vastbase=# select hll\_hash\_integer(1) <> hll\_hash\_integer(2);

?column?

----------

t

(1 row)

* ||
* 描述：可代表hll\_add、hll\_union、hll\_add\_rev三个函数的功能。
* 返回值类型：hll
* 示例：

--hll\_add

vastbase=# select hll\_empty() || hll\_hash\_integer(1);

?column?

----------------------------------------------------------------------------

x484c4c08000002002b0900000000000000f03f3e2921ff133fbaed3e2921ff133fbaed00

(1 row)

--hll\_add\_rev

vastbase=# select hll\_hash\_integer(1) || hll\_empty();

?column?

----------------------------------------------------------------------------

x484c4c08000002002b0900000000000000f03f3e2921ff133fbaed3e2921ff133fbaed00

(1 row)

--hll\_union

vastbase=# select (hll\_empty() || hll\_hash\_integer(1)) || (hll\_empty() || hll\_hash\_integer(2));

?column?

--------------------------------------------------------------------------------------------

x484c4c10002000002b090000000000000000400000000000000000b3ccc49320cca1ae3e2921ff133fbaed00

(1 row)

* #
* 描述：计算出hll的Dintinct值, 同hll\_cardinality函数。
* 返回值类型：int
* 示例：

vastbase=# select #(hll\_empty() || hll\_hash\_integer(1));

?column?

----------

1

(1 row)

#### SEQUENCE函数

序列函数为用户从序列对象中获取后续的序列值提供了简单的多用户安全的方法。

* nextval(regclass)
* 描述：递增序列并返回新值。
* fig: **说明：**
* 为了避免从同一个序列获取值的并发事务被阻塞，nextval操作不会回滚；也就是说，一旦一个值已经被抓取，那么就认为它已经被用过了，并且不会再被返回。即使该操作处于事务中，当事务之后中断，或者如果调用查询结束不使用该值，也是如此。这种情况将在指定值的顺序中留下未使用的“空洞”。因此，Vastbase序列对象不能用于获得“无间隙”序列。
* fig: **须知：**
* nextval函数只能在主机上执行，备机不支持执行此函数。
* 返回类型：numeric
* nextval函数有两种调用方式（其中第二种调用方式目前不支持Sequence命名中有特殊字符"."的情况），如下：
* 示例1：

vastbase=# select nextval('seqDemo');

nextval

---------

2

(1 row)

示例2：

vastbase=# select seqDemo.nextval;

nextval

---------

2

(1 row)

* currval(regclass)
* 返回当前会话里最近一次nextval返回的指定的sequence的数值。如果当前会话还没有调用过指定的sequence的nextval，那么调用currval将会报错。
* 返回类型：numeric
* currval函数有两种调用方式（其中第二种调用方式目前不支持Sequence命名中有特殊字符"."的情况），如下：
* 示例1：

vastbase=# select currval('seq1');

currval

---------

2

(1 row)

示例2：

vastbase=# select seq1.currval;

currval

---------

2

(1 row)

* lastval()
* 描述：返回当前会话里最近一次nextval返回的数值。这个函数等效于currval，只是它不用序列名为参数，它抓取当前会话里面最近一次nextval使用的序列。如果当前会话还没有调用过nextval，那么调用lastval将会报错。
* 返回类型：numeric
* 示例：

vastbase=# select lastval();

lastval

---------

2

(1 row)

* setval(regclass,numeric)
* 描述：设置序列的当前数值。
* 返回类型：numeric
* 示例：

vastbase=# select setval('seqDemo',1);

setval

--------

1

(1 row)

* setval(regclass, numeric, Boolean)
* 描述：设置序列的当前数值以及is\_called标志。
* 返回类型：numeric
* 示例：

vastbase=# select setval('seqDemo',1,true);

setval

--------

1

(1 row)

fig: **说明：**

Setval后当前会话会立刻生效，但如果其他会话有缓存的序列值，只能等到缓存值用尽才能感知Setval的作用。所以为了避免序列值冲突，setval要谨慎使用。

因为序列是非事务的，setval造成的改变不会由于事务的回滚而撤销。

fig: **须知：**

nextval函数只能在主机上执行，备机不支持执行此函数。

* pg\_sequence\_last\_value(sequence\_oid oid, OUT cache\_value int16, OUT last\_value int16)
* 描述：获取指定sequence的参数，包含缓存值，当前值。
* 返回类型：int16，int16

#### 范围函数和操作符

**范围操作符**

* =
* 描述：等于
* 示例：

vastbase=# SELECT int4range(1,5) = '[1,4]'::int4range AS RESULT;

result

--------

t

(1 row)

* <>
* 描述：不等于
* 示例：

vastbase=# SELECT numrange(1.1,2.2) <> numrange(1.1,2.3) AS RESULT;

result

--------

t

(1 row)

* <
* 描述：小于
* 示例：

vastbase=# SELECT int4range(1,10) < int4range(2,3) AS RESULT;

result

--------

t

(1 row)

* >
* 描述：大于
* 示例：

vastbase=# SELECT int4range(1,10) > int4range(1,5) AS RESULT;

result

--------

t

(1 row)

* <=
* 描述：小于或等于
* 示例：

vastbase=# SELECT numrange(1.1,2.2) <= numrange(1.1,2.2) AS RESULT;

result

--------

t

(1 row)

* >=
* 描述：大于或等于
* 示例：

vastbase=# SELECT numrange(1.1,2.2) >= numrange(1.1,2.0) AS RESULT;

result

--------

t

(1 row)

* @>
* 描述：包含范围
* 示例：

vastbase=# SELECT int4range(2,4) @> int4range(2,3) AS RESULT;

result

--------

t

(1 row)

* @>
* 描述：包含元素
* 示例：

vastbase=# SELECT '[2011-01-01,2011-03-01)'::tsrange @> '2011-01-10'::timestamp AS RESULT;

result

--------

t

(1 row)

* <@
* 描述：范围包含于
* 示例：

vastbase=# SELECT int4range(2,4) <@ int4range(1,7) AS RESULT;

result

--------

t

(1 row)

* <@
* 描述：元素包含于
* 示例：

vastbase=# SELECT 42 <@ int4range(1,7) AS RESULT;

result

--------

f

(1 row)

* &&
* 描述：重叠（有共同点）
* 示例：

vastbase=# SELECT int8range(3,7) && int8range(4,12) AS RESULT;

result

--------

t

(1 row)

* <<
* 描述：范围值是否比另一个范围值的最小值还小（没有交集）。
* 示例：

vastbase=# SELECT int8range(1,10) << int8range(100,110) AS RESULT;

result

--------

t

(1 row)

* >>
* 描述：范围值是否比另一个范围值的最大值还大（没有交集）。
* 示例：

vastbase=# SELECT int8range(50,60) >> int8range(20,30) AS RESULT;

result

--------

t

(1 row)

* &<
* 描述：范围值的最大值是否不超过另一个范围值的最大值。
* 示例：

vastbase=# SELECT int8range(1,20) &< int8range(18,20) AS RESULT;

result

--------

t

(1 row)

* &>
* 描述：范围值的最小值是否不小于另一个范围值的最小值。
* 示例：

vastbase=# SELECT int8range(7,20) &> int8range(5,10) AS RESULT;

result

--------

t

(1 row)

* -|-
* 描述：相邻
* 示例：

vastbase=# SELECT numrange(1.1,2.2) -|- numrange(2.2,3.3) AS RESULT;

result

--------

t

(1 row)

* +
* 描述：并集
* 示例：

vastbase=# SELECT numrange(5,15) + numrange(10,20) AS RESULT;

result

--------

[5,20)

(1 row)

* \*
* 描述：交集
* 示例：

vastbase=# SELECT int8range(5,15) \* int8range(10,20) AS RESULT;

result

---------

[10,15)

(1 row)

* -
* 描述：差集
* 示例：

vastbase=# SELECT int8range(5,15) - int8range(10,20) AS RESULT;

result

--------

[5,10)

(1 row)

简单的比较操作符<、>、<=和>=先比较下界，只有下界相等时才比较上界。

<<、>>和-|-操作符当包含空范围时也会返回false；也就是，不认为空范围在其他范围之前或之后。

并集和差集操作符的执行结果无法包含两个不相交的子范围。

**范围函数**

* numrange(numeric, numeric, [text])
* 描述：表示一个范围。
* 返回类型：范围元素类型
* 示例：

vastbase=# SELECT numrange(1.1,2.2) AS RESULT;

result

--------

[1.1,2.2)

(1 row)

vastbase=# SELECT numrange(1.1,2.2, '()') AS RESULT;

result

--------

(1.1,2.2)

(1 row)

* lower(anyrange)
* 描述：范围的下界。
* 返回类型：范围元素类型
* 示例：

vastbase=# SELECT lower(numrange(1.1,2.2)) AS RESULT;

result

--------

1.1

(1 row)

* upper(anyrange)
* 描述：范围的上界。
* 返回类型：范围元素类型
* 示例：

vastbase=# SELECT upper(numrange(1.1,2.2)) AS RESULT;

result

--------

2.2

(1 row)

* isempty(anyrange)
* 描述：范围是否为空。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT isempty(numrange(1.1,2.2)) AS RESULT;

result

--------

f

(1 row)

* lower\_inc(anyrange)
* 描述：是否包含下界。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT lower\_inc(numrange(1.1,2.2)) AS RESULT;

result

--------

t

(1 row)

* upper\_inc(anyrange)
* 描述：是否包含上界。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT upper\_inc(numrange(1.1,2.2)) AS RESULT;

result

--------

f

(1 row)

* lower\_inf(anyrange)
* 描述：下界是否为无穷。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT lower\_inf('(,)'::daterange) AS RESULT;

result

--------

t

(1 row)

* upper\_inf(anyrange)
* 描述：上界是否为无穷。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT upper\_inf('(,)'::daterange) AS RESULT;

result

--------

t

(1 row)

如果范围是空或者需要的界限是无穷的，lower和upper函数将返回null。lower\_inc、upper\_inc、lower\_inf和upper\_inf函数均对空范围返回false。

* elem\_contained\_by\_range(anyelement, anyrange)
* 描述：判断元素是否在范围内。
* 返回类型：Boolean
* 示例：

vastbase=# SELECT elem\_contained\_by\_range('2', numrange(1.1,2.2));

elem\_contained\_by\_range

-------------------------

t

(1 row)

#### 窗口函数

**窗口函数**

列存表目前只支持rank(expression)和row\_number(expression)两个函数。

窗口函数与OVER语句一起使用。OVER语句用于对数据进行分组，并对组内元素进行排序。窗口函数用于给组内的值生成序号。

fig: **说明：**

窗口函数中的order by后面必须跟字段名，若order by后面跟数字，该数字会被按照常量处理，因此对目标列没有起到排序的作用。

* RANK()
* 描述：RANK函数为各组内值生成跳跃排序序号，其中，相同的值具有相同序号。
* 返回值类型：BIGINT
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, rank() OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | rank

-------+---------------+------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 8

1 | 2 | 8

1 | 2 | 8

1 | 2 | 8

1 | 2 | 8

1 | 2 | 8

1 | 2 | 8

1 | 3 | 15

1 | 3 | 15

1 | 3 | 15

1 | 3 | 15

1 | 3 | 15

1 | 3 | 15

1 | 3 | 15

1 | 4 | 22

1 | 4 | 22

1 | 4 | 22

1 | 4 | 22

1 | 4 | 22

1 | 4 | 22

1 | 4 | 22

1 | 5 | 29

1 | 5 | 29

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

(42 rows)

* ROW\_NUMBER()
* 描述：ROW\_NUMBER函数为各组内值生成连续排序序号，其中，相同的值其序号也不相同。
* 返回值类型：BIGINT
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, Row\_number() OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | row\_number

-------+---------------+------------

1 | 1 | 1

1 | 1 | 2

1 | 1 | 3

1 | 1 | 4

1 | 1 | 5

1 | 1 | 6

1 | 1 | 7

1 | 2 | 8

1 | 2 | 9

1 | 2 | 10

1 | 2 | 11

1 | 2 | 12

1 | 2 | 13

1 | 2 | 14

1 | 3 | 15

1 | 3 | 16

1 | 3 | 17

1 | 3 | 18

1 | 3 | 19

1 | 3 | 20

1 | 3 | 21

1 | 4 | 22

1 | 4 | 23

1 | 4 | 24

1 | 4 | 25

1 | 4 | 26

1 | 4 | 27

1 | 4 | 28

1 | 5 | 29

1 | 5 | 30

2 | 5 | 1

2 | 5 | 2

2 | 5 | 3

2 | 5 | 4

2 | 5 | 5

2 | 6 | 6

2 | 6 | 7

2 | 6 | 8

2 | 6 | 9

2 | 6 | 10

2 | 6 | 11

2 | 6 | 12

(42 rows)

* DENSE\_RANK()
* 描述：DENSE\_RANK函数为各组内值生成连续排序序号，其中，相同的值具有相同序号。
* 返回值类型：BIGINT
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, dense\_rank() OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | dense\_rank

-------+---------------+------------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 5 | 5

1 | 5 | 5

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

(42 rows)

* PERCENT\_RANK()
* 描述：PERCENT\_RANK函数为各组内对应值生成相对序号，即根据公式 (rank - 1) / (total rows - 1)计算所得的值。其中rank为该值依据RANK函数所生成的对应序号，totalrows为该分组内的总元素个数。
* 返回值类型：DOUBLE PRECISION
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, percent\_rank() OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | percent\_rank

-------+---------------+------------------

1 | 1 | 0

1 | 1 | 0

1 | 1 | 0

1 | 1 | 0

1 | 1 | 0

1 | 1 | 0

1 | 1 | 0

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 2 | .241379310344828

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 3 | .482758620689655

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 4 | .724137931034483

1 | 5 | .96551724137931

1 | 5 | .96551724137931

2 | 5 | 0

2 | 5 | 0

2 | 5 | 0

2 | 5 | 0

2 | 5 | 0

2 | 6 | .454545454545455

2 | 6 | .454545454545455

2 | 6 | .454545454545455

2 | 6 | .454545454545455

2 | 6 | .454545454545455

2 | 6 | .454545454545455

2 | 6 | .454545454545455

(42 rows)

* CUME\_DIST()
* 描述：CUME\_DIST函数为各组内对应值生成累积分布序号。即根据公式(小于等于当前值的数据行数)/(该分组总行数totalrows)计算所得的相对序号。
* 返回值类型：DOUBLE PRECISION
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, cume\_dist() OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim e\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | cume\_dist

-------+---------------+------------------

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 1 | .233333333333333

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 2 | .466666666666667

1 | 3 | .7

1 | 3 | .7

1 | 3 | .7

1 | 3 | .7

1 | 3 | .7

1 | 3 | .7

1 | 3 | .7

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 4 | .933333333333333

1 | 5 | 1

1 | 5 | 1

2 | 5 | .416666666666667

2 | 5 | .416666666666667

2 | 5 | .416666666666667

2 | 5 | .416666666666667

2 | 5 | .416666666666667

2 | 6 | 1

2 | 6 | 1

2 | 6 | 1

2 | 6 | 1

2 | 6 | 1

2 | 6 | 1

2 | 6 | 1

(42 rows)

* NTILE(num\_buckets integer)
* 描述：NTILE函数根据num\_buckets integer将有序的数据集合平均分配到num\_buckets所指定数量的桶中，并将桶号分配给每一行。分配时应尽量做到平均分配。
* 返回值类型：INTEGER
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, ntile(3) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | ntile

-------+---------------+-------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 3 | 2

1 | 3 | 2

1 | 3 | 2

1 | 3 | 2

1 | 3 | 2

1 | 3 | 2

1 | 3 | 3

1 | 4 | 3

1 | 4 | 3

1 | 4 | 3

1 | 4 | 3

1 | 4 | 3

1 | 4 | 3

1 | 4 | 3

1 | 5 | 3

1 | 5 | 3

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 1

2 | 5 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 3

2 | 6 | 3

2 | 6 | 3

2 | 6 | 3

(42 rows)

* LAG(value any [, offset integer [, default any ]])
* 描述：LAG函数为各组内对应值生成滞后值。即当前值对应的行数往前偏移offset位后所得行的value值作为序号。若经过偏移后行数不存在，则对应结果取为default值。若无指定，在默认情况下，offset取为1，default值取为NULL。
* 返回值类型：与参数数据类型相同。
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, lag(d\_moy,3,null) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | lag

-------+---------------+-----

1 | 1 |

1 | 1 |

1 | 1 |

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 5 | 1

1 | 5 | 1

2 | 5 |

2 | 5 |

2 | 5 |

2 | 5 | 2

2 | 5 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

2 | 6 | 2

(42 rows)

* LEAD(value any [, offset integer [, default any ]])
* 描述：LEAD函数为各组内对应值生成提前值。即当前值对应的行数向后偏移offset位后所得行的value值作为序号。若经过向后偏移后行数超过当前组内的总行数，则对应结果取为default值。若无指定，在默认情况下，offset取为1，default值取为NULL。
* 返回值类型：与参数数据类型相同。
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, lead(d\_fy\_week\_seq,2) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | lead

-------+---------------+------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 2

1 | 1 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 2

1 | 2 | 3

1 | 2 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 3

1 | 3 | 4

1 | 3 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 4

1 | 4 | 5

1 | 4 | 5

1 | 5 |

1 | 5 |

2 | 5 | 5

2 | 5 | 5

2 | 5 | 5

2 | 5 | 6

2 | 5 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 | 6

2 | 6 |

2 | 6 |

(42 rows)

* FIRST\_VALUE(value any)
* 描述：FIRST\_VALUE函数取各组内的第一个值作为返回结果。
* 返回值类型：与参数数据类型相同。
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, first\_value(d\_fy\_week\_seq) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 7 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | first\_value

-------+---------------+-------------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 5 | 1

1 | 5 | 1

2 | 5 | 5

2 | 5 | 5

2 | 5 | 5

2 | 5 | 5

2 | 5 | 5

2 | 6 | 5

2 | 6 | 5

2 | 6 | 5

2 | 6 | 5

2 | 6 | 5

2 | 6 | 5

2 | 6 | 5

(42 rows)

* LAST\_VALUE(value any)
* 描述：LAST\_VALUE函数取各组内的最后一个值作为返回结果。
* 返回值类型：与参数数据类型相同。
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, last\_value(d\_moy) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 6 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | last\_value

-------+---------------+------------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 5 | 1

1 | 5 | 1

2 | 5 | 2

2 | 5 | 2

2 | 5 | 2

2 | 5 | 2

2 | 5 | 2

(35 rows)

* NTH\_VALUE(value any, nth integer)
* 描述：NTH\_VALUE函数返回该组内的第nth行作为结果。若该行不存在，则默认返回NULL。
* 返回值类型：与参数数据类型相同。
* 示例：

vastbase=# SELECT d\_moy, d\_fy\_week\_seq, nth\_value(d\_fy\_week\_seq,6) OVER(PARTITION BY d\_moy ORDER BY d\_fy\_week\_seq) FROM tpcds.date\_dim WHERE d\_moy < 4 AND d\_fy\_week\_seq < 6 ORDER BY 1,2;

d\_moy | d\_fy\_week\_seq | nth\_value

-------+---------------+-----------

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 1 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 2 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 3 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 4 | 1

1 | 5 | 1

1 | 5 | 1

2 | 5 |

2 | 5 |

2 | 5 |

2 | 5 |

2 | 5 |

(35 rows)

#### 账本数据库的函数

* get\_dn\_hist\_relhash(text, text)
* 描述：返回指定防篡改用户表的表级数据hash值。该函数仅供分布式使用。
* 参数类型：text
* 返回值类型：hash16
* ledger\_hist\_check(text, text)
* 描述：校验指定防篡改用户表的表级数据hash值与其对应历史表hash一致性。
* 参数类型：text
* 返回值类型：Boolean
* ledger\_hist\_repair(text, text)
* 描述：修复指定防篡改用户表对应的历史表hash值，使之与用户表hash一致，返回hash差值。
* 参数类型：text
* 返回值类型：hash16
* ledger\_hist\_archive(text, text)
* 描述：归档指定防篡改用户表对应的历史表至审计日志目录中hist\_back文件夹下。
* 参数类型：text
* 返回值类型：Boolean
* ledger\_gchain\_check(text, text)
* 描述：校验指定防篡改用户表对应的历史表hash与全局历史表对应的relhash一致性。
* 参数类型：text
* 返回值类型：Boolean
* ledger\_gchain\_repair(text, text)
* 描述：修复验指定防篡改用户表在全局历史表中的relhash，使之与其历史表hash一致，返回hash差值。
* 参数类型：text
* 返回值类型：hash16
* ledger\_gchain\_archive(void)
* 描述：归档全局历史表至审计日志目录中hist\_back文件夹下。
* 参数类型：void
* 返回值类型：Boolean
* hash16in(cstring)
* 描述：将输入16进制字符串转化成内部hash16形式。
* 参数类型：cstring
* 返回值类型：hash16
* hash16out(hash16)
* 描述：将内部hash16类型的数据转码转化为16进制cstring类型。
* 参数类型：hash16
* 返回值类型：cstring
* hash32in(cstring)
* 描述：将输入16进制字符串（32个字符）转化成内部类型hash32形式。
* 参数类型：cstring
* 返回值类型：hash32
* hash32out(hash32)
* 描述：将内部hash32类型的数据转码转化为16进制cstring类型。
* 参数类型：cstring
* 返回值类型：hash32

#### 密态等值的函数

* byteawithoutorderwithequalcolin(cstring)
* 描述：将输入转码转化成内部byteawithoutorderwithequalcol形式。
* 参数类型：cstring
* 返回值类型：byteawithoutorderwithequalcol
* byteawithoutorderwithequalcolout(byteawithoutorderwithequalcol)
* 描述：将内部byteawithoutorderwithequalcol类型的数据转码转化为cstring类型。
* 参数类型：byteawithoutorderwithequalcol
* 返回值类型：cstring
* byteawithoutorderwithequalcolsend(byteawithoutorderwithequalcol)
* 描述：将byteawithoutorderwithequalcol类型的数据转码转化为bytea类型。
* 参数类型：byteawithoutorderwithequalcol
* 返回值类型：bytea
* byteawithoutorderwithequalcolrecv(internal)
* 描述：将byteawithoutorderwithequalcol类型的数据转码转化为byteawithoutorderwithequalcol类型。
* 参数类型：internal
* 返回值类型：byteawithoutorderwithequalcol
* byteawithoutorderwithequalcoltypmodin(\_cstring)
* 描述：将byteawithoutorderwithequalcol类型的数据转码转化为byteawithoutorderwithequalcol类型。
* 参数类型：\_cstring
* 返回值类型：int4
* byteawithoutorderwithequalcoltypmodout(int4)
* 描述：将int4类型的数据转码转化为cstring类型。
* 参数类型：int4
* 返回值类型：cstring
* byteawithoutordercolin(cstring)
* 描述：将输入转码转化成内部byteawithoutordercolin形式。
* 参数类型：cstring
* 返回值类型：byteawithoutordercol
* byteawithoutordercolout(byteawithoutordercol)
* 描述：将内部byteawithoutordercol类型的数据转码转化为cstring类型。
* 参数类型：byteawithoutordercol
* 返回值类型：cstring
* byteawithoutordercolsend(byteawithoutordercol)
* 描述：将byteawithoutordercol类型的数据转码转化为bytea类型。
* 参数类型：byteawithoutordercol
* 返回值类型：bytea
* byteawithoutordercolrecv(internal)
* 描述：将byteawithoutordercol类型的数据转码转化为byteawithoutordercol类型。
* 参数类型：internal
* 返回值类型：byteawithoutordercol
* byteawithoutorderwithequalcolcmp(byteawithoutorderwithequalcol, byteawithoutorderwithequalcol)
* 描述：比较两个byteawithoutorderwithequalcol类型的数据大小，若第一个参数小于第二个参数，返回-1，若等于，返回0，若大于，则返回1。
* 参数类型：byteawithoutorderwithequalcol, byteawithoutorderwithequalcol
* 返回值类型：int4
* byteawithoutorderwithequalcolcmpbytear(byteawithoutorderwithequalcol, bytea)
* 描述：比较byteawithoutorderwithequalcol和bytea数据大小，若第一个参数小于第二个参数，返回-1，若等于，返回0，若大于，则返回1。
* 参数类型：byteawithoutorderwithequalcol, bytea
* 返回值类型：int4
* byteawithoutorderwithequalcolcmpbyteal(bytea, byteawithoutorderwithequalcol)
* 描述：比较bytea和byteawithoutorderwithequalcol数据大小，若第一个参数小于第二个参数，返回-1，若等于，返回0，若大于，则返回1。
* 参数类型：byteawithoutorderwithequalcol, bytea
* 返回值类型：int4
* byteawithoutorderwithequalcoleq(byteawithoutorderwithequalcol, byteawithoutorderwithequalcol)
* 描述：比较两个byteawithoutorderwithequalcol类型的数据是否相同，相同则返回true，否则返回false。
* 参数类型：byteawithoutorderwithequalcol, bytea
* 返回值类型：bool
* byteawithoutorderwithequalcoleqbyteal(bytea, byteawithoutorderwithequalcol)
* 描述：比较bytea和byteawithoutorderwithequalcol数据是否相同，相同则返回true，否则返回false。
* 参数类型：bytea，byteawithoutorderwithequalcol
* 返回值类型：bool
* byteawithoutorderwithequalcoleqbytear(byteawithoutorderwithequalcol, bytea)
* 描述：比较byteawithoutorderwithequalcol和bytea数据是否相同，相同则返回true，否则返回false。
* 参数类型：byteawithoutorderwithequalcol, bytea
* 返回值类型：bool
* byteawithoutorderwithequalcolne(byteawithoutorderwithequalcol, byteawithoutorderwithequalcol)
* 描述：比较两个byteawithoutorderwithequalcol类型的数据是否不相同，不相同则返回true，否则返回false。
* 参数类型：byteawithoutorderwithequalcol, byteawithoutorderwithequalcol
* 返回值类型：bool
* byteawithoutorderwithequalcolnebyteal(bytea, byteawithoutorderwithequalcol)
* 描述：比较bytea和byteawithoutorderwithequalcol数据是否相同，相同则返回true，否则返回false。
* 参数类型：bytea，byteawithoutorderwithequalcol
* 返回值类型：bool
* byteawithoutorderwithequalcolnebytear(byteawithoutorderwithequalcol, bytea)
* 描述：比较byteawithoutorderwithequalcol和bytea数据是否不相同，相同则返回true，否则返回false。
* 参数类型：byteawithoutorderwithequalcol, bytea
* 返回值类型：bool
* hll\_hash\_byteawithoutorderwithequalcol(byteawithoutorderwithequalcol)
* 描述：返回byteawithoutorderwithequalcol的hll哈希值。
* 参数类型：byteawithoutorderwithequalcol
* 返回值类型：hll\_hashval
* 由于byteawithoutorderwithequalcolin的实现会对cek进行查找，并且判断是否为正常加密后的数据类型。因此如果用户输入数据的格式不为加密后的数据格式，并且在本地不存在对应cek的情况下，会返回错误。

vastbase=# SELECT \* FROM byteawithoutorderwithequalcolsend('x907219912381298461289346129'::byteawithoutorderwithequalcol);

ERROR: cek with OID 596711794 not found

LINE 1: SELECT \* FROM byteawithoutorderwithequalcolsend('x907219912...

^

vastbase=# SELECT \* FROM byteawithoutordercolout('x90721901999999999999912381298461289346129');

ERROR: cek with OID 2566986098 not found

LINE 1: SELECT \* FROM byteawithoutordercolout('x9072190199999999999...

SELECT \* FROM byteawithoutorderwithequalcolrecv('x90721901999999999999912381298461289346129'::byteawithoutorderwithequalcol);

ERROR: cek with OID 2566986098 not found

^

vastbase=# SELECT \* FROM byteawithoutorderwithequalcolsend('x90721901999999999999912381298461289346129'::byteawithoutorderwithequalcol);

ERROR: cek with OID 2566986098 not found

LINE 1: SELECT \* FROM byteawithoutorderwithequalcolsend('x907219019...

^

#### 返回集合的函数

**序列号生成函数**

* generate\_series(start, stop)
* 描述：生成一个数值序列，从start到stop，步长为1。
* 参数类型：int、bigint、numeric
* 返回值类型：setof int、setof bigint、setof numeric（与参数类型相同）
* generate\_series(start, stop, step)
* 描述：生成一个数值序列，从start到stop，步长为step。
* 参数类型：int、bigint、numeric
* 返回值类型：setof int、setof bigint、setof numeric（与参数类型相同）
* generate\_series(start, stop, step interval)
* 描述：生成一个数值序列，从start到stop，步长为step。
* 参数类型：timestamp或timestamp with time zone
* 返回值类型：setof timestamp或setof timestamp with time zone（与参数类型相同）

如果step是正数且start大于stop，则返回零行。相反，如果step是负数且start小于stop，则也返回零行。如果输入是NULL，同样产生零行。如果step为零则是一个错误。

示例：

vastbase=# SELECT \* FROM generate\_series(2,4);

generate\_series

-----------------

2

3

4

(3 rows)

vastbase=# SELECT \* FROM generate\_series(5,1,-2);

generate\_series

-----------------

5

3

1

(3 rows)

vastbase=# SELECT \* FROM generate\_series(4,3);

generate\_series

-----------------

(0 rows)

--这个示例应用于date-plus-integer操作符。

vastbase=# SELECT current\_date + s.a AS dates FROM generate\_series(0,14,7) AS s(a);

dates

------------

2017-06-02

2017-06-09

2017-06-16

(3 rows)

vastbase=# SELECT \* FROM generate\_series('2008-03-01 00:00'::timestamp, '2008-03-04 12:00', '10 hours');

generate\_series

---------------------

2008-03-01 00:00:00

2008-03-01 10:00:00

2008-03-01 20:00:00

2008-03-02 06:00:00

2008-03-02 16:00:00

2008-03-03 02:00:00

2008-03-03 12:00:00

2008-03-03 22:00:00

2008-03-04 08:00:00

(9 rows)

**下标生成函数**

* generate\_subscripts(array anyarray, dim int)
* 描述：生成一系列包括给定数组的下标。
* 返回值类型：setof int
* generate\_subscripts(array anyarray, dim int, reverse boolean)
* 描述：生成一系列包括给定数组的下标。当reverse为真时，该系列则以相反的顺序返回。
* 返回值类型：setof int

generate\_subscripts是一个为给定数组中的指定维度生成有效下标集的函数。如果数组中没有所请求的维度或者NULL数组，返回零行（但是会给数组元素为空的返回有效下标）。示例：

--基本用法。

vastbase=# SELECT generate\_subscripts('{NULL,1,NULL,2}'::int[], 1) AS s;

s

---

1

2

3

4

(4 rows)

--unnest一个2D数组。

vastbase=# CREATE OR REPLACE FUNCTION unnest2(anyarray)

RETURNS SETOF anyelement AS $$

SELECT $1[i][j]

FROM generate\_subscripts($1,1) g1(i),

generate\_subscripts($1,2) g2(j);

$$ LANGUAGE sql IMMUTABLE;

vastbase=# SELECT \* FROM unnest2(ARRAY[[1,2],[3,4]]);

unnest2

---------

1

2

3

4

(4 rows)

--删除函数。

vastbase=# DROP FUNCTION unnest2;

#### 条件表达式函数

**条件表达式函数**

* coalesce(expr1, expr2, ..., exprn)
* 描述：
* 返回参数列表中第一个非NULL的参数值。
* COALESCE(expr1, expr2) 等价于CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END。
* 示例：

vastbase=# SELECT coalesce(NULL,'hello');

coalesce

----------

hello

(1 row)

备注：

* 如果表达式列表中的所有表达式都等于NULL，则本函数返回NULL。
* 它常用于在显示数据时用缺省值替换NULL。
* 和CASE表达式一样，COALESCE不会计算不需要用来判断结果的参数；即在第一个非空参数右边的参数不会被计算。
* decode(base\_expr, compare1, value1, Compare2,value2, … default)
* 描述：把base\_expr与后面的每个compare(n) 进行比较，如果匹配返回相应的value(n)。如果没有发生匹配，则返回default。
* 示例：

vastbase=# SELECT decode('A','A',1,'B',2,0);

case

------

1

(1 row)

* nullif(expr1, expr2)
* 描述：当且仅当expr1和expr2相等时，NULLIF才返回NULL，否则它返回expr1。
* nullif(expr1, expr2) 逻辑上等价于CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END。
* 示例：

vastbase=# SELECT nullif('hello','world');

nullif

--------

hello

(1 row)

备注：

如果两个参数的数据类型不同，则：

* 若两种数据类型之间存在隐式转换，则以其中优先级较高的数据类型为基准将另一个参数隐式转换成该类型，转换成功则进行计算，转换失败则返回错误。如：

vastbase=# SELECT nullif('1234'::VARCHAR,123::INT4);

nullif

--------

1234

(1 row)

vastbase=# SELECT nullif('1234'::VARCHAR,'2012-12-24'::DATE);

ERROR: invalid input syntax for type timestamp: "1234"

* 若两种数据类型之间不存在隐式转换，则返回错误 。如：

vastbase=# SELECT nullif(TRUE::BOOLEAN,'2012-12-24'::DATE);

ERROR: operator does not exist: boolean = timestamp without time zone

LINE 1: SELECT nullif(TRUE::BOOLEAN,'2012-12-24'::DATE) FROM sys\_dummy;

HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.

* nvl( expr1 , expr2 )
* 描述：
* 如果expr1为NULL，则返回expr2。
* 如果expr1非NULL，则返回expr1。
* 示例：

vastbase=# SELECT nvl('hello','world');

nvl

-------

hello

(1 row)

备注：参数expr1和expr2可以为任意类型，当NVL的两个参数不属于同类型时，看第二个参数是否可以向第一个参数进行隐式转换。如果可以则返回第一个参数类型，否则返回错误。

* greatest(expr1 [, ...])
* 描述：获取并返回参数列表中值最大的表达式的值。
* 返回值类型：
* 示例：

vastbase=# SELECT greatest(1\*2,2-3,4-1);

greatest

----------

3

(1 row)

vastbase=# SELECT greatest('HARRY', 'HARRIOT', 'HAROLD');

greatest

----------

HARRY

(1 row)

* least(expr1 [, ...])
* 描述：获取并返回参数列表中值最小的表达式的值。
* 示例：

vastbase=# SELECT least(1\*2,2-3,4-1);

least

-------

-1

(1 row)

vastbase=# SELECT least('HARRY','HARRIOT','HAROLD');

least

--------

HAROLD

(1 row)

* EMPTY\_BLOB()
* 描述：使用EMPTY\_BLOB在INSERT或UPDATE语句中初始化一个BLOB变量，取值为NULL。
* 返回值类型：BLOB
* 示例：

--新建表

vastbase=# CREATE TABLE blob\_tb(b blob,id int);

--插入数据

vastbase=# INSERT INTO blob\_tb VALUES (empty\_blob(),1);

--删除表

vastbase=# DROP TABLE blob\_tb;

备注：使用DBE\_LOB.GET\_LENGTH求得的长度为0。

#### 统计信息函数

统计信息函数根据访问对象分为两种类型：针对某个数据库进行访问的函数，以数据库中每个表或索引的OID作为参数，标识需要报告的数据库；针对某个服务器进行访问的函数，以一个服务器进程号为参数，其范围从1到当前活跃服务器的数目。

* pg\_stat\_get\_db\_conflict\_tablespace(oid)
* 描述：由于恢复与数据库中删除的表空间发生冲突而取消的查询数。
* 返回值类型：bigint
* pg\_control\_group\_config
* 描述：在当前节点上打印cgroup配置。
* 返回值类型：record
* pg\_stat\_get\_db\_stat\_reset\_time(oid)
* 描述：上次重置数据库统计信息的时间。首次连接到每个数据库期间初始化为系统时间。当您在数据库上调用pg\_stat\_reset以及针对其中的任何表或索引执行pg\_stat\_reset\_single\_table\_counters时，重置时间都会更新。
* 返回值类型：timestamptz
* pg\_stat\_get\_function\_total\_time(oid)
* 描述：该函数花费的总挂钟时间，以微秒为单位。包括花费在此函数调用其它函数上的时间。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_returned(oid)
* 描述：当前事务中参数为表时通过顺序扫描读取的行数，或参数为索引时返回的索引条目数。
* 返回值类型：bigint
* pg\_lock\_status()
* 描述：查询打开事务所持有的锁信息，所有用户均可执行该函数。
* 返回值类型：返回字段可参考PG\_LOCKS视图返回字段，该视图是通过查询本函数得到的结果。
* pg\_stat\_get\_xact\_numscans(oid)
* 描述：当前事务中参数为表时执行的顺序扫描次数，或参数为索引时执行的索引扫描次数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_blocks\_fetched(oid)
* 描述：当前事务中对表或索引的磁盘块获取请求数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_blocks\_hit(oid)
* 描述：当前事务中对缓存中找到的表或索引的磁盘块获取请求数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_function\_calls(oid)
* 描述：在当前事务中调用该函数的次数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_function\_self\_time(oid)
* 描述：在当前事务中仅花费在此函数上的时间，不包括花费在此函数内部调用其它函数上的时间。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_function\_total\_time(oid)
* 描述：当前事务中该函数所花费的总挂钟时间（以微秒为单位），包括花费在此函数内部调用其它函数上的时间。
* 返回值类型：bigint
* pg\_stat\_get\_wal\_senders()
* 描述：在主机端查询walsender信息。
* 返回值类型：setofrecord
* 返回字段说明如下：
* **表 1** 返回字段说明

| **字段名称** | **字段类型** | **字段说明** |
| --- | --- | --- |
| pid | bigint | walsender的线程号。 |
| sender\_pid | integer | walsender的pid相对的轻量级线程号。 |
| local\_role | text | 主节点类型。 |
| peer\_role | text | 备节点类型。 |
| peer\_state | text | 备节点状态。 |
| state | text | walsender状态。 |
| catchup\_start | timestamp with time zone | catchup启动时间。 |
| catchup\_end | timestamp with time zone | catchup结束时间。 |
| sender\_sent\_location | text | 主节点发送位置。 |
| sender\_write\_location | text | 主节点落盘位置。 |
| sender\_flush\_location | text | 主节点flush磁盘位置。 |
| sender\_replay\_location | text | 主节点redo位置。 |
| receiver\_received\_location | text | 备节点接收位置。 |
| receiver\_write\_location | text | 备节点落盘位置。 |
| receiver\_flush\_location | text | 备节点flush磁盘位置。 |
| receiver\_replay\_location | text | 备节点redo磁盘位置。 |
| sync\_percent | text | 同步百分比。 |
| sync\_state | text | 同步状态。 |
| sync\_priority | text | 同步复制的优先级。 |
| sync\_most\_available | text | 最大可用模式设置。 |
| channel | text | walsender信道信息。 |

* get\_paxos\_replication\_info()
* 描述：查询Paxos模式下主机或备机的复制状态。
* 返回值类型：setofrecord
* 返回字段说明如下：
* **表 2** 返回字段说明

| **字段名称** | **字段类型** | **字段说明** |
| --- | --- | --- |
| paxos\_write\_location | text | 已经写入DCF的XLog位置。 |
| paxos\_commit\_location | text | 已经在DCF中达成一致的XLog位置。 |
| local\_write\_location | text | 节点的落盘位置。 |
| local\_flush\_location | text | 节点的flush磁盘位置。 |
| local\_replay\_location | text | 节点的redo磁盘位置。 |
| dcf\_replication\_info | text | 节点的DCF模块信息。 |

* pg\_stat\_get\_stream\_replications()
* 描述：查询主备复制状态。
* 返回值类型：setofrecord
* 返回值说明如下。
* **表 3** 返回值说明

| **返回参数** | **返回参数类型** | **返回参数说明** |
| --- | --- | --- |
| local\_role | text | 本地角色。 |
| static\_connections | integer | 连接统计。 |
| db\_state | text | 数据库状态。 |
| detail\_information | text | 详细信息。 |

* pg\_stat\_get\_db\_numbackends(oid)
* 描述：处理该数据库活跃的服务器进程数目。
* 返回值类型：integer
* pg\_stat\_get\_db\_xact\_commit(oid)
* 描述：数据库中已提交事务的数量。
* 返回值类型：bigint
* pg\_stat\_get\_db\_xact\_rollback(oid)
* 描述：数据库中回滚事务的数量。
* 返回值类型：bigint
* pg\_stat\_get\_db\_blocks\_fetched(oid)
* 描述：数据库中磁盘块抓取请求的总数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_blocks\_hit(oid)
* 描述：数据库在缓冲区中找到的磁盘块抓取请求的总数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_tuples\_returned(oid)
* 描述：为数据库返回的Tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_tuples\_fetched(oid)
* 描述：为数据库中获取的Tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_tuples\_inserted(oid)
* 描述：在数据库中插入Tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_tuples\_updated(oid)
* 描述：在数据库中更新的Tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_tuples\_deleted(oid)
* 描述：数据库中删除Tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_conflict\_lock(oid)
* 描述：数据库中锁冲突的数量。
* 返回值类型：bigint
* pg\_stat\_get\_db\_deadlocks(oid)
* 描述：数据库中死锁的数量。
* 返回值类型：bigint
* pg\_stat\_get\_numscans(oid)
* 描述：如果参数是一个表，则顺序扫描读取的行数目。如果参数是一个索引，则返回索引行的数目。
* 返回值类型：bigint
* pg\_stat\_get\_role\_name(oid)
* 描述：根据用户oid获取用户名。仅sysadmin和monitor admin用户可以访问。
* 返回值类型：text
* 示例：

vastbase=# select pg\_stat\_get\_role\_name(10);

pg\_stat\_get\_role\_name

-----------------------

aabbcc

(1 row)

* pg\_stat\_get\_tuples\_returned(oid)
* 描述：如果参数是一个表，则顺序扫描读取的行数目。如果参数是一个索引，则返回的索引行的数目。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_fetched(oid)
* 描述：如果参数是一个表，则位图扫描抓取的行数目。如果参数是一个索引，则用简单索引扫描抓取的行数目。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_inserted(oid)
* 描述：插入表中行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_updated(oid)
* 描述：在表中已更新行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_deleted(oid)
* 描述：从表中删除行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_changed(oid)
* 描述：该表上一次analyze或autoanalyze之后插入、更新、删除行的总数量。
* 返回值类型：bigint
* pg\_stat\_get\_tuples\_hot\_updated(oid)
* 描述：表热更新的行数。
* 返回值类型：bigint
* pg\_stat\_get\_live\_tuples(oid)
* 描述：表活行数。
* 返回值类型：bigint
* pg\_stat\_get\_dead\_tuples(oid)
* 描述：表死行数。
* 返回值类型：bigint
* pg\_stat\_get\_blocks\_fetched(oid)
* 描述：表或者索引的磁盘块抓取请求的数量。
* 返回值类型：bigint
* pg\_stat\_get\_blocks\_hit(oid)
* 描述：在缓冲区中找到的表或者索引的磁盘块请求数目。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_tuples\_inserted(oid)
* 描述：插入相应表分区中行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_tuples\_updated(oid)
* 描述：在相应表分区中已更新行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_tuples\_deleted(oid)
* 描述：从相应表分区中删除行的数量。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_tuples\_changed(oid)
* 描述：该表分区上一次analyze或autoanalyze之后插入、更新、删除行的总数量。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_live\_tuples(oid)
* 描述：分区表活行数。
* 返回值类型：bigint
* pg\_stat\_get\_partition\_dead\_tuples(oid)
* 描述：分区表死行数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_fetched(oid)
* 描述：事务中扫描的tuple行数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_inserted(oid)
* 描述：表相关的活跃子事务中插入的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_deleted(oid)
* 描述：表相关的活跃子事务中删除的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_hot\_updated(oid)
* 描述：表相关的活跃子事务中热更新的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_tuples\_updated(oid)
* 描述：表相关的活跃子事务中更新的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_partition\_tuples\_inserted(oid)
* 描述：表分区相关的活跃子事务中插入的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_partition\_tuples\_deleted(oid)
* 描述：表分区相关的活跃子事务中删除的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_partition\_tuples\_hot\_updated(oid)
* 描述：表分区相关的活跃子事务中热更新的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_xact\_partition\_tuples\_updated(oid)
* 描述：表分区相关的活跃子事务中更新的tuple数。
* 返回值类型：bigint
* pg\_stat\_get\_last\_vacuum\_time(oid)
* 描述：用户在该表上最后一次手动启动清理或者autovacuum线程启动清理的时间。
* 返回值类型：timestamptz
* pg\_stat\_get\_last\_autovacuum\_time(oid)
* 描述：autovacuum守护进程在该表上最后一次启动清理的时间。
* 返回值类型：timestamptz
* pg\_stat\_get\_vacuum\_count(oid)
* 描述：用户在该表上启动清理的次数。
* 返回值类型：bigint
* pg\_stat\_get\_autovacuum\_count(oid)
* 描述：autovacuum守护进程在该表上启动清理的次数。
* 返回值类型：bigint
* pg\_stat\_get\_last\_analyze\_time(oid)
* 描述：用户在该表上最后一次手动启动分析或者autovacuum线程启动分析的时间。
* 返回值类型：timestamptz
* pg\_stat\_get\_last\_autoanalyze\_time(oid)
* 描述：autovacuum守护进程在该表上最后一次启动分析的时间。
* 返回值类型：timestamptz
* pg\_stat\_get\_analyze\_count(oid)
* 描述：用户在该表上启动分析的次数。
* 返回值类型：bigint
* pg\_stat\_get\_autoanalyze\_count(oid)
* 描述：autovacuum守护进程在该表上启动分析的次数。
* 返回值类型：bigint
* pg\_total\_autovac\_tuples(bool,bool)
* 描述：返回total autovac相关的tuple记录，如nodename,nspname,relname以及各类tuple的IUD信息，入参分别为：是否查询relation信息，是否查询local信息。
* 返回值类型：setofrecord
* 返回参数说明如下。
* **表 4** 返回参数说明

| **返回参数** | **返回参数类型** | **返回参数说明** |
| --- | --- | --- |
| nodename | name | 节点名称。 |
| nspname | name | 名称空间名称。 |
| relname | name | 表、索引、视图等对象名称。 |
| partname | name | 分区名称。 |
| n\_dead\_tuples | bigint | 表分区内的死行数。 |
| n\_live\_tuples | bigint | 表分区内的活行数。 |
| changes\_since\_analyze | bigint | analyze产生改变的数量。 |

* pg\_autovac\_status(oid)
* 描述：返回和autovac状态相关的参数信息，如nodename,nspname,relname,analyze,vacuum设置，analyze/vacuum阈值, analyze/vacuum tuple数等。仅sysadmin可以使用该函数。
* 返回值类型：setofrecord
* 返回值参数说明如下。

**表 5** 返回值参数说明

| **返回参数** | **返回参数类型** | **返回参数说明** |
| --- | --- | --- |
| nspname | text | 名称空间名称。 |
| relname | text | 表、索引、视图等对象名称。 |
| nodename | text | 节点名称。 |
| doanalyze | Boolean | 是否执行analyze。 |
| anltuples | bigint | analyze tuple数量。 |
| anlthresh | bigint | analyze阈值。 |
| dovacuum | Boolean | 是否执行vacuum。 |
| vactuples | bigint | vacuum tuple数量。 |
| vacthresh | bigint | vacuum阈值。 |

* pg\_autovac\_timeout(oid)
* 描述：返回某个表做autovac连续超时的次数，表信息非法或node信息异常返回NULL。
* 返回值类型：bigint
* pg\_stat\_get\_last\_data\_changed\_time(oid)
* 描述：insert/update/delete, exchange/truncate/drop partition在该表上最后一次操作的时间，PG\_STAT\_ALL\_TABLES(参考：开发者指南->系统表和系统视图->系统视图->PG\_STAT\_ALL\_TABLES章节)视图last\_data\_changed列的数据是通过该函数求值，在表数量很大的场景中，通过视图获取表数据最后修改时间的性能较差，建议直接使用该函数获取表数据的最后修改时间。
* 返回值类型：timestamptz
* pg\_stat\_set\_last\_data\_changed\_time(oid)
* 描述：手动设置该表上最后一次insert/update/delete, exchange/truncate/drop partition操作的时间。
* 返回值类型：void
* pg\_backend\_pid()
* 描述：当前会话的服务器线程的线程ID。
* 返回值类型：integer
* pg\_stat\_get\_activity(integer)
* 描述：返回一个关于带有特殊PID的后台进程的记录信息，当参数为NULL时，则返回每个活动的后台进程的记录。返回结果不包含connection\_info列。初始用户、系统管理员和monadmin可以查看所有的数据，普通用户只能查询自己的结果。
* 示例：

vastbase=# select \* from pg\_stat\_get\_activity(139881386280704);

datid | pid | sessionid | usesysid | application\_name | state | query | waiting | xact\_start | query\_start |

backend\_start | state\_change | client\_addr | client\_hostname | client\_port | enqueue | query\_id | srespool | global\_sessionid | unique\_sql\_id | trace\_id

-------+-----------------+-----------+----------+------------------+--------+------------------------------------------------------+---------+-------------------------------+-------------------------------+-----

16545 | 139881386280704 | 69 | 10 | vsql | active | select \* from pg\_stat\_get\_activity(139881386280704); | f | 2022-01-18 19:43:05.167718+08 | 2022-01-18 19:43:05.167718+08 | 2022

-01-18 19:42:33.513507+08 | 2022-01-18 19:43:05.16773+08 | | | -1 | | 72620543991624410 | default\_pool | 1938253334#69#0 | 3751941862 |

(1 row)

返回值类型：setofrecord

返回参数说明如下。

**表 6** 返回参数说明

| **返回参数** | **返回参数类型** | **返回参数说明** |
| --- | --- | --- |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| pid | bigint | 后台线程ID。 |
| sessionid | bigint | 会话ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| application\_name | text | 连接到该后台的应用名。 |
| state | text | 该后台当前总体状态。 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| waiting | Boolean | 如果后台当前正等待锁则为true。 |
| xact\_start | timestamp with time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。  如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestamp with time zone | 开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。 |
| backend\_start | timestamp with time zone | 该过程开始的时间，即当客户端连接服务器时。 |
| state\_change | timestamp with time zone | 上次状态改变的时间。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| enqueue | text | 该字段暂不支持。 |
| query\_id | bigint | 查询语句的ID。 |
| srespool | name | 资源池名字。 |
| global\_sessionid | text | 全局会话id。 |
| unique\_sql\_id | bigint | 语句的unique sql id。 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

* pg\_stat\_get\_activity\_with\_conninfo(integer)
* 描述：返回一个关于带有特殊PID的后台进程的记录信息，当参数为NULL时，则返回每个活动的后台进程的记录。初始用户、系统管理员和monadmin可以查看所有的数据，普通用户只能查询自己的结果。
* 返回值类型：setofrecord
* 返回值说明如下。

**表 7** 返回值说明

| **返回值** | **返回值类型** | **返回值说明** |
| --- | --- | --- |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| pid | bigint | 后台线程ID。 |
| sessionid | bigint | 会话ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| application\_name | text | 连接到该后台的应用名。 |
| state | text | 改后台当前总体状态 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| waiting | Boolean | 如果后台当前正等待锁则为true |
| xact\_start | timestamp with time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestamp with time zone | 开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。 |
| backend\_start | timestamp with time zone | 该过程开始的时间，即当客户端连接服务器时。 |
| state\_change | timestamp with time zone | 上次状态改变的时间。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| enqueue | text | 该字段暂不支持。 |
| query\_id | bigint | 查询语句的ID。 |
| connection\_info | text | json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息。 |
| srespool | name | 资源池名字。 |
| global\_sessionid | text | 全局会话ID。 |
| unique\_sql\_id | bigint | 语句的unique sql id。 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

* pg\_user\_iostat(text)
* 描述：显示和当前用户执行作业正在运行时的IO负载管理相关信息。
* 返回值类型：record
* 函数返回字段说明如下：

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| userid | oid | 用户id。 |
| min\_curr\_iops | int4 | 当前该用户io在数据库节点中的最小值。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| max\_curr\_iops | int4 | 当前该用户io在数据库节点中的最大值。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| min\_peak\_iops | int4 | 该用户io峰值中，数据库节点的最小值。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| max\_peak\_iops | int4 | 该用户io峰值中，数据库节点的最大值。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| io\_limits | int4 | 用户指定的资源池所设置的io\_limits。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| io\_priority | text | 该用户所设io\_priority。对于行存，以万次/s为单位；对于列存，以次/s为单位。 |
| curr\_io\_limits | int4 | 使用io\_priority管控io时的实时io\_limits值。 |

* pg\_stat\_get\_function\_calls(oid)
* 描述：函数已被调用次数。
* 返回值类型：bigint
* pg\_stat\_get\_function\_self\_time(oid)
* 描述：只有在此函数上所花费的时间。此函数调用其它函数上花费的时间被排除在外。
* 返回值类型：bigint
* pg\_stat\_get\_backend\_idset()
* 描述：设置当前活动的服务器进程数（从1到活动服务器进程的数量）。
* 返回值类型：setofinteger
* pg\_stat\_get\_backend\_pid(integer)
* 描述：给定的服务器线程的线程ID。
* 返回值类型：bigint
* pg\_stat\_get\_backend\_dbid(integer)
* 描述：给定服务器进程的数据库ID。
* 返回值类型：oid
* pg\_stat\_get\_backend\_userid(integer)
* 描述：给定服务器进程的用户ID。
* 返回值类型：oid
* pg\_stat\_get\_backend\_activity(integer)
* 描述：给定服务器进程的当前活动查询，仅在调用者是系统管理员或被查询会话的用户，并且打开track\_activities的时候才能获得结果。
* 返回值类型：text
* pg\_stat\_get\_backend\_waiting(integer)
* 描述：如果给定服务器进程在等待某个锁，并且调用者是系统管理员或被查询会话的用户，并且打开track\_activities的时候才返回真。
* 返回值类型：Boolean
* pg\_stat\_get\_backend\_activity\_start(integer)
* 描述：给定服务器进程当前正在执行的查询的起始时间，仅在调用者是系统管理员或被查询会话的用户，并且打开track\_activities的时候才能获得结果。
* 返回值类型：timestampwithtimezone
* pg\_stat\_get\_backend\_xact\_start(integer)
* 描述：给定服务器进程当前正在执行的事务的开始时间，但只有当前用户是系统管理员或被查询会话的用户，并且打开track\_activities的时候才能获得结果。
* 返回值类型：timestampwithtimezone
* pg\_stat\_get\_backend\_start(integer)
* 描述：给定服务器进程启动的时间，如果当前用户不是系统管理员或被查询的后端的用户，则返回NULL。
* 返回值类型：timestampwithtimezone
* pg\_stat\_get\_backend\_client\_addr(integer)
* 描述：连接到给定客户端后端的IP地址。如果是通过Unix域套接字连接的则返回NULL；如果当前用户不是系统管理员或被查询会话的用户，也返回NULL。
* 返回值类型：inet
* pg\_stat\_get\_backend\_client\_port(integer)
* 描述：连接到给定客户端后端的TCP端口。如果是通过Unix域套接字连接的则返回-1；如果当前用户不是系统管理员或被查询会话的用户，也返回NULL。
* 返回值类型：integer
* pg\_stat\_get\_bgwriter\_timed\_checkpoints()
* 描述：后台写进程开启定时检查点的时间（因为checkpoint\_timeout时间已经过期了）。
* 返回值类型：bigint
* pg\_stat\_get\_bgwriter\_requested\_checkpoints()
* 描述：后台写进程开启基于后端请求的检查点的时间，因为已经超过了checkpoint\_segments或因为已经执行了CHECKPOINT。
* 返回值类型：bigint
* pg\_stat\_get\_bgwriter\_buf\_written\_checkpoints()
* 描述：在检查点期间后台写进程写入的缓冲区数目。
* 返回值类型：bigint
* pg\_stat\_get\_bgwriter\_buf\_written\_clean()
* 描述：为日常清理脏块，后台写进程写入的缓冲区数目。
* 返回值类型：bigint
* pg\_stat\_get\_bgwriter\_maxwritten\_clean()
* 描述：后台写进程停止清理扫描的时间，因为已经写入了更多的缓冲区（相比bgwriter\_lru\_maxpages参数声明的缓冲区数）。
* 返回值类型：bigint
* pg\_stat\_get\_buf\_written\_backend()
* 描述：后端进程写入的缓冲区数，因为它们需要分配一个新的缓冲区。
* 返回值类型：bigint
* pg\_stat\_get\_buf\_alloc()
* 描述：分配的总缓冲区数。
* 返回值类型：bigint
* pg\_stat\_clear\_snapshot()
* 描述：清理当前的统计快照。
* 返回值类型：void
* pg\_stat\_reset()
* 描述：为当前数据库重置统计计数器为0（需要系统管理员权限）。
* 返回值类型：void
* pg\_stat\_reset\_shared(text)
* 描述：重置shared cluster每个节点当前数据统计计数器为0（需要系统管理员权限）。
* 返回值类型：void
* pg\_stat\_reset\_single\_table\_counters(oid)
* 描述：为当前数据库中的一个表或索引重置统计为0（需要系统管理员权限）。
* 返回值类型：void
* pg\_stat\_reset\_single\_function\_counters(oid)
* 描述：为当前数据库中的一个函数重置统计为0（需要系统管理员权限）。
* 返回值类型：void
* pg\_stat\_session\_cu(int, int, int)
* 描述：获取当前节点所运行session的CU命中统计信息。
* 返回值类型：record
* pg\_stat\_get\_cu\_mem\_hit(oid)
* 描述：获取当前节点当前数据库中一个列存表的CU内存命中次数。
* 返回值类型：bigint
* pg\_stat\_get\_cu\_hdd\_sync(oid)
* 描述：获取当前节点当前数据库中一个列存表从磁盘同步读取CU次数。
* 返回值类型：bigint
* pg\_stat\_get\_cu\_hdd\_asyn(oid)
* 描述：获取当前节点当前数据库中一个列存表从磁盘异步读取CU次数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_cu\_mem\_hit(oid)
* 描述：获取当前节点一个数据库CU内存命中次数。
* 返回值类型：bigint
* pg\_stat\_get\_db\_cu\_hdd\_sync(oid)
* 描述：获取当前节点一个数据库从磁盘同步读取CU次数。
* 返回值类型：bigint
* fenced\_udf\_process(integer)
* 描述：查看本地UDF Master和Work进程数。入参为1时查看master进程数，入参为2时查看worker进程数，入参为3时杀死所有worker进程。
* 返回值类型：text
* total\_cpu()
* 描述：获取当前节点使用的CPU时间，单位是jiffies。
* 返回值类型：bigint
* mot\_global\_memory\_detail()
* 描述：检查MOT全局内存的大小，主要包括数据和索引。
* 返回值类型：record
* mot\_local\_memory\_detail()
* 描述：检查MOT局部内存的大小，主要包括数据和索引。
* 返回值类型：record
* mot\_session\_memory\_detail()
* 描述：检查所有会话对MOT内存的使用情况。
* 返回值类型：record
* total\_memory()
* 描述：获取当前节点使用的虚拟内存大小，单位KB。
* 返回值类型：bigint
* pg\_stat\_get\_db\_cu\_hdd\_asyn(oid)
* 描述：获取当前节点一个数据库从磁盘异步读取CU次数。
* 返回值类型：bigint
* pg\_stat\_bad\_block(text, int, int, int, int, int, timestamp with time zone, timestamp with time zone)
* 描述：获取当前节点自启动后，读取出现Page/CU的损坏信息。
* 例: select \* from pg\_stat\_bad\_block();
* 返回值类型：record
* pg\_stat\_bad\_block\_clear()
* 描述：清理节点记录的读取出现的Page/CU损坏信息（需要系统管理员权限）。
* 返回值类型：void
* gs\_respool\_exception\_info(pool text)
* 描述：查看某个资源池关联的查询规则信息。
* 返回值类型：record
* gs\_control\_group\_info(pool text)
* 描述：查看资源池关联的控制组信息
* 返回值类型：record
* 返回信息如下：

| **属性** | **属性值** | **描述** |
| --- | --- | --- |
| name | class\_a:workload\_a1 | class和workload名称 |
| class | class\_a | Class控制组名称 |
| workload | workload\_a1 | Workload控制组名称 |
| type | DEFWD | 控制组类型（Top、CLASS、BAKWD、DEFWD、TSWD） |
| gid | 87 | 控制组id |
| shares | 30 | 占父节点CPU资源的百分比 |
| limits | 0 | 占父节点CPU核数的百分比 |
| rate | 0 | Timeshare中的分配比例 |
| cpucores | 0-3 | CPU核心数 |

* gs\_all\_control\_group\_info()
* 描述：查看数据库内所有的控制组信息。
* 返回值类型：record
* gs\_get\_control\_group\_info()
* 描述：查看所有的控制组信息。
* 返回值类型：record
* get\_instr\_workload\_info(integer)
* 描述：获取数据库主节点上事务量信息，事务时间信息。
* 返回值类型：record

| **属性** | **属性值** | **描述** |
| --- | --- | --- |
| resourcepool\_oid | 10 | 资源池的oid(逻辑同负载等价) |
| commit\_counter | 4 | 前端事务commit数量 |
| rollback\_counter | 1 | 前端事务rollback数量 |
| resp\_min | 949 | 前端事务最小响应时间（单位：微秒） |
| resp\_max | 201891 | 前端事务最大响应时间（单位：微秒） |
| resp\_avg | 43564 | 前端事务平均响应时间(单位：微秒) |
| resp\_total | 217822 | 前端事务总响应时间（单位：微秒） |
| bg\_commit\_counter | 910 | 后端事务commit数量 |
| bg\_rollback\_counter | 0 | 后端事务rollback数量 |
| bg\_resp\_min | 97 | 后端事务最小响应时间（单位：微秒） |
| bg\_resp\_max | 678080687 | 后端事务最大响应时间（单位：微秒） |
| bg\_resp\_avg | 327847884 | 后端事务平均响应时间（单位：微秒） |
| bg\_resp\_total | 298341575300 | 后端事务总响应时间（单位：微秒） |

* pv\_instance\_time()
* 描述：获取当前节点上各个关键阶段的时间消耗。
* 返回值类型：record

| **Stat\_name属性** | **属性值** | **描述** |
| --- | --- | --- |
| DB\_TIME | 1062385 | 所有线程端到端的墙上时间（WALL TIME）消耗总和(单位： 微秒) |
| CPU\_TIME | 311777 | 所有线程CPU时间消耗总和(单位： 微秒) |
| EXECUTION\_TIME | 380037 | 消耗在执行器上的时间总和(单位： 微秒) |
| PARSE\_TIME | 6033 | 消耗在SQL解析上的时间总和(单位： 微秒) |
| PLAN\_TIME | 173356 | 消耗在执行计划生成上的时间总和(单位： 微秒) |
| REWRITE\_TIME | 2274 | 消耗在查询重写上的时间总和(单位： 微秒) |
| PL\_EXECUTION\_TIME | 0 | 消耗在PL/SQL执行上的时间总和(单位： 微秒) |
| PL\_COMPILATION\_TIME | 557 | 消耗在SQL编译上的时间总和(单位： 微秒) |
| NET\_SEND\_TIME | 1673 | 消耗在网络发送上的时间总和(单位： 微秒) |
| DATA\_IO\_TIME | 426622 | 消耗在数据读写上的时间总和(单位： 微秒) |

* DBE\_PERF.get\_global\_instance\_time()
* 描述：提供vastbase各个关键阶段的时间消耗，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* get\_instr\_unique\_sql()
* 描述：获取当前节点的执行语句（归一化SQL）信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* reset\_unique\_sql(text, text, bigint)
* 描述：重置系统执行语句（归一化SQL）信息，执行该函数必须具有sysadmin权限。第一个参数取值范围“global/local”，global表示清理所有节点上的信息，local表示只清理当前节点；第二参数取值范围“ALL/BY\_USERID/BY\_CNID”，ALL表示清理所有信息，BY\_USERID表示通过指定USERID清理只属于该用户的sql信息，BY\_CNID表示清理系统中涉及到该数据库主节点的sql信息；第三个参数表示具体的CNID和USERID，如果第二个参数为ALL，第三个参数不起作用，可以取任意值。
* 返回值类型：boolean
* fig: **说明：**
* 此函数中所说节点指分布式节点，当前vastbase为集中式数据库，global与local功能一致，取值范围不支持BY\_CNID。
* get\_instr\_wait\_event(NULL)
* 描述：获取当前节点event等待的统计信息。
* 返回值类型：record
* get\_instr\_user\_login()
* 描述：获取当前节点的用户登入登出次数信息，查询该函数必须具有sysadmin或者monitor admin权限。
* 返回值类型：record
* get\_instr\_rt\_percentile(integer)
* 描述：获取数据库SQL响应时间P80,P95分布信息。
* 返回值类型：record
* get\_node\_stat\_reset\_time()
* 描述：获取当前节点的统计信息重置（重启，主备倒换，数据库删除）时间。
* 返回值类型：record
* DBE\_PERF.get\_global\_os\_runtime()
* 描述：显示当前操作系统运行的状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_os\_threads()
* 描述：提供vastbase中所有正常节点下的线程状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_workload\_sql\_count()
* 描述：提供vastbase中不同负载SELECT，UPDATE，INSERT，DELETE，DDL，DML，DCL计数信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_workload\_sql\_elapse\_time()
* 描述：提供vastbase中不同负载SELECT，UPDATE，INSERT，DELETE，响应时间信息（TOTAL,AVG, MIN, MAX），查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_workload\_transaction()
* 描述：获取vastbase内所有节点上的事务量信息，事务时间信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_session\_stat()
* 描述：获取vastbase节点上的会话状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* fig: **说明：**
* 状态信息有17项：commit、rollback、sql、table\_scan、blocks\_fetched、physical\_read\_operation、
* shared\_blocks\_dirtied、local\_blocks\_dirtied、shared\_blocks\_read、local\_blocks\_read、
* blocks\_read\_time、blocks\_write\_time、sort\_imemory、sort\_idisk、cu\_mem\_hit、
* cu\_hdd\_sync\_read、cu\_hdd\_asyread。
* DBE\_PERF.get\_global\_session\_time()
* 描述：提供vastbase各节点各个关键阶段的时间消耗，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_session\_memory()
* 描述：汇聚各节点的Session级别的内存使用情况，包含执行作业在数据节点上Postgres线程和Stream线程分配的所有内存，单位为MB，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_session\_memory\_detail()
* 描述：汇聚各节点的线程的内存使用情况，以MemoryContext节点来统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* create\_wlm\_session\_info(int flag)
* 描述：将当前内存中记录的TopSQL查询语句级别相关统计信息清理。该函数只有管理员用户可以执行。
* 返回值类型：int
* pg\_stat\_get\_wlm\_session\_info(int flag)
* 描述：获取当前内存中记录的TopSQL查询语句级别相关统计信息，当传入的参数不为0时，会将这部分信息从内存中清理掉。该函数只有system admin和monitor admin用户可以执行。
* 返回值类型：record
* gs\_paxos\_stat\_replication()
* 描述：在主机端查询备机信息。目前只支持集中式DCF模式。
* 返回值类型：setofrecord
* 返回字段说明如下：

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **字段说明** |
| local\_role | text | 发送日志节点的角色。 |
| peer\_role | text | 接收日志节点的角色。 |
| local\_dcf\_role | text | 发送日志节点的DCF角色。 |
| peer\_dcf\_role | text | 接收日志节点的DCF角色。 |
| peer\_state | text | 接收日志节点的状态。 |
| sender\_write\_location | text | 发送日志节点写到xlog buffer的位置。 |
| sender\_commit\_location | text | 发送日志节点DCF日志达成一致性点。 |
| sender\_flush\_location | text | 发送日志节点写到xlog disk的位置。 |
| sender\_replay\_location | text | 发送日志节点replay的位置。 |
| receiver\_write\_location | text | 接收日志节点写到xlog buffer的位置。 |
| receiver\_commit\_location | text | 接收日志节点DCF日志达成一致性点。 |
| receiver\_flush\_location | text | 接收日志节点写到xlog disk的位置。 |
| receiver\_replay\_location | text | 接收日志节点重演xlog的位置。 |
| sync\_percent | text | 同步百分比。 |
| dcf\_run\_mode | int4 | DCF同步模式。 |
| channel | text | 信道信息。 |

* gs\_wlm\_get\_resource\_pool\_info(int)
* 描述：获取所有用户的资源使用统计信息，入参为int类型可以为可以为任意int值或NULL。
* 返回值类型：record
* gs\_wlm\_get\_all\_user\_resource\_info()
* 描述：获取所有用户的资源使用统计信息。
* 返回值类型：record
* gs\_wlm\_get\_user\_info(int)
* 描述：获取所有用户的相关信息，入参为int类型可以为可以为任意int值或NULL。该函数只有sysadmin权限的用户可以执行。
* 返回值类型：record
* gs\_wlm\_get\_workload\_records()
* 描述：获取动态负载管理下的所有作业信息，该函数只在动态负载管理开的情况下有效。
* 返回值类型：record
* gs\_wlm\_readjust\_user\_space()
* 描述：修正所有用户的存储空间使用情况。该函数只有管理员用户可以执行。
* 返回值类型：record
* gs\_wlm\_readjust\_user\_space\_through\_username(text name)
* 描述：修正指定用户的存储空间使用情况。该函数普通用户只能修正自己的使用情况，只有管理员用户可以修正所有用户的使用情况。当name指定位“0000”，表示需要修正所有用户的使用情况。
* 返回值类型：record
* gs\_wlm\_readjust\_user\_space\_with\_reset\_flag(text name, boolean isfirst)
* 描述：修正指定用户的存储空间使用情况。入参isfirst为true表示从0开始统计，否则从上一次结果继续统计。该函数普通用户只能修正自己的使用情况，只有管理员用户可以修正所有用户的使用情况。当name指定位“0000”，表示需要修正所有用户的使用情况。
* 返回值类型：record
* gs\_wlm\_session\_respool(bigint)
* 描述：获取当前所有后台线程的session resouce pool相关信息，入参为bigint类型可以为可以为任意bigint值或NULL。
* 返回值类型：record
* gs\_wlm\_get\_session\_info()
* 描述：目前该接口已废弃，暂不可用。
* gs\_wlm\_get\_user\_session\_info()
* 描述：目前该接口已废弃，暂不可用。
* gs\_io\_wait\_status()
* 描述：目前该接口不支持单机和集中式，暂不可用。
* global\_stat\_get\_hotkeys\_info()
* 描述：获取整个数据库实例中热点key的统计情况。目前该接口不支持单机和集中式，暂不可用。
* global\_stat\_clean\_hotkeys()
* 描述：清理整个数据库实例中热点key的统计信息。目前该接口不支持单机和集中式，暂不可用。
* DBE\_PERF.get\_global\_session\_stat\_activity()
* 描述：汇聚vastbase内各节点上正在运行的线程相关的信息，查询该函数必须具有monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_thread\_wait\_status()
* 描述：汇聚所有节点上工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况，查询该函数必须具有sysadmin和monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_operator\_history\_table()
* 描述：汇聚当前用户数据库主节点上执行作业结束后的算子相关记录（持久化），查询该函数必须具有sysadmin和monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_operator\_history()
* 描述：汇聚当前用户数据库主节点上执行作业结束后的算子相关记录，查询该函数必须具有sysadmin和monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_operator\_runtime()
* 描述：汇聚当前用户数据库主节点上执行作业实时的算子相关记录，查询该函数必须具有sysadmin和monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statement\_complex\_history()
* 描述：汇聚当前用户数据库主节点上复杂查询的历史记录，查询该函数必须具有monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statement\_complex\_history\_table()
* 描述：汇聚当前用户数据库主节点上复杂查询的历史记录（持久化），查询该函数必须具有monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statement\_complex\_runtime()
* 描述：汇聚当前用户数据库主节点上复杂查询的实时信息，查询该函数必须具有sysadmin和monadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_memory\_node\_detail()
* 描述：汇聚所有节点某个数据库节点内存使用情况，查询该函数必须具有monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_shared\_memory\_detail()
* 描述：汇聚所有节点已产生的共享内存上下文的使用信息，查询该函数必须具有monitoradmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_all\_indexes()
* 描述：汇聚所有节点当前数据库中的每个索引行，显示特定索引的I/O的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_all\_tables()
* 描述：显示汇聚各节点数据中每个表（包括TOAST表）的一行的统计信息
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_all\_tables()
* 描述：显示各节点数据中每个表（包括TOAST表）的一行的统计信息。
* 返回值类型：record
* DBE\_PERF.get\_local\_toastname\_and\_toastindexname()
* 描述：提供本地toast表的name和index和其关联表的对应关系，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_all\_indexes()
* 描述：统计所有节点当前数据库中的每个索引行，显示特定索引的I/O的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_all\_sequences()
* 描述：提供命名空间中所有sequences的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_all\_tables()
* 描述：汇聚各节点的数据库中每个表I/O的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_all\_tables()
* 描述：统计vastbase内数据库中每个表I/O的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_local\_toast\_relation()
* 描述：提供本地toast表的name和其关联表的对应关系，查询该函数必须具有sysadmin权限
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_sys\_indexes()
* 描述：汇聚各节点的命名空间中所有系统表索引的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_sys\_indexes()
* 描述：统计各节点的命名空间中所有系统表索引的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_sys\_sequences()
* 描述：提供命名空间中所有系统表为sequences的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_sys\_tables()
* 描述：提供各节点的命名空间中所有系统表的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_sys\_tables()
* 描述：vastbase内汇聚命名空间中所有系统表的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_user\_indexes()
* 描述：各节点的命名空间中所有用户关系表索引的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_user\_indexes()
* 描述：vastbase内汇聚命名空间中所有用户关系表索引的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_user\_sequences()
* 描述：显示各节点的命名空间中所有用户的sequences的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statio\_user\_tables()
* 描述：显示各节点的命名空间中所有用户关系表的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statio\_user\_tables()
* 描述：vastbase内汇聚命名空间中所有用户关系表的IO状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_stat\_db\_cu()
* 描述：视图查询vastbase各个节点，每个数据库的CU命中情况，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_all\_indexes()
* 描述：汇聚所有节点数据库中每个索引的统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_all\_indexes()
* 描述：统计所有节点数据库中每个索引的统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_sys\_tables()
* 描述：汇聚各节点pg\_catalog、information\_schema模式的所有命名空间中系统表的统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_sys\_tables()
* 描述：统计各节点pg\_catalog、information\_schema模式的所有命名空间中系统表的统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_sys\_indexes()
* 描述：汇聚各节点pg\_catalog、information\_schema模式中所有系统表的索引状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_sys\_indexes()
* 描述：统计各节点pg\_catalog、information\_schema模式中所有系统表的索引状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_user\_tables()
* 描述：汇聚所有命名空间中用户自定义普通表的状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_user\_tables()
* 描述：统计所有命名空间中用户自定义普通表的状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_user\_indexes()
* 描述：汇聚所有数据库中用户自定义普通表的索引状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_user\_indexes()
* 描述：统计所有数据库中用户自定义普通表的索引状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_database()
* 描述：汇聚所有节点数据库统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_database\_conflicts()
* 描述：统计所有节点数据库统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_xact\_all\_tables()
* 描述：汇聚命名空间中所有普通表和toast表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_xact\_all\_tables()
* 描述：统计命名空间中所有普通表和toast表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_xact\_sys\_tables()
* 描述：汇聚所有节点命名空间中系统表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_xact\_sys\_tables()
* 描述：统计所有节点命名空间中系统表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_xact\_user\_tables()
* 描述：汇聚所有节点命名空间中用户表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_stat\_xact\_user\_tables()
* 描述：统计所有节点命名空间中用户表的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_user\_functions()
* 描述：汇聚所有节点命名空间中用户定义函数的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_xact\_user\_functions()
* 描述：统计所有节点命名空间中用户定义函数的事务状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_stat\_bad\_block()
* 描述：汇聚所有节点表、索引等文件的读取失败信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_file\_redo\_iostat()
* 描述：统计所有节点表、索引等文件的读取失败信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_file\_iostat()
* 描述：汇聚所有节点数据文件IO的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_locks()
* 描述：汇聚所有节点的锁信息，查询该函数必须具有sysadmin和monadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_replication\_slots()
* 描述：汇聚所有节点上逻辑复制信息，查询该函数必须具有sysadmin和monadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_bgwriter\_stat()
* 描述：汇聚所有节点后端写进程活动的统计信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_replication\_stat()
* 描述：汇聚各节点日志同步状态信息，如发起端发送日志位置，收端接收日志位置等，查询该函数必须具有sysadmin和monadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_transactions\_running\_xacts()
* 描述：汇聚各节点运行事务的信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_transactions\_running\_xacts()
* 描述：统计各节点运行事务的信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_transactions\_prepared\_xacts()
* 描述：汇聚各节点当前准备好进行两阶段提交的事务的信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_transactions\_prepared\_xacts()
* 描述：统计各节点当前准备好进行两阶段提交的事务的信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_statement()
* 描述：汇聚各节点历史执行语句状态信息，查询该函数必须具有monitor admin和sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_statement\_count()
* 描述：汇聚各节点SELECT，UPDATE，INSERT，DELETE，响应时间信息（TOTAL,AVG, MIN, MAX），查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_config\_settings()
* 描述：汇聚各节点GUC参数配置信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_wait\_events()
* 描述：汇聚各节点wait events状态信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_statement\_responsetime\_percentile()
* 描述：获取vastbaseSQL响应时间P80，P95分布信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_summary\_user\_login()
* 描述：统计vastbase各节点用户登入登出次数信息，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.get\_global\_record\_reset\_time()
* 描述：汇聚vastbase统计信息重置（重启，主备倒换，数据库删除）时间，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.track\_memory\_context(context\_list text)
* 描述：设置需要统计内存申请详细信息的内存上下文。入参为内存上下文的名称，使用“，”分隔，如“ThreadTopMemoryContext, SessionCacheMemoryContext”，注意该内存上下文名称是上下文敏感的。此外，单个内存上下文的长度为63，超过的部分会被截断。而且一次能够统计的内存上下文上限为16个，设置超过16个内存上下文会设置失败。每一次调用该函数都会将上次统计的结果清空，当入参指定为“”时，表示取消该统计功能。只有初始用户（super user）或者具有monadmin权限的用户可以执行该函数。
* 返回值类型：boolean
* DBE\_PERF.track\_memory\_context\_detail()
* 描述：获取DBE\_PERF.track\_memory\_context函数指定的内存上下文的内存申请详细信息。返回值的定义见视图DBE\_PERF.track\_memory\_context\_detail。只有初始用户（super user）或者具有monadmin权限的用户可以执行该函数。
* 返回值类型：record
* pg\_stat\_get\_mem\_mbytes\_reserved(tid)
* 描述：统计资源管理相关变量值，仅用于定位问题使用。
* 参数：线程id。
* 返回值类型：text
* gs\_wlm\_user\_resource\_info(name text)
* 描述：查询具体某个用户的资源限额和资源使用情况。
* 返回值类型：record
* pg\_stat\_get\_file\_stat()
* 描述：通过对数据文件IO的统计，反映数据的IO性能，用以发现IO操作异常等性能问题。
* 返回值类型: record
* pg\_stat\_get\_redo\_stat()
* 描述：用于统计会话线程日志回放情况。
* 返回值类型: record
* pg\_stat\_get\_status(int8)
* 描述：可以检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。
* 返回值类型：record
* get\_local\_rel\_iostat()
* 描述：查询当前节点的数据文件IO状态累计值。
* 返回值类型：record
* DBE\_PERF.get\_global\_rel\_iostat()
* 描述：汇聚所有节点数据文件IO的统计，查询该函数必须具有sysadmin权限。
* 返回值类型：record
* DBE\_PERF.global\_threadpool\_status()
* 描述：显示在所有节点上的线程池中工作线程及会话的状态信息。函数返回信息具体字段参考：开发者指南->schema->DBE\_PERF Schema->Session/Thread->GLOBAL\_THREADPOOL\_STATUS章节。
* 返回值类型：record
* remote\_bgwriter\_stat()
* 描述：显示数据库所有实例的bgwriter线程刷页信息，候选buffer链中页面个数，buffer淘汰信息（本节点除外、DN上不可使用）。
* 返回值类型：record
* pv\_os\_run\_info
* 描述：显示当前操作系统运行的状态信息，具体字段信息参考：开发者指南->系统表和系统视图->系统视图->GS\_OS\_RUN\_INFO章节。
* 参数：nan
* 返回值类型：setof record
* pv\_session\_stat
* 描述：以会话线程或AutoVacuum线程为单位，统计会话状态信息，具体字段信息参考：开发者指南->系统表和系统视图->系统视图->GS\_SESSION\_STAT章节。
* 参数：nan
* 返回值类型：setof record
* pv\_session\_time
* 描述：用于统计会话线程的运行时间信息，及各执行阶段所消耗时间，具体字段信息参考：开发者指南->系统表和系统视图->系统视图->GS\_SESSION\_TIME章节)。
* 参数：nan
* 返回值类型：setof record
* pg\_stat\_get\_db\_temp\_bytes
* 描述：用于统计通过数据库查询写入临时文件的数据总量。计算所有临时文件，不论为什么创建临时文件，而且不管log\_temp\_files设置。
* 参数：oid
* 返回值类型：bigint
* pg\_stat\_get\_db\_temp\_files
* 描述：通过数据库查询创建的临时文件数量。计算所有临时文件，不论为什么创建临时文件（比如排序或者哈希），而且不管log\_temp\_files设置。
* 参数：oid
* 返回值类型：bigint
* remote\_candidate\_stat()
* 描述：用于显示数据库所有实例的检查点信息和各类日志刷页情况（本节点除外），集中式不支持。
* 返回值类型：record
* dbe\_perf.gs\_stat\_activity\_timeout(int)
* 描述：获取当前节点上执行时间超过超时阈值的查询作业信息。需要GUC参数track\_activities设置为on才能正确返回结果。超时阈值的取值范围是0~2147483。
* 返回值类型：setof record

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| database | name | 用户会话连接的数据库名称。 |
| pid | bigint | 后台线程ID。 |
| sessionid | bigint | 会话ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| application\_name | text | 连接到该后台的应用名。 |
| query | text | 该后台正在执行的查询。 |
| xact\_start | timestamptz | 启动当前事务的时间。 |
| query\_start | timestamptz | 开始当前查询的时间。 |
| query\_id | bigint | 查询语句ID。 |

* gs\_wlm\_user\_resource\_info(name text)
* 描述：查询具体某个用户的资源限额和资源使用情况。普通用户只能查询到自己相关的信息，管理员权限的用户可以查看全部用户的信息。
* 返回值类型：record
* create\_wlm\_instance\_statistics\_info
* 描述：将当前实例的历史监控数据进行持久化保存。
* 参数：nan
* 返回值类型：integer
* gs\_session\_memory
* 描述：统计Session级别的内存使用情况，包含执行作业在数据节点上Postgres线程和Stream线程分配的所有内存。
* fig: **说明：**
* 若GUC参数enable\_memory\_limit(参考：开发者指南->GUC参数说明->资源消耗->内存章节的enable\_memory\_limit参数)=off，该函数不能使用。
* 返回值类型：record
* **表 8** 返回值说明

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| init\_mem | integer | 当前正在执行作业进入执行器前已分配的内存，单位MB。 |
| used\_mem | integer | 当前正在执行作业已分配的内存，单位MB。 |
| peak\_mem | integer | 当前正在执行作业已分配的内存峰值，单位MB。 |

* gs\_wlm\_persistent\_user\_resource\_info()
* 描述：将当前所有的用户资源使用统计信息归档到gs\_wlm\_user\_resource\_history系统表中，只有sysadmin有权限查询。
* 返回值类型：record
* create\_wlm\_operator\_info(int flag)
* 描述：将当前内存中记录的TopSQL算子级别相关统计信息清理，当传入的参数大于0时，会将这部分信息归档到gs\_wlm\_operator\_info和gs\_wlm\_ec\_operator\_info中，否则不会归档。该函数只有sysadmin权限的用户可以执行。
* 返回值类型：int
* GS\_ALL\_NODEGROUP\_CONTROL\_GROUP\_INFO(text)
* 描述：提供了所有逻辑数据库实例的控制组信息。该函数在调用的时候需要指定要查询逻辑数据库实例的名称。例如要查询'installation'逻辑数据库实例的控制组信息：

SELECT \* FROM GS\_ALL\_NODEGROUP\_CONTROL\_GROUP\_INFO('installation')

返回值类型：record

函数返回字段如下：

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | text | 控制组的名称。 |
| type | text | 控制组的类型。 |
| gid | bigint | 控制组ID。 |
| classgid | bigint | Workload所属Class的控制组ID。 |
| class | text | Class控制组。 |
| workload | text | Workload控制组。 |
| shares | bigint | 控制组分配的CPU资源配额。 |
| limits | bigint | 控制组分配的CPU资源限额。 |
| wdlevel | bigint | Workload控制组层级。 |
| cpucores | text | 控制组使用的CPU核的信息。 |

* gs\_total\_nodegroup\_memory\_detail
* 描述：返回当前数据库逻辑数据库使用内存的信息，单位为MB得到一个逻辑数据
* 库。
* 返回值类型：setof record
* local\_redo\_time\_count()
* 描述：返回本节点各个回放线程的各个流程的耗时统计（仅在备机上有有效数据）。
* 返回值如下：
* local\_redo\_time\_count返回参数说明。

| **字段名** | **描述** |
| --- | --- |
| thread\_name | 线程名字 |
| step1\_total | step1的总时间，每个线程对应的流程如下：  极致RTO：   * batch redo：从队列中获取一条日志 * redo manager：从队列中获取一条日志 * redo worker：从队列中获取一条日志 * trxn manager：从队列中读取一条日志 * trxn worker：从队列中读取一条日志 * read worker：从文件中读取一次xlog page（整体） * read page worker：从队列中获取一个日志 * startup：从队列中获取一个日志   并行回放：   * page redo：从队列中获取一条日志 * startup：读取一条日志 |
| step1\_count | step1的统计次数 |
| step2\_total | step2的总时间，每个线程对应的流程如下：  极致RTO：   * batch redo：处理日志（整体） * redo manager：处理日志（整体） * redo worker：处理日志（整体） * trxn manager：处理日志（整体） * trxn worker：处理日志（整体） * redo worker：读取xlog page耗时 * read page worker：生成和发送lsn forwarder * startup：check stop(是否回放到指定位置）   并行回放：   * page redo：处理日志（整体） * startup：check stop(是否回放到指定位置） |
| step2\_count | step2的统计次数 |
| step3\_total | step3的总时间，每个线程对应的流程如下：  极致RTO：   * batch redo：更新standbystate * redo manager：数据日志处理 * redo worker：回放page也日志（整体） * trxn manager：更新flush lsn * trxn worker：回放日志处理 * redo worker：推进xlog segment * read page worker：获取一个新的item * startup：redo delay（延迟回放特性等待时间）   并行回放：   * page redo：更新standbystate * startup：redo delay（延迟回放特性等待时间） |
| step3\_count | step3的统计次数 |
| step4\_total | step4的总时间，每个线程对应的流程如下：  极致RTO：   * batch redo：解析xlog * redo manager：DDL处理 * redo worker：读取数据page页 * trxn manager：同步等待时间 * trxn worker：更新本线程lsn * read page worker：将日志放入分发线程 * startup：分发（整体）   并行回放：   * page redo：undo 日志回放 * startup：分发（整体） |
| step4\_count | step4的统计次数 |
| step5\_total | step5的总时间，每个线程对应的流程如下：  极致RTO：   * batch redo：分发给redo manager * redo manager：分发给redo worker * redo worker：回放数据page页的日志 * trxn manager：分发给trxn worker * trxn worker：强同步wait时间 * read page worker：更新本线程lsn * startup：日志decode   并行回放：   * page redo：sharetrxn 日志回放 * startup：日志回放 |
| step5\_count | step5的统计次数 |
| step6\_total | step6的总时间，每个线程对应的流程如下：  极致RTO：   * redo worker：回放非数据页page日志 * trxn manager：全局lsn更新 * read page worker：日志crc校验   并行回放：   * page redo：synctrxn 日志回放 * startup：强同步等待 |
| step6\_count | step6的统计次数 |
| step7\_total | step7的总时间，每个线程对应的流程如下：  极致RTO：  redo worker：fsm更新  并行回放：  page redo：single 日志回放 |
| step7\_count | step7的统计次数 |
| step8\_total | step8的总时间，每个线程对应的流程如下：  极致RTO：  redo worker：强同步等待  并行回放：  page redo：all workers do 日志回放 |
| step8\_count | step8的统计次数 |
| step9\_total | step9的总时间，每个线程对应的流程如下：  极致RTO：  无  并行回放：  page redo：muliti workers do 日志回放 |
| step9\_count | step9的统计次数 |

* local\_xlog\_redo\_statics()
* 描述：返回本节点已经回放的各个类型类型的日志统计信息（仅在备机上有有效数据）。
* 返回值如下：
* **表 9** local\_xlog\_redo\_statics返回参数说明

| **字段名** | **描述** |
| --- | --- |
| xlog\_type | 日志类型 |
| rmid | resource manager id |
| info | xlog operation |
| num | 日志个数 |
| extra | 针对page回放日志和xact日志有有效值。page页回放日志标识从磁盘读取page的个数。xact日志表示删除文件的个数。 |

* gs\_get\_shared\_memctx\_detail(text)
* 描述：返回指定内存上下文上的内存申请的详细信息，包含每一处内存申请所在的文件、行号和大小（同一文件同一行大小会做累加）。只支持查询通过pg\_shared\_memory\_detail视图查询出来的内存上下文，入参为内存上下文名称（即pg\_shared\_memory\_detail返回结果的contextname列）。查询该函数必须具有sysadmin权限或者monitor admin权限。
* 返回值类型：setof record

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| file | text | 申请内存所在文件的文件名。 |
| line | int8 | 申请内存所在文件的代码行号。 |
| size | int8 | 申请的内存大小，同一文件同一行多次申请会做累加。 |

fig: **说明：**

该视图不支持release版本小型化场景。

* gs\_get\_session\_memctx\_detail(text)
* 描述：返回指定内存上下文上的内存申请的详细信息，包含每一处内存申请所在的文件、行号和大小（同一文件同一行大小会做累加）。仅在线程池模式下生效。只支持查询通过pv\_session\_memory\_context视图查询出来的内存上下文，入参为内存上下文名称（即pv\_session\_memory\_context返回结果的contextname列）。查询该函数必须具有sysadmin权限或者monitor admin权限。
* 返回值类型：setof record

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| file | text | 申请内存所在文件的文件名。 |
| line | int8 | 申请内存所在文件的代码行号。 |
| size | int8 | 申请的内存大小，同一文件同一行多次申请会做累加。 |

fig: **说明：**

该视图仅在线程池模式下生效，且该视图不支持release版本小型化场景。

* gs\_get\_thread\_memctx\_detail(tid,text)
* 描述：返回指定内存上下文上的内存申请的详细信息，包含每一处内存申请所在的文件，行号和大小（同一文件同一行大小会做累加）。只支持查询通过pv\_thread\_memory\_context视图查询出来的内存上下文，第一个入参为线程id（即pv\_thread\_memory\_context返回数据的tid列），第二个参数为内存上下文名称（即pv\_thread\_memory\_context返回数据的contextname列）。查询该函数必须具有sysadmin权限或者monitor admin权限。
* 返回值类型：setof record

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| file | text | 申请内存所在文件的文件名。 |
| line | int8 | 申请内存所在文件的代码行号。 |
| size | int8 | 申请的内存大小，同一文件同一行多次申请会做累加。 |

fig: **说明：**

该视图不支持release版本小型化场景。

#### 触发器函数

* pg\_get\_triggerdef(oid)
* 描述：获取触发器的定义信息。
* 参数：待查触发器的OID。
* 返回值类型：text
* 示例:

vastbase=# select pg\_get\_triggerdef(oid) from pg\_trigger;

pg\_get\_triggerdef

------------------------------------------------------------------------

CREATE TRIGGER tg1 BEFORE INSERT ON gtest26 FOR EACH STATEMENT EXECUTE PROCEDURE gtest\_trigger\_func()

CREATE TRIGGER tg03 AFTER INSERT ON gtest26 FOR EACH ROW WHEN ((new.a IS NOT NULL)) EXECUTE PROCEDURE gtest\_trigger\_func()

(2 rows)

* pg\_get\_triggerdef(oid, boolean)
* 描述：获取触发器的定义信息。
* 参数：待查触发器的OID及是否以pretty方式展示。
* fig: **说明：**

仅在创建trigger时指定WHEN条件的情况下，布尔类型参数才生效。

返回值类型：text

示例:

vastbase=# select pg\_get\_triggerdef(oid,true) from pg\_trigger;

pg\_get\_triggerdef

--------------------------------------------------------------------

CREATE TRIGGER tg1 BEFORE INSERT ON gtest26 FOR EACH STATEMENT EXECUTE PROCEDURE gtest\_trigger\_func()

CREATE TRIGGER tg03 AFTER INSERT ON gtest26 FOR EACH ROW WHEN (new.a IS NOT NULL) EXECUTE PROCEDURE gtest\_trigger\_func()

(2 rows)

vastbase=# select pg\_get\_triggerdef(oid,false) from pg\_trigger;

pg\_get\_triggerdef

--------------------------------------------------------------------

CREATE TRIGGER tg1 BEFORE INSERT ON gtest26 FOR EACH STATEMENT EXECUTE PROCEDURE gtest\_trigger\_func()

CREATE TRIGGER tg03 AFTER INSERT ON gtest26 FOR EACH ROW WHEN ((new.a IS NOT NULL)) EXECUTE PROCEDURE gtest\_trigger\_func()

(2 rows)

#### 提示信息函数

* report\_application\_error
* 描述：PL执行过程中，可以使用此函数来抛ERROR。
* 返回值类型：void
* **表 1** report\_application\_error参数说明

| **参数** | **类型** | **说明** | **是否必选** |
| --- | --- | --- | --- |
| log | text | error消息的内容。 | 是 |
| code | int4 | error消息对应的error code，范围为：-20999 ~ -20000。 | 否 |

#### 全局临时表函数

* pg\_get\_gtt\_relstats(relOid)
* 描述：显示当前会话指定的全局临时表的基本信息。
* 参数：全局临时表的OID。
* 返回值类型：record
* 示例:

vastbase=# select \* from pg\_get\_gtt\_relstats(74069);

relfilenode | relpages | reltuples | relallvisible | relfrozenxid | relminmxid

-------------+----------+-----------+---------------+--------------+------------

74069 | 58 | 13000 | 0 | 11151 | 0

(1 row)

* pg\_get\_gtt\_statistics(relOid, attnum, ''::text)
* 描述：显示当前会话指定的全局临时表的单列统计信息。
* 参数：全局临时表的OID和属性attnum。
* 返回值类型：record
* 示例:

vastbase=# select \* from pg\_get\_gtt\_statistics(74069,1,''::text);

starelid | starelkind | staattnum | stainherit | stanullfrac | stawidth | stadistinct | stakind1 | stakind2 | stakind3 | stakind4 | stakind5 | staop1 | staop2 | staop3 | staop4 | staop5 | stanumbers1 | stanumbers2 | stanumbers3 | stanu

mbers4 | stanumbers5 |

stavalues1

| stavalues2 | stavalues3 | stavalues4 | stavalues5 | stadndistinct | staextinfo

----------+------------+-----------+------------+-------------+----------+-------------+----------+----------+----------+----------+----------+--------+--------+--------+--------+--------+-------------+-------------+-------------+------

-------+-------------+----------------------------------------------------------------------

--------------------------------------------------------------------------------------------

------------------------------------------------------------------------+------------+------------+------------+------------+---------------+------------

74069 | c | 1 | f | 0 | 4 | -1 | 2 | 3 | 0 | 0 | 0 | 97 | 97 | 0 | 0 | 0 | | {1} | |

| | {1,130,260,390,520,650,780,910,1040,1170,1300,1430,1560,1690,1820,1950,2080,2210,2340,2470,2600,2730,2860,2990,3120,3250,3380,3510,3640,3770,3900,4030,4160,4290,4420,4550,4680,4810,4940,5070,5200,5330,5460,5590,57

20,5850,5980,6110,6240,6370,6500,6630,6760,6890,7020,7150,7280,7410,7540,7670,7800,7930,8060,8190,8320,8450,8580,8710,8840,8970,9100,9230,9360,9490,9620,9750,9880,10010,10140,10270,10400,10530,10660,10790,10920,11050,11180,11310,11440,1

1570,11700,11830,11960,12090,12220,12350,12480,12610,12740,12870,13000} | | | | | 0 |

(1 row)

* pg\_gtt\_attached\_pid(relOid)
* 描述：显示正在使用指定全局临时表的所有线程pid。
* 参数：全局临时表的OID。
* 返回值类型：record
* 示例:

vastbase=# select \* from pg\_gtt\_attached\_pid(74069);

relid | pid

-------+-----------------

74069 | 139648170456832

74069 | 139648123270912

(2 rows)

* dbe\_perf.get\_global\_full\_sql\_by\_timestamp(start\_timestamp timestamp, end\_timestamp timestamp)
* 描述：获取实例级的全量SQL(Full SQL)信息。
* 返回值类型：record
* **表 1** dbe\_perf.get\_global\_full\_sql\_by\_timestamp参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| start\_timestamp | timestamp | SQL启动时间范围的开始时间点。 |
| end\_timestamp | timestamp | SQL启动时间范围的结束时间点。 |

* dbe\_perf.get\_global\_slow\_sql\_by\_timestamp(start\_timestamp timestamp, end\_timestamp timestamp)
* 描述：获取实例级的慢SQL(Slow SQL)信息。
* 返回值类型：record
* **表 2** dbe\_perf.get\_global\_slow\_sql\_by\_timestamp参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| start\_timestamp | timestamp | SQL启动时间范围的开始时间点。 |
| end\_timestamp | timestamp | SQL启动时间范围的结束时间点。 |

* statement\_detail\_decode(detail text, format text, pretty bool)
* 解析全量/慢SQL语句中的details字段的信息。
* **表 3** statement\_detail\_decode参数说明

| **参数** | **类型** | **描述** |
| --- | --- | --- |
| detail | text | SQL语句产生的事件的集合（不可读）。 |
| format | text | 解析输出格式，取值为plaintext或json。 |
| pretty | bool | 当format为plaintext时，是否以优雅的格式展示：   * true表示通过“\n”分隔事件。 * false表示通过“，”分隔事件。 |

* pg\_list\_gtt\_relfrozenxids()
* 描述：显示各会话的冻结事务xid。
* pid=0的行，显示所有会话中最老的冻结事务xid。
* 参数：无。
* 返回值类型：record
* 示例:

vastbase=# select \* from pg\_list\_gtt\_relfrozenxids();

pid | relfrozenxid

-----------------+--------------

139648123270912 | 11151

139648170456832 | 11155

0 | 11151

(3 rows)

#### 故障注入系统函数

* gs\_fault\_inject(int64, text, text, text, text, text)
* 描述：该函数不能调用，调用时会报WARNING信息："unsupported fault injection"，并不会对数据库产生任何影响和改变。
* 参数：int64注入故障类型（0：CLOG扩展页面，1：读取CLOG页面，2：强制死锁）。
* text第二个入参在第一入参为2的模式下若为“1”则死锁，其余不死锁；第二个入参在第一入参为0，1时，表示CLOG开始扩展或读取的起始页面号。
* text第三个入参在第一入参为0，1时，表示扩展或读取的页面个数。
* text第四到六入参为预留参数。
* 返回值类型：int64

#### AI特性函数

* gs\_index\_advise(text)
* 描述：针对单条查询语句推荐索引。
* 参数：SQL语句字符串
* 返回值类型：record
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI子功能->INDEX-ADVISOR索引推荐章节的单query索引推荐。
* hypopg\_create\_index(text)
* 描述：创建虚拟索引。
* 参数：创建索引语句的字符串
* 返回值类型：record
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI功能->INDEX-ADVISOR索引推荐章节的虚拟索引。
* hypopg\_display\_index()
* 描述：显示所有创建的虚拟索引信息。
* 参数：无
* 返回值类型：record
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI功能->INDEX-ADVISOR索引推荐章节的虚拟索引。
* hypopg\_drop\_index(oid)
* 描述：删除指定的虚拟索引。
* 参数：索引的oid
* 返回值类型：bool
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI功能->INDEX-ADVISOR索引推荐章节的虚拟索引。
* hypopg\_reset\_index()
* 描述：清除所有虚拟索引。
* 参数：无
* 返回值类型：无
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI功能->INDEX-ADVISOR索引推荐章节的虚拟索引。
* hypopg\_estimate\_size(oid)
* 描述：估计指定索引创建所需的空间大小。
* 参数：索引的oid
* 返回值类型：int8
* 示例请参见：开发者指南->AI特性->AI4DB数据库自治运维->DBMIND的AI功能->INDEX-ADVISOR索引推荐章节的虚拟索引。
* check\_engine\_status(ip text, port text)
* 描述：测试给定的ip和port上是否有predictor engine提供服务。
* 参数：predictor engine的ip地址和端口号。
* 返回值类型：text
* 示例请参见：开发者指南->AI特性->AI in DB：数据库内AI功能->智能Explain：SQL语句查询时间预测->使用指导。
* encode\_plan\_node(optname text, orientation text, strategy text, options text, dop int8, quals text, projection text)
* 描述：对入参的计划算子信息进行编码。
* 参数：计划算子信息。
* 返回值类型：text。
* fig: **说明：**
* 该函数为内部功能调用函数，不建议用户直接使用。
* model\_train\_opt(template text, model text)
* 描述：训练给定的查询性能预测模型。
* 参数：性能预测模型的模板名和模型名。
* 返回值类型：tartup\_time\_accuracy FLOAT8、 total\_time\_accuracy FLOAT8、 rows\_accuracy FLOAT8、 peak\_memory\_accuracy FLOAT8
* 示例请参见：开发者指南->AI特性->AI in DB：数据库内AI功能->智能Explain：SQL语句查询时间预测->使用指导。
* track\_model\_train\_opt(ip text, port text)
* 描述：返回给定ip和port predictor engine的训练日志地址。
* 参数：predictor engine的ip地址和端口号。
* 返回值类型：text
* 示例请参见：开发者指南->AI特性->AI in DB：数据库内AI功能->智能Explain：SQL语句查询时间预测->使用指导。
* encode\_feature\_perf\_hist(datname text)
* 描述：将目标数据库已收集的历史计划算子进行编码。
* 参数：数据库名。
* 返回值类型：queryid bigint、 plan\_node\_id int、 parent\_node\_id int、 left\_child\_id int、 right\_child\_id int, encode text、 startup\_time bigint、 total\_time bigint、 rows bigint、 peak\_memory int
* 示例请参见：开发者指南->AI特性->AI in DB：数据库内AI功能->智能Explain：SQL语句查询时间预测->使用指导。
* gather\_encoding\_info(datname text)
* 描述：调用encode\_feature\_perf\_hist，将编码好的数据进行持久化保存。
* 参数：数据库名。
* 返回值类型：int
* 示例请参见：开发者指南->AI特性->AI in DB：数据库内AI功能->智能Explain：SQL语句查询时间预测->使用指导。
* db4ai\_predict\_by\_bool (text, VARIADIC "any")
* 描述：获取返回值为布尔型的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：bool
* db4ai\_predict\_by\_float4(text, VARIADIC "any")
* 描述：获取返回值为float4的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：float
* db4ai\_predict\_by\_float8(text, VARIADIC "any")
* 描述：获取返回值为float8的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：float
* db4ai\_predict\_by\_int32(text, VARIADIC "any")
* 描述：获取返回值为int32的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：int
* db4ai\_predict\_by\_int64(text, VARIADIC "any")
* 描述：获取返回值为int64的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：int
* db4ai\_predict\_by\_numeric(text, VARIADIC "any")
* 描述：获取返回值为numeric的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：numeric
* db4ai\_predict\_by\_text(text, VARIADIC "any")
* 描述：获取返回值为字符型的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：text
* db4ai\_predict\_by\_float8\_array(text, VARIADIC "any")
* 描述：获取返回值为字符型的模型进行模型推断任务。此函数为内部调用函数，建议直接使用语法PREDICT BY进行推断任务。
* 参数：模型名称和推断任务的输入列。
* 返回值类型：text
* gs\_explain\_model(text)
* 描述：获取返回值为字符型的模型进行模型解析文本化任务。
* 参数：模型名称。
* 返回值类型：text

#### 动态数据脱敏函数

fig: **说明：**

该函数为内部功能调用函数。

* creditcardmasking(col text, letter char default 'x')
* 描述：将col字符串后四位之前的数字使用letter替换。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* basicmailmasking(col text, letter char default 'x')
* 描述：将col字符串中第一个'@'之前的字符使用letter替换。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* fullmailmasking(col text, letter char default 'x')
* 描述：将col字符串中出现最后一个'.'之前的字符(除'@'外)使用letter替换。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* alldigitsmasking(col text, letter char default '0')
* 描述：将col字符串中出现的数字使用letter替换。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* shufflemasking(col text)
* 描述：将col字符串中的字符乱序排列。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* randommasking(col text)
* 描述：将col字符串中的字符随机化。
* 参数：待替换的字符串、替换字符。
* 返回值类型：text
* regexpmasking
* 描述：脱敏策略的内部函数，对字符进行正则表达式替换。
* 参数：col text, reg text, replace\_text text, pos INTEGER default 0, reg\_len INTEGER default -1
* 返回值类型：text

#### 内部函数

Vastbase中下列函数使用了内部数据类型，用户无法直接调用，在此章节列出。

* 选择率计算函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| areajoinsel | areasel | arraycontjoinsel | arraycontsel | contjoinsel | contsel | eqjoinsel |
| eqsel | iclikejoinsel | iclikesel | icnlikejoinsel | icnlikesel | icregexeqjoinsel | icregexeqsel |
| icregexnejoinsel | icregexnesel | likejoinsel | likesel | neqjoinsel | neqsel | nlikejoinsel |
| nlikesel | positionjoinsel | positionsel | regexeqjoinsel | regexeqsel | regexnejoinsel | regexnesel |
| scalargtjoinsel | scalargtsel | scalarltjoinsel | scalarltsel | tsmatchjoinsel | tsmatchsel | - |

* 统计信息收集函数

|  |  |  |
| --- | --- | --- |
| array\_typanalyze | range\_typanalyze | ts\_typanalyze |
| local\_rto\_stat |  |  |

* 排序内部功能函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bpchar\_sortsupport | bytea\_sortsupport | date\_sortsupport | numeric\_sortsupport | timestamp\_sortsupport |

* 全文检索内部功能函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dispell\_init | dispell\_lexize | dsimple\_init | dsimple\_lexize | dsnowball\_init | dsnowball\_lexize | dsynonym\_init |
| dsynonym\_lexize | gtsquery\_compress | gtsquery\_consistent | gtsquery\_decompress | gtsquery\_penalty | gtsquery\_picksplit | gtsquery\_same |
| gtsquery\_union | ngram\_end | ngram\_lextype | ngram\_start | pound\_end | pound\_lextype | pound\_start |
| prsd\_end | prsd\_headline | prsd\_lextype | prsd\_start | thesaurus\_init | thesaurus\_lexize | zhprs\_end |
| zhprs\_getlexeme | zhprs\_lextype | zhprs\_start | - | - | - | - |

* 内部类型处理函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| abstimerecv | euc\_jis\_2004\_to\_utf8 | int2recv | line\_recv | oidvectorrecv\_extend | tidrecv | utf8\_to\_koi8u |
| anyarray\_recv | euc\_jp\_to\_mic | int2vectorrecv | lseg\_recv | path\_recv | time\_recv | utf8\_to\_shift\_jis\_2004 |
| array\_recv | euc\_jp\_to\_sjis | int4recv | macaddr\_recv | pg\_node\_tree\_recv | time\_transform | utf8\_to\_sjis |
| ascii\_to\_mic | euc\_jp\_to\_utf8 | int8recv | mic\_to\_ascii | point\_recv | timestamp\_recv | utf8\_to\_uhc |
| ascii\_to\_utf8 | euc\_kr\_to\_mic | internal\_out | mic\_to\_big5 | poly\_recv | timestamp\_transform | utf8\_to\_win |
| big5\_to\_euc\_tw | euc\_kr\_to\_utf8 | interval\_recv | mic\_to\_euc\_cn | pound\_nexttoken | timestamptz\_recv | uuid\_recv |
| big5\_to\_mic | euc\_tw\_to\_big5 | interval\_transform | mic\_to\_euc\_jp | prsd\_nexttoken | timetz\_recv | varbit\_recv |
| big5\_to\_utf8 | euc\_tw\_to\_mic | iso\_to\_koi8r | mic\_to\_euc\_kr | range\_recv | tintervalrecv | varbit\_transform |
| bit\_recv | euc\_tw\_to\_utf8 | iso\_to\_mic | mic\_to\_euc\_tw | rawrecv | tsqueryrecv | varchar\_transform |
| boolrecv | float4recv | iso\_to\_win1251 | mic\_to\_iso | record\_recv | tsvectorrecv | varcharrecv |
| box\_recv | float8recv | iso\_to\_win866 | mic\_to\_koi8r | regclassrecv | txid\_snapshot\_recv | void\_recv |
| bpcharrecv | gb18030\_to\_utf8 | iso8859\_1\_to\_utf8 | mic\_to\_latin1 | regconfigrecv | uhc\_to\_utf8 | win\_to\_utf8 |
| btoidsortsupport | gbk\_to\_utf8 | iso8859\_to\_utf8 | mic\_to\_latin2 | regdictionaryrecv | unknownrecv | win1250\_to\_latin2 |
| bytearecv | gin\_extract\_tsvector | johab\_to\_utf8 | mic\_to\_latin3 | regoperatorrecv | utf8\_to\_ascii | win1250\_to\_mic |
| byteawithoutorderwithequalcolrecv | gtsvector\_compress | json\_recv | mic\_to\_latin4 | regoperrecv | utf8\_to\_big5 | win1251\_to\_iso |
| cash\_recv | gtsvector\_consistent | koi8r\_to\_iso | mic\_to\_sjis | regprocedurerecv | utf8\_to\_euc\_cn | win1251\_to\_koi8r |
| charrecv | gtsvector\_decompress | koi8r\_to\_mic | mic\_to\_win1250 | regprocrecv | utf8\_to\_euc\_jis\_2004 | win1251\_to\_mic |
| cidr\_recv | gtsvector\_penalty | koi8r\_to\_utf8 | mic\_to\_win1251 | regtyperecv | utf8\_to\_euc\_jp | win1251\_to\_win866 |
| cidrecv | gtsvector\_picksplit | koi8r\_to\_win1251 | mic\_to\_win866 | reltimerecv | utf8\_to\_euc\_kr | win866\_to\_iso |
| circle\_recv | gtsvector\_same | koi8r\_to\_win866 | namerecv | shift\_jis\_2004\_to\_euc\_jis\_2004 | utf8\_to\_euc\_tw | win866\_to\_koi8r |
| cstring\_recv | gtsvector\_union | koi8u\_to\_utf8 | ngram\_nexttoken | shift\_jis\_2004\_to\_utf8 | utf8\_to\_gb18030 | win866\_to\_mic |
| date\_recv | hll\_recv | latin1\_to\_mic | numeric\_recv | sjis\_to\_euc\_jp | utf8\_to\_gbk | win866\_to\_win1251 |
| domain\_recv | hll\_trans\_recv | latin2\_to\_mic | numeric\_transform | sjis\_to\_mic | utf8\_to\_iso8859 | xidrecv |
| euc\_cn\_to\_mic | hstore\_recv | latin2\_to\_win1250 | nvarchar2recv | sjis\_to\_utf8 | utf8\_to\_iso8859\_1 | xidrecv4 |
| euc\_cn\_to\_utf8 | inet\_recv | latin3\_to\_mic | oidrecv | smalldatetime\_recv | utf8\_to\_johab | xml\_recv |
| euc\_jis\_2004\_to\_shift\_jis\_2004 | int1recv | latin4\_to\_mic | oidvectorrecv | textrecv | utf8\_to\_koi8r | cstore\_tid\_out |
| i16toi1 | int16 | int16\_bool | int16eq | int16div | int16ge | int16gt |
| int16in | int16le | int16lt | int16mi | int16mul | int16ne | int16out |
| int16pl | int16recv | int16send | numeric\_bool | int2vectorin\_extend | int2vectorout\_extend | int2vectorrecv\_extend |
| int2vectorsend\_extend | jsonb\_in | jsonb\_out | jsonb\_recv | jsonb\_send | complex\_array\_in | bool\_int1 |
| bool\_int2 | bool\_int8 | bpchar\_float4 | bpchar\_float8 | bpchar\_int4 | bpchar\_int8 | bpchar\_numeric |
| bpchar\_timestamp | f4toi1 | f8toi1 | float4\_bpchar | float4\_text | float4\_varchar | float8\_bpchar |
| float8\_text | float8\_varchar | i1tof4 | i1tof8 | i1toi2 | i1toi4 | i1toi8 |
| i2toi1 | i4toi1 | i8toi1 | int1\_avg\_accum | int1\_bool | int1\_bpchar | int1\_numeric |
| int1\_nvarchar2 | int1\_text | int1\_varchar | int1abs | int1and | int1cmp | int1div |
| int1eq | int1ge | int1gt | int1in | int1inc | int1larger | int1le |
| int1lt | int1mi | int1mod | int1mul | int1ne | int1not | int1or |
| int1out | int1pl | int1shl | int1shr | int1smaller | int1um | int1up |
| int1xor | int2\_bool | int2\_bpchar | int2\_text | int2\_varchar | int4\_bpchar | int4\_text |
| int4\_varchar | int8\_bool | int8\_bpchar | int8\_text | int8\_varchar | job\_submit | numeric\_bpchar |
| numeric\_int1 | numeric\_text | numeric\_varchar | nvarchar2in | nvarchar2out | nvarchar2send | oidvectorin\_extend |
| oidvectorout\_extend | oidvectorsend\_extend | rawcmp | raweq | rawge | rawgt | rawin |
| rawle | rawlike | rawlt | rawne | rawnlike | rawout | rawsend |
| text\_float4 | text\_float8 | text\_int1 | text\_int2 | text\_int4 | text\_int8 | text\_numeric |
| timestamp\_text | timestamp\_varchar | varchar\_float4 | varchar\_float8 | varchar\_int4 | varchar\_int8 | varchar\_numeric |
| xidout4 | xidsend4 | calculate\_quantile\_of | calculate\_value\_at | large\_seq\_rollback\_ntree | large\_seq\_upgrade\_ntree | - |

* 聚合操作内部函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| array\_agg\_finalfn | array\_agg\_transfn | bytea\_string\_agg\_finalfn | bytea\_string\_agg\_transfn | date\_list\_agg\_noarg2\_transfn | date\_list\_agg\_transfn | float4\_list\_agg\_noarg2\_transfn |
| float4\_list\_agg\_transfn | float8\_list\_agg\_noarg2\_transfn | float8\_list\_agg\_transfn | int2\_list\_agg\_noarg2\_transfn | int2\_list\_agg\_transfn | int4\_list\_agg\_noarg2\_transfn | int4\_list\_agg\_transfn |
| int8\_list\_agg\_noarg2\_transfn | int8\_list\_agg\_transfn | interval\_list\_agg\_noarg2\_transfn | interval\_list\_agg\_transfn | list\_agg\_finalfn | list\_agg\_noarg2\_transfn | list\_agg\_transfn |
| median\_float8\_finalfn | median\_interval\_finalfn | median\_transfn | mode\_final | numeric\_list\_agg\_noarg2\_transfn | numeric\_list\_agg\_transfn | ordered\_set\_transition |
| percentile\_cont\_float8\_final | percentile\_cont\_interval\_final | string\_agg\_finalfn | string\_agg\_transfn | timestamp\_list\_agg\_noarg2\_transfn | timestamp\_list\_agg\_transfn | timestamptz\_list\_agg\_noarg2\_transfn |
| timestamptz\_list\_agg\_transfn | checksumtext\_agg\_transfn | - | - | - | - | - |
| json\_agg\_finalfn | json\_agg\_transfn | json\_object\_agg\_finalfn | json\_object\_agg\_transfn | - | - | - |

* 哈希内部功能函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| hashbeginscan | hashbuild | hashbuildempty | hashbulkdelete | hashcostestimate | hashendscan | hashgetbitmap |
| hashgettuple | hashinsert | hashmarkpos | hashmerge | hashrescan | hashrestrpos | hashvacuumcleanup |
| hashvarlena | - | - | - | - | - | - |

* Btree索引内部功能函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| cbtreebuild | cbtreecanreturn | cbtreecostestimate | cbtreegetbitmap | cbtreegettuple | btbeginscan | btbuild |
| btbuildempty | btbulkdelete | btcanreturn | btcostestimate | btendscan | btfloat4sortsupport | btfloat8sortsupport |
| btgetbitmap | btgettuple | btinsert | btint2sortsupport | btint4sortsupport | btint8sortsupport | btmarkpos |
| btmerge | btnamesortsupport | btrescan | btrestrpos | bttextsortsupport | btvacuumcleanup | cbtreeoptions |

* GiST索引内部功能函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| gist\_box\_compress | gist\_box\_consistent | gist\_box\_decompress | gist\_box\_penalty | gist\_box\_picksplit | gist\_box\_same | gist\_box\_union |
| gist\_circle\_compress | gist\_circle\_consistent | gist\_point\_compress | gist\_point\_consistent | gist\_point\_distance | gist\_poly\_compress | gist\_poly\_consistent |
| gistbeginscan | gistbuild | gistbuildempty | gistbulkdelete | gistcostestimate | gistendscan | gistgetbitmap |
| gistinsert | gistmarkpos | gistmerge | gistrescan | gistrestrpos | gistvacuumcleanup | range\_gist\_compress |
| range\_gist\_decompress | range\_gist\_penalty | range\_gist\_picksplit | range\_gist\_same | range\_gist\_union | spg\_kd\_choose | spg\_kd\_config |
| spg\_kd\_picksplit | spg\_quad\_choose | spg\_quad\_config | spg\_quad\_inner\_consistent | spg\_quad\_leaf\_consistent | spg\_quad\_picksplit | spg\_text\_choose |
| spg\_text\_inner\_consistent | spg\_text\_leaf\_consistent | spg\_text\_picksplit | spgbeginscan | spgbuild | spgbuildempty | spgbulkdelete |
| spgcostestimate | spgendscan | spggetbitmap | spggettuple | spginsert | spgmarkpos | spgmerge |
| spgrestrpos | spgvacuumcleanup | - | - | - | - | - |

* Gin索引内部功能函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| gin\_cmp\_prefix | gin\_extract\_tsquery | gin\_tsquery\_consistent | gin\_tsquery\_triconsistent | ginarrayconsistent | ginarrayextract | ginarraytriconsistent |
| ginbeginscan | ginbuild | ginbuildempty | ginbulkdelete | gincostestimate | ginendscan | gingetbitmap |
| gininsert | ginmarkpos | ginmerge | ginqueryarrayextract | ginrescan | ginrestrpos | ginvacuumcleanup |
| cginbuild | cgingetbitmap | - | - | - | - | - |
| gin\_compare\_jsonb | gin\_consistent\_jsonb | gin\_consistent\_jsonb\_hash | gin\_extract\_jsonb  gin\_extract\_jsonb\_hash | cginbuild | gin\_extract\_jsonb\_query | gin\_triconsistent\_jsonb |

* Psort索引内部函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| psortbuild | psortcanreturn | psortcostestimate | psortgetbitmap | psortgettuple |

* Ubtree索引内部函数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ubtbeginscan | ubtbuild | ubtbuildempty | ubtbulkdelete | ubtcanreturn |
| ubtcostestimate | ubtendscan | ubtgetbitmap | ubtgettuple | ubtinsert |
| ubtmarkpos | ubtmerge | ubtoptions | ubtrescan | ubtrestrpos |
| ubtvacuumcleanup |  |  |  |  |

* plpgsql内部函数

plpgsql\_inline\_handler

* 集合相关内部函数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| array\_indexby\_delete | array\_indexby\_length | array\_integer\_deleteidx | array\_integer\_exists | array\_integer\_first | array\_integer\_last |
| array\_integer\_next | array\_integer\_prior | array\_varchar\_deleteidx | array\_varchar\_exists | array\_varchar\_first | array\_varchar\_last |
| array\_varchar\_next | array\_varchar\_prior | - | - | - | - |

* 外表相关内部函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dist\_fdw\_handler | roach\_handler | streaming\_fdw\_handler | dist\_fdw\_validator | file\_fdw\_handler | file\_fdw\_validator | log\_fdw\_handler |

* 主数据库节点远程读取备数据库节点数据页辅助函数
* gs\_read\_block\_from\_remote用于读取非段页式表文件的页面。默认只有初始化用户可以查看，其余用户需要赋权后才可以使用。
* pg\_read\_binary\_file\_blocks用于读取数据，非压缩表返回实际数据，压缩表返回压缩后的数据。默认只有初始化用户可以查看，其余用户需要赋权后才可以使用。
* gs\_read\_segment\_block\_from\_remote用于读取段页式表文件的页面。默认只有初始化用户可以查看，其余用户需要赋权后才可以使用。
* 主数据库节点远程读取备数据库节点数据文件辅助函数
* gs\_read\_file\_from\_remote用于读取指定的文件。gs\_repair\_file利用gs\_read\_file\_size\_from\_remote函数获取文件大小后，依赖这个函数将远端文件逐段读取。默认只有初始化用户可以查看，其余用户需要赋权后才可以使用。
* gs\_read\_file\_size\_from\_remote用于读取指定文件的大小。用于读取指定文件的大小，gs\_repair\_file函数修复文件时，要先获取远端关于这个文件的大小，用于校验本地文件缺失的文件信息，然后将缺失的文件逐个修复。默认只有初始化用户可以查看，其余用户需要赋权后才可以使用。
* 账本数据库函数
* get\_dn\_hist\_relhash
* AI特性函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| create\_snapshot | create\_snapshot\_internal | prepare\_snapshot\_internal | prepare\_snapshot | manage\_snapshot\_internal | archive\_snapshot | publish\_snapshot |
| purge\_snapshot\_internal | purge\_snapshot | sample\_snapshot |  |  |  |  |

* PKG\_SERVICE函数

isubmit\_on\_nodes、submit\_on\_nodes

* 其他函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| to\_tsvector\_for\_batch | value\_of\_percentile | session\_context | bind\_variable | job\_update | job\_cancel | job\_finish |
| similar\_escape | table\_skewness （不可用） | timetz\_text | time\_text | reltime\_text | abstime\_text | \_pg\_keysequal |
| analyze\_query  (不可用） | analyze\_workload （不可用） | ssign\_table\_type | gs\_comm\_proxy\_thread\_status | gs\_txid\_oldestxmin | pg\_cancel\_session | pg\_stat\_segment\_space\_info |
| remote\_segment\_space\_info | set\_cost\_params | set\_weight\_params | start\_collect\_workload | tdigest\_in | tdigest\_merge | tdigest\_merge\_to\_one |
| tdigest\_mergep | tdigest\_out | pg\_get\_delta\_info | disable\_conn | - | - | - |

#### Global SysCache特性函数

* gs\_gsc\_table\_detail(database\_id dafault NULL, rel\_id default NULL)
* 描述：查看数据库里全局系统缓存的表元数据。调用该函数的用户需要具有SYSADMIN权限。
* 参数：指定需要查看全局系统缓存的数据库和表，database\_id默认值NULL或者-1表示所有的数据库，0表示共享表，其他数字表示指定数据库及共享表，rel\_id表示指定表的oid，默认值NULL或者-1表示所有的表，其他值表示指定的表，database\_id不存在会报错，rel\_id不存在结果为空。
* 返回值类型：Tuple
* 示例:

select \* from gs\_gsc\_table\_detail(-1) limit 1;

database\_oid | database\_name | reloid | relname | relnamespace | reltype | reloftype | relowner | relam | relfilenode | reltablespace | relhasindex | relisshared | relkind | relnatts | relhasoids | relhaspkey | parttype | tdhasuids | attnames | extinfo

--------------+---------------+--------+-------------------------+--------------+---------+-----------+----------+-------+-------------+---------------+-------------+-------------+---------+----------+------------+------------+----------+-----------+-----------+---------

0 | | 2676 | pg\_authid\_rolname\_index | 11 | 0 | 0 | 10 | 403 | 0 | 1664 | f | t | i | 1 | f | f | n | f | 'rolname' |

(1 row)

* gs\_gsc\_catalog\_detail(database\_id dafault NULL, rel\_id default NULL)
* 描述：查看数据库里全局系统缓存的系统表行信息。调用该函数的用户需要具有SYSADMIN权限。
* 参数：指定需要查看全局系统缓存的数据库和表，database\_id 默认值NULL或者-1表示所有的数据库，0表示共享表，其他数字表示指定数据库及共享表， rel\_id表示指定表的id，仅包含所有有系统缓存的系统表，默认值NULL或者-1表示所有的表，database\_id不存在会报错，rel\_id不存在结果为空。
* 返回值类型：Tuple
* 示例：

vastbase=#

select \* from gs\_gsc\_catalog\_detail(16574, 1260);

database\_id | database\_name | rel\_id | rel\_name | cache\_id | self | ctid | infomask | infomask2 | hash\_value | refcount

-------------+---------------+--------+-----------+----------+--------+--------+----------+-----------+------------+----------

0 | | 1260 | pg\_authid | 10 | (0, 9) | (0, 9) | 10507 | 26 | 531311568 | 10

0 | | 1260 | pg\_authid | 11 | (0, 4) | (0, 4) | 2313 | 26 | 365368336 | 1

0 | | 1260 | pg\_authid | 11 | (0, 9) | (0, 9) | 10507 | 26 | 3911517328 | 10

0 | | 1260 | pg\_authid | 11 | (0, 7) | (0, 7) | 2313 | 26 | 1317799983 | 1

0 | | 1260 | pg\_authid | 11 | (0, 5) | (0, 5) | 2313 | 26 | 3664347448 | 1

0 | | 1260 | pg\_authid | 11 | (0, 1) | (0, 1) | 2313 | 26 | 276477273 | 1

0 | | 1260 | pg\_authid | 11 | (0, 3) | (0, 3) | 2313 | 26 | 2465837659 | 1

0 | | 1260 | pg\_authid | 11 | (0, 8) | (0, 8) | 2313 | 26 | 3205288035 | 1

0 | | 1260 | pg\_authid | 11 | (0, 6) | (0, 6) | 2313 | 26 | 131811687 | 1

0 | | 1260 | pg\_authid | 11 | (0, 2) | (0, 2) | 2313 | 26 | 1226484587 | 1

(10 rows)

* gs\_gsc\_clean(database\_id dafault NULL)
* 描述：清理global syscache的缓存，需要注意，正在使用中的数据不会被清理。调用该函数的用户需要具有SYSADMIN权限。
* 参数：指定需要清理全局系统缓存的数据库，默认值NULL或者-1表示清理所有的数据库全局系统缓存，0表示只淘汰共享表的全局系统缓存，其他数字表示淘汰清理指定数据库以及共享表的全局系统缓存，database\_id不存在会报错。
* 返回值类型：bool
* 示例：

vastbase=# select \* from gs\_gsc\_clean();

gs\_gsc\_clean

--------------

t

(1 row)

* gs\_gsc\_dbstat\_info(database\_id default NULL)
* 描述：获取本地节点的GSC的内存统计信息，包括tuple、relation、partition的缓存查询，命中，加载、失效、占用空间信息，DB级别的淘汰信息，线程引用信息，内存占用信息。可以用于定位性能问题，例如当发现hits/searches数组远小于1时，可能是global\_syscache\_threshold设置太小，导致查询命中率下降。调用该函数的用户需要具有SYSADMIN权限。
* 参数：指定需要查看的数据库全局系统缓存统计信息，NULL或者-1表示查看所有的数据库，0表示只查看共享表信息，其他数字表示查看指定的数据库和共享表的信息。不合法的输入值，databse\_id不存在会报错。
* 返回值类型：Tuple
* 示例：

vastbase=# select \* from gs\_gsc\_dbstat\_info();

database\_id | database\_name | tup\_searches | tup\_hits | tup\_miss | tup\_count | tup\_dead | tup\_memory | rel\_searches | rel\_hits | rel\_mis

s | rel\_count | rel\_dead | rel\_memory | part\_searches | part\_hits | part\_miss | part\_count | part\_dead | part\_memory | total\_memory | swa

pout\_count | refcount

-------------+---------------+--------------+----------+----------+-----------+----------+------------+--------------+----------+--------

--+-----------+----------+------------+---------------+-----------+-----------+------------+-----------+-------------+--------------+----

-----------+----------

0 | | 300 | 235 | 31 | 22 | 2 | 9752 | 598 | 108 | 1

8 | 18 | 0 | 77720 | 0 | 0 | 0 | 0 | 0 | 0 | 752912 |

0 | 0

16574 | postgres | 3368 | 2289 | 329 | 273 | 0 | 92593 | 1113 | 524 | 4

8 | 48 | 0 | 340456 | 0 | 0 | 0 | 0 | 0 | 0 | 4124792 |

0 | 10

(2 rows)

#### 数据损坏检测修复函数

* gs\_verify\_data\_file(verify\_segment bool)
* 描述：校验当前实例当前库是否存在文件丢失的情况。校验只包括数据表主文件是否有中间段丢失的情况。默认参数是false，表示不校验段页式表数据文件。参数设置为true时仅校验段页式表文件。默认只有初始化用户、具有sysadmin属性的用户以及在运维模式下具有运维管理员属性的用户可以查看，其余用户需要赋权后才可以使用。
* 返回的结果：
* 非段页式表：rel\_oid和rel\_name是对应文件的表oid和表名，miss\_file\_path表示丢失文件的相对路径。
* 段页式表：因所有表存放在相同文件中，所以rel\_oid和rel\_name无法显示具体表的信息。对于段页式表，如果第一个文件损坏，不会检查出后面的.1 .2等文件。例如3、3.1、3.2损坏，只能检查出3损坏。当段页式文件不足5个时，使用函数检测时，未生成的文件也会校验出来，例如只有1和2文件，校验段页式时，也会检测出3，4，5文件。以下示例，第一个是校验非段页式表的示例，第二是校验段页式表的示例。
* 参数说明：
* verify\_segment

指定文件校验的范围。false校验非段页式表；true校验段页式表。

取值范围：true和false，默认是false。

* 返回值类型：record
* 示例：
* 校验非段页式表

vastbase=# select \* from gs\_verify\_data\_file();

node\_name | rel\_oid | rel\_name | miss\_file\_path

------------------+---------+--------------+------------------

dn\_6001\_6002\_6003 | 16554 | test | base/16552/24745

* 校验段页式表

vastbase=# select \* from gs\_verify\_data\_file(true);

node\_name | rel\_oid | rel\_name | miss\_file\_path

-------------------+---------+----------+----------------

dn\_6001\_6002\_6003 | 0 | none | base/16573/2

* gs\_repair\_file(tableoid Oid，path text, timeout int)
* 描述：根据传入的参数修复文件，仅支持有正常主备连接的主DN使用。参数依据gs\_verify\_data\_file函数返回的oid和路径填写。段页式表tableoid赋值为0到4,294,967,295的任意值（内部校验根据文件路径判断是否是段页式表文件，段页式表文件则不使用tableoid）。修复成功返回值为true，修复失败会显示具体失败原因。默认只有在主DN节点上，使用初始化用户、具有sysadmin属性的用户以及在运维模式下具有运维管理员属性的用户可以查看，其余用户需要赋权后才可以使用。
* fig: **注意：**
  1. 当DN实例上存在文件损坏时，进行升主会校验出错，报PANIC退出无法升主，为正常现象。可在其他DN升主后，通过备DN自动修复进行修复。
  2. 当文件存在但是大小为0时，此时不会去修复该文件，若想要修复该文件，需要将为0的文件删除后再修复。
  3. 删除文件需要等文件fd自动关闭后再修复，人工操作可以执行重启进程、主备切换命令。

参数说明：

* tableoid
* 要修复的文件对应的表oid，依据gs\_verify\_data\_file函数返回的列表中rel\_oid一列填写。
* 取值范围： oid，0 - 4294967295。注意：输入负值等都会被强制转成非负整数类型。
* path
* 需要修复的文件路径，依据gs\_verify\_data\_file函数返回的列表中miss\_file\_path一列填写。
* 取值范围：字符串。
* timeout
* 等待备DN回放的时长，修复文件需要等待备DN回放到当前主DN对应的位置，根据备DN回放所需时长设定。
* 取值范围：60s - 3600s。

返回值类型：bool

示例：

vastbase=# select \* from gs\_repair\_file(16554,'base/16552/24745',360);

gs\_repair\_file

----------------

t

* local\_bad\_block\_info()
* 描述：显示本实例页面损坏的情况。从磁盘读取页面，发现页面CRC校验失败时进行记录。默认只有初始化用户、具有sysadmin属性的用户、具有监控管理员属性的用户以及在运维模式下具有运维管理员属性的用户、以及监控用户可以查看，其余用户需要赋权后才可以使用。
* 显示信息：file\_path是损坏文件的相对路径，如果是段页式表，则显示的是逻辑信息，不是实际的物理文件信息。block\_num是该文件损坏的具体页面号，页面号从0开始。check\_time表示发现页面损坏的时间。repair\_time表示修复页面的时间。
* 返回值类型：record
* 示例：

vastbase=# select \* from local\_bad\_block\_info();

node\_name | spc\_node | db\_node | rel\_node| bucket\_node | fork\_num | block\_num | file\_path | check\_time | repair\_time

-----------------+-------+--------+--------+--------------+----------+-----------+-----------------+--------------------------+-------------------------------

dn\_6001\_6002\_6003| 1663 | 16552 | 24745 | -1 | 0 | 0 | base/16552/24745 | 2022-01-13 20:19:08.385004+08 | 2022-01-13 20:19:08.407314+08

* local\_clear\_bad\_block\_info()
* 描述：清理local\_bad\_block\_info中已修复页面的数据，也就是repair\_time不为空的信息。默认只有初始化用户、具有sysadmin属性的用户以及在运维模式下具有运维管理员属性的用户、以及监控用户可以查看，其余用户需要赋权后才可以使用。
* 返回值类型：bool
* 示例：

vastbase=# select \* from local\_clear\_bad\_block\_info();

result

--------

t

* gs\_verify\_and\_tryrepair\_page (path text, blocknum oid, verify\_mem bool, is\_segment bool)
* 描述：校验本实例指定页面的情况。默认只有在主DN节点上，使用初始化用户、具有sysadmin属性的用户以及在运维模式下具有运维管理员属性的用户可以查看，其余用户需要赋权后才可以使用。
* 返回的结果信息：disk\_page\_res表示磁盘上页面的校验结果；mem\_page\_res表示内存中页面的校验结果；is\_repair表示在校验的过程中是否触发修复功能，t表示已修复，f表示未修复。
* 注意：当DN实例上存在页面损坏时，进行升主会校验出错，报PANIC退出无法升主，为正常现象。可在其他DN升主后，通过备DN自动修复进行修复。
* 参数说明：
* path

损坏文件的路径。依据local\_bad\_block\_info中file\_path一列填 写。

取值范围：字符串。

* blocknum

损坏文件的页号。依据local\_bad\_block\_info中block\_num一列填 写。

取值范围：Oid，0 - 4294967295。注意：输入负值等都会被强制 转成非负整数类型。

* verify\_mem

指定是否校验内存中的指定页面。设定为false时，只校验磁盘上 的页面。设置为true时，校验内存中的页面和磁盘上的页面。如 果发现磁盘上页面损坏，会将内存中的页面做一个基本信息校验 刷盘，修复磁盘上页面。如果校验内存页面时发现页面不在内存 中，会经内存接口读取磁盘上的页面。此过程中如果磁盘页面有 问题，则会触发远程读自动修复功能。

取值范围：bool，true和false。

* is\_segment

是否是段页式表。根据local\_bad\_block\_info中的bucket\_node列 值决定。如果bucket\_node为-1时，表示不是段页式表，将i s\_segment设置为false；非-1的情况将is\_segment设置为true。

取值范围：bool，true和false。

返回值类型：record

* 示例：

vastbase=# select \* from gs\_verify\_and\_tryrepair\_page('base/16552/24745',0,false,false);

node\_name | path | blocknum | disk\_page\_res | mem\_page\_res | is\_repair

------------------+------------------+------------+-----------------------------+---------------+----------

dn\_6001\_6002\_6003 | base/16552/24745 | 0 | page verification succeeded.| | f

* gs\_repair\_page(path text, blocknum oid, is\_segment bool, timeout int)
* 描述：修复本实例指定页面，仅支持有正常主备连接的主DN使用。页面修复成功返回true，修复过程中出错会有报错信息提示。默认只有在主DN节点上，使用初始化用户、具有sysadmin属性的用户以及在运维模式下具有运维管理员属性的用户可以查看，其余用户需要赋权后才可以使用。
* 注意：当DN实例上存在页面损坏时，进行升主会校验出错，报PANIC退出无法升主，为正常现象。可在其他DN升主后，通过备DN自动修复进行修复。
* 参数说明：
* path

损坏页面的路径。根据local\_bad\_block\_info中file\_path一列设 置，或者是gs\_verify\_and\_tryrepair\_page函数中path一列设置。

取值范围：字符串。

* blocknum

损坏页面的页面号。根据local\_bad\_block\_info中block\_num一列 设置，或者是gs\_verify\_and\_tryrepair\_page函数中blocknum一 列设置。

取值范围：Oid，0 - 4294967295。注意：输入负值等都会被强制 转成非负整数类型。

* is\_segment

是否是段页式表。根据local\_bad\_block\_info中的bucket\_node列 值决定，如果bucket\_node为-1时，表示不是段页式表，将 is\_segment设置为false；非-1的情况将is\_segment设置为 true。

取值范围：bool，true或者false。

* timeout

等待备DN回放的时长。修复页面需要等待备DN回放到当前主 DN对应的位置，根据备DN回放所需时长设定。

取值范围：60s - 3600s。

返回值类型：bool

* 示例：

vastbase=# select \* from gs\_repair\_page('base/16552/24745',0,false,60);

result

--------

t

#### 废弃函数

Vastbase中下列函数在最新版本中已废弃：

|  |  |  |
| --- | --- | --- |
| gs\_wlm\_get\_session\_info | gs\_wlm\_get\_user\_session\_info | pgxc\_get\_csn |
| pgxc\_get\_stat\_dirty\_tables | pgxc\_get\_thread\_wait\_status | pgxc\_gtm\_snapshot\_status |
| pgxc\_is\_committed | pgxc\_lock\_for\_backup | pgxc\_lock\_for  \_sp\_database |
| pgxc\_lock\_for\_transfer | pgxc\_log\_comm\_status | pgxc\_max\_datanode\_size |
| pgxc\_node\_str | pgxc\_pool\_check | pgxc\_pool  \_connection\_status |
| pgxc\_pool\_reload | pgxc\_prepared\_xact | pgxc\_snapshot\_status |
| pgxc\_stat\_dirty\_tables | pgxc\_unlock\_for  \_sp\_database | pgxc\_unlock\_  for\_transfer |
| pgxc\_version | array\_extend | prepare\_statement\_status |
| remote\_rto\_stat | dbe\_perf.global\_slow  \_query\_info | dbe\_perf.global\_slow  \_query\_info\_bytime |
| dbe\_perf.global  \_slow\_query\_history | pg\_stat\_get\_pooler  \_status | pg\_stat\_get\_wlm  \_node\_resource\_info |
| pg\_stat\_get\_wlm  \_session\_info\_internal | DBE\_PERF.get\_wlm  \_controlgroup\_ng\_config() | DBE\_PERF.get\_wlm  \_user\_resource\_runtime() |
| global\_space\_shrink | pg\_pool\_validate | gs\_stat\_ustore |
| table\_skewness(text) | table\_skewness(text, text, text) | - |

#### HashFunc函数

* hash\_array(anyarray)
* 描述：数组哈希，将数组的元素通过哈希函数得到结果，并返回合并结果。
* 参数：数据类型为anyarray。
* 返回值类型：integer
* 示例：

vastbase=# select hash\_array(ARRAY[[1,2,3],[1,2,3]]);

hash\_array

------------

-382888479

(1 row)

* hash\_numeric(numeric)
* 描述：计算Numeric类型的数据的hash值。
* 参数：Numeric类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hash\_numeric(30);

hash\_numeric

--------------

-282860963

(1 row)

* hash\_range(anyrange)
* 描述：计算range的哈希值。
* 参数：anyrange类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hash\_range(numrange(1.1,2.2));

hash\_range

------------

683508754

(1 row)

* hashbpchar(character)
* 描述：计算bpchar的哈希值。
* 参数：character类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hashbpchar('hello');

hashbpchar

-------------

-1870292951

(1 row)

* hashchar(char)
* 描述：char和布尔数据转换为哈希值。
* 参数：char类型的数据或者bool类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hashbpchar('hello');

hashbpchar

-------------

-1870292951

(1 row)

vastbase=# select hashchar('true');

hashchar

------------

1686226652

(1 row)

* hashenum(anyenum)
* 描述：枚举类型转哈希值。
* 参数：anyenum类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# CREATE TYPE b1 AS ENUM('good', 'bad', 'ugly');

CREATE TYPE

vastbase=# call hashenum('good'::b1);

hashenum

------------

1821213359

(1 row)

* hashfloat4(real)
* 描述：float4转哈希值。
* 参数：real类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hashfloat4(12.1234);

hashfloat4

------------

1398514061

(1 row)

* hashfloat8(double precision)
* 描述：float8转哈希值。
* 参数：double precision类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hashfloat8(123456.1234);

hashfloat8

------------

1673665593

(1 row)

* hashinet(inet)
* 描述：支持inet / cidr上的哈希索引的功能。返回传入inet的hash值。
* 参数：inet类型的数据。
* 返回值类型：integer
* 示例：

vastbase=# select hashinet('127.0.0.1'::inet);

hashinet

-------------

-1435793109

(1 row)

* hashint1(tinyint)
* 描述：INT1转哈希值。
* 参数：tinyint类型的数据。
* 返回值类型：uint32
* 示例：

vastbase=# select hashint1(20);

hashint1

-------------

-2014641093

(1 row)

* hashint2(smallint)
* 描述：INT2转哈希值。
* 参数：smallint类型的数据。
* 返回值类型：uint32
* 示例：

vastbase=# select hashint2(20000);

hashint2

------------

-863179081

(1 row)

### 表达式

#### 简单表达式

**逻辑表达式**

逻辑表达式的操作符和运算规则，请参见：开发者指南->SQL语法参考->SQL基本要素->函数和操作符->逻辑操作符。

**比较表达式**

常用的比较操作符，请参见：开发者指南->SQL语法参考->SQL基本要素->函数和操作符->比较操作符。

除比较操作符外，还可以使用以下句式结构：

* BETWEEN操作符
* a BETWEEN x AND y等效于a >= x AND a <= y
* a NOT BETWEEN x AND y等效于a < x OR a > y
* 检查一个值是不是null，可使用：
* expression IS NULL
* expression IS NOT NULL
* 或者与之等价的句式结构，但不是标准的：
* expression ISNULL
* expression NOTNULL
* fig: **须知：**
* 不要写expression=NULL或expression<>(!=)NULL，因为NULL代表一个未知的值，不能通过该表达式判断两个未知值是否相等。
* is distinct from/is not distinct from
* is distinct from
* A和B的数据类型、值不完全相同时为true。
* A和B的数据类型、值完全相同时为false。
* 将空值视为相同。
* is not distinct from
* A和B的数据类型、值不完全相同时为false。
* A和B的数据类型、值完全相同时为true。
* 将空值视为相同。

**伪列**

ROWNUM

ROWNUM是一个伪列，它返回一个数字，表示从查询中获取结果的行编号。第一行的ROWNUM为1，第二行的为2，依此类推。

ROWNUM的返回类型为numeric。ROWNUM可以用于限制查询返回的总行数，例如下面语句限制查询从Students表中返回最多10条记录。

select \* from Students where rownum <= 10;

**示例**

vastbase=# SELECT 2 BETWEEN 1 AND 3 AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 2 >= 1 AND 2 <= 3 AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 2 NOT BETWEEN 1 AND 3 AS RESULT;

result

----------

f

(1 row)

vastbase=# SELECT 2 < 1 OR 2 > 3 AS RESULT;

result

----------

f

(1 row)

vastbase=# SELECT 2+2 IS NULL AS RESULT;

result

----------

f

(1 row)

vastbase=# SELECT 2+2 IS NOT NULL AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 2+2 ISNULL AS RESULT;

result

----------

f

(1 row)

vastbase=# SELECT 2+2 NOTNULL AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 2+2 IS DISTINCT FROM NULL AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 2+2 IS NOT DISTINCT FROM NULL AS RESULT;

result

----------

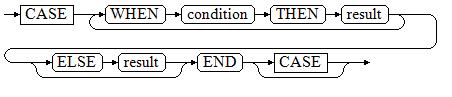
f

(1 row)

#### 条件表达式

在执行SQL语句时，可通过条件表达式筛选出符合条件的数据。

条件表达式主要有以下几种：

* CASE
* CASE表达式是条件表达式，类似于其他编程语言中的CASE语句。
* CASE表达式的语法图请参考图1。
* **图 1** case::=
* 
* CASE子句可以用于合法的表达式中。condition是一个返回BOOLEAN数据类型的表达式：
* 如果结果为真，CASE表达式的结果就是符合该条件所对应的result。
* 如果结果为假，则以相同方式处理随后的WHEN或ELSE子句。
* 如果各WHEN condition都不为真，表达式的结果就是在ELSE子句执行的result。如果省略了ELSE子句且没有匹配的条件，结果为NULL。
* 示例：

vastbase=#CREATE TABLE tpcds.case\_when\_t1(CW\_COL1 INT);

vastbase=#INSERT INTO tpcds.case\_when\_t1 VALUES (1), (2), (3);

vastbase=#SELECT \* FROM tpcds.case\_when\_t1;

cw\_col1

---------

1

2

3

(3 rows)

vastbase=#SELECT CW\_COL1, CASE WHEN CW\_COL1=1 THEN 'one' WHEN CW\_COL1=2 THEN 'two' ELSE 'other' END FROM tpcds.case\_when\_t1 ORDER BY 1;

cw\_col1 | case

---------+-------

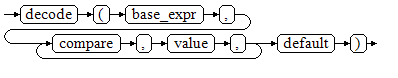
1 | one

2 | two

3 | other

(3 rows)

vastbase=#DROP TABLE tpcds.case\_when\_t1;

* DECODE
* DECODE的语法图请参考图2.
* **图 2** decode::=
* 
* 将表达式base\_expr与后面的每个compare(n) 进行比较，如果匹配返回相应的value(n)。如果没有发生匹配，则返回default。
* 示例请参见条件表达式函数 (参见：开发者指南->SQL语法参考->函数与操作符->条件表达式函数 章节)。

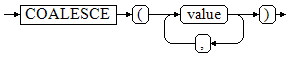
vastbase=#SELECT DECODE('A','A',1,'B',2,0);

case

------

1

(1 row)

* COALESCE
* COALESCE的语法图请参见图3。
* **图 3** coalesce::=
* 
* COALESCE返回它的第一个非NULL的参数值。如果参数都为NULL，则返回NULL。它常用于在显示数据时用缺省值替换NULL。和CASE表达式一样，COALESCE只计算用来判断结果的参数，即在第一个非空参数右边的参数不会被计算。
* 示例：

vastbase=#CREATE TABLE tpcds.c\_tabl(description varchar(10), short\_description varchar(10), last\_value varchar(10)) ;

vastbase=#INSERT INTO tpcds.c\_tabl VALUES('abc', 'efg', '123');

vastbase=#INSERT INTO tpcds.c\_tabl VALUES(NULL, 'efg', '123');

vastbase=#INSERT INTO tpcds.c\_tabl VALUES(NULL, NULL, '123');

vastbase=#SELECT description, short\_description, last\_value, COALESCE(description, short\_description, last\_value) FROM tpcds.c\_tabl ORDER BY 1, 2, 3, 4;

description | short\_description | last\_value | coalesce

-------------+-------------------+------------+----------

abc | efg | 123 | abc

| efg | 123 | efg

| | 123 | 123

(3 rows)

vastbase=#DROP TABLE tpcds.c\_tabl;

如果description不为NULL，则返回description的值，否则计算下一个参数short\_description；如果short\_description不为NULL，则返回short\_description的值，否则计算下一个参数last\_value；如果last\_value不为NULL，则返回last\_value的值，否则返回（none）。

vastbase=#SELECT COALESCE(NULL,'Hello World');

coalesce

---------------

Hello World

(1 row)

* NULLIF
* NULLIF的语法图请参见图4。
* **图 4** nullif::=
* IMG_256
* 只有当value1和value2相等时，NULLIF才返回NULL。否则它返回value1。
* 示例：

vastbase=#CREATE TABLE tpcds.null\_if\_t1 (

NI\_VALUE1 VARCHAR(10),

NI\_VALUE2 VARCHAR(10)

);

vastbase=#INSERT INTO tpcds.null\_if\_t1 VALUES('abc', 'abc');

vastbase=#INSERT INTO tpcds.null\_if\_t1 VALUES('abc', 'efg');

vastbase=#SELECT NI\_VALUE1, NI\_VALUE2, NULLIF(NI\_VALUE1, NI\_VALUE2) FROM tpcds.null\_if\_t1 ORDER BY 1, 2, 3;

ni\_value1 | ni\_value2 | nullif

-----------+-----------+--------

abc | abc |

abc | efg | abc

(2 rows)

vastbase=#DROP TABLE tpcds.null\_if\_t1;

如果value1等于value2则返回NULL，否则返回value1。

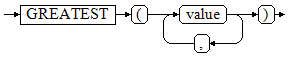
vastbase=#SELECT NULLIF('Hello','Hello World');

nullif

--------

Hello

(1 row)

* GREATEST（最大值），LEAST（最小值）
* GREATEST的语法图请参见图5。
* **图 5** greatest::=
* 
* 从一个任意数字表达式的列表里选取最大的数值。

vastbase=#SELECT greatest(9000,155555,2.01);

greatest

----------

155555

(1 row)

LEAST的语法图请参见图6。

**图 6** least::=



从一个任意数字表达式的列表里选取最小的数值。

以上的数字表达式必须都可以转换成一个普通的数据类型，该数据类型将是结果类型。

列表中的NULL值将被忽略。只有所有表达式的结果都是NULL的时候，结果才是NULL。

vastbase=#SELECT least(9000,2);

least

-------

2

(1 row)

示例请参见：开发者指南->SQL语法参考->SQL基本要素->函数和操作符->条件表达式函数章节。

* NVL
* NVL的语法图请参见图7。
* **图 7** nvl::=
* IMG_256
* 如果value1为NULL则返回value2，如果value1非NULL，则返回value1。
* 示例：

vastbase=#SELECT nvl(null,1);

nvl

-----

1

(1 row)

vastbase=#SELECT nvl ('Hello World' ,1);

nvl

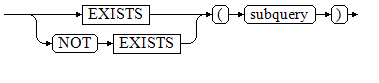
---------------

Hello World

(1 row)

#### 子查询表达式

子查询表达式主要有以下几种：

* EXISTS/NOT EXISTS
* EXISTS/NOT EXISTS的语法图请参见图1。
* **图 1** EXISTS/NOT EXISTS::=
* 
* EXISTS的参数是一个任意的SELECT语句，或者说子查询。系统对子查询进行运算以判断它是否返回行。如果它至少返回一行，则EXISTS结果就为“真”；如果子查询没有返回任何行，EXISTS的结果是“假”。
* 这个子查询通常只是运行到能判断它是否可以生成至少一行为止，而不是等到全部结束。
* 示例：

vastbase=# SELECT sr\_reason\_sk,sr\_customer\_sk FROM tpcds.store\_returns WHERE EXISTS (SELECT d\_dom FROM tpcds.date\_dim WHERE d\_dom = store\_returns.sr\_reason\_sk and sr\_customer\_sk <10);

sr\_reason\_sk | sr\_customer\_sk

--------------+----------------

13 | 2

22 | 5

17 | 7

25 | 7

3 | 7

31 | 5

7 | 7

14 | 6

20 | 4

5 | 6

10 | 3

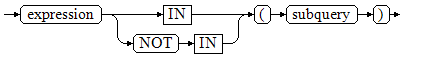
1 | 5

15 | 2

4 | 1

26 | 3

(15 rows)

* IN/NOT IN
* IN/NOT IN的语法请参见图2。
* **图 2** IN/NOT IN::=
* 
* 右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式对子查询结果的每一行进行一次计算和比较。如果找到任何相等的子查询行，则IN结果为“真”。如果没有找到任何相等行，则结果为“假”（包括子查询没有返回任何行的情况）。
* 表达式或子查询行里的NULL遵照SQL处理布尔值和NULL组合时的规则。如果两个行对应的字段都相等且非空，则这两行相等；如果任意对应字段不等且非空，则这两行不等；否则结果是未知（NULL）。如果每一行的结果都是不等或NULL ，并且至少有一个NULL ，则IN的结果是NULL 。
* 示例：

vastbase=# SELECT sr\_reason\_sk,sr\_customer\_sk FROM tpcds.store\_returns WHERE sr\_customer\_sk IN (SELECT d\_dom FROM tpcds.date\_dim WHERE d\_dom < 10);

sr\_reason\_sk | sr\_customer\_sk

--------------+----------------

10 | 3

26 | 3

22 | 5

31 | 5

1 | 5

32 | 5

32 | 5

4 | 1

15 | 2

13 | 2

33 | 4

20 | 4

33 | 8

5 | 6

14 | 6

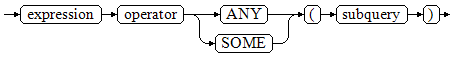
17 | 7

3 | 7

25 | 7

7 | 7

(19 rows)

* ANY/SOME
* ANY/SOME的语法图请参见图3。
* **图 3** any/some::=
* 
* 右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较，其结果必须是布尔值。如果至少获得一个真值，则ANY结果为“真”。如果全部获得假值，则结果是“假”（包括子查询没有返回任何行的情况）。SOME是ANY的同义词。IN与ANY可以等效替换 。
* 示例：

vastbase=# SELECT sr\_reason\_sk,sr\_customer\_sk FROM tpcds.store\_returns WHERE sr\_customer\_sk < ANY (SELECT d\_dom FROM tpcds.date\_dim WHERE d\_dom < 10);

sr\_reason\_sk | sr\_customer\_sk

--------------+----------------

26 | 3

17 | 7

32 | 5

32 | 5

13 | 2

31 | 5

25 | 7

5 | 6

7 | 7

10 | 3

1 | 5

14 | 6

4 | 1

3 | 7

22 | 5

33 | 4

20 | 4

33 | 8

15 | 2

(19 rows)

* ALL
* ALL的语法请参见图4。
* **图 4** all::=
* all
* 右边是一个圆括弧括起来的子查询，它必须只返回一个字段。左边表达式使用operator对子查询结果的每一行进行一次计算和比较，其结果必须是布尔值。如果全部获得真值，ALL结果为“真”（包括子查询没有返回任何行的情况）。如果至少获得一个假值，则结果是“假”。
* 示例：

vastbase=# SELECT sr\_reason\_sk,sr\_customer\_sk FROM tpcds.store\_returns WHERE sr\_customer\_sk < all(SELECT d\_dom FROM tpcds.date\_dim WHERE d\_dom < 10);

sr\_reason\_sk | sr\_customer\_sk

--------------+----------------

(0 rows)

#### 数组表达式

**IN**

expression **IN** (value [, ...])

右侧括号中的是一个表达式列表。左侧表达式的结果与表达式列表的内容进行比较。如果列表中的内容符合左侧表达式的结果，则IN的结果为true。如果没有相符的结果，则IN的结果为false。

示例如下：

vastbase=# SELECT 8000+500 IN (10000, 9000) AS RESULT;

result

----------

f

(1 row)

如果表达式结果为null，或者表达式列表不符合表达式的条件且右侧表达式列表返回结果至少一处为空，则IN的返回结果为null，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

**NOT IN**

expression **NOT IN** (value [, ...])

右侧括号中的是一个表达式列表。左侧表达式的结果与表达式列表的内容进行比较。如果在列表中的内容没有符合左侧表达式结果的内容，则NOT IN的结果为true。如果有符合的内容，则NOT IN的结果为false。

示例如下：

vastbase=# SELECT 8000+500 NOT IN (10000, 9000) AS RESULT;

result

----------

t

(1 row)

如果查询语句返回结果为空，或者表达式列表不符合表达式的条件且右侧表达式列表返回结果至少一处为空，则NOT IN的返回结果为null，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

fig: **说明：**

在所有情况下X NOT IN Y等价于NOT(X IN Y)。

**ANY/SOME (array)**

expression operator **ANY** (array expression)

expression operator **SOME** (array expression)

vastbase=# SELECT 8000+500 < SOME (array[10000,9000]) AS RESULT;

result

----------

t

(1 row)

vastbase=# SELECT 8000+500 < ANY (array[10000,9000]) AS RESULT;

result

----------

t

(1 row)

右侧括号中的是一个数组表达式，它必须产生一个数组值。左侧表达式的结果使用操作符对数组表达式的每一行结果都进行计算和比较，比较结果必须是布尔值。

* 如果对比结果至少获取一个真值，则ANY的结果为true。
* 如果对比结果没有真值，则ANY的结果为false。
* 如果结果没有真值，并且数组表达式生成至少一个值为null，则ANY的值为NULL，而不是false。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

fig: **说明：**

SOME是ANY的同义词。

**ALL (array)**

expression operator **ALL** (array expression)

右侧括号中的是一个数组表达式，它必须产生一个数组值。左侧表达式的结果使用操作符对数组表达式的每一行结果都进行计算和比较，比较结果必须是布尔值。

* 如果所有的比较结果都为真值（包括数组不含任何元素的情况），则ALL的结果为true。
* 如果存在一个或多个比较结果为假值，则ALL的结果为false。
* 如果数组表达式产生一个NULL数组，则ALL的结果为NULL。如果左边表达式的值为NULL ，则ALL的结果通常也为NULL（某些不严格的比较操作符可能得到不同的结果）。另外，如果右边的数组表达式中包含null元素并且比较结果没有假值，则ALL的结果将是NULL（某些不严格的比较操作符可能得到不同的结果），而不是真。这样的处理方式和SQL返回空值的布尔组合规则是一致的。

vastbase=# SELECT 8000+500 < ALL (array[10000,9000]) AS RESULT;

result

----------

t

(1 row)

#### 行表达式

**语法格式**

row\_constructor operator row\_constructor

* 两边都是一个行构造器，两行值必须具有相同数目的字段，每一行都进行比较，行比较允许使用=、<>、<、<=、>=等操作符，或其中一个相似的语义符。
* =<>和别的操作符使用略有不同。如果两行值的所有字段都是非空并且相等，则认为两行是相等的；如果两行值的任意字段为非空并且不相等，则认为两行是不相等的；否则比较结果是未知的（null）。
* 对于<、<=、>、> =的情况下，行中元素从左到右依次比较，直到遇到一对不相等的元素或者一对为空的元素。如果这对元素中存在至少一个null值，则比较结果是未知的（null），否则这对元素的比较结果为最终的结果。

**示例**

vastbase=# SELECT ROW(1,2,NULL) < ROW(1,3,0) AS RESULT;

result

----------

t

(1 row)

### 事务控制

事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。

**启动事务**

Vastbase通过START TRANSACTION和BEGIN语法启动事务，请参见：开发者指南->SQL语法参考->SQL语法->START TRANSACTION章节和BEGIN章节。

**设置事务**

Vastbase通过SET TRANSACTION或者SET LOCAL TRANSACTION语法设置事务，请参见：开发者指南->SQL语法参考->SQL语法->SET TRANSACTION章节。

**提交事务**

Vastbase通过COMMIT或者END可完成提交事务的功能，即提交事务的所有操作，请参见：开发者指南->SQL语法参考->SQL语法->COMMIT | END章节。

**回滚事务**

回滚是在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销。请参见：开发者指南->SQL语法参考->SQL语法->ROLLBACK章节。

fig: **说明：**

数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，如果其中有一个语句失败，那么整个请求都将会被回滚。

### 系统操作

Vastbase通过SQL语句执行不同的系统操作，比如：设置变量、显示执行计划和垃圾收集等操作。

**设置变量**

设置会话或事务中需要使用的各种参数，请参见：开发者指南->SQL语法参考->SQL语法->SET章节。

**显示执行计划**

显示Vastbase为SQL语句规划的执行计划，请参见：开发者指南->SQL语法参考->SQL语法->EXPLAIN章节。

**事务日志检查点**

预写式日志（WAL）缺省时在事务日志中每隔一段时间放置一个检查点。CHECKPOINT强制立即进行检查，而不是等到下一次调度时的检查点。请参见：开发者指南->SQL语法参考->SQL语法->CHECKPOINT章节。

**垃圾收集**

进行垃圾收集以及可选择的对数据库进行分析。请参见：开发者指南->SQL语法参考->SQL语法->VACUUM章节。

**收集统计信息**

收集与数据库中表内容相关的统计信息。请参见：开发者指南->SQL语法参考->SQL语法->ANALYZE和ANALYSE章节。

**设置当前事务的约束检查模式**

设置当前事务里的约束检查的特性。请参见：开发者指南->SQL语法参考->SQL语法->SET CONSTRAINTS。

**关闭当前数据库节点**

关闭当前数据库节点，请参见：开发者指南->SQL语法参考->SQL语法->SHUTDOWN。

## DML语法一览表

**功能描述**

DML（Data Manipulation Language数据操作语言），用于对数据库表中的数据进行操作。如：插入、更新、查询、删除。

**插入数据**

插入数据是往数据库表中添加一条或多条记录，请参见：开发者指南->SQL语法参考->SQL语法->INSERT。

**修改数据**

修改数据是修改数据库表中的一条或多条记录，请参见：开发者指南->SQL语法参考->SQL语法->UPDATE。

**查询数据**

数据库查询语句SELECT是用于在数据库中检索适合条件的信息，请参见：开发者指南->SQL语法参考->SQL语法->SELECT。

**删除数据**

Vastbase提供了两种删除表数据的语句：删除表中指定条件的数据，请参见：开发者指南->SQL语法参考->SQL语法->DELETE；或删除表的所有数据，请参见：开发者指南->SQL语法参考->SQL语法->TRUNCATE。

TRUNCATE快速地从表中删除所有行，它和在每个表上进行无条件的DELETE有同样的效果，不过因为它不做表扫描，因而快得多。在大表上最有用。

**拷贝数据**

Vastbase提供了在表和文件之间拷贝数据的语句，请参见：开发者指南->SQL语法参考->SQL语法->COPY。

**锁定表**

Vastbase提供了多种锁模式用于控制对表中数据的并发访问，请参见：开发者指南->SQL语法参考->SQL语法->LOCK。

**调用函数**

Vastbase提供了三个用于调用函数的语句，它们在语法结构上没有差别，请参见：开发者指南->SQL语法参考->SQL语法->CALL。

**操作会话**

用户与数据库之间建立的连接称为会话，请参考下表：

会话相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 修改会话 | ALTER SESSION(参见：开发者指南->SQL语法参考->SQL语法->ALTER SESSION章节) |
| 结束会话 | ALTER SYSTEM KILL SESSION(参见：开发者指南->SQL语法参考->SQL语法->ALTER SYSTEM KILL SESSION) |

## DDL语法一览表

**功能描述**

DDL（Data Definition Language数据定义语言），用于定义或修改数据库中的对象。如：表、索引、视图等。



Vastbase不支持数据库主节点不完整时进行DDL操作。例如：Vastbase中有1个数据库主节点故障时执行新建数据库、表等操作都会失败。

**定义客户端加密主密钥**

客户端加密主密钥主要用于密态数据库特性，用来加密列加密密钥(cek)。客户端加密主密钥定义主要包括创建客户端加密主密钥以及删除客户端加密主密钥。所涉及的SQL语句，请参考下表：

客户端加密主密钥定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建客户端加密主密钥 | CREATE CLIENT MASTER KEY(参见：开发者指南->SQL语法参考->SQL语法->CREATE-CLIENT-MASTER-KEY章节) |
| 删除客户端加密主密钥 | DROP CLIENT MASTER KEY(参见：开发者指南->SQL语法参考->SQL语法->DROP CLIENT MASTER KEY章节) |

**定义列加密密钥**

列加密密钥主要用于密态数据库特性中，用来加密数据。列加密密钥定义主要包括创建列加密密钥以及删除列加密密钥。所涉及的SQL语句，请参考下表：

列加密密钥定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建列加密密钥 | CREATE COLUMN ENCRYPTION KEY(参见：开发者指南->SQL语法参考->SQL语法->CREATE COLUMN ENCRYPTION KEY章节) |
| 删列加密密钥 | DROP COLUMN ENCRYPTION KEY(参见：开发者指南->SQL语法参考->SQL语法->DROP COLUMN ENCRYPTION KEY章节) |

**定义数据库**

数据库是组织、存储和管理数据的仓库，而数据库定义主要包括：创建数据库、修改数据库属性，以及删除数据库。所涉及的SQL语句，请参考下表：

数据库定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建数据库 | CREATE DATABASE(参见：开发者指南->SQL语法参考->SQL语法->CREATE DATABASE章节) |
| 修改数据库属性 | ALTER DATABASE(参见：开发者指南->SQL语法参考->SQL语法->ALTER DATABASE章节) |
| 删除数据库 | DROP DATABASE(参见：开发者指南->SQL语法参考->SQL语法->DROP DATABASE章节) |

**定义模式**

模式是一组数据库对象的集合，主要用于控制对数据库对象的访问。所涉及的SQL语句，请参考下表：

模式定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建模式 | CREATE SCHEMA(参见：开发者指南->SQL语法参考->SQL语法->CREATE-SCHEMA章节) |
| 修改模式属性 | ALTER SCHEMA(参见：开发者指南->SQL语法参考->SQL语法->ALTER SCHEMA章节) |
| 删除模式 | DROP SCHEMA(参见：开发者指南->SQL语法参考->SQL语法->DROP SCHEMA章节) |

**定义表空间**

表空间用于管理数据对象，与磁盘上的一个目录对应。所涉及的SQL语句，请参考下表：

表空间定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建表空间 | CREATE TABLESPACE(参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLESPACE章节) |
| 修改表空间属性 | ALTER TABLESPACE(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLESPACE章节) |
| 删除表空间 | DROP TABLESPACE(参见：开发者指南->SQL语法参考->SQL语法->DROP TABLESPACE章节) |

**定义表**

表是数据库中的一种特殊数据结构，用于存储数据对象以及对象之间的关系。所涉及的SQL语句，请参考下表：

表定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建表 | CREATE TABLE(参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLE章节) |
| 修改表属性 | ALTER TABLE(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLE章节) |
| 删除表 | DROP TABLE(参见：开发者指南->SQL语法参考->SQL语法->DROP TABLE章节) |

**定义分区表**

分区表是一种逻辑表，数据是由普通表存储的，主要用于提升查询性能。所涉及的SQL语句，请参考下表：

分区表定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建分区表 | CREATE TABLE PARTITION(参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLE PARTITION章节) |
| 创建分区 | ALTER TABLE PARTITION(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLE PARTITION章节) |
| 修改分区表属性 | ALTER TABLE PARTITION(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLE-PARTITION章节) |
| 删除分区 | ALTER TABLE PARTITION(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLE PARTITION章节) |
| 删除分区表 | DROP TABLE(参见：开发者指南->SQL语法参考->SQL语法->DROP TABLE章节) |

**定义索引**

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。所涉及的SQL语句，请参考下表：

索引定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建索引 | CREATE INDEX(参见：开发者指南->SQL语法参考->SQL语法->CREATE INDEX章节) |
| 修改索引属性 | ALTER INDEX(参见：开发者指南->SQL语法参考->SQL语法->ALTER INDEX章节) |
| 删除索引 | DROP INDEX(参见：开发者指南->SQL语法参考->SQL语法->DROP-INDEX章节) |
| 重建索引 | REINDEX(参见：开发者指南->SQL语法参考->SQL语法->REINDEX章节) |

**定义存储过程**

存储过程是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名称并给出参数（如果该存储过程带有参数）来执行它。所涉及的SQL语句，请参考下表：

存储过程定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建存储过程 | CREATE PROCEDURE(参见：开发者指南->SQL语法参考->SQL语法->CREATE PROCEDURE章节) |
| 删除存储过程 | DROP PROCEDURE(参见：开发者指南->SQL语法参考->SQL语法->DROP PROCEDURE章节) |

**定义函数**

在Vastbase中，它和存储过程类似，也是一组SQL语句集，使用上没有差别。所涉及的SQL语句，请参考下表：

函数定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建函数 | CREATE FUNCTION(参见：开发者指南->SQL语法参考->SQL语法->CREATE FUNCTION章节) |
| 修改函数属性 | ALTER FUNCTION(参见：开发者指南->SQL语法参考->SQL语法->ALTER FUNCTION章节) |
| 删除函数 | DROP FUNCTION(参见：开发者指南->SQL语法参考->SQL语法->DROP FUNCTION章节) |

**定义视图**

视图是从一个或几个基本表中导出的虚表，可用于控制用户对数据访问，请参考下表：

视图定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建视图 | CREATE VIEW(参见：开发者指南->SQL语法参考->SQL语法->CREATE VIEW章节) |
| 删除视图 | DROP VIEW(参见：开发者指南->SQL语法参考->SQL语法->DROP VIEW章节) |

**定义包**

包是模块化的思想，由包头（package specification）和包体(package body)组成，用来分类管理存储过程和函数，类似于Java、C++等语言中的类，请参考下表：

包定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建包 | CREATE PACKAGE(参见：开发者指南->SQL语法参考->SQL语法->CREATE PACKAGE章节) |
| 删除包 | DROP PACKAGE(参见：开发者指南->SQL语法参考->SQL语法->DROP PACKAGE章节) |
| 修改包属性 | ALTER PACKAGE(参见：开发者指南->SQL语法参考->SQL语法->ALTER PACKAGE章节) |

**定义游标**

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化，请参考下表：

游标定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建游标 | CURSOR(参见：开发者指南->SQL语法参考->SQL语法->CURSOR) |
| 移动游标 | MOVE(参见：开发者指南->SQL语法参考->SQL语法->MOVE) |
| 从游标中提取数据 | FETCH(参见：开发者指南->SQL语法参考->SQL语法->FETCH章节) |
| 关闭游标 | CLOSE(参见：开发者指南->SQL语法参考->SQL语法->CLOSE章节) |

**定义聚合函数**

聚合函数定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建一个新的聚合函数 | CREATE AGGREGATE(参见：开发者指南->SQL语法参考->SQL语法->CREATE AGGREGATE章节) |
| 修改聚合函数 | ALTER AGGREGATE(参见：开发者指南->SQL语法参考->SQL语法->ALTER AGGREGATE章节) |
| 删除聚合函数 | DROP AGGREGATE(参见：开发者指南->SQL语法参考->SQL语法->DROP-AGGREGATE章节) |

**定义数据类型转换**

数据类型定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建一个新的用户自定义数据类型转换 | CREATE CAST(参见：开发者指南->SQL语法参考->SQL语法->CREATE CAST章节) |
| 删除用户自定义数据类型转换 | DROP CAST(参见：开发者指南->SQL语法参考->SQL语法->DROP-CAST章节) |

**定义插件扩展**

插件扩展定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建一个新的插件扩展 | CREATE EXTENSION(参见：开发者指南->SQL语法参考->SQL语法->CREATE EXTENSION章节) |
| 修改插件扩展 | ALTER EXTENSION(参见：开发者指南->SQL语法参考->SQL语法->ALTER EXTENSION章节) |
| 删除插件扩展 | DROP EXTENSION(参见：开发者指南->SQL语法参考->SQL语法->DROP EXTENSION章节) |

**定义操作符**

操作符定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建一个新的操作符 | CREATE OPERATOR(参见：开发者指南->SQL语法参考->SQL语法->CREATE OPERATOR章节) |
| 修改操作符 | ALTER OPERATOR(参见：开发者指南->SQL语法参考->SQL语法->ALTER OPERATOR章节) |
| 删除操作符 | DROP OPERATOR(参见：开发者指南->SQL语法参考->SQL语法->DROP OPERATOR章节) |

**定义过程语言**

过程语言定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 修改过程语言 | ALTER LANGUAGE(参见：开发者指南->SQL语法参考->SQL语法->ALTER LANGUAGE章节) |

**定义数据类型**

数据类型定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建一个新的数据类型 | CREATE TYPE(参见：开发者指南->SQL语法参考->SQL语法->CREATE TYPE章节) |
| 修改数据类型 | ALTER TYPE(参见：开发者指南->SQL语法参考->SQL语法->ALTER TYPE章节) |
| 删除数据类型 | DROP TYPE(参见：开发者指南->SQL语法参考->SQL语法->DROP TYPE章节) |

## DCL语法一览表

DCL（Data Control Language数据控制语言），是用来创建用户角色、设置或更改数据库用户或角色权限的语句。

**定义角色**

角色是用来管理权限的，从数据库安全的角度考虑，可以把所有的管理和操作权限划分到不同的角色上。所涉及的SQL语句，请参考下表：

角色定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建角色 | CREATE ROLE(参见：开发者指南->SQL语法参考->SQL语法->CREATE-ROLE章节) |
| 修改角色属性 | ALTER ROLE(参见：开发者指南->SQL语法参考->SQL语法->ALTER ROLE章节) |
| 删除角色 | DROP ROLE(参见：开发者指南->SQL语法参考->SQL语法->DROP ROLE章节) |

**定义用户**

用户是用来登录数据库的，通过对用户赋予不同的权限，可以方便地管理用户对数据库的访问及操作。所涉及的SQL语句，请参考下表：

用户定义相关SQL

| **功能** | **相关SQL** |
| --- | --- |
| 创建用户 | CREATE USER(参见：开发者指南->SQL语法参考->SQL语法->CREATE USER章节) |
| 修改用户属性 | ALTER USER(参见：开发者指南->SQL语法参考->SQL语法->ALTER-USER章节) |
| 删除用户 | DROP USER(参见：开发者指南->SQL语法参考->SQL语法->DROP USER章节) |

**授权**

Vastbase提供了针对数据对象和角色授权的语句，请参见：开发者指南->SQL语法参考->SQL语法->GRANT章节。

**收回权限**

Vastbase提供了收回权限的语句，请参见：开发者指南->SQL语法参考->SQL语法->REVOKE章节。

**设置默认权限**

Vastbase允许设置应用于将来创建的对象的权限，请参见：开发者指南->SQL语法参考->SQL语法->ALTER DEFAULT PRIVILEGES章节。

**关闭当前节点**

Vastbase支持使用shutdown命令关闭当前数据库节点，请参见：开发者指南->SQL语法参考->SQL语法->SHUTDOWN章节。

## SQL语法

SQL 是一门声明性语言，它是数据库用户与数据库交互的方式。

### ABORT

**功能描述**

回滚当前事务并且撤销所有当前事务中所做的更改。

作用等同于ROLLBACK(参见：开发者指南->SQL语法参考->SQL语法->ROLLBACK章节)，早期SQL有用ABORT，现在推荐使用ROLLBACK。

**注意事项**

在事务外部执行ABORT语句不会影响事务的执行，但是会抛出一个NOTICE信息。

**语法格式**

ABORT [ WORK | TRANSACTION ] ;

**参数说明**

**WORK | TRANSACTION**

可选关键字，除了增加可读性没有其他任何作用。

**示例**

--创建表customer\_demographics\_t1。

vastbase=# CREATE TABLE customer\_demographics\_t1

(

CD\_DEMO\_SK INTEGER NOT NULL,

CD\_GENDER CHAR(1) ,

CD\_MARITAL\_STATUS CHAR(1) ,

CD\_EDUCATION\_STATUS CHAR(20) ,

CD\_PURCHASE\_ESTIMATE INTEGER ,

CD\_CREDIT\_RATING CHAR(10) ,

CD\_DEP\_COUNT INTEGER ,

CD\_DEP\_EMPLOYED\_COUNT INTEGER ,

CD\_DEP\_COLLEGE\_COUNT INTEGER

)

WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE)

;

--插入记录。

vastbase=# INSERT INTO customer\_demographics\_t1 VALUES(1920801,'M', 'U', 'DOCTOR DEGREE', 200, 'GOOD', 1, 0,0);

--开启事务。

vastbase=# START TRANSACTION;

--更新字段值。

vastbase=# UPDATE customer\_demographics\_t1 SET cd\_education\_status= 'Unknown';

--终止事务，上面所执行的更新会被撤销掉。

vastbase=# ABORT;

--查询数据。

vastbase=# SELECT \* FROM customer\_demographics\_t1 WHERE cd\_demo\_sk = 1920801;

cd\_demo\_sk | cd\_gender | cd\_marital\_status | cd\_education\_status | cd\_purchase\_estimate | cd\_credit\_rating | cd\_dep\_count | cd\_dep\_employed\_count | cd\_dep\_college\_count

------------+-----------+-------------------+----------------------+----------------------+------------------+--------------+-----------------------+----------------------

1920801 | M | U | DOCTOR DEGREE | 200 | GOOD | 1 | 0 | 0

(1 row)

--删除表。

vastbase=# DROP TABLE customer\_demographics\_t1;

**相关链接**

SET TRANSACTION(参见：开发者指南->SQL语法参考->SQL语法->SET TRANSACTION章节)，COMMIT | END(参见：开发者指南->SQL语法参考->SQL语法->COMMIT END章节)，ROLLBACK(参见：开发者指南->SQL语法参考->SQL语法->ROLLBACK章节)

### ALTER AUDIT POLICY

**功能描述**

修改统一审计策略。

**注意事项**

* 开启三权分立后，只有安全管理员才能进行此操作。
* 需要打开enable\_security\_policy开关统一审计策略才可以生效。

**语法格式**

ALTER AUDIT POLICY [ IF EXISTS ] policy\_name { ADD | REMOVE } { [ privilege\_audit\_clause ] [ access\_audit\_clause ] };

ALTER AUDIT POLICY [ IF EXISTS ] policy\_name MODIFY ( filter\_group\_clause );

ALTER AUDIT POLICY [ IF EXISTS ] policy\_name DROP FILTER;

ALTER AUDIT POLICY [ IF EXISTS ] policy\_name COMMENTS policy\_comments;

ALTER AUDIT POLICY [ IF EXISTS ] policy\_name { ENABLE | DISABLE };

* privilege\_audit\_clause：

PRIVILEGES { DDL | ALL }

* access\_audit\_clause：

ACCESS { DML | ALL }

* filter\_group\_clause：

FILTER ON { ( FILTER\_TYPE ( filter\_value [, ... ] ) ) [, ... ] }

**参数说明**

* **policy\_name**
* 审计策略名称，需要唯一，不可重复。
* 取值范围：字符串，要符合标识符的命名规范。
* **DDL**
* 指的是针对数据库执行如下操作时进行审计，目前支持：CREATE、ALTER、DROP、ANALYZE、COMMENT、GRANT、REVOKE、SET、SHOW、LOGIN\_ANY、LOGIN\_FAILURE、LOGIN\_SUCCESS、LOGOUT。
* **ALL**
* 指的是上述DDL支持的所有对数据库的操作。
* **DML**
* 指的是针对数据库执行如下操作时进行审计，目前支持：SELECT、COPY、DEALLOCATE、DELETE、EXECUTE、INSERT、PREPARE、REINDEX、TRUNCATE、UPDATE。
* **FILTER\_TYPE**
* 指定审计策略的过滤信息，过滤类型包括：IP、ROLES、APP。
* **filter\_value**
* 指具体过滤信息内容。
* **policy\_comments**
* 用于记录策略相关的描述信息。
* **ENABLE|DISABLE**
* 可以打开或关闭统一审计策略。若不指定ENABLE|DISABLE，语句默认为ENABLE。

**示例**

请参见：开发者指南->SQL语法参考->SQL语法->CREATE AUDIT POLICY章节的示例。

**相关链接**

CREATE AUDIT POLICY(参见：开发者指南->SQL语法参考->SQL语法->CREATE-AUDIT-POLICY章节)，DROP AUDIT POLICY(参见：开发者指南->SQL语法参考->SQL语法->DROP AUDIT POLICY章节)。

### ALTER AGGREGATE

**功能描述**

修改一个聚合函数的定义。

**注意事项**

要使用 ALTER AGGREGATE ，你必须是该聚合函数的所有者。 要改变一个聚合函数的模式，你必须在新模式上有 CREATE 权限。要改变所有者，你必须是新所有角色的一个直接或间接成员，并且该角色必须在聚合函数的模式上有 CREATE 权限。（这些限制强制了修改该所有者不会做任何通过删除和重建聚合函数不能做的事情。不过，具有AYSADMIN权限用户可以用任何方法任意更改聚合函数的所属关系）。

**语法格式**

ALTER AGGREGATE name ( argtype [ , ... ] ) RENAME TO new\_name

ALTER AGGREGATE name ( argtype [ , ... ] ) OWNER TO new\_owner

ALTER AGGREGATE name ( argtype [ , ... ] ) SET SCHEMA new\_schema

**参数说明**

* **name**
* 现有的聚合函数的名称（可以有模式修饰）。
* **argtype**
* 聚合函数操作的输入数据类型。要引用一个零参数聚合函数，可以写入\*代替输入数据类型列表。
* **new\_name**
* 聚合函数的新名字。
* **new\_owner**
* 聚合函数的新所有者。
* **new\_schema**
* 聚合函数的新模式。

**示例**

把一个接受integer 类型参数的聚合函数myavg重命名为 my\_average ：

ALTER AGGREGATE myavg(integer) RENAME TO my\_average;

把一个接受integer 类型参数的聚合函数myavg的所有者改为joe ：

ALTER AGGREGATE myavg(integer) OWNER TO joe;

把一个接受integer 类型参数的聚合函数myavg移动到模式myschema里：

ALTER AGGREGATE myavg(integer) SET SCHEMA myschema;

**兼容性**

SQL标准里没有ALTER AGGREGATE语句。

### ALTER DATA SOURCE

**功能描述**

修改Data Source对象的属性和内容。

属性有：名称和属主；内容有：类型、版本和连接选项。

**注意选项**

* 只有初始用户/系统管理员/属主才拥有修改Data Source的权限。
* 修改属主时，新的属主用户必须是初始用户或系统管理员。
* 当在OPTIONS中出现password选项时，需要保证Vastbase每个节点的$GAUSSHOME/bin目录下存在datasource.key.cipher和datasource.key.rand文件，如果不存在这两个文件，请使用vb\_guc工具生成并发布到每个节点的$GAUSSHOME/bin目录下。

**语法格式**

ALTER DATA SOURCE src\_name

[TYPE 'type\_str']

[VERSION {'version\_str' | NULL}]

[OPTIONS ( {[ ADD | SET | DROP ] optname ['optvalue']} [, ...] )];

ALTER DATA SOURCE src\_name RENAME TO src\_new\_name;

ALTER DATA SOURCE src\_name OWNER TO new\_owner;

**参数说明**

* **src\_name**
* 待修改的Data Source的名称。
* 取值范围：字符串，需要符合标识符的命名规范。
* **TYPE**
* 将Data Source原来的TYPE修改为指定值。
* 取值范围：空串或非空字符串。
* **VERSION**
* 将Data Source原来的VERSION修改为指定值。
* 取值范围：空串或非空字符串或NULL。
* **OPTIONS**
* 修改OPTIONS中的字段：增加（ADD）、修改（SET）、删除（DROP），且字段名称optname需唯一，具体要求如下：
* 增加字段：ADD可以省略，待增加字段不能已经存在了；
* 修改字段：SET不可省略，待修改字段必须存在；
* 删除字段：DROP不可省略，待删除字段必须存在，且不能指定optvalue；
* **src\_new\_name**
* 新的Data Source名称。
* 取值范围：字符串，需符合标识符命名规范。
* **new\_user**
* 对象的新属主。
* 取值范围：字符串，有效的用户名。

**示例**

--创建一个空Data Source对象。

vastbase=# CREATE DATA SOURCE ds\_test1;

--修改名称。

vastbase=# ALTER DATA SOURCE ds\_test1 RENAME TO ds\_test;

--修改属主。

vastbase=# CREATE USER user\_test1 IDENTIFIED BY 'Gs@123456';

vastbase=# ALTER USER user\_test1 WITH SYSADMIN;

vastbase=# ALTER DATA SOURCE ds\_test OWNER TO user\_test1;

--修改TYPE和VERSION。

vastbase=# ALTER DATA SOURCE ds\_test TYPE 'MPPDB\_TYPE' VERSION 'XXX';

--添加字段。

vastbase=# ALTER DATA SOURCE ds\_test OPTIONS (add dsn 'gaussdb', username 'test\_user');

--修改字段。

vastbase=# ALTER DATA SOURCE ds\_test OPTIONS (set dsn 'unknown');

--删除字段。

vastbase=# ALTER DATA SOURCE ds\_test OPTIONS (drop username);

--删除Data Source和user对象。

vastbase=# DROP DATA SOURCE ds\_test;

vastbase=# DROP USER user\_test1;

**相关链接**

CREATE DATA SOURCE (参见：开发者指南->SQL语法参考->SQL语法->CREATE DATA SOURCE 章节)，DROP DATA SOURCE(参见：开发者指南->SQL语法参考->SQL语法->DROP DATA SOURCE 章节)

### ALTER DATABASE

**功能描述**

修改数据库的属性，包括它的名称、所有者、连接数限制、对象隔离属性等。

**注意事项**

* 只有数据库的所有者或者被授予了数据库ALTER权限的用户才能执行ALTER DATABASE命令，系统管理员默认拥有此权限。针对所要修改属性的不同，还有以下权限约束：
* 修改数据库名称，必须拥有CREATEDB权限。
* 修改数据库所有者，当前用户必须是该database的所有者或者系统管理员，必须拥有CREATEDB权限，且该用户是新所有者角色的成员。
* 修改数据库默认表空间，必须拥有新表空间的CREATE权限。这个语句会从物理上将一个数据库原来缺省表空间上的表和索引移至新的表空间。注意不在缺省表空间的表和索引不受此影响。
* 不能重命名当前使用的数据库，如果需要重新命名，须连接至其他数据库上。

**语法格式**

* 修改数据库的最大连接数。

ALTER DATABASE database\_name

[ [ WITH ] CONNECTION LIMIT connlimit ];

* 修改数据库名称。

ALTER DATABASE database\_name

RENAME TO new\_name;

* 修改数据库所属者。

ALTER DATABASE database\_name

OWNER TO new\_owner;

* 修改数据库默认表空间。

ALTER DATABASE database\_name

SET TABLESPACE new\_tablespace;

* 修改数据库指定会话参数值。

ALTER DATABASE database\_name

SET configuration\_parameter { { TO | = } { value | DEFAULT } | FROM CURRENT };

* 数据库配置参数重置。

ALTER DATABASE database\_name RESET

{ configuration\_parameter | ALL };

* 修改数据库对象隔离属性。

ALTER DATABASE database\_name [ WITH ] { ENABLE | DISABLE } PRIVATE OBJECT;

fig: **说明：**

* 修改数据库的对象隔离属性时须连接至该数据库，否则无法更改。
* 新创建的数据库，对象隔离属性默认是关闭的。当开启数据库对象隔离属性后，普通用户只能查看有权访问的对象（表、函数、视图、字段等）。对象隔离特性对管理员用户不生效，当开启对象隔离特性后，管理员也可以查看到全量的数据库对象。

**参数说明**

* **database\_name**
* 需要修改属性的数据库名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **connlimit**
* 数据库可以接收的最大并发连接数（管理员用户连接除外）。
* 取值范围：整数，建议填写1~50的整数。-1（缺省）表示没有限制。
* **new\_name**
* 数据库的新名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **new\_owner**
* 数据库的新所有者。
* 取值范围：字符串，有效的用户名。
* **new\_tablespace**
* 数据库新的默认表空间，该表空间为数据库中已经存在的表空间。默认的表空间为pg\_default。
* 取值范围：字符串，有效的表空间名。
* **configuration\_parameter**
* **value**
* 把指定的数据库会话参数值设置为给定的值。如果value是DEFAULT或者RESET，则在新的会话中使用系统的缺省设置。OFF关闭设置。
* 取值范围：字符串
* DEFAULT
* OFF
* RESET
* **FROM CURRENT**
* 根据当前会话连接的数据库设置该参数的值。
* **RESET configuration\_parameter**
* 重置指定的数据库会话参数值。
* **RESET ALL**
* 重置全部的数据库会话参数值。

* 修改数据库默认表空间，会将旧表空间中的所有表和索引转移到新表空间中，该操作不会影响其他非默认表空间中的表和索引。
* 修改的数据库会话参数值，将在下一次会话中生效。

**示例**

请参考CREATE DATABASE (参见：开发者指南->SQL语法参考->SQL语法->CREATE DATABASE 章节)的示例。

**相关链接**

CREATE DATABASE (参见：开发者指南->SQL语法参考->SQL语法->CREATE DATABASE 章节)，DROP DATABASE (参见：开发者指南->SQL语法参考->SQL语法->DROP DATABASE 章节)

### ALTER DEFAULT PRIVILEGES

**功能描述**

设置应用于将来创建的对象的权限（这不会影响分配到已有对象中的权限）。

**注意事项**

目前只支持表（包括视图）、序列、函数、类型、密态数据库客户端主密钥和列加密密钥的权限更改。

**语法格式**

ALTER DEFAULT PRIVILEGES

[ FOR { ROLE | USER } target\_role [, ...] ]

[ IN SCHEMA schema\_name [, ...] ]

abbreviated\_grant\_or\_revoke;

* 其中abbreviated\_grant\_or\_revoke子句用于指定对哪些对象进行授权或回收权限。

grant\_on\_tables\_clause

| grant\_on\_sequences\_clause

| grant\_on\_functions\_clause

| grant\_on\_types\_clause

| grant\_on\_client\_master\_keys\_clause

| grant\_on\_column\_encryption\_keys\_clause

| revoke\_on\_tables\_clause

| revoke\_on\_sequences\_clause

| revoke\_on\_functions\_clause

| revoke\_on\_types\_clause

| revoke\_on\_client\_master\_keys\_clause

| revoke\_on\_column\_encryption\_keys\_clause

* 其中grant\_on\_tables\_clause子句用于对表授权。

GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM }

[, ...] | ALL [ PRIVILEGES ] }

ON TABLES

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中grant\_on\_sequences\_clause子句用于对序列授权。

GRANT { { SELECT | UPDATE | USAGE | ALTER | DROP | COMMENT }

[, ...] | ALL [ PRIVILEGES ] }

ON SEQUENCES

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中grant\_on\_functions\_clause子句用于对函数授权。

GRANT { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }

ON FUNCTIONS

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中grant\_on\_types\_clause子句用于对类型授权。

GRANT { { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }

ON TYPES

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中grant\_on\_client\_master\_keys\_clause子句用于对客户端主密钥授权。

GRANT { { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }

ON CLIENT\_MASTER\_KEYS

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中grant\_on\_column\_encryption\_keys\_clause子句用于对列加密密钥授权。

GRANT { { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }

ON COLUMN\_ENCRYPTION\_KEYS

TO { [ GROUP ] role\_name | PUBLIC } [, ...]

[ WITH GRANT OPTION ]

* 其中revoke\_on\_tables\_clause子句用于回收表对象的权限。

REVOKE [ GRANT OPTION FOR ]

{ { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM }

[, ...] | ALL [ PRIVILEGES ] }

ON TABLES

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

* 其中revoke\_on\_sequences\_clause子句用于回收序列的权限。

REVOKE [ GRANT OPTION FOR ]

{ { SELECT | UPDATE | USAGE | ALTER | DROP | COMMENT }

[, ...] | ALL [ PRIVILEGES ] }

ON SEQUENCES

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

* 其中revoke\_on\_functions\_clause子句用于回收函数的权限。

REVOKE [ GRANT OPTION FOR ]

{ {EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }

ON FUNCTIONS

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

* 其中revoke\_on\_types\_clause子句用于回收类型的权限。

REVOKE [ GRANT OPTION FOR ]

{ { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }

ON TYPES

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

* 其中revoke\_on\_client\_master\_keys\_clause子句用于回收客户端主密钥的权限。

REVOKE [ GRANT OPTION FOR ]

{ { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }

ON CLIENT\_MASTER\_KEYS

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

* 其中revoke\_on\_column\_encryption\_keys\_clause子句用于回收列加密密钥的权限。

REVOKE [ GRANT OPTION FOR ]

{ { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }

ON COLUMN\_ENCRYPTION\_KEYS

FROM { [ GROUP ] role\_name | PUBLIC } [, ...]

[ CASCADE | RESTRICT | CASCADE CONSTRAINTS ]

**参数说明**

* **target\_role**
* 已有角色的名称。如果省略FOR ROLE/USER，则缺省值为当前角色/用户。
* 取值范围：已有角色的名称。
* **schema\_name**
* 现有模式的名称。
* target\_role必须有schema\_name的CREATE权限。
* 取值范围：现有模式的名称。
* **role\_name**
* 被授予或者取消权限角色的名称。
* 取值范围：已存在的角色名称。

fig: **须知：**

如果想删除一个被赋予了默认权限的角色，有必要恢复改变的缺省权限或者使用DROP OWNED BY来为角色脱离缺省的权限记录。

**示例**

--创建模式tpcds

vastbase=# CREATE SCHEMA tpcds;

--将创建在模式tpcds里的所有表（和视图）的SELECT权限授予每一个用户。

vastbase=# ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds GRANT SELECT ON TABLES TO PUBLIC;

--创建用户普通用户jack。

vastbase=# CREATE USER jack PASSWORD 'Bigdata@123';

--将tpcds下的所有表的插入权限授予用户jack。

vastbase=# ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds GRANT INSERT ON TABLES TO jack;

--撤销上述权限。

vastbase=# ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds REVOKE SELECT ON TABLES FROM PUBLIC;

vastbase=# ALTER DEFAULT PRIVILEGES IN SCHEMA tpcds REVOKE INSERT ON TABLES FROM jack;

--删除用户jack。

vastbase=# DROP USER jack;

**相关链接**

GRANT (参见：开发者指南->SQL语法参考->SQL语法->GRANT章节)，REVOKE(参见：开发者指南->SQL语法参考->SQL语法->REVOKE章节)

### ALTER DIRECTORY

**功能描述**

对directory属性进行修改。

**注意事项**

* 目前只支持修改directory属主。
* 当enable\_access\_server\_directory=off时，只允许初始用户修改directory属主；当enable\_access\_server\_directory=on时，具有SYSADMIN权限的用户和directory对象的属主可以修改directory，且要求该用户是新属主的成员。

**语法格式**

ALTER DIRECTORY directory\_name

OWNER TO new\_owner;

**参数描述**

**directory\_name**

需要修改的目录名称，范围为已经存在的目录名称。

**示例**

--创建目录。

vastbase=# CREATE OR REPLACE DIRECTORY dir as '/tmp/';

--修改目录的owner。

vastbase=# ALTER DIRECTORY dir OWNER TO system;

--删除目录。

vastbase=# DROP DIRECTORY dir;

**相关链接**

CREATE DIRECTORY (参见：开发者指南->SQL语法参考->SQL语法->CREATE DIRECTORY章节)，DROP DIRECTORY(参见：开发者指南->SQL语法参考->SQL语法->DROP DIRECTORY章节)

### ALTER EXTENSION

**功能描述**

修改插件扩展。

**注意事项**

ALTER EXTENSION 修改一个已安装的扩展的定义。这里有几种方式：

* UPDATE
* 这种方式更新这个扩展到一个新的版本。这个扩展必须满足一个适用的更新脚本（或者一系列脚本）这样就能修改当前安装版本到一个要求的版本。
* SET SCHEMA
* 这种方式移动扩展对象到另一个模式。这个扩展必须relocatable才能使命令成功。
* ADD member\_object
* 这种方式添加一个已存在对象到扩展。这主要在扩展更新脚本上有用。 这个对象接着会被视为扩展的成员; 显而易见,该对象只能通过取消扩展来取消 。
* DROP member\_object
* 这个方式从扩展上移除一个成员对象。 主要在扩展更新脚本上有用。这个对象没有被取消, 只是从扩展里分开了。
* 您必须拥有扩展来使用 ALTER EXTENSION。 这个 ADD/DROP 方式要求添加/删除对象的所有权。

**语法格式**

ALTER EXTENSION name UPDATE [ TO new\_version ]

ALTER EXTENSION name SET SCHEMA new\_schema

ALTER EXTENSION name ADD member\_object

ALTER EXTENSION name DROP member\_object

where member\_object is:

FOREIGN TABLE object\_name |

FUNCTION function\_name ( [ [ argmode ] [ argname ] argtype [, ...] ] ) |

[ PROCEDURAL ] LANGUAGE object\_name |

SCHEMA object\_name |

SERVER object\_name |

TABLE object\_name |

TEXT SEARCH CONFIGURATION object\_name |

TYPE object\_name |

VIEW object\_name

**参数说明**

* **name**
* 已安装扩展的名称。
* **new\_version**
* 扩展的新版本。可以通过被标识符和字面字符重写。如果不指定的扩展的新版本，ALTER EXTENSION UPDATE会更新到扩展的控制文件中显示的默认版本。
* **new\_schema**
* 扩展的新模式。
* **object\_name**
* **function\_name**
* 从扩展里被添加或移除的对象的名称。包含表、聚合 、域、外链表、函数、操作符、操作符类、操作符族、序列、文本搜索对象、类型和能被模式合格的视图的名称。
* **argmode**
* 这个函数参数的模型：IN、OUT、INOUT或者VARIADIC。如果省略的话，默认值为IN。ALTER EXTENSION 不关心OUT参数 ，因为确认函数的一致性只需要输入参数，因此列出IN、INOUT和VARIADIC参数就足够了。
* **argname**
* 函数参数的名称。ALTER EXTENSION不关心参数名称，确认函数的一致性只需要参数数据类型。
* **argtype**
* 函数参数的数据类型（可以有模式修饰）。

**示例**

1、创建扩展

create extension "uuid-ossp";

2、更新 "uuid-ossp" 扩展到版本 2.0：

ALTER EXTENSION "uuid-ossp" UPDATE TO '2.0';

3、创建模式utils

CREATE SCHEMA utils;

4、更新 "uuid-ossp"扩展的模式为utils：

ALTER EXTENSION "uuid-ossp" SET SCHEMA utils;

5、创建函数

create function length(p\_one text, p\_other text)

returns text

as

$$

select case

when length(p\_one) >= length(p\_other) then p\_one

else p\_other

end

$$

language sql

immutable;

6、添加length函数给 "uuid-ossp" 扩展：

ALTER EXTENSION "uuid-ossp" ADD FUNCTION length(p\_one text, p\_other text);

7、从"uuid-ossp" 扩展扩展删除length函数：

ALTER EXTENSION "uuid-ossp" DROP FUNCTION length(p\_one text, p\_other text);

### ALTER FOREIGN TABLE

**功能描述**

对外表进行修改。

**语法格式**

ALTER FOREIGN TABLE [ IF EXISTS ] table\_name

OPTIONS ( {[ ADD | SET | DROP ] option ['value']}[, ... ]);

ALTER FOREIGN TABLE [ IF EXISTS ] table\_name

ALTER column\_name OPTIONS;

**参数说明**

* **table\_name**
* 需要修改的外表名称。
* 取值范围：已存在的外表名。
* **option**
* 改变外表或者外表字段的选项。ADD、SET和DROP指定执行的操作。如果没有显式设置，那么默认为ADD。选项的名字不允许重复（尽管表选项和表字段选项可以有相同的名字）。选项的名称和值也会通过外部数据封装器的类库进行校验。
* oracle\_fdw支持的options包括：
* **table**
  + - oracle server侧的表名。需要同oracle系统表中记录的表名完全一致，通常是由大写字符组成。
* **schema**
  + - 表所对应的schema（或owner）。需要与oracle系统表中记录的表名完全一致，通常是由大写字符组成。
* mysql\_fdw支持的options包括：
* **dbname**
  + - MySQL的database名称。
* **table\_name**
  + - MySQL侧的表名。
* postgres\_fdw支持的options包括：
* **schema\_name**
  + - 远端server的schema名称。如果不指定的话，将使用外表自身的schema名称作为远端的schema名称。
* **table\_name**
  + - 远端server的表名。如果不指定的话，将使用外表自身的表名作为远端的表名。
* **column\_name**
  + - 远端server的表的列名。如果不指定的话，将使用外表自身的列名作为远端的的表的列名。
* file\_fdw支持的options包括：
* filename

指定要读取的文件，必需的参数，且必须是一个绝对路径名。

* format

远端server的文件格式，支持text/csv/binary/fixed四种格式，和COPY语句的FORMAT选项相同。

* header

指定的文件是否有标题行，与COPY语句的HEADER选项相同。

-delimiter

指定文件的分隔符，与COPY的DELIMITER选项相同。

-quote

指定文件的引用字符，与COPY的QUOTE选项相同。

-escape

指定文件的转义字符，与COPY的ESCAPE选项相同。

-null

指定文件的null字符串，与COPY的NULL选项相同。

-encoding

指定文件的编码，与COPY的ENCODING选项相同。

-force\_not\_null

这是一个布尔选项。如果为真，则声明字段的值不应该匹配空字符串（也就是，文件级别null选项）。与COPY的 FORCE\_NOT\_NULL选项里的字段相同。

* **value**
* option的新值。

**相关链接**

CREATE FOREIGN TABLE(参见：开发者指南->SQL语法参考->SQL语法->CREATE FOREIGN TABLE章节)，DROP FOREIGN TABLE(参见：开发者指南->SQL语法参考->SQL语法->DROP FOREIGN TABLE章节)

### ALTER FUNCTION

**功能描述**

修改自定义函数的属性。

**注意事项**

只有函数的所有者或者被授予了函数ALTER权限的用户才能执行ALTER FUNCTION命令，系统管理员默认拥有该权限。针对所要修改属性的不同，还有以下权限约束：

* 如果函数中涉及对临时表相关的操作，则无法使用ALTER FUNCTION。
* 修改函数的所有者或修改函数的模式，当前用户必须是该函数的所有者或者系统管理员，且该用户是新所有者角色的成员。
* 只有系统管理员和初始化用户可以将function的schema修改成public。

**语法格式**

* 修改自定义函数的附加参数。

ALTER FUNCTION function\_name ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

action [ ... ] [ RESTRICT ];

其中附加参数action子句语法为。

{CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT}

| {IMMUTABLE | STABLE | VOLATILE}

| {NOT FENCED | FENCED}

| [ NOT ] LEAKPROOF

| {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER}

| AUTHID { DEFINER | CURRENT\_USER }

| COST execution\_cost

| ROWS result\_rows

| SET configuration\_parameter {{ TO | = } { value | DEFAULT }| FROM CURRENT}

| RESET {configuration\_parameter| ALL}

* 修改自定义函数的名称。

ALTER FUNCTION funname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

RENAME TO new\_name;

* 修改自定义函数的所属者。

ALTER FUNCTION funname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

OWNER TO new\_owner;

* 修改自定义函数的模式。

ALTER FUNCTION funname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

SET SCHEMA new\_schema;

**参数说明**

* **function\_name**
* 要修改的函数名称。
* 取值范围：已存在的函数名。
* **argmode**
* 标识该参数是输入、输出参数。
* 取值范围：IN/OUT/INOUT/VARIADIC。
* **argname**
* 参数名称。
* 取值范围：字符串，符合标识符命名规范。
* **argtype**
* 函数参数的类型。
* **CALLED ON NULL INPUT**
* 表明该函数的某些参数是NULL的时候可以按照正常的方式调用。缺省时与指定此参数的作用相同。
* **RETURNS NULL ON NULL INPUT**
* **STRICT**
* STRICT用于指定如果函数的某个参数是NULL，此函数总是返回NULL。如果声明了这个参数，则如果存在NULL参数时不会执行该函数；而只是自动假设一个NULL结果。
* RETURNS NULL ON NULL INPUT和STRICT的功能相同。
* **IMMUTABLE**
* 表示该函数在给出同样的参数值时总是返回同样的结果。
* **STABLE**
* 表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。
* **VOLATILE**
* 表示该函数值可以在一次表扫描内改变，不会做任何优化。
* **LEAKPROOF**
* 表示该函数没有副作用，指出参数只包括返回值。LEAKPROOF只能由系统管理员设置。
* **EXTERNAL**
* （可选）目的是和SQL兼容，这个特性适合于所有函数，而不仅是外部函数。
* **SECURITY INVOKER**
* **AUTHID CURRENT\_USER**
* 表明该函数将以调用它的用户的权限执行。缺省时与指定此参数的作用相同。
* SECURITY INVOKER和AUTHID CURRENT\_USER的功能相同。
* **SECURITY DEFINER**
* **AUTHID DEFINER**
* 声明该函数将以创建它的用户的权限执行。
* AUTHID DEFINER和SECURITY DEFINER的功能相同。
* **COST execution\_cost**
* 用来估计函数的执行成本。
* execution\_cost以cpu\_operator\_cost为单位。
* 取值范围：正数
* **ROWS result\_rows**
* 估计函数返回的行数。用于函数返回的是一个集合。
* 取值范围：正数，默认值是1000行。
* **configuration\_parameter**
* **value**

把指定的数据库会话参数值设置为给定的值。如果value是 DEFAULT或者RESET，则在新的会话中使用系统的缺省设置。 OFF关闭设置。

取值范围：字符串

* + - DEFAULT
    - OFF
    - RESET

指定默认值。

* **from current**

取当前会话中的值设置为configuration\_parameter的值。

* **new\_name**
* 函数的新名称。要修改函数的所属模式，必须拥有新模式的CREATE权限。
* 取值范围：字符串，符合标识符命名规范。
* **new\_owner**
* 函数的新所有者。要修改函数的所有者，新所有者必须拥有该函数所属模式的CREATE权限。
* 取值范围：已存在的用户角色。
* **new\_schema**
* 函数的新模式。
* 取值范围：已存在的模式。

**示例**

请参见CREATE FUNCTION的示例。

**相关链接**

CREATE FUNCTION (参见：开发者指南->SQL语法参考->SQL语法->CREATE FUNCTION章节)，DROP FUNCTION (参见：开发者指南->SQL语法参考->SQL语法->DROP FUNCTION 章节)

### ALTER GLOBAL CONFIGURATION

**功能描述**

新增、修改系统表gs\_global\_config，增加key-value值。

**注意事项**

* 仅支持数据库初始用户运行此命令。
* 不支持创建修改关键字为weak\_password。

**语法格式**

ALTER GLOBAL CONFIGURATION with(paraname=value,paraname=value...);

**参数说明**

* **paraname**
* 参数名称，text类型。
* **value**
* 参数值，text类型。

### ALTER GROUP

**功能描述**

修改一个用户组的属性。

**注意事项**

ALTER GROUP是ALTER ROLE的别名，非SQL标准语法，不推荐使用，建议用户直接使用ALTER ROLE替代。

**语法格式**

* 向用户组中添加用户。

ALTER GROUP group\_name

ADD USER user\_name [, ... ];

* 从用户组中删除用户。

ALTER GROUP group\_name

DROP USER user\_name [, ... ];

* 修改用户组的名称。

ALTER GROUP group\_name

RENAME TO new\_name;

**参数说明**

请参见：开发者指南->SQL语法参考->SQL语法->ALTER ROLE的参数说明。

**示例**

--创建用户lche、jim

CREATE USER lche PASSWORD 'Aa@12345';

CREATE USER jim PASSWORD 'Bb@12345';

--创建GROUP

CREATE GROUP super\_users IDENTIFIED BY 'Aa@12345';

--向用户组中添加用户。

ALTER GROUP super\_users ADD USER lche, jim;

--从用户组中删除用户。

ALTER GROUP super\_users DROP USER jim;

--修改用户组的名称。

ALTER GROUP super\_users RENAME TO normal\_users;

**相关链接**

ALTER GROUP(参见：开发者指南->SQL语法参考->SQL语法->ALTER GROUP章节)，DROP GROUP(参见：开发者指南->SQL语法参考->SQL语法->DROP GROUP章节)，ALTER ROLE(参见：开发者指南->SQL语法参考->SQL语法->ALTER ROLE章节)

### ALTER INDEX

**功能描述**

ALTER INDEX用于修改现有索引的定义。

它有几种子形式：

* IF EXISTS
* 如果指定的索引不存在，则发出一个notice而不是error。
* RENAME TO
* 只改变索引的名称。对存储的数据没有影响。
* SET TABLESPACE
* 这个选项会改变索引的表空间为指定表空间，并且把索引相关的数据文件移动到新的表空间里。
* SET ( { STORAGE\_PARAMETER = value } [, ...] )
* 改变索引的一个或多个索引方法特定的存储参数。 需要注意的是索引内容不会被这个命令立即修改，根据参数的不同，可能需要使用REINDEX重建索引来获得期望的效果。
* RESET ( { storage\_parameter } [, ...] )
* 重置索引的一个或多个索引方法特定的存储参数为缺省值。与SET一样，可能需要使用REINDEX来完全更新索引。
* [ MODIFY PARTITION index\_partition\_name ] UNUSABLE
* 用于设置表或者索引分区上的索引不可用。
* REBUILD [ PARTITION index\_partition\_name ]
* 用于重建表或者索引分区上的索引。
* RENAME PARTITION
* 用于重命名索引分区。
* MOVE PARTITION
* 用于修改索引分区的所属表空间。

**注意事项**

只有索引的所有者或者拥有索引所在表的INDEX权限的用户有权限执行此命令，系统管理员默认拥有此权限。

**语法格式**

* 重命名表索引的名称。

ALTER INDEX [ IF EXISTS ] index\_name

RENAME TO new\_name;

* 修改表索引的所属空间。

ALTER INDEX [ IF EXISTS ] index\_name

SET TABLESPACE tablespace\_name;

* 修改表索引的存储参数。

ALTER INDEX [ IF EXISTS ] index\_name

SET ( {storage\_parameter = value} [, ... ] );

* 重置表索引的存储参数。

ALTER INDEX [ IF EXISTS ] index\_name

RESET ( storage\_parameter [, ... ] ) ;

* 设置表索引或索引分区不可用。

ALTER INDEX [ IF EXISTS ] index\_name

[ MODIFY PARTITION index\_partition\_name ] UNUSABLE;

fig: **说明：**

列存表不支持该语法。

* 重建表索引或索引分区。

ALTER INDEX index\_name

REBUILD [ PARTITION index\_partition\_name ];

* 重命名索引分区。

ALTER INDEX [ IF EXISTS ] index\_name

RENAME PARTITION index\_partition\_name TO new\_index\_partition\_name;

* 修改索引分区的所属表空间。

ALTER INDEX [ IF EXISTS ] index\_name

MOVE PARTITION index\_partition\_name TABLESPACE new\_tablespace;

**参数说明**

* **index\_name**
* 要修改的索引名。
* **new\_name**
* 新的索引名。
* 取值范围：字符串，且符合标识符命名规范。
* **tablespace\_name**
* 表空间的名称。
* 取值范围：已存在的表空间。
* **storage\_parameter**
* 索引方法特定的参数名。
* **value**
* 索引方法特定的存储参数的新值。根据参数的不同，这可能是一个数字或单词。
* **new\_index\_partition\_name**
* 新索引分区名。
* **index\_partition\_name**
* 索引分区名。
* **new\_tablespace**
* 新表空间。

**示例**

请参见：开发者指南->SQL语法参考->SQL语法->CREATE INDEX的示例。

**相关链接**

CREATE INDEX(参见：开发者指南->SQL语法参考->SQL语法->CREATE-INDEX章节)，DROP INDEX(参见：开发者指南->SQL语法参考->SQL语法->DROP INDEX章节)，REINDEX(参见：开发者指南->SQL语法参考->SQL语法->REINDEX章节)

### ALTER LANGUAGE

**功能描述**

修改一个过程语言的定义。单机和集中式暂不支持创建过程语言，所以暂不支持修改过程语言。

**语法格式**

ALTER [ PROCEDURAL ] LANGUAGE name RENAME TO new\_name ALTER [ PROCEDURAL ] LANGUAGE name OWNER TO new\_owner

**参数说明**

* **name**
* 语言的名字。
* **new\_name**
* 语言的新名字。
* **new\_owner**
* 语言的新的所有者。

**兼容性**

SQL标准里没有ALTER LANGUAGE语句。

### ALTER LARGE OBJECT

**功能描述**

ALTER LARGE OBJECT用于更改一个large object的定义。它的唯一的功能是分配一个新的所有者。

**注意事项**

使用ALTER LARGE OBJECT必须是系统管理员或者是其所有者。

**语法格式**

ALTER LARGE OBJECT large\_object\_oid

OWNER TO new\_owner;

**参数说明**

* **large\_object\_oid**
* 要被变large object的OID 。
* 取值范围：已存在的大对象名。
* **OWNER TO new\_owner**
* large object新的所有者。
* 取值范围：已存在的用户名/角色名。

**示例**

1、创建LARGE OBJECT。

SELECT pg\_catalog.lo\_create('100001');

2、将LARGE OBJECT所有者改为vastbase。

ALTER LARGE OBJECT 100001 OWNER TO vastbase;

### ALTER MASKING POLICY

**功能描述**

修改脱敏策略。

**注意事项**

* 只有poladmin、sysadmin或初始用户才能执行此操作。
* 需要打开enable\_security\_policy开关脱敏策略才可以生效。

**语法格式**

* 修改策略描述：

ALTER MASKING POLICY policy\_name COMMENTS policy\_comments;

* 修改脱敏方式：

ALTER MASKING POLICY policy\_name [ADD | REMOVE | MODIFY] masking\_actions[, ...]\*;

其中masking\_action:

masking\_function ON LABEL(label\_name[, ...]\*)

* 修改脱敏策略生效场景：

ALTER MASKING POLICY policy\_name MODIFY(FILTER ON FILTER\_TYPE(filter\_value[, ...]\*)[, ...]\*);

* 移除脱敏策略生效场景，使策略对所用场景生效：

ALTER MASKING POLICY policy\_name DROP FILTER;

* 修改脱敏策略开启/关闭：

ALTER MASKING POLICY policy\_name [ENABLE | DISABLE];

**参数说明**

* **policy\_name**
* 脱敏策略名称，需要唯一，不可重复。
* 取值范围：字符串，要符合标识符的命名规范。
* **policy\_comments**
* 需要为脱敏策略添加或修改的描述信息。
* **masking\_function**
* 指的是预置的八种脱敏方式或者用户自定义的函数，支持模式。
* maskall不是预置函数，硬编码在代码中，不支持df展示。
* 预置时脱敏方式如下：

maskall | randommasking | creditcardmasking | basicemailmasking | fullemailmasking | shufflemasking | alldigitsmasking | regexpmasking

* **label\_name**
* 资源标签名称。
* **FILTER\_TYPE**
* 指定脱敏策略的过滤信息，过滤类型包括：IP、ROLES、APP。
* **filter\_value**
* 指具体过滤信息内容，例如具体的IP、具体的APP名称、具体的用户名。
* **ENABLE|DISABLE**
* 可以打开或关闭脱敏策略。若不指定ENABLE|DISABLE，语句默认为ENABLE。

**示例**

--创建dev\_mask和bob\_mask用户。

vastbase=# CREATE USER dev\_mask PASSWORD 'dev@1234';

vastbase=# CREATE USER bob\_mask PASSWORD 'bob@1234';

--创建一个表tb\_for\_masking

vastbase=# CREATE TABLE tb\_for\_masking(col1 text, col2 text, col3 text);

--创建资源标签标记敏感列col1

vastbase=# CREATE RESOURCE LABEL mask\_lb1 ADD COLUMN(tb\_for\_masking.col1);

--创建资源标签标记敏感列col2

vastbase=# CREATE RESOURCE LABEL mask\_lb2 ADD COLUMN(tb\_for\_masking.col2);

--对访问敏感列col1的操作创建脱敏策略

vastbase=# CREATE MASKING POLICY maskpol1 maskall ON LABEL(mask\_lb1);

--为脱敏策略maskpol1添加描述

vastbase=# ALTER MASKING POLICY maskpol1 COMMENTS 'masking policy for tb\_for\_masking.col1';

--修改脱敏策略maskpol1，新增一项脱敏方式

vastbase=# ALTER MASKING POLICY maskpol1 ADD randommasking ON LABEL(mask\_lb2);

--修改脱敏策略maskpol1，移除一项脱敏方式

vastbase=# ALTER MASKING POLICY maskpol1 REMOVE randommasking ON LABEL(mask\_lb2);

--修改脱敏策略maskpol1，修改一项脱敏方式

vastbase=# ALTER MASKING POLICY maskpol1 MODIFY randommasking ON LABEL(mask\_lb1);

--修改脱敏策略maskpol1使之仅对用户dev\_mask和bob\_mask,客户端工具为psql和vsql，IP地址为'10.20.30.40', '127.0.0.0/24'场景生效。

vastbase=# ALTER MASKING POLICY maskpol1 MODIFY (FILTER ON ROLES(dev\_mask, bob\_mask), APP(psql, vsql), IP('10.20.30.40', '127.0.0.0/24'));

--修改脱敏策略maskpol1，使之对所有用户场景生效

vastbase=# ALTER MASKING POLICY maskpol1 DROP FILTER;

--禁用脱敏策略maskpol1

vastbase=# ALTER MASKING POLICY maskpol1 DISABLE;

**相关链接**

CREATE MASKING POLICY (参见：开发者指南->SQL语法参考->SQL语法->CREATE MASKING POLICY章节),DROP MASKING POLICY(参见：开发者指南->SQL语法参考->SQL语法->DROP MASKING POLICY章节)

### ALTER MATERIALIZED VIEW

**功能描述**

更改一个现有物化视图的多个辅助属性。

可用于ALTER MATERIALIZED VIEW的语句形式和动作是ALTER TABLE的一个子集，并且在用于物化视图时具有相同的含义。详见开发者指南->SQL语法参考->SQL语法->ALTER TABLE章节。

**注意事项**

* 只有物化视图的所有者有权限执行ALTER TMATERIALIZED VIEW命令，系统管理员默认拥有此权限。
* 不支持更改物化视图结构。

**语法格式**

* 修改物化视图的所属用户。

ALTER MATERIALIZED VIEW [ IF EXISTS ] mv\_name

OWNER TO new\_owner;

* 修改物化视图的列。

ALTER MATERIALIZED VIEW [ IF EXISTS ] mv\_name

RENAME [ COLUMN ] column\_name TO new\_column\_name;

* 重命名物化视图。

ALTER MATERIALIZED VIEW [ IF EXISTS ] mv\_name

RENAME TO new\_name;

**参数说明**

* **mv\_name**
* 一个现有物化视图的名称，可以用模式修饰。
* 取值范围：字符串，符合标识符命名规范。
* **column\_name**
* 一个新的或者现有的列的名称。
* 取值范围：字符串，符合标识符命名规范。
* **new\_column\_name**
* 一个现有列的新名称。
* **new\_owner**
* 该物化视图的新拥有者的用户名。
* **new\_name**
* 该物化视图的新名称。

**示例**

--把物化视图foo重命名为bar。

vastbase=# ALTER MATERIALIZED VIEW foo RENAME TO bar;

**相关链接**

CREATE MATERIALIZED VIEW (参见：开发者指南->SQL语法参考->SQL语法->CREATE MATERIALIZED VIEW章节)、CREATE INCREMENTAL MATERIALIZED VIEW (参见：开发者指南->SQL语法参考->SQL语法->CREATE INCREMENTAL MATERIALIZED VIEW章节)、DROP MATERIALIZED VIEW (参见：开发者指南->SQL语法参考->SQL语法->DROP MATERIALIZED VIEW章节)、REFRESH INCREMENTAL MATERIALIZED VIEW(参见：开发者指南->SQL语法参考->SQL语法->REFRESH INCREMENTAL MATERIALIZED VIEW章节) 、REFRESH MATERIALIZED VIEW (参见：开发者指南->SQL语法参考->SQL语法->REFRESH MATERIALIZED VIEW章节)

### ALTER PUBLICATION

**功能描述**

更改发布PUBLICATION的属性。

**注意事项**

发布的属主和系统管理员才能执行ALTER PUBLICATION。新所有者角色的直接或间接成员才可以改变所有者。新的所有者必须在当前数据库上拥有CREATE权限。此外，FOR ALL TABLES发布的新所有者必须是系统管理员。但是，系统管理员可以在避开这些限制的情况下更改发布的所有权。

**语法格式**

* 用指定的表替换当前发布的表。

ALTER PUBLICATION name SET TABLE table\_name [, ...]

* 从发布中添加一个或多个表。

ALTER PUBLICATION name ADD TABLE table\_name [, ...]

* 从发布中删除一个或多个表。

ALTER PUBLICATION name DROP TABLE table\_name [, ...]

* 改变在CREATE PUBLICATION中指定的所有发布属性，未提及的属性保留其之前的设置。

ALTER PUBLICATION name SET ( publication\_parameter [= value] [, ... ] )

* 更改发布的所有者。

ALTER PUBLICATION name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

* 更改发布的名称。

ALTER PUBLICATION name RENAME TO new\_name

**参数说明**

* **name**
* 待修改的发布的名称。
* **table\_name**
* 现有表的名称。
* **SET ( publication\_parameter [= value] [, ... ] )。**
* 该子句修改最初由CREATE PUBLICATION设置的发布参数。
* **new\_owner**
* 发布的新所有者的用户名。
* **new\_name**
* 发布的新名称。

**示例**

详情请参见CREATE-PUBLICATION的示例。

**相关链接**

CREATE PUBLICATION （参见：开发者指南->SQL语法参考->SQL语法->CREATE PUBLICATION 章节），DROP PUBLICATION （参见：开发者指南->SQL语法参考->SQL语法->DROP PUBLICATION 章节）。

### ALTER RESOURCE LABEL

**功能描述**

修改资源标签。

**注意事项**

只有poladmin、 sysadmin或初始用户才能执行此操作。

**语法格式**

ALTER RESOURCE LABEL label\_name (ADD|REMOVE)

label\_item\_list[, ...]\*;

* label\_item\_list：

resource\_type(resource\_path[, ...]\*)

* resource\_type：

TABLE | COLUMN | SCHEMA | VIEW | FUNCTION

**参数说明**

* **label\_name**
* 资源标签名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **resource\_type**
* 指的是要标记的数据库资源类型。
* **resource\_path**
* 指的是描述具体的数据库资源的路径。

**示例**

--创建基本表table\_for\_label。

vastbase=# CREATE TABLE table\_for\_label(col1 int, col2 text);

--创建资源标签table\_label。

vastbase=# CREATE RESOURCE LABEL table\_label ADD COLUMN(table\_for\_label.col1);

--将col2添加至资源标签table\_label中

vastbase=# ALTER RESOURCE LABEL table\_label ADD COLUMN(table\_for\_label.col2)

--将资源标签table\_label中的一项移除

vastbase=# ALTER RESOURCE LABEL table\_label REMOVE COLUMN(table\_for\_label.col1);

**相关链接**

CREATE RESOURCE LABEL(参见：开发者指南->SQL语法参考->SQL语法->CREATE RESOURCE LABEL章节)，DROP RESOURCE LABEL(参见：开发者指南->SQL语法参考->SQL语法->DROP RESOURCE LABEL章节)。

### ALTER ROLE

**功能描述**

修改角色属性。

**语法格式**

* 修改角色的权限。

ALTER ROLE role\_name [ [ WITH ] option [ ... ] ];

其中权限项子句option为：

{CREATEDB | NOCREATEDB}

| {CREATEROLE | NOCREATEROLE}

| {INHERIT | NOINHERIT}

| {AUDITADMIN | NOAUDITADMIN}

| {SYSADMIN | NOSYSADMIN}

| {MONADMIN | NOMONADMIN}

| {OPRADMIN | NOOPRADMIN}

| {POLADMIN | NOPOLADMIN}

| {USEFT | NOUSEFT}

| {LOGIN | NOLOGIN}

| {REPLICATION | NOREPLICATION}

| {INDEPENDENT | NOINDEPENDENT}

| {VCADMIN | NOVCADMIN}

| {PERSISTENCE | NOPERSISTENCE}

| CONNECTION LIMIT connlimit

| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password' [EXPIRED]

| [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY 'password' [ REPLACE 'old\_password' | EXPIRED ]

| VALID BEGIN 'timestamp'

| VALID UNTIL 'timestamp'

| RESOURCE POOL 'respool'

| USER GROUP 'groupuser'

| PERM SPACE 'spacelimit'

| TEMP SPACE 'tmpspacelimit'

| SPILL SPACE 'spillspacelimit'

| NODE GROUP logic\_cluster\_name

| ACCOUNT { LOCK | UNLOCK }

| PGUSER

* 修改角色的名称。

ALTER ROLE role\_name

RENAME TO new\_name;

* 锁定或解锁。

ALTER ROLE role\_name

ACCOUNT { LOCK | UNLOCK };

* 设置角色的配置参数。

ALTER ROLE role\_name [ IN DATABASE database\_name ]

SET configuration\_parameter {{ TO | = } { value | DEFAULT } | FROM CURRENT};

* 重置角色的配置参数。

ALTER ROLE role\_name

[ IN DATABASE database\_name ] RESET {configuration\_parameter|ALL};

**参数说明**

* **role\_name**

现有角色名。

* 取值范围：已存在的用户名。
* **IN DATABASE database\_name**

表示修改角色在指定数据库上的参数。

* **SET configuration\_parameter**

设置角色的参数。ALTER ROLE中修改的会话参数只针对指定的角色， 且在下一次该角色启动的会话中有效。

* 取值范围：configuration\_parameter和value的取值请参见：开发者指南->SQL语法参考->SQL语法->SET章节。
* DEFAULT：表示清除configuration\_parameter参数的值，configuration\_parameter参数的值将继承本角色新产生的SESSION的默认值。
* FROM CURRENT：取当前会话中的值设置为configuration\_parameter参数的值。
* **RESET configuration\_parameter/ALL**

清除configuration\_parameter参数的值。与SET configuration\_parameter TO DEFAULT的效果相同。

* 取值范围：ALL表示清除所有参数的值。
* **ACCOUNT LOCK | ACCOUNT UNLOCK**
* ACCOUNT LOCK：锁定帐户，禁止登录数据库。
* ACCOUNT UNLOCK：解锁帐户，允许登录数据库。
* **PGUSER**

当前版本不允许修改角色的PGUSER属性。

* **PASSWORD/IDENTIFIED BY 'password'**

重置或修改用户密码。除了初始用户外其他管理员或普通用户修改自己 的密码需要输入正确的旧密码。只有初始用户、系统管理员 （sysadmin）或拥有创建用户（CREATEROLE）权限的用户才可以重置 普通用户密码，无需输入旧密码。初始用户可以重置系统管理员的密码， 系统管理员不允许重置其他系统管理员的密码。

* **EXPIRED**

设置密码失效。只有初始用户、系统管理员（sysadmin）或拥有创建用 户（CREATEROLE）权限的用户才可以设置用户密码失效，其中系统管 理员也可以设置自己或其他系统管理员密码失效。不允许设置初始用户 密码失效。密码失效的用户可以登录数据库但不能执行查询操作，只有 修改密码或由管理员重置密码后才可以恢复正常查询操作。

其他参数请参见“CREATE ROLE”的参数说明部分。

**注意事项**

只允许修改用户的标识，但是不允许修改属性标识。

**示例**

* 修改角色manager的密码为abcd@123。

ALTER ROLE manager IDENTIFIED BY 'abcd@123' REPLACE 'Bigdata@123';

* 修改角色manager为系统管理员。

ALTER ROLE manager SYSADMIN;

### ALTER ROW LEVEL SECURITY POLICY

**功能描述**

对已存在的行访问控制策略（包括行访问控制策略的名称、行访问控制指定的用户、行访问控制的策略表达式）进行修改。

**注意事项**

表的所有者或管理员用户才能进行此操作。

**语法格式**

ALTER [ ROW LEVEL SECURITY ] POLICY [ IF EXISTS ] policy\_name ON table\_name RENAME TO new\_policy\_name;

ALTER [ ROW LEVEL SECURITY ] POLICY policy\_name ON table\_name

[ TO { role\_name | PUBLIC } [, ...] ]

[ USING ( using\_expression ) ];

**参数说明**

* **policy\_name**
* 行访问控制策略名称。
* **table\_name**
* 行访问控制策略的表名。
* **new\_policy\_name**
* 新的行访问控制策略名称。
* **role\_name**
* 行访问控制策略应用的数据库用户，可以指定多个用户，PUBLIC表示应用到所有用户。
* **using\_expression**
* 行访问控制的表达式，返回值为boolean类型。

**示例**

--创建数据表all\_data

vastbase=# CREATE TABLE all\_data(id int, role varchar(100), data varchar(100));

--创建行访问控制策略，当前用户只能查看用户自身的数据

vastbase=# CREATE ROW LEVEL SECURITY POLICY all\_data\_rls ON all\_data USING(role = CURRENT\_USER);

vastbase=# d+ all\_data

Table "public.all\_data"

Column | Type | Modifiers | Storage | Stats target | Description

--------+------------------------+-----------+----------+--------------+-------------

id | integer | | plain | |

role | character varying(100) | | extended | |

data | character varying(100) | | extended | |

Row Level Security Policies:

POLICY "all\_data\_rls"

USING (((role)::name = "current\_user"()))

Has OIDs: no

Location Nodes: ALL DATANODES

Options: orientation=row, compression=no

--修改行访问控制all\_data\_rls的名称

vastbase=# ALTER ROW LEVEL SECURITY POLICY all\_data\_rls ON all\_data RENAME TO all\_data\_new\_rls;

--修改行访问控制策略影响的用户

vastbase=# ALTER ROW LEVEL SECURITY POLICY all\_data\_new\_rls ON all\_data TO alice, bob;

vastbase=# d+ all\_data

Table "public.all\_data"

Column | Type | Modifiers | Storage | Stats target | Description

--------+------------------------+-----------+----------+--------------+-------------

id | integer | | plain | |

role | character varying(100) | | extended | |

data | character varying(100) | | extended | |

Row Level Security Policies:

POLICY "all\_data\_new\_rls"

TO alice,bob

USING (((role)::name = "current\_user"()))

Has OIDs: no

Location Nodes: ALL DATANODES

Options: orientation=row, compression=no, enable\_rowsecurity=true

--修改行访问控制策略表达式

vastbase=# ALTER ROW LEVEL SECURITY POLICY all\_data\_new\_rls ON all\_data USING (id > 100 AND role = current\_user);

vastbase=# d+ all\_data

Table "public.all\_data"

Column | Type | Modifiers | Storage | Stats target | Description

--------+------------------------+-----------+----------+--------------+-------------

id | integer | | plain | |

role | character varying(100) | | extended | |

data | character varying(100) | | extended | |

Row Level Security Policies:

POLICY "all\_data\_new\_rls"

TO alice,bob

USING (((id > 100) AND ((role)::name = "current\_user"())))

Has OIDs: no

Location Nodes: ALL DATANODES

Options: orientation=row, compression=no, enable\_rowsecurity=true

**相关链接**

CREATE ROW LEVEL SECURITY POLICY(参见：开发者指南->SQL语法参考->SQL语法->CREATE ROW LEVEL SECURITY POLICY章节)，DROP ROW LEVEL SECURITY POLICY(参见：开发者指南->SQL语法参考->SQL语法->DROP ROW LEVEL SECURITY POLICY章节)

### ALTER PROCEDURE

**功能描述**

修改自定义存储过程的属性。

**注意事项**

只有存储过程的所有者或者被授予了存储过程ALTER权限的用户才能执行ALTER PROCEDURE命令，系统管理员默认拥有该权限。针对所要修改属性的不同，还有以下权限约束：

* 如果存储过程中涉及对临时表相关的操作，则无法使用ALTER PROCEDURE。
* 修改存储过程的所有者或修改存储过程的模式，当前用户必须是该存储过程的所有者或者系统管理员，且该用户是新所有者角色的成员。
* 只有系统管理员和初始化用户可以将procedure的schema修改成public。

**语法格式**

* 修改自定义存储过程的附加参数。

ALTER PROCEDURE procedure\_name ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

action [ ... ] [ RESTRICT ];

其中自定义存储过程的附加参数action子句语法为：

[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER

SET configuration\_parameter { TO | = } { value | DEFAULT }

SET configuration\_parameter FROM CURRENT

RESET configuration\_parameter

RESET ALL

* 修改自定义存储过程的名称。

ALTER PROCEDURE proname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

RENAME TO new\_name;

* 修改自定义存储过程的所属者。

ALTER PROCEDURE proname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

OWNER TO new\_owner;

* 修改自定义存储过程的模式。

ALTER PROCEDURE proname ( [ { [ argname ] [ argmode ] argtype} [, ...] ] )

SET SCHEMA new\_schema;

* 修改自定义存储过程的插件。

ALTER PROCEDURE name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]

[ NO ] DEPENDS ON EXTENSION extension\_name;

**参数说明**

* **procedure\_name**
* 要修改的存储过程名称。
* 取值范围：已存在的存储过程名。
* **argmode**
* 标识该参数是输入、输出参数。
* 取值范围：IN/OUT/INOUT/VARIADIC。
* **argname**
* 参数名称。
* 取值范围：字符串，符合标识符命名规范。
* **argtype**
* 存储过程参数的类型。
* **new\_name**
* 存储过程的新名称。要修改存储过程的所属模式，必须拥有新模式的CREATE权限。
* 取值范围：字符串，符合标识符命名规范。
* **new\_owner**
* 存储过程的新所有者。要修改存储过程的所有者，新所有者必须拥有该存储过程所属模式的CREATE权限。
* 取值范围：已存在的用户角色。
* **new\_schema**
* 存储过程的新模式。
* 取值范围：已存在的模式。
* **extension\_name**

已安装扩展的名称。

**示例**

1、定义存储过程为SQL查询。

CREATE PROCEDURE func\_add\_sql(integer, integer) RETURNS integer  
   AS 'select $1 + $2;'  
   LANGUAGE SQL  
   IMMUTABLE  
   RETURNS NULL ON NULL INPUT;

2、利用参数名用 PL/pgSQL 自增一个整数。

CREATE OR REPLACE PROCEDURE func\_increment\_plsql(i integer) RETURNS integer AS $$  
       BEGIN  
               RETURN i + 1;  
       END;  
$$ LANGUAGE plpgsql;

3、返回RECORD类型

CREATE OR REPLACE PROCEDURE func\_increment\_sql(i int, out result\_1 bigint, out result\_2 bigint)  
returns SETOF RECORD  
as $$  
begin  
   result\_1 = i + 1;  
   result\_2 = i \* 10;  
return next;  
end;  
$$language plpgsql;

4、返回一个包含多个输出参数的记录。

CREATE PROCEDURE func\_dup\_sql(in int, out f1 int, out f2 text)  
   AS $$ SELECT $1, CAST($1 AS text) || ' is text' $$  
   LANGUAGE SQL;  
      
SELECT \* FROM func\_dup\_sql(42);

返回结果为：

f1 |     f2  
----+------------  
 42 | 42 is text  
(1 row)

5、计算两个整数的和，并返回结果。如果输入为null，则返回null。

CREATE PROCEDURE func\_add\_sql2(num1 integer, num2 integer) RETURN integer  
AS  
BEGIN   
RETURN num1 + num2;  
END;  
/

6、修改函数func\_add\_sql2的执行规则为IMMUTABLE，即参数不变时返回相同结果。

ALTER PROCEDURE func\_add\_sql2(INTEGER, INTEGER) IMMUTABLE;

7、将函数func\_add\_sql2的名称修改为add\_two\_number。

ALTER PROCEDURE func\_add\_sql2(INTEGER, INTEGER) RENAME TO add\_two\_number

8、将函数add\_two\_number的属者改为vastbase。

ALTER PROCEDURE add\_two\_number(INTEGER, INTEGER) OWNER TO vastbase;

9、删除函数。

DROP PROCEDURE add\_two\_number;  
DROP PROCEDURE func\_increment\_sql;  
DROP PROCEDURE func\_dup\_sql;  
DROP PROCEDURE func\_increment\_plsql;  
DROP PROCEDUREfunc\_add\_sql;

**相关链接**

CREATE PROCEDURE(参见：开发者指南->SQL语法参考->SQL语法->CREATE PROCEDURE章节)，DROP PROCEDURE(参见：开发者指南->SQL语法参考->SQL语法->DROP PROCEDURE章节)

### ALTER SCHEMA

**功能描述**

修改模式属性。

**注意事项**

* 只有模式的所有者或者被授予了模式ALTER权限的用户有权限执行ALTER SCHEMA命令，系统管理员默认拥有此权限。但要修改模式的所有者，当前用户必须是该模式的所有者或者系统管理员，且该用户是新所有者角色的成员。
* 对于系统模式pg\_catalog，只允许初始用户修改模式的所有者。

**语法格式**

* 修改模式的防篡改属性。

ALTER SCHEMA schema\_name { WITH | WITHOUT } BLOCKCHAIN

* 修改模式的名称。

ALTER SCHEMA schema\_name

RENAME TO new\_name;

* 修改模式的所有者。

ALTER SCHEMA schema\_name

OWNER TO new\_owner;

**参数说明**

* **schema\_name**
* 现有模式的名称。
* 取值范围：已存在的模式名。
* **RENAME TO new\_name**
* 修改模式的名称。非系统管理员要改变模式的名称，则该用户必须在此数据库上有CREATE权限。
* new\_name：模式的新名称。
* 取值范围：字符串，要符合标识符命名规范。
* **OWNER TO new\_owner**
* 修改模式的所有者。非系统管理员要改变模式的所有者，该用户还必须是新的所有角色的直接或间接成员， 并且该成员必须在此数据库上有CREATE权限。
* new\_owner：模式的新所有者。
* 取值范围：已存在的用户名/角色名。
* **{ WITH | WITHOUT } BLOCKCHAIN**
* 修改模式的防篡改属性。具有防篡改属性模式下的普通行存表均为防篡改历史表，不包括外表、临时表、系统表。当该模式下不包含任何表时才可修改防篡改属性。另外，不支持临时表模式。toast表模式、dbe\_perf模式、blockchain模式修改防篡改属性。

**示例**

--创建模式ds。

vastbase=# CREATE SCHEMA ds;

--将当前模式ds更名为ds\_new。

vastbase=# ALTER SCHEMA ds RENAME TO ds\_new;

--创建用户jack。

vastbase=# CREATE USER jack PASSWORD 'xxxxxxxxx';

--将DS\_NEW的所有者修改为jack。

vastbase=# ALTER SCHEMA ds\_new OWNER TO jack;

--删除用户jack和模式ds\_new。

vastbase=# DROP SCHEMA ds\_new;

vastbase=# DROP USER jack;

**相关链接**

CREATE SCHEMA (参见：开发者指南->SQL语法参考->SQL语法->CREATE SCHEMA 章节)，DROP SCHEMA (参见：开发者指南->SQL语法参考->SQL语法->DROP SCHEMA章节)

### ALTER SEQUENCE

**功能描述**

修改一个现有的序列的参数。

**注意事项**

* 只有序列的所有者或者被授予了序列ALTER权限的用户才能执行ALTER SEQUENCE命令，系统管理员默认拥有该权限。但要修改序列的所有者，当前用户必须是该序列的所有者或者系统管理员，且该用户是新所有者角色的成员。
* 当前版本仅支持修改拥有者、归属列和最大值。若要修改其他参数，可以删除重建，并用Setval函数恢复当前值。
* ALTER SEQUENCE MAXVALUE不支持在事务、函数和存储过程中使用。
* 修改序列的最大值后，会清空该序列在所有会话的cache。
* 如果Sequence被创建时使用了LARGE标识，则ALTER时也需要使用LARGE标识。
* ALTER SEQUENCE会阻塞nextval、setval、currval和lastval的调用。

**语法格式**

* 修改序列归属列

ALTER SEQUENCE [ IF EXISTS ] name

[MAXVALUE maxvalue | NO MAXVALUE | NOMAXVALUE | CACHE cache]

[ OWNED BY { table\_name.column\_name | NONE } ] ;

* 修改序列的拥有者

ALTER SEQUENCE [ IF EXISTS ] name OWNER TO new\_owner;

**参数说明**

* **name**
* 将要修改的序列名称。
* **IF EXISTS**
* 当序列不存在时使用该选项不会出现错误消息，仅有一个通知。
* **CACHE**
* 为了快速访问，而在内存中预先存储序列号的个数。如果没有指定，将保持旧的缓冲值。
* **OWNED BY**
* 将序列和一个表的指定字段进行关联。这样，在删除那个字段或其所在表的时候会自动删除已关联的序列。
* 如果序列已经和表有关联后，使用这个选项后新的关联关系会覆盖旧的关联。
* 关联的表和序列的所有者必须是同一个用户，并且在同一个模式中。
* 使用OWNED BY NONE将删除任何已经存在的关联。
* **new\_owner**
* 序列新所有者的用户名。用户要修改序列的所有者，必须是新角色的直接或者间接成员，并且那个角色必须有序列所在模式上的CREATE权限。

**示例**

--创建一个名为serial的递增序列，从101开始。

vastbase=# CREATE SEQUENCE serial START 101;

--创建一个表,定义默认值。

vastbase=# CREATE TABLE T1(C1 bigint default nextval('serial'));

--将序列serial的归属列变为T1.C1。

vastbase=# ALTER SEQUENCE serial OWNED BY T1.C1;

--删除序列和表。

vastbase=# DROP SEQUENCE serial cascade;

vastbase=# DROP TABLE T1;

**相关链接**

CREATE SEQUENCE(参见：开发者指南->SQL语法参考->SQL语法->CREATE SEQUENCE 章节)，DROP SEQUENCE(参见：开发者指南->SQL语法参考->SQL语法->DROP SEQUENCE 章节)

### ALTER SERVER

**功能描述**

增加、修改和删除一个现有server的参数。已有server可以从pg\_foreign\_server系统表中查询。

**注意事项**

只有SERVER的所有者或者被授予了SERVER的ALTER权限的用户才可以执行ALTER SERVER命令，系统管理员默认拥有该权限。但要修改SERVER的所有者，当前用户必须是该SERVER的所有者或者系统管理员，且该用户是新所有者角色的成员。

**语法格式**

* 修改外部服务的参数。

ALTER SERVER server\_name [ VERSION 'new\_version' ]

[ OPTIONS ( {[ ADD | SET | DROP ] option ['value']} [, ... ] ) ];

在OPTIONS选项里，ADD、SET和DROP指定要执行的操作，未指定时默认为ADD操作。option和value为对应操作的参数。

* 修改外部服务的名称。

ALTER SERVER server\_name

RENAME TO new\_name;

**参数说明**

* **server\_name**
* 所修改的server的名称。
* **new\_version**
* 修改后server的新版本名称。
* **OPTIONS**
* 更改该服务器的选项。ADD、SET和 DROP指定要执行的动作。如果没有显式地指定操作， 将会假定为ADD。选项名称必须唯一，名称和值也会使用该服务器的外部数据包装器库进行验证。
* oracle\_fdw支持的options包括：
* **dbserver**

远端oracle数据库的连接字符串。

* **isolation\_level** （默认值为serializable）

oracle数据库的事务隔离级别。

取值范围：serializable、 read\_committed 、 read\_only

* mysql\_fdw支持的options包括：
* **host** （默认值为 127.0.0.1）

MySQL Server/MariaDB的地址。

* **port** （默认值为 3306）

MySQL Server/MariaDB侦听的端口号。

* postgres\_fdw支持的options同libpq支持的连接参数一致。需要注意的是，以下几个options不支持修改：
* **user**和**password**

用户名和密码将在创建user mapping时指定。

* **client\_encoding**

将自动获取本地server的编码方式并设置该值。

* **application\_name**

总是设置成postgres\_fdw。

* 除了libpq支持的连接参数外，还额外提供3个options：
* **use\_remote\_estimate**

控制postgres\_fdw是否发出EXPLAIN命令以获取运行消耗估 算。默认值为false。

* **fdw\_startup\_cost**

执行一个外表扫描时的启动耗时估算。这个值通常包含建立连接、 远端对请求的分析和生成计划的耗时。默认值为100。

* **fdw\_typle\_cost**

在远端服务器上对每一个元组进行扫描时的额外消耗。这个值通 常表示数据在server间传输的额外消耗。默认值为0.01。

* **new\_name**
* 修改后server的新名称。

**相关链接**

CREATE SERVER (参见：开发者指南->SQL语法参考->SQL语法->CREATE SERVER章节)，DROP SERVER (参见：开发者指南->SQL语法参考->SQL语法->DROP SERVER章节)

### ALTER SESSION

**功能描述**

ALTER SESSION命令用于定义或修改那些对当前会话有影响的条件或参数。修改后的会话参数会一直保持，直到断开当前会话。

**注意事项**

* 如果执行SET TRANSACTION之前没有执行START TRANSACTION，则事务立即结束，命令无法显示效果。
* 可以用START TRANSACTION里面声明所需要的transaction\_mode(s)的方法来避免使用SET TRANSACTION。

**语法格式**

* 设置会话的事务参数。

ALTER SESSION SET [ SESSION CHARACTERISTICS AS ] TRANSACTION

{ ISOLATION LEVEL { READ COMMITTED } | { READ ONLY | READ WRITE } } [, ...] ;

* 设置会话的其他运行时参数。

ALTER SESSION SET

{{config\_parameter { { TO | = } { value | DEFAULT }

| FROM CURRENT }}

| TIME ZONE time\_zone

| CURRENT\_SCHEMA schema

| NAMES encoding\_name

| ROLE role\_name PASSWORD 'password'

| SESSION AUTHORIZATION { role\_name PASSWORD 'password' | DEFAULT | XML OPTION { DOCUMENT | CONTENT }

} ;

**参数说明**

修改会话涉及到的参数说明请参见SET语法 (参见：开发者指南->SQL语法参考->SQL语法->SET章节)中的参数说明。

**示例**

-- 创建模式ds。

vastbase=# CREATE SCHEMA ds;

--设置模式搜索路径。

vastbase=# SET SEARCH\_PATH TO ds, public;

--设置日期时间风格为传统的POSTGRES风格（日在月前）。

vastbase=# SET DATESTYLE TO postgres, dmy;

--设置当前会话的字符编码为UTF8。

vastbase=# ALTER SESSION SET NAMES 'UTF8';

--设置时区为加州伯克利。

vastbase=# SET TIME ZONE 'PST8PDT';

--设置时区为意大利。

vastbase=# SET TIME ZONE 'Europe/Rome';

--设置当前模式。

vastbase=# ALTER SESSION SET CURRENT\_SCHEMA TO tpcds;

--设置XML OPTION为DOCUMENT。

vastbase=# ALTER SESSION SET XML OPTION DOCUMENT;

--创建角色joe，并设置会话的角色为joe。

vastbase=# CREATE ROLE joe WITH PASSWORD 'xxxxxxxxx';

vastbase=# ALTER SESSION SET SESSION AUTHORIZATION joe PASSWORD 'xxxxxxxxx';

--切换到默认用户。

vastbase=> ALTER SESSION SET SESSION AUTHORIZATION default;

--删除ds模式。

vastbase=# DROP SCHEMA ds;

--删除joe。

vastbase=# DROP ROLE joe;

**相关链接**

SET (参见：开发者指南->SQL语法参考->SQL语法->SET章节)

### ALTER SUBSCRIPTION

**功能描述**

ALTER SUBSCRIPTION可以修改在CREATE SUBSCRIPTION中指定的订阅属性。

**注意事项**

订阅的所有者才能执行ALTER SUBSCRIPTION，并且新的所有者必须是系统管理员。

**语法格式**

* 更新订阅的连接信息。

ALTER SUBSCRIPTION name CONNECTION 'conninfo'

* 更新订阅的发布端的发布名称。

ALTER SUBSCRIPTION name SET PUBLICATION publication\_name [, ...]

* 更新订阅的发布端的发布名称。

ALTER SUBSCRIPTION name REFRESH PUBLICATION [ WITH ( refresh\_option [= value] [, ... ] ) ]

* 激活订阅。

ALTER SUBSCRIPTION name ENABLE

* 更新CREATE SUBSCRIPTION中定义的属性。

ALTER SUBSCRIPTION name SET ( subscription\_parameter [= value] [, ... ] )

* 更新订阅的属主。

ALTER SUBSCRIPTION name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

* 修改订阅的名称。

ALTER SUBSCRIPTION name RENAME TO new\_name

**参数说明**

* **name**
* 要修改属性的订阅的名称。
* **CONNECTION 'conninfo'**
* 该子句修改最初由CREATE SUBSCRIPTION设置的连接属性。
* **ENABLE (boolean)**
* 指定订阅是否应该主动复制，或者是否应该只是设置，但尚未启动。默认值是true。
* **SET ( subscription\_parameter [= value] [, ... ] )**
* 该子句修改原先由CREATE SUBSCRIPTION设置的参数。允许的选项是slot\_name和synchronous\_commit。
* 如果创建订阅时设置enabled为false，则slot\_name将被强制设置为NONE，即空值，即使用户指定了slot\_name的值，复制槽也不存在。
* 将enabled参数的值由false改为true，即启用订阅时，将会连接发布端创建复制槽，此时如果用户未指定slot\_name参数的值，则会使用默认值，即对应的订阅的名称。
* 当enabled为true，即订阅处于正常使用状态，不能修改slot\_name为空，但可以修改复制槽的名称为其他非空合法名称。
* **REFRESH PUBLICATION**
* 从发布端获取缺少的表信息。这将开始复制自上次调用REFRESH PUBLICATION或从CREATE SUBSCRIPTION以来添加到订阅发布中的表。
* refresh\_option指定了刷新操作的附加选项。支持的选项有：
* copy\_data (boolean)
* 指定在复制启动后是否应复制正在订阅的发布中的现有数据。默认值是true。（以前订阅的表不会被复制）
* **new\_owner**
* 订阅的新所有者的用户名。
* **new\_name**
* 订阅的新名称。

**示例**

请参见CREATE SUBSCRIPTION的示例。

**相关链接**

CREATE SUBSCRIPTION (参见：开发者指南->SQL语法参考->SQL语法->CREATE SUBSCRIPTION 章节)，DROP SUBSCRIPTION(参见：开发者指南->SQL语法参考->SQL语法->DROP SUBSCRIPTION 章节)

### ALTER SYNONYM

**功能描述**

修改SYNONYM对象的属性。

**注意事项**

* 目前仅支持修改SYNONYM对象的属主。
* 只有系统管理员有权限修改SYNONYM对象的属主信息。
* 新属主必须具有SYNONYM对象所在模式的CREATE权限。

**语法格式**

ALTER SYNONYM synonym\_name OWNER TO new\_owner;

**参数描述**

* **synonym**
* 待修改的同义词名字，可以带模式名。
* 取值范围：字符串，需要符合标识符的命名规范。
* **new\_owner**
* 同义词对象的新所有者。
* 取值范围：字符串，有效的用户名。

**示例**

1、创建同义词t1。

CREATE OR REPLACE SYNONYM t1 FOR t1;

2、创建新用户u1，将授予public模式的CREATE权限。

CREATE USER u1 PASSWORD 'user@111';

GRANT CREATE ON SCHEMA public to u1;

3、修改同义词t1的owner为u1。

ALTER SYNONYM t1 OWNER TO u1;

4、删除同义词t1。

DROP SYNONYM t1;

5、删除用户u1。

DROP USER u1 CASCADE;

**相关链接**

CREATE SYNONYM (参见：开发者指南->SQL语法参考->SQL语法->CREATE SYNONYM 章节)，DROP SYNONYM(参见：开发者指南->SQL语法参考->SQL语法->DROP SYNONYM章节)

### ALTER SYSTEM KILL SESSION

**功能描述**

ALTER SYSTEM KILL SESSION命令用于结束一个会话。

**注意事项**

无。

**语法格式**

ALTER SYSTEM KILL SESSION 'session\_sid, serial' [ IMMEDIATE ];

**参数说明**

* **session\_sid, serial**
* 会话的SID和SERIAL（获取方法请参考示例）。
* **IMMEDIATE**
* 表明会话将在命令执行后立即结束。

**示例**

--查询会话信息。

vastbase=#

SELECT sa.sessionid AS sid,0::integer AS serial#,ad.rolname AS username FROM pg\_stat\_get\_activity(NULL) AS sa

LEFT JOIN pg\_authid ad ON(sa.usesysid = ad.oid)WHERE sa.application\_name <> 'JobScheduler';

sid | serial# | username

-----------------+---------+----------

140131075880720 | 0 | omm

140131025549072 | 0 | omm

140131073779472 | 0 | omm

140131071678224 | 0 | omm

140131125774096 | 0 |

140131127875344 | 0 |

140131113629456 | 0 |

140131094742800 | 0 |

(8 rows)

--结束SID为140131075880720的会话。

vastbase=# ALTER SYSTEM KILL SESSION '140131075880720,0' IMMEDIATE;

### ALTER SYSTEM SET

**功能描述**

ALTER SYSTEM SET命令用于设置POSTMASTER、SIGHUP、BACKEND级别的GUC参数。此命令会将参数写入配置文件，不同级别生效方式有所不同。

**注意事项**

* 此命令仅限初始用户和拥有sysadmin权限的用户才可使用。
* 不同级别GUC参数生效时间如下：
* POSTMASTER级别的GUC参数需要重启后才生效。
* BACKEND级别的GUC参数需要会话重新连接后才生效。
* SIGHUP级别的GUC参数立即生效（需要等待线程重新加载参数，实际略微有延迟）。
* 通过配置audit\_set\_parameter参数 (参见：开发者指南->GUC参数说明->审计->操作审计章节的audit\_set\_parameter参数)，可以配置此操作是否被审计。
* 操作可被备机同步。
* 同vb\_guc一致，并不关注数据库是主或备节点、是否只读。
* 不可在事务中执行，因为此操作无法被回滚。
* 部分参数只能由初始用户修改，具体如下：
* audit\_copy\_exec, audit\_data\_format, audit\_database\_process, audit\_directory, audit\_dml\_state,
* audit\_dml\_state\_select, audit\_enabled, audit\_file\_remain\_threshold, audit\_file\_remain\_time,
* audit\_function\_exec, audit\_grant\_revoke, audit\_login\_logout, audit\_resource\_policy,
* audit\_rotation\_interval, audit\_rotation\_size, audit\_set\_parameter, audit\_space\_limit,
* audit\_system\_object, audit\_user\_locked, audit\_user\_violation,
* asp\_log\_directory, config\_file, data\_directory, enable\_access\_server\_directory,
* enable\_copy\_server\_files, external\_pid\_file, hba\_file, ident\_file, log\_directory, perf\_directory,
* query\_log\_directory, ssl\_ca\_file, ssl\_cert\_file, ssl\_crl\_file, ssl\_key\_file, stats\_temp\_directory,
* unix\_socket\_directory, unix\_socket\_group, unix\_socket\_permissions,
* krb\_caseins\_users, krb\_server\_keyfile, krb\_srvname, allow\_system\_table\_mods, enableSeparationOfDuty,
* modify\_initial\_password, password\_encryption\_type, password\_policy

**语法格式**

ALTER SYSTEM SET parameter TO value;

**参数说明**

* **parameter**
* GUC参数名。
* **value**
* GUC参数值。

**示例**

--设置SIGHUP级别参数audit\_enabled。

vastbase=# alter system set audit\_enabled to off;

ALTER SYSTEM SET

vastbase=# show audit\_enabled;

audit\_enabled

---------------

off

(1 row)

--设置POSTMASTER级别参数 enable\_thread\_pool，将在重启之后生效。

vastbase=# alter system set enable\_thread\_pool to on;

NOTICE: please restart the database for the POSTMASTER level parameter to take effect.

ALTER SYSTEM SET

### ALTER TABLE

**功能描述**

修改表，包括修改表的定义、重命名表、重命名表中指定的列、重命名表的约束、设置表的所属模式、添加/更新多个列、打开/关闭行访问控制开关。

**注意事项**

* 表的所有者被授予了表ALTER权限的用户或被授予ALTER ANY TABLE的用户有权限执行ALTER TABLE命令，系统管理员默认拥有此权限。但要修改表的所有者或者修改表的模式，当前用户必须是该表的所有者或者系统管理员，且该用户是新所有者角色的成员。
* 不能修改分区表的tablespace，但可以修改分区的tablespace。
* 不支持修改存储参数ORIENTATION。
* SET SCHEMA操作不支持修改为系统内部模式，当前仅支持用户模式之间的修改。
* 列存表只支持PARTIAL CLUSTER KEY、UNIQUE、PRIMARY KEY表级约束，不支持外键等表级约束。
* 列存表只支持添加字段ADD COLUMN、修改字段的数据类型ALTER TYPE、设置单个字段的收集目标SET STATISTICS、支持更改表名称、支持更改表空间、支持删除字段DROP COLUMN。对于添加的字段和修改的字段类型要求是列存支持的数据类型(参考：《[VASTDATA]Vastbase G100 V2.2兼容性手册->Msqk兼容性->数据类型》章节)ALTER TYPE的USING选项只支持常量表达式和涉及本字段的表达式，暂不支持涉及其他字段的表达式。
* 列存表支持的字段约束包括NULL、NOT NULL、DEFAULT常量值、UNIQUE和PRIMARY KEY；对字段约束的修改当前只支持对DEFAULT值的修改（SET DEFAULT）和删除（DROP DEFAULT），暂不支持对非空约束NULL/NOT NULL的修改。
* 不支持增加自增列，或者增加DEFAULT值中包含nextval()表达式的列。
* 不支持对外表、临时表开启行访问控制开关。
* 通过约束名删除PRIMARY KEY约束时，不会删除NOT NULL约束，如果有需要，请手动删除NOT NULL约束。
* 使用JDBC时，支持通过PrepareStatement对DEFAUTL值进行参数化设置。

**语法格式**

* 修改表的定义。
* ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name ) }
* action [, ... ];
* 其中具体表操作action可以是以下子句之一：
* column\_clause
* | ADD table\_constraint [ NOT VALID ]
* | ADD table\_constraint\_using\_index
* | VALIDATE CONSTRAINT constraint\_name
* | DROP CONSTRAINT [ IF EXISTS ] constraint\_name [ RESTRICT | CASCADE ]
* | CLUSTER ON index\_name
* | SET WITHOUT CLUSTER
* | SET ( {storage\_parameter = value} [, ... ] )
* | RESET ( storage\_parameter [, ... ] )
* | OWNER TO new\_owner
* | SET TABLESPACE new\_tablespace
* | SET {COMPRESS|NOCOMPRESS}
* | TO { GROUP groupname | NODE ( nodename [, ... ] ) }
* | ADD NODE ( nodename [, ... ] )
* | DELETE NODE ( nodename [, ... ] )
* | DISABLE TRIGGER [ trigger\_name | ALL | USER ]
* | ENABLE TRIGGER [ trigger\_name | ALL | USER ]
* | ENABLE REPLICA TRIGGER trigger\_name
* | ENABLE ALWAYS TRIGGER trigger\_name
* | DISABLE/ENABLE [ REPLICA | ALWAYS ] RULE
* | DISABLE ROW LEVEL SECURITY
* | ENABLE ROW LEVEL SECURITY
* | FORCE ROW LEVEL SECURITY
* | NO FORCE ROW LEVEL SECURITY
* | ENCRYPTION KEY ROTATION
* | SET WITH OIDS
* | SET WITHOUT OIDS
* | INHERIT parents
* | NO INHERIT parents
* | OF type\_name
* | NOT OF
* | REPLICA IDENTITY { DEFAULT | USING INDEX index\_name | FULL | NOTHING }

fig: **说明：**

* **ADD table\_constraint [ NOT VALID ]**

给表增加一个新的约束。

* **ADD table\_constraint\_using\_index**

根据已有唯一索引为表增加主键约束或唯一约束。

* **VALIDATE CONSTRAINT constraint\_name**

验证一个使用NOT VALID选项创建的检查类约束，通过扫描全表来保证所有记录都符合约束条件。如果约束已标记为有效时，什么操作也不会发生。

* **DROP CONSTRAINT [ IF EXISTS ] constraint\_name [ RESTRICT | CASCADE ]**

删除一个表上的约束。

* **CLUSTER ON index\_name**

为将来的CLUSTER（聚簇）操作选择默认索引。实际上并没有重新盘簇化处理该表。

* **SET WITHOUT CLUSTER**

从表中删除最新使用的CLUSTER索引。这样会影响将来那些没有声明索引的CLUSTER（聚簇）操作。

* **SET ( {storage\_parameter = value} [, ... ] )**

修改表的一个或多个存储参数。

* **RESET ( storage\_parameter [, ... ] )**

重置表的一个或多个存储参数。与SET一样，根据参数的不同可能需要重写表才能获得想要的效果。

* **OWNER TO new\_owner**

将表、序列、视图的属主改变成指定的用户。

* **SET TABLESPACE new\_tablespace**

这种形式将表空间修改为指定的表空间并将相关的数据文件移动到新的表空间。但是表上的所有索引都不会被移动，索引可以通过ALTER INDEX语法的SET TABLESPACE选项来修改索引的表空间。

* **SET {COMPRESS|NOCOMPRESS}**

修改表的压缩特性。表压缩特性的改变只会影响后续批量插入的数据的存储方式，对已有数据的存储毫无影响。也就是说，表压缩特性的修改会导致该表中同时存在着已压缩和未压缩的数据。行存表不支持压缩。

* **TO { GROUP groupname | NODE ( nodename [, ... ] ) }**

此语法仅在扩展模式（GUC参数support\_extended\_features为on时）下可用。该模式谨慎打开，主要供内部扩容工具使用，一般用户不应使用该模式。

* **ADD NODE ( nodename [, ... ] )**

此语法主要供内部扩容工具使用，一般用户不建议使用。

* **DELETE NODE ( nodename [, ... ] )**

此语法主要供内部缩容工具使用，一般用户不建议使用。

* **DISABLE TRIGGER [ trigger\_name | ALL | USER ]**

禁用trigger\_name所表示的单个触发器，或禁用所有触发器，或仅禁用用户触发器（此选项不包括内部生成的约束触发器，例如，可延迟唯一性和排除约束的约束触发器）。

应谨慎使用此功能，因为如果不执行触发器，则无法保证原先期望的约束的完整性。

* **ENABLE TRIGGER [ trigger\_name | ALL | USER ]**

启用trigger\_name所表示的单个触发器，或启用所有触发器，或仅启用用户触发器。

* **ENABLE REPLICA TRIGGER trigger\_name**

触发器触发机制受配置变量session\_replication\_role （参见：开发者指南->GUC参数说明->客户端连接缺省设置->语句行为章节）的影响，当复制角色为“origin”（默认值）或“local”时，将触发简单启用的触发器。

配置为ENABLE REPLICA的触发器仅在会话处于“replica”模式时触发。

* **ENABLE ALWAYS TRIGGER trigger\_name**

无论当前复制模式如何，配置为ENABLE ALWAYS的触发器都将触发。

* **DISABLE/ENABLE [ REPLICA | ALWAYS ] RULE**

配置属于表的重写规则,已禁用的规则对系统来说仍然是可见的，只是在查询重写期间不被应用。语义为关闭/启动规则。由于关系到视图的实现，ON SELECT规则不可禁用。 配置为ENABLE REPLICA的规则将会仅在会话为"replica" 模式时启动，而配置为ENABLE ALWAYS的触发器将总是会启动，不考虑当前复制模式。规则触发机制也受配置变量session\_replication\_role（参见：开发者指南->GUC参数说明->客户端连接缺省设置->语句行为章节）的影响，类似于上述触发器。

* **DISABLE/ENABLE ROW LEVEL SECURITY**

开启或关闭表的行访问控制开关。

当开启行访问控制开关时，如果未在该数据表定义相关行访问控制策略，数据表的行级访问将不受影响；如果关闭表的行访问控制开关，即使定义了行访问控制策略，数据表的行访问也不受影响。详细信息参见CREATE ROW LEVEL SECURITY POLICY (参见：开发者指南->SQL语法参考->SQL语法->CREATE ROW LEVEL SECURITY POLICY 章节)。

* **NO FORCE/FORCE ROW LEVEL SECURITY**

强制开启或关闭表的行访问控制开关。

默认情况，表所有者不受行访问控制特性影响，但当强制开启表的行访问控制开关时，表的所有者（不包含系统管理员用户）会受影响。系统管理员可以绕过所有的行访问控制策略，不受影响。

* **ENCRYPTION KEY ROTATION**
* 透明数据加密密钥轮转。只有在数据库开启透明加密功能，并且表的enable\_tde选项为on时才可以进行表的数据加密密钥轮转。执行密钥轮转操作后，系统会自动向KMS申请创建新的密钥。密钥轮转后，使用旧密钥加密的数据仍使用旧密钥解密，新写入的数据使用新密钥加密。为保证加密数据安全，用户可根据加密表的新增数据量大小定期更新密钥，建议更新周期为两到三年。
* **REPLICA IDENTITY {DEFAULT | USING INDEX index\_name | FULL | NOTHING}**

调整逻辑复制时写入WAL日志中的信息量，该选项仅在wal\_level配置为logical时才有效。 当原数据表发生更新时，默认的逻辑复制流只包含主键的历史记录，如果需要输出所需字段更新或删除的历史记录，可修改本参数。“DEFAULT”（非系统表的默认值）会记录主键字段的旧值。“USING INDEX”会记录名为index\_name索引包含的字段的旧值，索引的所有列必须NOT NULL。“FULL”记录了所有列的旧值。“NOTHING”（系统表默认值）不记录旧值的信息。

* **SET WITH OIDS**

在资料表中增加了一个OID系统栏位。如果资料表中已经有OID，则此语法什么都不改变。

* **SET WITHOUT OIDS**

从资料表中移除一处OID系统栏位。如果资料表中没有OID，则此语法什么都不改变。

* **INHERIT parent\_table**

将目标资料表加到指定的父资料表中成为新的子资料表。之后，针对父资料表的查询将会包含目标资料表的资料。要作为子资料表加入前，目标资料表必须已经包含父资料表的所有栏位。这些栏位必须具有可匹配的资料类别，并且如果他们在父资料表中具有NOT NULL的限制条件，那么他们必须在子资料表中也具有NOT NULL的限制条件。对于父资料表的所有CHECK限制条件，必须还有相对应的子资料表限制条件，除非父资料表中标记为不可继承。

* **NO INHERIT parent\_table**

从指定的父资料表的子资料表中产出目标资料表。针对父资料表的查询将不再包含从目标资料表中所产生的记录。

* **OF type\_name**

将表连接至一种复合类型，与CREATE TABLE OF选项创建表一样。表的字段的名称和类型必须精确匹配复合类型中的定义，不过oid系统字段允许不一样。表不能是从任何其他表继承的。这些限制确保CREATE TABLE OF选项允许一个相同的表定义。

* **NOT OF**

将一个与某类型进行关联的表进行关联的解除。

* **REPLICA IDENTITY { DEFAULT | USING INDEX index\_name | FULL | NOTHING }**

DEFAULT记录主键的列的旧值。USING INDEX记录命名索引覆盖的列的旧值，这些值必须是唯一的，不局部的，不可延迟的，并且仅包括标记为NOT NULL的列。FULL记录该行中所有列的旧值。NOTHING不记录有关旧行的信息。在所有情况下，除非该行的新旧版本中至少要记录的列之一不同，否则不会记录任何旧值。

其中列相关的操作column\_clause可以是以下子句之一：

ADD [ COLUMN ] column\_name data\_type [ compress\_mode ] [ COLLATE collation ] [ column\_constraint [ ... ] ]

| MODIFY column\_name data\_type

| MODIFY column\_name [ CONSTRAINT constraint\_name ] NOT NULL [ ENABLE ]

| MODIFY column\_name [ CONSTRAINT constraint\_name ] NULL

| DROP [ COLUMN ] [ IF EXISTS ] column\_name [ RESTRICT | CASCADE ]

| ALTER [ COLUMN ] column\_name [ SET DATA ] TYPE data\_type [ COLLATE collation ] [ USING expression ]

| ALTER [ COLUMN ] column\_name { SET DEFAULT expression | DROP DEFAULT }

| ALTER [ COLUMN ] column\_name { SET | DROP } NOT NULL

| ALTER [ COLUMN ] column\_name SET STATISTICS [PERCENT] integer

| ADD STATISTICS (( column\_1\_name, column\_2\_name [, ...] ))

| DELETE STATISTICS (( column\_1\_name, column\_2\_name [, ...] ))

| ALTER [ COLUMN ] column\_name SET ( {attribute\_option = value} [, ... ] )

| ALTER [ COLUMN ] column\_name RESET ( attribute\_option [, ... ] )

| ALTER [ COLUMN ] column\_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }

下是对于各种column\_clause子句含义的说明：

* **ADD [ COLUMN ] column\_name data\_type [ compress\_mode ] [ COLLATE collation ] [ column\_constraint [ ... ] ]**

向表中增加一个新的字段。用ADD COLUMN增加一个字段，所有表中现有行都初始化为该字段的缺省值（如果没有声明DEFAULT子句，值为NULL）。

* **ADD ( { column\_name data\_type [ compress\_mode ] } [, ...] )**

向表中增加多列。

* **MODIFY ( { column\_name data\_type | column\_name [ CONSTRAINT constraint\_name ] NOT NULL [ ENABLE ] | column\_name [ CONSTRAINT constraint\_name ] NULL } [, ...] )**

修改表已存在字段的数据类型。

* **DROP [ COLUMN ] [ IF EXISTS ] column\_name [ RESTRICT | CASCADE ]**

从表中删除一个字段，和这个字段相关的索引和表约束也会被自动删除。如果任何表之外的对象依赖于这个字段，必须声明CASCADE ，比如视图。

DROP COLUMN命令并不是物理上把字段删除，而只是简单地把它标记为对SQL操作不可见。随后对该表的插入和更新将在该字段存储一个NULL。因此，删除一个字段是很快的，但是它不会立即释放表在磁盘上的空间，因为被删除了的字段占据的空间还没有回收。这些空间将在执行VACUUM时而得到回收。

* **ALTER [ COLUMN ] column\_name [ SET DATA ] TYPE data\_type [ COLLATE collation ] [ USING expression ]**

改变表字段的数据类型。该字段涉及的索引和简单的表约束将被自动地转换为使用新的字段类型，方法是重新分析最初提供的表达式。

ALTER TYPE要求重写整个表的特性有时候是一个优点，因为重写的过程消除了表中没用的空间。比如，要想立刻回收被一个已经删除的字段占据的空间，最快的方法是

ALTER TABLE table ALTER COLUMN anycol TYPE anytype;

这里的anycol是任何在表中还存在的字段，而anytype是和该字段的原类型一样的类型。这样的结果是在表上没有任何可见的语意的变化，但是这个命令强制重写，这样就删除了不再使用的数据。

* **ALTER [ COLUMN ] column\_name { SET DEFAULT expression | DROP DEFAULT }**

为一个字段设置或者删除缺省值。请注意缺省值只应用于随后的INSERT命令，它们不会修改表中已经存在的行。也可以为视图创建缺省，这个时候它们是在视图的ON INSERT规则应用之前插入到INSERT句中的。

* **ALTER [ COLUMN ] column\_name { SET | DROP } NOT NULL**

修改一个字段是否允许NULL值或者拒绝NULL值。如果表在字段中包含非NULL，则只能使用SET NOT NULL。

* **ALTER [ COLUMN ] column\_name SET STATISTICS [PERCENT] integer**

为随后的ANALYZE操作设置针对每个字段的统计收集目标。目标的范围可以在0到10000之内设置。设置为-1时表示重新恢复到使用系统缺省的统计目标。

* **{ADD | DELETE} STATISTICS ((column\_1\_name, column\_2\_name [, ...]))**

用于添加和删除多列统计信息声明（不实际进行多列统计信息收集），以便在后续进行全表或全库analyze时进行多列统计信息收集。每组多列统计信息最多支持32列。不支持添加/删除多列统计信息声明的表：系统表、外表。

* **ALTER [ COLUMN ] column\_name SET ( {attribute\_option = value} [, ... ] )**
* **ALTER [ COLUMN ] column\_name RESET ( attribute\_option [, ... ] )**

设置/重置属性选项。

目前，属性选项只定义了n\_distinct和n\_distinct\_inherited。n\_distinct影响表本身的统计值，而n\_distinct\_inherited影响表及其继承子表的统计。目前，只支持SET/RESET n\_distinct参数，禁止SET/RESET n\_distinct\_inherited参数。

* **ALTER [ COLUMN ] column\_name SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }**

为一个字段设置存储模式。这个设置控制这个字段是内联保存还是保存在一个附属的表里，以及数据是否要压缩。仅支持对行存表的设置；对列存表没有意义，执行时报错。SET STORAGE本身并不改变表上的任何东西，只是设置将来的表操作时，建议使用的策略。

其中列约束column\_constraint为：

[ CONSTRAINT constraint\_name ]

{ NOT NULL |

NULL |

CHECK ( expression ) |

DEFAULT default\_expr |

UNIQUE index\_parameters |

PRIMARY KEY index\_parameters |

ENCRYPTED WITH ( COLUMN\_ENCRYPTION\_KEY = column\_encryption\_key, ENCRYPTION\_TYPE = encryption\_type\_value ) |

REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]

[ ON DELETE action ] [ ON UPDATE action ] } [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

其中列的压缩可选项compress\_mode为：

[ DELTA | PREFIX | DICTIONARY | NUMSTR | NOCOMPRESS ]

其中根据已有唯一索引为表增加主键约束或唯一约束table\_constraint\_using\_index为：

[ CONSTRAINT constraint\_name ]

{ UNIQUE | PRIMARY KEY } USING INDEX index\_name

[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

其中表约束table\_constraint为：

* [ CONSTRAINT constraint\_name ]
* { CHECK ( expression ) |
* UNIQUE ( column\_name [, ... ] ) index\_parameters |
* PRIMARY KEY ( column\_name [, ... ] ) index\_parameters |
* PARTIAL CLUSTER KEY ( column\_name [, ... ] }
* FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]
* [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] }
* [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

其中索引参数index\_parameters为：

* [ WITH ( {storage\_parameter = value} [, ... ] ) ]
* [ USING INDEX TABLESPACE tablespace\_name ]

重命名表。对名称的修改不会影响所存储的数据。

* ALTER TABLE [ IF EXISTS ] table\_name
* RENAME TO new\_table\_name;

重命名表中指定的列。

* ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name )}
* RENAME [ COLUMN ] column\_name TO new\_column\_name;

重命名表的约束。

* ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name ) }
* RENAME CONSTRAINT constraint\_name TO new\_constraint\_name;

设置表的所属模式。

* ALTER TABLE [ IF EXISTS ] table\_name
* SET SCHEMA new\_schema;
* fig: **说明：**
* 这种形式把表移动到另外一个模式。相关的索引、约束都跟着移动。目前序列不支持改变schema。 若该表拥有序列，需要将序列删除，重建，或者取消拥有关系， 才能将表schema更改成功。
* 要修改一个表的模式，用户必须在新模式上拥有CREATE权限。要把该表添加为一个父表的新子表，用户必须同时又是父表的所有者。要修改所有者，用户还必须是新的所有角色的直接或间接成员，并且该成员必须在此表的模式上有CREATE权限。这些限制规定了该用户不能做出了重建和删除表之外的事情。不过，系统管理员可以以任何方式修改任意表的所有权限。
* 除了RENAME和SET SCHEMA之外所有动作都可以捆绑在一个经过多次修改的列表中并行使用。比如，可以在一个命令里增加几个字段或修改几个字段的类型。对于大表，此种操作带来的效率提升更明显，原因在于只需要对该大表做一次处理。
* 增加一个CHECK或NOT NULL约束将会扫描该表，以保证现有的行符合约束要求。
* 用一个非空缺省值增加一个字段或者改变一个字段的现有类型会重写整个表。对于大表来说，这个操作可能会花很长时间，并且它还临时需要两倍的磁盘空间。

添加多个列。

* ALTER TABLE [ IF EXISTS ] table\_name
* ADD ( { column\_name data\_type [ compress\_mode ] [ COLLATE collation ] [ column\_constraint [ ... ] ]} [, ...] );

更新多个列。

* ALTER TABLE [ IF EXISTS ] table\_name
* MODIFY ( { column\_name data\_type | column\_name [ CONSTRAINT constraint\_name ] NOT NULL [ ENABLE ] | column\_name [ CONSTRAINT constraint\_name ] NULL } [, ...] );

**参数说明**

* **IF EXISTS**
* 如果不存在相同名称的表，不会抛出一个错误，而会发出一个通知，告知表不存在。
* **table\_name [\*] | ONLY table\_name | ONLY ( table\_name )**
* table\_name是需要修改的表名。
* 若声明了ONLY选项，则只有那个表被更改。若未声明ONLY，该表及其所有子表都将会被更改。另外，可以在表名称后面显示地增加\*选项来指定包括子表，即表示所有后代表都被扫描，这是默认行为。
* **constraint\_name**
* 要删除的现有约束的名称。
* **index\_name**
* 索引名称。
* **storage\_parameter**
* 表的存储参数的名称。
* 创建索引新增一个选项：
* parallel\_workers（int类型）

取值范围：[0,32]，0表示关闭并发。

表示创建索引时起的bgworker线程数量，例如2就表示将会起 2个bgworker线程并发创建索引。

如果未设置，启动bgworker线程数量与表大小相关，一般不

超过4个线程。

* hasuids（bool类型）

默认值：off

参数开启：更新表元组时，为元组分配表级唯一标识id。

* **new\_owner**
* 表新拥有者的名称。
* **new\_tablespace**
* 表所属新的表空间名称。
* **column\_name**、**column\_1\_name、 column\_2\_name**
* 现存的或新字段的名称。
* **data\_type**
* 新字段的类型，或者现存字段的新类型。
* **compress\_mode**
* 表字段的压缩可选项。该子句指定该字段优先使用的压缩算法。行存表不支持压缩。
* **collation**
* 字段排序规则名称。可选字段COLLATE指定了新字段的排序规则，如果省略，排序规则为新字段的默认类型。排序规则可以使用“select \* from pg\_collation;”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。
* **USING expression**
* USING子句声明如何从旧的字段值里计算新的字段值；如果省略，缺省从旧类型向新类型的赋值转换。如果从旧数据类型到新类型没有隐含或者赋值的转换，则必须提供一个USING子句。
* fig: **说明：**
* ALTER TYPE的USING选项实际上可以声明涉及该行旧值的任何表达式，即它可以引用除了正在被转换的字段之外其他的字段。这样，就可以用ALTER TYPE语法做非常普遍性的转换。因为这个灵活性，USING表达式并没有作用于该字段的缺省值（如果有的话），结果可能不是缺省表达式要求的常量表达式。这就意味着如果从旧类型到新类型没有隐含或者赋值转换的话，即使存在USING子句，ALTER TYPE也可能无法把缺省值转换成新的类型。在这种情况下，应该用DROP DEFAULT先删除缺省，执行ALTER TYPE，然后使用SET DEFAULT增加一个合适的新缺省值。类似的考虑也适用于涉及该字段的索引和约束。
* **NOT NULL | NULL**
* 设置列是否允许空值。
* **integer**
* 带符号的整数常值。当使用PERCENT时表示按照表数据的百分比收集统计信息，integer的取值范围为0-100。
* **attribute\_option**
* 属性选项。
* **PLAIN | EXTERNAL | EXTENDED | MAIN**
* 字段存储模式。
* PLAIN必需用于定长的数值（比如integer）并且是内联的、不压缩的。
* MAIN用于内联、可压缩的数据。
* EXTERNAL用于外部保存、不压缩的数据。使用EXTERNAL将令在text和bytea字段上的子字符串操作更快，但付出的代价是增加了存储空间。
* EXTENDED用于外部的压缩数据，EXTENDED是大多数支持非PLAIN存储的数据的缺省。
* **CHECK ( expression )**
* 每次将要插入的新行或者将要被更新的行必须使表达式结果为真才能成功，否则会抛出一个异常并且不会修改数据库。
* 声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。
* 目前，CHECK表达式不能包含子查询也不能引用除当前行字段之外的变量。
* **DEFAULT default\_expr**
* 给字段指定缺省值。
* 缺省表达式的数据类型必须和字段类型匹配。
* 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。
* **UNIQUE index\_parameters**
* **UNIQUE ( column\_name [, ... ] ) index\_parameters**
* UNIQUE约束表示表里的一个或多个字段的组合必须在全表范围内唯一。
* **PRIMARY KEY index\_parameters**
* **PRIMARY KEY ( column\_name [, ... ] ) index\_parameters**
* 主键约束表明表中的一个或者一些字段只能包含唯一（不重复）的非NULL值。
* **REFERENCES reftable [ ( refcolum ) ] [ MATCH matchtype ] [ ON DELETE action ] [ ON UPDATE action ] (column constraint)**
* **FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ] [ MATCH matchtype ] [ ON DELETE action ] [ ON UPDATE action ] (table constraint)**
* 外键约束要求新表中一列或多列构成的组应该只包含、匹配被参考表中被参考字段值。若省略refcolum，则将使用reftable的主键。被参考列应该是被参考表中的唯一字段或主键。外键约束不能被定义在临时表和永久表之间。
* 参考字段与被参考字段之间存在三种类型匹配，分别是：
* MATCH FULL：不允许一个多字段外键的字段为NULL，除非全部外键字段都是NULL。
* MATCH SIMPLE（缺省）：允许任意外键字段为NULL。
* MATCH PARTIAL：目前暂不支持。
* 另外，当被参考表中的数据发生改变时，某些操作也会在新表对应字段的数据上执行。ON DELETE子句声明当被参考表中的被参考行被删除时要执行的操作。ON UPDATE子句声明当被参考表中的被参考字段数据更新时要执行的操作。对于ON DELETE子句、ON UPDATE子句的可能动作：
* NO ACTION（缺省）：删除或更新时，创建一个表明违反外键约束的错误。若约束可推迟，且若仍存在任何引用行，那这个错误将会在检查约束的时候产生。
* RESTRICT：删除或更新时，创建一个表明违反外键约束的错误。与NO ACTION相同，只是动作不可推迟。
* CASCADE：删除新表中任何引用了被删除行的行，或更新新表中引用行的字段值为被参考字段的新值。
* SET NULL：设置引用字段为NULL。
* SET DEFAULT：设置引用字段为它们的缺省值。
* **DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE**
* 设置该约束是否可推迟。
* DEFERRABLE：可以推迟到事务结尾使用SET CONSTRAINTS命令检查。
* NOT DEFERRABLE：在每条命令之后马上检查。
* INITIALLY IMMEDIATE：那么每条语句之后就立即检查它。
* INITIALLY DEFERRED：只有在事务结尾才检查它。
  + fig: **说明：**
  + Ustore表不支持新增DEFERRABLE 以及 INITIALLY DEFERRED约束。
* **PARTIAL CLUSTER KEY**
* 局部聚簇存储，列存表导入数据时按照指定的列(单列或多列)，进行局部排序。
* **WITH ( {storage\_parameter = value} [, ... ] )**
* 为表或索引指定一个可选的存储参数。
* **tablespace\_name**
* 索引所在表空间的名称。
* **COMPRESS|NOCOMPRESS**
* NOCOMPRESS：如果指定关键字NOCOMPRESS则不会修改表的现有压缩特性。
* COMPRESS：如果指定COMPRESS关键字，则对该表进行批量插入元组时触发该特性。行存表不支持压缩。
* **new\_table\_name**
* 修改后新的表名称。
* **new\_column\_name**
* 表中指定列修改后新的列名称。
* **new\_constraint\_name**
* 修改后表约束的新名称。
* **new\_schema**
* 修改后新的模式名称。
* **CASCADE**
* 级联删除依赖于被依赖字段或者约束的对象（比如引用该字段的视图）。
* **RESTRICT**
* 如果字段或者约束还有任何依赖的对象，则拒绝删除该字段。这是缺省行为。
* **schema\_name**
* 表所在的模式名称。

**示例**

请参考CREATE TABLE的示例。

**相关链接**

CREATE TABLE 参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLE章节)，DROP TABLE(参见：开发者指南->SQL语法参考->SQL语法->DROP TABLE章节)

### ALTER TABLE PARTITION

**功能描述**

修改表分区，包括增删分区、切割分区、合成分区以及修改分区属性等。

**注意事项**

* 添加分区的表空间不能是PG\_GLOBAL。
* 添加分区的名称不能与该分区表已有分区的名称相同。
* 添加分区的分区键值要和分区表的分区键的类型一致。
* 若添加RANGE分区，添加分区键值要大于分区表中最后一个范围分区的上边界。
* 若添加LIST分区，添加分区键值不能与现有分区键值重复。
* 不支持添加HASH分区。
* 如果目标分区表中已有分区数达到了最大值1048575，则不能继续添加分区。
* 当分区表只有一个分区时，不能删除该分区。
* 选择分区使用PARTITION FOR()，括号里指定值个数应该与定义分区时使用的列个数相同，并且一一对应。
* Value分区表不支持相应的Alter Partition操作。
* 列存分区表不支持切割分区。
* 间隔分区表不支持添加分区。
* 哈希分区表不支持切割分区，不支持合成分区，不支持添加和删除分区。
* 列表分区表不支持切割分区，不支持合成分区。
* 只有分区表的所有者或者被授予了分区表ALTER权限的用户有权限执行ALTER TABLE PARTITION命令，系统管理员默认拥有此权限。

**语法格式**

修改表分区主语法如下：

ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name )}

action [, ... ];

其中action统指如下分区维护子语法。当存在多个分区维护子句时，保证了分区的连续性，无论这些子句的排序如何，Vastbase总会先执行DROP PARTITION再执行ADD PARTITION操作，最后顺序执行其它分区维护操作。

move\_clause |

exchange\_clause |

row\_clause |

merge\_clause |

modify\_clause |

split\_clause |

add\_clause |

drop\_clause |

truncate\_clause

* move\_clause子语法用于移动分区到新的表空间。

MOVE PARTITION { partion\_name | FOR ( partition\_value [, ...] ) } TABLESPACE tablespacename

* exchange\_clause子语法用于把普通表的数据迁移到指定的分区。

EXCHANGE PARTITION|SUBPARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) }

WITH TABLE {[ ONLY ] ordinary\_table\_name | ordinary\_table\_name \* | ONLY ( ordinary\_table\_name )}

[ { WITH | WITHOUT } VALIDATION ] [ VERBOSE ]

进行交换的普通表和分区必须满足如下条件：

* 普通表和分区的列数目相同，对应列的信息严格一致，包括：列名、列的数据类型、列约束、列的Collation信息、列的存储参数、列的压缩信息等。
* 普通表和分区的表压缩信息严格一致。
* 普通表和分区的索引个数相同，且对应索引的信息严格一致。
* 普通表和分区的表约束个数相同，且对应表约束的信息严格一致。
* 普通表不可以是临时表，分区表只能是范围分区表，列表分区表，哈希分区表。
* 普通表和分区表上不可以有动态数据脱敏，行访问控制约束。
* 列表分区表，哈希分区表不能是列存储。
* List/Hash/Range类型分区表支持exchange\_clause。
* 完成交换后，普通表和分区的数据被置换，同时普通表和分区的表空间信息被置换。此时，普通表和分区的统计信息变得不可靠，需要对普通表和分区重新执行analyze。
* 由于非分区键不能建立本地唯一索引，只能建立全局唯一索引，所以如果普通表含 有唯一索引时，会导致不能交换数据。
* row\_clause子语法用于设置分区表的行迁移开关。

{ ENABLE | DISABLE } ROW MOVEMENT

* merge\_clause子语法用于把多个分区合并成一个分区。

MERGE PARTITIONS { partition\_name } [, ...] INTO PARTITION partition\_name [ TABLESPACE tablespacename ] [ UPDATE GLOBAL INDEX ]

* modify\_clause子语法用于设置分区索引是否可用。

MODIFY PARTITION partition\_name { UNUSABLE LOCAL INDEXES | REBUILD UNUSABLE LOCAL INDEXES }

* split\_clause子语法用于把一个分区切割成多个分区。

SPLIT PARTITION { partition\_name | FOR ( partition\_value [, ...] ) } { split\_point\_clause | no\_split\_point\_clause } [ UPDATE GLOBAL INDEX ]

* 指定切割点split\_point\_clause的语法为。

AT ( partition\_value ) INTO ( PARTITION partition\_name [ TABLESPACE tablespacename ] , PARTITION partition\_name [ TABLESPACE tablespacename ] )

**须知：**

* 列存分区表不支持切割分区。
* 切割点的大小要位于正在被切割的分区的分区键范围内，指定切割点的方式只能把一个分区切割成两个新分区。
* 不指定切割点no\_split\_point\_clause的语法为。

INTO { ( partition\_less\_than\_item [, ...] ) | ( partition\_start\_end\_item [, ...] ) }

**须知：**

* 不指定切割点的方式，partition\_less\_than\_item指定的第一个新分区的分区键要大于正在被切割的分区的前一个分区（如果存在的话）的分区键，partition\_less\_than\_item指定的最后一个分区的分区键要等于正在被切割的分区的分区键大小。
* 不指定切割点的方式，partition\_start\_end\_item指定的第一个新分区的起始点（如果存在的话）必须等于正在被切割的分区的前一个分区（如果存在的话）的分区键，partition\_start\_end\_item指定的最后一个分区的终止点（如果存在的话）必须等于正在被切割的分区的分区键。
* partition\_less\_than\_item支持的分区键个数最多为4，而partition\_start\_end\_item仅支持1个分区键，其支持的数据类型参见PARTITION BY RANGE(parti...] (参见：开发者指南->SQL语法参考->SQL语法->CREATE-TABLE-PARTITION章节)。
* 在同一语句中partition\_less\_than\_item和partition\_start\_end\_item两者不可同时使用；不同split语句之间没有限制。
* 分区项partition\_less\_than\_item的语法为。

PARTITION partition\_name VALUES LESS THAN ( { partition\_value | MAXVALUE } [, ...] ) [ TABLESPACE tablespacename ]

* 分区项partition\_start\_end\_item的语法为，其约束参见START END语法描述(参见：开发者指南->SQL语法参考->SQL语法->CREATE-TABLE-PARTITION章节)。

PARTITION partition\_name {{START(partition\_value) END (partition\_value) EVERY (interval\_value)} |{START(partition\_value) END ({partition\_value | MAXVALUE})} |{START(partition\_value)} |{END({partition\_value |MAXVALUE})}} [TABLESPACE tablespace\_name]

* add\_clause子语法用于为指定的分区表添加一个或多个分区。

ADD PARTITION ( partition\_col1\_name = partition\_col1\_value [, partition\_col2\_name = partition\_col2\_value ] [, ...] )

[ LOCATION 'location1' ]

[ PARTITION (partition\_colA\_name = partition\_colA\_value [, partition\_colB\_name = partition\_colB\_value ] [, ...] ) ]

[ LOCATION 'location2' ]

ADD {partition\_less\_than\_item | partition\_start\_end\_item| partition\_list\_item }

分区项partition\_list\_item的语法如下。

PARTITION partition\_name VALUES (list\_values\_clause) [ TABLESPACE tablespacename ]

* partition\_list\_item仅支持的1个分区键，其支持的数据类型参见PARTITION BY LIST(partit...)(参见：开发者指南->SQL语法参考->SQL语法->CREATE-TABLE-PARTITION章节)。间隔/哈希分区表不支持添加分区。
* drop\_clause子语法用于删除分区表中的指定分区。

DROP PARTITION { partition\_name | FOR ( partition\_value [, ...] ) } [ UPDATE GLOBAL INDEX ]

哈希分区表不支持删除分区。

* truncate\_clause子语法用于清空分区表中的指定分区。

TRUNCATE PARTITION { partition\_name | FOR ( partition\_value [, ...] ) } [ UPDATE GLOBAL INDEX ]

* 修改表分区名称的语法。

ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name )}

RENAME PARTITION { partion\_name | FOR ( partition\_value [, ...] ) } TO partition\_new\_name;

**参数说明**

* **table\_name**
* 分区表名。
* 取值范围：已存在的分区表名。
* **partition\_name**
* 分区名。
* 取值范围：已存在的分区名。
* **tablespacename**
* 指定分区要移动到哪个表空间。
* 取值范围：已存在的表空间名。
* **partition\_value**
* 分区键值。
* 通过PARTITION FOR ( partition\_value [, ...] )子句指定的这一组值，可以唯一确定一个分区。
* 取值范围：需要进行重命名的分区的分区键的取值范围。
* **UNUSABLE LOCAL INDEXES**
* 设置该分区上的所有索引不可用。
* **REBUILD UNUSABLE LOCAL INDEXES**
* 重建该分区上的所有索引。
* **ENABLE/DISABLE ROW MOVEMET**
* 行迁移开关。
* 如果进行UPDATE操作时，更新了元组在分区键上的值，造成了该元组所在分区发生变化，就会根据该开关给出报错信息，或者进行元组在分区间的转移。
* 取值范围：
* ENABLE：打开行迁移开关。
* DISABLE：关闭行迁移开关。
* 默认是打开状态。
* **ordinary\_table\_name**
* 进行迁移的普通表的名称。
* 取值范围：已存在的普通表名。
* **{ WITH | WITHOUT } VALIDATION**
* 在进行数据迁移时，是否检查普通表中的数据满足指定分区的分区键范围。
* 取值范围：
* WITH：对于普通表中的数据要检查是否满足分区的分区键范围，如果有数据不满足，则报错。
* WITHOUT：对于普通表中的数据不检查是否满足分区的分区键范围。
* 默认是WITH状态。
* 由于检查比较耗时，特别是当数据量很大的情况下更甚。所以在保证当前普通表中的数据满足分区的分区键范围时，可以加上WITHOUT来指明不进行检查。
* **VERBOSE**
* 在VALIDATION是WITH状态时，如果检查出普通表有不满足要交换分区的分区键范围的数据，那么把这些数据插入到正确的分区，如果路由不到任何分区，再报错。
* fig: **须知：**
* 只有在VALIDATION是WITH状态时，才可以指定VERBOSE。
* **partition\_new\_name**
* 分区的新名称。
* 取值范围：字符串，要符合标识符的命名规范。

**示例**

此处示例为exchange\_clause子语法，其余请参考CREATE TABLE PARTITION的示例。

vastbase=# create table t\_range\_list

( id number,

partition\_key int,

subpartition\_key int,

col2 varchar2(10) )

partition by range(partition\_key)

subpartition by list(subpartition\_key)(

partition p1 values less than (100)

(subpartition sub\_1\_1 values (10),

subpartition sub\_1\_2 values (20)),

partition p2 values less than(200)

(subpartition sub\_2\_1 values (10),

subpartition sub\_2\_2 values (20)),

partition p3 values less than (300)

(subpartition sub\_3\_1 values (10),

subpartition sub\_3\_2 values (20)));

vastbase=# INSERT INTO t\_range\_list VALUES(1,50,10,'sub\_1\_1');

INSERT INTO t\_range\_list VALUES(2,150,20,'sub\_2\_2');

INSERT INTO t\_range\_list VALUES(3,250,10,'sub\_3\_1');

vastbase=#create table t\_exchange

( id number,

partition\_key int,

subpartition\_key int,

col2 varchar2(10) );

vastbase=# INSERT INTO t\_exchange VALUES(1,40,10,'sub\_1\_1');

INSERT 0 1

vastbase=# ALTER TABLE t\_range\_list exchange subpartition sub\_1\_1 with table t\_exchange;

ALTER TABLE

**相关链接**

CREATE TABLE PARTITION (参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLE PARTITION章节)，DROP TABLE(参见：开发者指南->SQL语法参考->SQL语法->DROP TABLE章节)

### ALTER TABLE SUBPARTITION

**功能描述**

修改二级分区表分区，包括增删分区、清空分区、切割分区等。

**注意事项**

* 目前二级分区表只支持增删分区、清空分区、切割分区。
* 添加分区的表空间不能是PG\_GLOBAL。
* 添加分区的名称不能与该分区表已有一级分区和二级分区的名称相同。
* 添加分区的分区键值要和分区表的分区键的类型一致。
* 若添加RANGE分区，添加分区键值要大于分区表中最后一个范围分区的上边界。若需要在有MAXVALUE分区的表上新增分区，建议使用SPLIT语法。
* 若添加LIST分区，添加分区键值不能与现有分区键值重复。若需要在有DEFAULT分区的表上新增分区，建议使用SPLIT语法。
* 不支持添加HASH分区。只有一种情况例外，二级分区表的二级分区方式为HASH且一级分区方式不是HASH，此时支持新增一级分区并创建对应的二级分区。
* 如果目标分区表中已有分区数达到了最大值1048575，则不能继续添加分区。
* 当分区表只有一个一级分区或二级分区时，不能删除该分区。
* 不支持删除HASH分区。
* 选择分区使用PARTITION FOR()，括号里指定值个数应该与定义分区时使用的列个数相同，并且一一对应。
* 切割分区只能对二级分区（叶子节点）进行切割，被切割分区只能是Range、List分区策略，不支持切割hash分区策略。List分区策略只能是default分区才能被切割。
* 只有分区表的所有者或者被授予了分区表ALTER权限的用户有权限执行ALTER TABLE PARTITION命令，系统管理员默认拥有此权限。
* 如果alter语句不带有UPDATE GLOBAL INDEX，那么原有的GLOBAL索引将失效，查询时将使用其他索引进行查询；如果alter语句带有UPDATEGLOBAL INDEX，原有的GLOBAL索引仍然有效，并且索引功能正确。

**语法格式**

* 修改表分区主语法。

ALTER TABLE [ IF EXISTS ] { table\_name [\*] | ONLY table\_name | ONLY ( table\_name )}

action [, ... ];

其中action统指如下分区维护子语法。

add\_clause |

drop\_clause |

split\_clause |

truncate\_clause

* add\_clause子语法用于为指定的分区表添加一个或多个分区。语法可以作用在一级分区上。

ADD {partition\_less\_than\_item | partition\_list\_item } [ ( subpartition\_definition\_list ) ]

也可以作用在二级分区上。

MODIFY PARTITION partition\_name ADD subpartition\_definition

其中，分区项partition\_less\_than\_item为RANGE分区定义语法，具体语法如下。

PARTITION partition\_name VALUES LESS THAN ( partition\_value | MAXVALUE ) [ TABLESPACE tablespacename ]

分区项partition\_list\_item为LIST分区定义语法，具体语法如下。

PARTITION partition\_name VALUES ( partition\_value [, ...] | DEFAULT ) [ TABLESPACE tablespacename ]

subpartition\_definition\_list为1到多个二级分区subpartition\_definition对象，subpartition\_definition具体语法如下。

SUBPARTITION subpartition\_name [ VALUES LESS THAN ( partition\_value | MAXVALUE ) | VALUES ( partition\_value [, ...] | DEFAULT )] [ TABLESPACE tablespace ]

fig: **须知：**

若一级分区为HASH分区，不支持以ADD形式新增一级分区；若二级分区为HASH分区，不支持以MODIFY形式新增二级分区。

* drop\_clause子语法用于删除分区表中的指定分区。语法可以作用在一级分区上。

DROP PARTITION { partition\_name | FOR ( partition\_value ) } [ UPDATE GLOBAL INDEX ]

也可以作用在二级分区上。

DROP SUBPARTITION { subpartition\_name | FOR ( partition\_value, subpartition\_value ) } [ UPDATE GLOBAL INDEX ]

fig: **须知：**

* 若一级分区为HASH分区，不支持删除一级分区；若二级分区为HASH分区，不支持删除二级分区。
* 不支持删除唯一子分区。
* split\_clause子语法用于把一个分区切割成多个分区。

SPLIT SUBPARTITION { subpartition\_name} { split\_point\_clause } [ UPDATE GLOBAL INDEX ]

指定切割点split\_point\_clause的语法为：

AT ( subpartition\_value ) INTO ( SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] , SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] )

fig: **须知：**

* 切割点的大小要位于正在被切割的分区的分区键范围内。
* 只能把一个分区切割成两个新分区。
* 指定Range分区策略切割点split\_point\_clause的语法为：

AT ( subpartition\_value ) INTO ( SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] , SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] )

指定List分区策略切割点split\_point\_clause的语法为：

VALUES ( subpartition\_value ) INTO ( SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] , SUBPARTITION subpartition\_name [ TABLESPACE tablespacename ] )

* 切割点的大小要位于正在被切割的分区的分区键范围内。
* 只能把一个分区切割成两个新分区。
* Range分区策略切割点是把当前分区以此切割点分割为两个分区（小于此分割点为一个分区，大于此分割点为另一个分区），所以Range分区策略切割点只能为一个。List分区策略切割点可以为多个，但不超过64个，即把这些切割点从当前分区的边界值提取出来作为一个新分区，当前分区剩余边界值作为另一个新分区。
* truncate\_clause子语法用于清空分区表中的指定分区。

TRUNCATE SUBPARTITION { subpartition\_name } [ UPDATE GLOBAL INDEX ]

**参数说明**

* **table\_name**
* 分区表名。
* 取值范围：已存在的分区表名。
* **subpartition\_name**
* 二级分区名。
* 取值范围：已存在的二级分区名。
* **tablespacename**
* 指定分区要移动到哪个表空间。
* 取值范围：已存在的表空间名。

**示例**

请参考CREATE TABLE SUBPARTITION (参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLE PARTITION章节)的示例。

### ALTER TABLESPACE

**功能描述**

修改表空间的属性。

**注意事项**

* 只有表空间的所有者或者被授予了表空间ALTER权限的用户有权限执行ALTER TABLESPACE命令，系统管理员默认拥有此权限。但要修改表空间的所有者，当前用户必须是该表空间的所有者或系统管理员，且该用户是新所有者角色的成员。
* 要修改表空间的所有者A为B，则A必须是B的直接或者间接成员。
* fig: **说明：**
* 如果new\_owner与old\_owner一致，此处不再校验当前执行操作的用户是否具有修改权限，而直接显示ALTER成功。

**语法格式**

* 重命名表空间的语法。
* ALTER TABLESPACE tablespace\_name
* RENAME TO new\_tablespace\_name;
* 设置表空间所有者的语法。
* ALTER TABLESPACE tablespace\_name
* OWNER TO new\_owner;
* 设置表空间属性的语法。
* ALTER TABLESPACE tablespace\_name
* SET ( {tablespace\_option = value} [, ... ] );
* 重置表空间属性的语法。
* ALTER TABLESPACE tablespace\_name
* RESET ( { tablespace\_option } [, ...] );
* 设置表空间限额的语法。
* ALTER TABLESPACE tablespace\_name
* RESIZE MAXSIZE { UNLIMITED | 'space\_size'};

**参数说明**

* **tablespace\_name**
* 要修改的表空间。
* 取值范围：已存在的表空间名。
* **new\_tablespace\_name**
* 表空间的新名称。
* 新名称不能以“PG\_”开头。
* 取值范围：字符串，符合标识符命名规范。
* **new\_owner**
* 表空间的新所有者。
* 取值范围：已存在的用户名。
* **tablespace\_option**
* 设置或者重置表空间的参数。
* 取值范围：
* seq\_page\_cost：设置优化器计算一次顺序获取磁盘页面的开销。缺省为1.0。
* random\_page\_cost：设置优化器计算一次非顺序获取磁盘页面的开销。缺省为4.0。

fig: **说明：**

* + - random\_page\_cost是相对于seq\_page\_cost的取值，等于或者小于seq\_page\_cost时毫无意义。
    - 默认值为4.0的前提条件是，优化器采用索引来扫描表数据，并且表数据在cache中命中率可以90%左右。
    - 如果表数据空间要比物理内存小，那么减小该值到一个适当水平；相反地，如果表数据在cache中命中率要低于90%，那么适当增大该值。
    - 如果采用了类似于SSD的随机访问代价较小的存储器，可以适当减小该值，以反映真正的随机扫描代价。
* value的取值范围：正的浮点类型。
* **RESIZE MAXSIZE**
* 重新设置表空间限额的数值。
* 取值范围：
* UNLIMITED，该表空间不设置限额。
* 由space\_size来确定，其格式参考CREATE TABLESPACE(参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLESPACE章节)。

fig: **说明：**

* + - 若调整后的限额值比当前表空间实际使用的值要小，调整操作可以执行成功，后续用户需要将该表空间的使用值降低到新限额值之下，才能继续往该表空间中写入数据。
    - 修改参数MAXSIZE时也可使用：

ALTER TABLESPACE tablespace\_name RESIZE MAXSIZE

{ 'UNLIMITED' | 'space\_size'};

**示例**

请参考CREATE TABLESPACE （参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLESPACE 章节）的示例。

**相关链接**

CREATE TABLESPACE (参见：开发者指南->SQL语法参考->SQL语法->CREATE TABLESPACE 章节)，DROP TABLESPACE (参见：开发者指南->SQL语法参考->SQL语法->DROP TABLESPACE 章节)

### ALTER TEXT SEARCH CONFIGURATION

**功能描述**

更改文本搜索配置的定义。用户可以将映射从字串类型调整为字典，或者改变配置的名称或者所有者，或者修改搜索配置的配置参数。

ADD MAPPING FOR选项为文本搜索配置增加字串类型映射；如果ADD MAPPING FOR后面任何一个字串类型的映射已经存在于此文本搜索配置中，那么系统将会报错。

ALTER MAPPING FOR选项会首先清除已有的字串类型映射，然后添加指定的字串类型映射。

ALTER MAPPING REPLACE ... WITH ... 与ALTER MAPPING FOR ... REPLACE ... WITH ...选项会直接使用new\_dictionary替换old\_dictionary。需要注意的是，只有pg\_ts\_config\_map系统表中存在maptokentype与old\_dictionary对应关系的元组时，才能更新成功，否则不会成功，也不会有任何提示信息返回。

DROP MAPPING FOR选项会删除当前文本搜索配置中指定的字串类型映射。 如果没有指定IF EXISTS选项，当DROP MAPPING FOR选项指定的字串类型映射在文本搜索配置中不存在时，数据库会报错。

**注意事项**

* 当一个搜索配置已经被引用（如被用来创建索引），则不允许用户修改此文本搜索配置。
* 要使用ALTER TEXT SEARCH CONFIGURATION，用户必须是配置的所有者。

**语法格式**

* 增加文本搜索配置字串类型映射语法

ALTER TEXT SEARCH CONFIGURATION name

ADD MAPPING FOR token\_type [, ... ] WITH dictionary\_name [, ... ];

* 修改文本搜索配置字典语法

ALTER TEXT SEARCH CONFIGURATION name

ALTER MAPPING FOR token\_type [, ... ] REPLACE old\_dictionary WITH new\_dictionary;

* 修改文本搜索配置字串类型语法

ALTER TEXT SEARCH CONFIGURATION name

ALTER MAPPING FOR token\_type [, ... ] WITH dictionary\_name [, ... ];

* 更改文本搜索配置字典语法

ALTER TEXT SEARCH CONFIGURATION name

ALTER MAPPING REPLACE old\_dictionary WITH new\_dictionary;

* 删除文本搜索配置字串类型映射语法

ALTER TEXT SEARCH CONFIGURATION name

DROP MAPPING [ IF EXISTS ] FOR token\_type [, ... ];

* 重命名文本搜索配置所有者语法

ALTER TEXT SEARCH CONFIGURATION name OWNER TO new\_owner;

* 重命名文本搜索配置名称语法

ALTER TEXT SEARCH CONFIGURATION name RENAME TO new\_name;

* 重命名文本搜索配置命名空间语法

ALTER TEXT SEARCH CONFIGURATION name SET SCHEMA new\_schema;

* 修改文本搜索配置属性语法

ALTER TEXT SEARCH CONFIGURATION name SET ( { configuration\_option = value } [, ...] );

* 重置文本搜索配置属性语法

ALTER TEXT SEARCH CONFIGURATION name RESET ( {configuration\_option} [, ...] );

**参数说明**

* **name**
* 已有文本搜索配置的名称（可以有模式修饰）。
* **token\_type**
* 与配置的语法解析器关联的字串类型的名称。详细信息参见解析器 (参见：开发者指南->SQL语法参考->全文检索->解析器章节)。
* **dictionary\_name**
* 文本搜索字典名称。 如果有多个字典，则它们会按指定的顺序搜索。
* **old\_dictionary**
* 映身中拟被替换的文本搜索字典名称。
* **new\_dictionary**
* 替换old\_dictionary的文本搜索字典的名称。
* **new\_owner**
* 文本搜索配置的新所有者。
* **new\_name**
* 文本搜索配置的新名称。
* **new\_schema**
* 文本搜索配置的新模式名。
* **configuration\_option**
* 文本搜索配置项。详细信息参见CREATE TEXT SEARCH CONFIGURATION (参见：开发者指南->SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节)。
* **value**
* 文本搜索配置项的值。

**示例**

--创建文本搜索配置。

vastbase=# CREATE TEXT SEARCH CONFIGURATION english\_1 (parser=default);

CREATE TEXT SEARCH CONFIGURATION

--增加文本搜索配置字串类型映射语法。

vastbase=# ALTER TEXT SEARCH CONFIGURATION english\_1 ADD MAPPING FOR word WITH simple,english\_stem;

ALTER TEXT SEARCH CONFIGURATION

--增加文本搜索配置字串类型映射语法。

vastbase=# ALTER TEXT SEARCH CONFIGURATION english\_1 ADD MAPPING FOR email WITH english\_stem, french\_stem;

ALTER TEXT SEARCH CONFIGURATION

--查询文本搜索配置相关信息。

vastbase=# SELECT b.cfgname,a.maptokentype,a.mapseqno,a.mapdict,c.dictname FROM pg\_ts\_config\_map a,pg\_ts\_config b, pg\_ts\_dict c WHERE a.mapcfg=b.oid AND a.mapdict=c.oid AND b.cfgname='english\_1' ORDER BY 1,2,3,4,5;

cfgname | maptokentype | mapseqno | mapdict | dictname

-----------+--------------+----------+---------+--------------

english\_1 | 2 | 1 | 3765 | simple

english\_1 | 2 | 2 | 12960 | english\_stem

english\_1 | 4 | 1 | 12960 | english\_stem

english\_1 | 4 | 2 | 12964 | french\_stem

(4 rows)

--增加文本搜索配置字串类型映射语法。

vastbase=# ALTER TEXT SEARCH CONFIGURATION english\_1 ALTER MAPPING REPLACE french\_stem with german\_stem;

ALTER TEXT SEARCH CONFIGURATION

--查询文本搜索配置相关信息。

vastbase=# SELECT b.cfgname,a.maptokentype,a.mapseqno,a.mapdict,c.dictname FROM pg\_ts\_config\_map a,pg\_ts\_config b, pg\_ts\_dict c WHERE a.mapcfg=b.oid AND a.mapdict=c.oid AND b.cfgname='english\_1' ORDER BY 1,2,3,4,5;

cfgname | maptokentype | mapseqno | mapdict | dictname

-----------+--------------+----------+---------+--------------

english\_1 | 2 | 1 | 3765 | simple

english\_1 | 2 | 2 | 12960 | english\_stem

english\_1 | 4 | 1 | 12960 | english\_stem

english\_1 | 4 | 2 | 12966 | german\_stem

(4 rows)

请参见CREATE TEXT SEARCH CONFIGURATION (参见：开发者指南->SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节)的示例。

**相关链接**

CREATE TEXT SEARCH CONFIGURATION(参见：开发者指南->SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节)， DROP TEXT SEARCH CONFIGURATION(参见：开发者指南->SQL语法参考->SQL语法->DROP TEXT SEARCH CONFIGURATION章节)

### ALTER TEXT SEARCH DICTIONARY

**功能描述**

修改全文检索词典的相关定义，包括参数、名称、所有者以及模式等。

**注意事项**

* 预定义词典不支持ALTER操作。
* 只有词典的所有者可以执行ALTER操作，系统管理员默认拥有此权限。
* 创建或修改词典之后，任何对于filepath路径下用户自定义的词典定义文件的修改，将不会影响到数据库中的词典。如果需要在数据库中使用这些修改，需使用ALTER TEXT SEARCH DICTIONARY语句更新对应词典的定义文件。

**语法格式**

* 修改词典定义。

ALTER TEXT SEARCH DICTIONARY name (

option [ = value ] [, ... ]

);

* 重命名词典。

ALTER TEXT SEARCH DICTIONARY name RENAME TO new\_name;

* 设置词典的所属模式。

ALTER TEXT SEARCH DICTIONARY name SET SCHEMA new\_schema;

* 修改词典的所属者。

ALTER TEXT SEARCH DICTIONARY name OWNER TO new\_owner;

**参数说明**

* **name**
* 已存在的词典名（可指定模式名，否则默认在当前模式下）。
* 取值范围：已存在的词典名。
* **option**
* 要修改的参数名。与template对应，不同的词典类型具有不同的参数列表，且与指定顺序无关。详细参数说明请见option(参见：开发者指南->SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节的option)。
* fig: **说明：**
* 不支持修改词典的TEMPLATE参数值。
* 不支持仅修改FILEPATH参数而不修改对应的词典定义文件参数。
* 词典定义文件的文件名仅支持小写字母、数据、下划线混合。
* **value**
* 要修改的参数值。如果省略等号（=）和value，则表示删除该option的先前设置，使用默认值。
* 取值范围：对应option定义。
* **new\_name**
* 词典的新名称。
* 取值范围：符合标识符命名规范的字符串，且最大长度不超过63个字符。
* **new\_owner**
* 词典新的所有者。
* 取值范围：已存在的用户。
* **new\_schema**
* 词典的新模式。
* 取值范围：已存在的模式。

**示例**

--更改Snowball类型字典的停用词定义，其他参数保持不变。

vastbase=# ALTER TEXT SEARCH DICTIONARY my\_dict ( StopWords = newrussian, FilePath = 'file:///home/dicts' );

--更改Snowball类型字典的Language参数，并删除停用词定义。

vastbase=# ALTER TEXT SEARCH DICTIONARY my\_dict ( Language = dutch， StopWords );

--更新词典定义，不实际更改任何内容。

vastbase=# ALTER TEXT SEARCH DICTIONARY my\_dict ( dummy );

**相关链接**

CREATE TEXT SEARCH DICTIONARY（参见：开发者指南->SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节），DROP TEXT SEARCH DICTIONARY（参见：开发者指南->SQL语法参考->SQL语法->DROP TEXT SEARCH CONFIGURATION章节）

### ALTER TRIGGER

**功能描述**

修改触发器名称。

fig: **说明：**

目前只支持修改名称。

**注意事项**

只有触发器所在表的所有者可以执行ALTER TRIGGER操作，系统管理员默认拥有此权限。

**语法格式**

ALTER TRIGGER trigger\_name ON table\_name RENAME TO new\_name;

**参数说明**

* **trigger\_name**
* 要修改的触发器名称。
* 取值范围：已存在的触发器。
* **table\_name**
* 要修改的触发器所在的表名称。
* 取值范围：已存在的含触发器的表。
* **new\_name**
* 修改后的新名称。
* 取值范围：符合标识符命名规范的字符串，最大长度不超过63个字符，且不能与所在表上其他触发器同名。

**示例**

请参见CREATE TRIGGER (参见：开发者指南->SQL语法参考->SQL语法->CREATE TRIGGER章节)的示例。

**相关链接**

CREATE TRIGGER (参见：开发者指南->SQL语法参考->SQL语法->CREATE TRIGGER章节)，DROP TRIGGER (参见：开发者指南->SQL语法参考->SQL语法->DROP TRIGGER章节)，ALTER TABLE(参见：开发者指南->SQL语法参考->SQL语法->ALTER TABLE章节)

### ALTER TYPE

**功能描述**

修改一个类型的定义。

**注意事项**

只有类型的所有者或者被授予了类型ALTER权限的用户可以执行ALTER TYPE命令，系统管理员默认拥有此权限。但要修改类型的所有者或者修改类型的模式，当前用户必须是该类型的所有者或者系统管理员，且该用户是新所有者角色的成员。

**语法格式**

* 修改类型。

ALTER TYPE name action [, ... ]

ALTER TYPE name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

ALTER TYPE name RENAME ATTRIBUTE attribute\_name TO new\_attribute\_name [ CASCADE | RESTRICT ]

ALTER TYPE name RENAME TO new\_name

ALTER TYPE name SET SCHEMA new\_schema

ALTER TYPE name ADD VALUE [ IF NOT EXISTS ] new\_enum\_value [ { BEFORE | AFTER } neighbor\_enum\_value ]

ALTER TYPE name RENAME VALUE existing\_enum\_value TO new\_enum\_value

where action is one of:

ADD ATTRIBUTE attribute\_name data\_type [ COLLATE collation ] [ CASCADE | RESTRICT ]

DROP ATTRIBUTE [ IF EXISTS ] attribute\_name [ CASCADE | RESTRICT ]

ALTER ATTRIBUTE attribute\_name [ SET DATA ] TYPE data\_type [ COLLATE collation ] [ CASCADE | RESTRICT ]

* 给复合类型增加新的属性。

ALTER TYPE name ADD ATTRIBUTE attribute\_name data\_type [ COLLATE collation ] [ CASCADE | RESTRICT ]

* 从复合类型删除一个属性。

ALTER TYPE name DROP ATTRIBUTE [ IF EXISTS ] attribute\_name [ CASCADE | RESTRICT ]

* 改变一种复合类型中某个属性的类型。

ALTER TYPE name ALTER ATTRIBUTE attribute\_name [ SET DATA ] TYPE data\_type [ COLLATE collation ] [ CASCADE | RESTRICT ]

* 改变类型的所有者。

ALTER TYPE name OWNER TO { new\_owner | CURRENT\_USER | SESSION\_USER }

* 改变类型的名称或是一个复合类型中的一个属性的名称。

ALTER TYPE name RENAME TO new\_name

ALTER TYPE name RENAME ATTRIBUTE attribute\_name TO new\_attribute\_name [ CASCADE | RESTRICT ]

* 将类型移至一个新的模式中。

ALTER TYPE name SET SCHEMA new\_schema

* 为枚举类型增加一个新值。

ALTER TYPE name ADD VALUE [ IF NOT EXISTS ] new\_enum\_value [ { BEFORE | AFTER } neighbor\_enum\_value ]

* 重命名枚举类型的一个标签值。

ALTER TYPE name RENAME VALUE existing\_enum\_value TO new\_enum\_value

**参数说明**

* **name**
* 一个需要修改的现有的类型的名称(可以有模式修饰) 。
* **new\_name**
* 该类型的新名称。
* **new\_owner**
* 新所有者的用户名 。
* **new\_schema**
* 该类型的新模式 。
* **attribute\_name**
* 拟增加、更改或删除的属性的名称。
* **new\_attribute\_name**
* 拟改名的属性的新名称。
* **data\_type**
* 拟新增属性的数据类型或是拟更改的属性的新类型名。
* **new\_enum\_value**
* 枚举类型新增加的标签值，是一个非空的长度不超过63个字节的字符串。
* **neighbor\_enum\_value**
* 一个已有枚举标签值，新值应该被增加在紧接着该枚举值之前或者之后的位置上。
* **existing\_enum\_value**
* 现有的要重命名的枚举值，是一个非空的长度不超过63个字节的字符串
* **CASCADE**
* 自动级联更新需更新类型以及相关联的记录和继承它们的子表。
* **RESTRICT**
* 如果需联动更新类型是已更新类型的关联记录，则拒绝更新。这是缺省选项。
* fig: **须知：**
* ADD ATTRIBUTE、DROP ATTRIBUTE和ALTER ATTRIBUTE选项可以组合成一个列表同时处理。 例如，在一条命令中同时增加几个属性或是更改几个属性的类型是可以实现的。
* 要修改一个类型的模式，必须在新模式上拥有CREATE权限。 要修改所有者，必须是新的所有角色的直接或间接成员， 并且该成员必须在此类型的模式上有CREATE权限。 （这些限制强制了修改所有者不会做任何通过删除和重建类型不能做的事情。 不过，系统管理员可以以任何方式修改任意类型的所有权。） 要增加一个属性或是修改一个属性的类型，也必须有该类型的USAGE权限。

**示例**

请参考CREATE TYPE (参见：开发者指南->SQL语法参考->SQL语法->CREATE TYPE 章节)的示例。

**相关链接**

CREATE TYPE(参见：开发者指南->SQL语法参考->SQL语法->CREATE TYPE 章节)，DROP TYPE(参见：开发者指南->SQL语法参考->SQL语法->DROP TYPE 章节)

### ALTER USER

**功能描述**

修改数据库用户的属性。

**注意事项**

ALTER USER中修改的会话参数只针对指定的用户，且在下一次会话中有效。

**语法格式**

* 修改用户的权限等信息。

ALTER USER user\_name [ [ WITH ] option [ ... ] ];

其中option子句为：

{ CREATEDB | NOCREATEDB }

| { CREATEROLE | NOCREATEROLE }

| { INHERIT | NOINHERIT }

| { AUDITADMIN | NOAUDITADMIN }

| { SYSADMIN | NOSYSADMIN }

| {MONADMIN | NOMONADMIN}

| {OPRADMIN | NOOPRADMIN}

| {POLADMIN | NOPOLADMIN}

| { USEFT | NOUSEFT }

| { LOGIN | NOLOGIN }

| { REPLICATION | NOREPLICATION }

| {INDEPENDENT | NOINDEPENDENT}

| {VCADMIN | NOVCADMIN}

| {PERSISTENCE | NOPERSISTENCE}

| CONNECTION LIMIT connlimit

| [ ENCRYPTED | UNENCRYPTED ] PASSWORD { 'password' [EXPIRED] | DISABLE | EXPIRED }

| [ ENCRYPTED | UNENCRYPTED ] IDENTIFIED BY { 'password' [ REPLACE 'old\_password' | EXPIRED ] | DISABLE }

| VALID BEGIN 'timestamp'

| VALID UNTIL 'timestamp'

| RESOURCE POOL 'respool'

| PERM SPACE 'spacelimit'

| PGUSER

* 修改用户名。

ALTER USER user\_name

RENAME TO new\_name;

* 锁定或解锁。

ALTER USER user\_name

ACCOUNT { LOCK | UNLOCK };

* 修改与用户关联的指定会话参数值。

ALTER USER user\_name

SET configuration\_parameter { { TO | = } { value | DEFAULT } | FROM CURRENT };

* 重置与用户关联的指定会话参数值。

ALTER USER user\_name

RESET { configuration\_parameter | ALL };

**参数说明**

* **user\_name**
* 现有用户名。
* 取值范围：已存在的用户名。
* **new\_password**
* 新密码。
* 密码规则如下：
* 不能与当前密码相同。
* 密码默认不少于8个字符。
* 不能与用户名及用户名倒序相同。
* 至少包含大写字母（A-Z）、小写字母（a-z）、数字（0-9）、非字母数字字符（限定为~!@#$%^&\*()-\_=+|[{}];:,<.>/?）四类字符中的三类字符。
* 取值范围：字符串。
* **old\_password**
* 旧密码。
* **ACCOUNT LOCK | ACCOUNT UNLOCK**
* ACCOUNT LOCK：锁定帐户，禁止登录数据库。
* ACCOUNT UNLOCK：解锁帐户，允许登录数据库。
* **PGUSER**
* 当前版本不允许修改用户的PGUSER属性。

其他参数请参见CREATE ROLE (参见：开发者指南->SQL语法参考->SQL语法->CREATE ROLE章节)和ALTER ROLE (参见：开发者指南->SQL语法参考->SQL语法->ALTER ROLE章节)的参数说明。

**示例**

请参考CREATE USER(参见：开发者指南->SQL语法参考->SQL语法->CREATE USER章节)的示例。

**相关链接**

CREATE ROLE (参见：开发者指南->SQL语法参考->SQL语法->CREATE ROLE章节)，CREATE USER(参见：开发者指南->SQL语法参考->SQL语法->CREATE USER章节)，DROP USER(参见：开发者指南->SQL语法参考->SQL语法->DROP USER章节)

### ALTER USER MAPPING

**功能描述**

更改一个用户映射的定义。

**注意事项**

当在OPTIONS中出现password选项时，需要保证vastbase每个节点的$GAUSSHOME/bin目录下存在usermapping.key.cipher和usermapping.key.rand文件，如果不存在这两个文件，请使用vb\_guc工具生成并发布到每个节点的$GAUSSHOME/bin目录下。

**语法格式**

ALTER USER MAPPING FOR { user\_name | USER | CURRENT\_USER | PUBLIC }

SERVER server\_name

OPTIONS ( [ ADD | SET | DROP ] option ['value'] [, ... ] )

在OPTIONS选项里，ADD、SET和DROP指定要执行的操作，未指定时默认为ADD操作。option和value为对应操作的参数及参数值。

**参数说明**

* **user\_name**
* 该映射的用户名。
* CURRENT\_USER和USER匹配当前用户的名称。PUBLIC被用来匹配系统中所有当前以及未来的用户名。
* **server\_name**
* 该用户映射的服务器名。
* **OPTIONS**
* 为该用户映射更改选项。新选项会覆盖任何之前指定的选项。ADD、SET和DROP指定要被执行的动作。如果没有显式地指定操作，将假定为ADD。选项名称必须为唯一，该服务器的外部数据包装器也会验证选项。
* oracle\_fdw支持的options包括：
* **user**

oracle server的用户名。

* **password**

oracle用户对应的密码。

* mysql\_fdw支持的options包括：
* **username**

MySQL Server/MariaDB的用户名。

* **password**

MySQL Server/MariaDB用户对应的密码。

* postgres\_fdw支持的options包括：
* **user**

远端vastbase数据库用户的用户名。

* **password**

远端vastbase数据库用户对应的密码。

**相关链接**

CREATE USER MAPPING (参见：开发者指南->SQL语法参考->SQL语法->CREATE USER MAPPING 章节)，DROP USER MAPPING (参见：开发者指南->SQL语法参考->SQL语法->DROP USER MAPPING 章节)

### ALTER VIEW

**功能描述**

ALTER VIEW更改视图的各种辅助属性。（如果用户是更改视图的查询定义，要使用CREATE OR REPLACE VIEW。）

**注意事项**

只有视图的所有者或者被授予了视图ALTER权限的用户才可以执行ALTER VIEW命令，系统管理员默认拥有该权限。针对所要修改属性的不同，对其还有以下权限约束：

* 修改视图的模式，当前用户必须是视图的所有者或者系统管理员，且要有新模式的CREATE权限。
* 修改视图的所有者，当前用户必须是视图的所有者或者系统管理员，且该用户必须是新所有者角色的成员，并且此角色必须有视图所在模式的CREATE权限。

**语法格式**

* 设置视图列的默认值。

ALTER VIEW [ IF EXISTS ] view\_name

ALTER [ COLUMN ] column\_name SET DEFAULT expression;

* 取消列视图列的默认值。

ALTER VIEW [ IF EXISTS ] view\_name

ALTER [ COLUMN ] column\_name DROP DEFAULT;

* 修改视图的所有者。

ALTER VIEW [ IF EXISTS ] view\_name

OWNER TO new\_owner;

* 重命名视图。

ALTER VIEW [ IF EXISTS ] view\_name

RENAME TO new\_name;

* 设置视图的所属模式。

ALTER VIEW [ IF EXISTS ] view\_name

SET SCHEMA new\_schema;

* 设置视图的选项。

ALTER VIEW [ IF EXISTS ] view\_name

SET ( { view\_option\_name [ = view\_option\_value ] } [, ... ] );

* 重置视图的选项。

ALTER VIEW [ IF EXISTS ] view\_name

RESET ( view\_option\_name [, ... ] );

**参数说明**

* **IF EXISTS**
* 使用这个选项，如果视图不存在时不会产生错误，仅有会有一个提示信息。
* **view\_name**
* 视图名称，可以用模式修饰。
* 取值范围：字符串，符合标识符命名规范。
* **column\_name**
* 可选的名称列表，视图的字段名。如果没有给出，字段名取自查询中的字段名。
* 取值范围：字符串，符合标识符命名规范。
* **SET/DROP DEFAULT**
* 设置或删除一个列的缺省值，该参数暂无实际意义。
* **new\_owner**
* 视图新所有者的用户名称。
* **new\_name**
* 视图的新名称。
* **new\_schema**
* 视图的新模式。
* **view\_option\_name [ = view\_option\_value ]**
* 该子句为视图指定一个可选的参数。
* 目前view\_option\_name支持的参数仅有security\_barrier，当VIEW试图提供行级安全时，应使用该参数。
* 取值范围：Boolean类型，TRUE、FALSE。

**示例**

--创建测试schema。

vastbase=# CREATE SCHEMA stest;

--创建测试表stest.customer。

vastbase=# CREATE TABLE stest.customer(c\_customer\_sk int);

--插入测试数据。

vastbase=# INSERT INTO stest.customer values(1);

vastbase=# INSERT INTO stest.customer values(2);

vastbase=# INSERT INTO stest.customer values(3);

--创建一个由c\_customer\_sk小于10的内容组成的视图。

vastbase=# CREATE VIEW stest.customer\_details\_view\_v1 AS

SELECT \* FROM stest.customer

WHERE c\_customer\_sk < 10;

--修改视图名称。

vastbase=# ALTER VIEW stest.customer\_details\_view\_v1 RENAME TO customer\_details\_view\_v2;

--修改视图所属schema。

vastbase=# ALTER VIEW stest.customer\_details\_view\_v2 SET schema public;

--删除视图。

vastbase=# DROP VIEW public.customer\_details\_view\_v2;

**相关链接**

CREATE VIEW(参考：开发者指南->SQL语法参考->SQL语法->CREATE-VIEW章节)，DROP VIEW(参考：开发者指南->SQL语法参考->SQL语法->DROP-VIEW章节)

### ANALYZE和ANALYSE

**功能描述**

ANALYZE和ANALYSE用于收集与数据库中普通表内容相关的统计信息，统计结果存储在系统表PG\_STATISTIC下。执行计划生成器会使用这些统计数据，以确定最有效的执行计划。

如果没有指定参数，ANALYZE会分析当前数据库中的每个表和分区表。同时也可以通过指定table\_name、column和partition\_name参数把分析限定在特定的表、列或分区表中。

ANALYZE VERIFY和ANALYSE VERIFY用于检测数据库中普通表（行存表、列存表）的数据文件是否损坏。

**注意事项**

* ANALYZE非临时表不能在一个匿名块、事务块、函数或存储过程内被执行。支持存储过程中ANALYZE临时表，不支持统计信息回滚操作。
* ANALYZE VERIFY 操作处理的大多为异常场景检测需要使用RELEASE版本。
* ANALYZE VERIFY 场景不触发远程读，因此远程读参数不生效。对于关键系统表出现错误被系统检测出页面损坏时，将直接报错不再继续检测。

**语法格式**

* 收集表的统计信息。

{ ANALYZE | ANALYSE } [ VERBOSE ]

[ table\_name [ ( column\_name [, ...] ) ] ];

* 收集分区表的统计信息。

{ ANALYZE | ANALYSE } [ VERBOSE ]

[ table\_name [ ( column\_name [, ...] ) ] ]

PARTITION ( patrition\_name ) ;

普通分区表目前支持针对某个分区的统计信息的语法，在功能上不支持针对某个分区的统计信息收集。

* 收集多列统计信息。

{ANALYZE | ANALYSE} [ VERBOSE ]

table\_name (( column\_1\_name, column\_2\_name [, ...] ));

**说明：**

* 收集多列统计信息时，请设置GUC参数default\_statistics\_target为负数， 以使用百分比采样方式。
* 每组多列统计信息最多支持32列。
* 不支持收集多列统计信息的表：系统表。
* 检测当前库的数据文件。

{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE};

**说明：**

* Fast模式校验时，需要对校验的表有并发的DML操作，会导致校验过程中有误报的问题，因为当前Fast模式是直接从磁盘上读取，并发有其他线程修改文件时，会导致获取的数据不准确，建议离线操作。
* 支持对全库进行操作，由于涉及的表较多，建议以重定向保存结果，命令如下所示：

vsql -d database -p port -f "verify.sql"> verify\_warning.txt 2>&1

* 对外提示NOTICE只核对外可见的表，内部表的检测会包含在它所依赖的外部表，不对外显示和呈现。
* 此命令的处理可容错ERROR级别的处理。由于debug版本的Assert可能会导致core无法继续执行命令，建议在release模式下操作。
* 对于全库操作时，当关键系统表出现损坏则直接报错，不再继续执行。
* 不支持临时表和unlog表。
* 检测表和索引的数据文件。

{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE} table\_name|index\_name [CASCADE];

**说明：**

* 支持对普通表的操作和索引表的操作，但不支持对索引表index使用CASCADE操作。原因是由于CASCADE模式用于处理主表的所有索引表，当单独对索引表进行检测时，无需使用CASCADE模式。
* 不支持临时表和unlog表。
* 对于主表的检测会同步检测主表的内部表，例如toast表、cudesc表等。
* 当提示索引表损坏时，建议使用reindex命令进行重建索引操作。
* 检测表分区的数据文件。

{ANALYZE | ANALYSE} VERIFY {FAST|COMPLETE} table\_name PARTITION {(patrition\_name)}[CASCADE];

**说明：**

* 支持对表的单独分区进行检测操作，但不支持对索引表index使用CASCADE操作。
* 不支持临时表和unlog表。
* 全局分区索引适配支持ANALYZE VERIFY FAST。

**参数说明**

* VERBOSE：启用显示进度信息，如果指定了VERBOSE，ANALYZE发出进度信息，表明目前正在处理的表。各种有关表的统计信息也会打印出来。
* FAST|COMPLETE：对于行存表，FAST模式下主要对于行存表的CRC和page header进行校验，如果校验失败则会告警; 而COMPLETE模式下，则主要对行存表的指针、tuple进行解析校验。 对于列存表，FAST模式下主要对于列存表的CRC和magic进行校验，如果校验失败则会告警; 而COMPLETE模式下，则主要对列存表的CU进行解析校验。
* table\_name：需要分析的特定表的表名（可能会带模式名），如果省略，将对数据库中的所有表（非外部表）进行分析。对于ANALYZE收集统计信息，目前仅支持行存表、列存表。
* 取值范围：已有的表名。
* index\_name需要分析的特定索引表的表名（可能会带模式名）。
* 取值范围：已有的表名。
* CASCADE：CASCADE模式下会对当前表的所有索引进行检测处理。
* partition\_name：如果table为分区表，在关键字PARTITION后面指定分区名partition\_name表示分析该分区表的统计信息。目前语法上支持分区表做ANALYZE，但功能实现上暂不支持对指定分区统计信息的分析。
* 取值范围：表的某一个分区名。

**示例**

1、创建测试表。

CREATE TABLE customer\_info

(

WR\_RETURNED\_DATE\_SK INTEGER ,

WR\_RETURNED\_TIME\_SK INTEGER ,

WR\_ITEM\_SK INTEGER NOT NULL,

WR\_REFUNDED\_CUSTOMER\_SK INTEGER

) ;

2、创建分区表。

CREATE TABLE customer\_par

(

WR\_RETURNED\_DATE\_SK INTEGER ,

WR\_RETURNED\_TIME\_SK INTEGER ,

WR\_ITEM\_SK INTEGER NOT NULL,

WR\_REFUNDED\_CUSTOMER\_SK INTEGER

)

PARTITION BY RANGE(WR\_RETURNED\_DATE\_SK)

(

PARTITION P1 VALUES LESS THAN(2452275),

PARTITION P2 VALUES LESS THAN(2452640),

PARTITION P3 VALUES LESS THAN(2453000),

PARTITION P4 VALUES LESS THAN(MAXVALUE)

)

ENABLE ROW MOVEMENT;

3、使用ANALYZE语句更新统计信息。

ANALYZE customer\_info;

4、使用ANALYZE VERBOSE语句更新统计信息，并输出表的相关信息。

ANALYZE VERBOSE customer\_info;

返回结果为：

INFO: analyzing "public.customer\_info"(node1 pid=336454)

INFO: ANALYZE INFO : "customer\_info": scanned 0 of 0 pages, containing 0 live rows and 0 dead rows; 0 rows in sample, 0 estimated total rows(node1 pid=336454)

ANALYZE

5、删除表。

DROP TABLE customer\_info;

DROP TABLE customer\_par;

### ANONYMOUS BLOCK

**功能描述**

定义一个新的匿名块。

**注意事项**

无

**语法格式**

[DECLARE [declare\_statements]]

BEGIN

execution\_staements

END;

/

**参数说明**

* declare\_statements

声明变量，包括变量名和变量类型，如“sales\_cnt int”。

* execution\_statements

匿名块中要执行的语句。

取值范围：DML操作(数据操纵操作：select、insert、delete、 update)或系统表中已注册的函数名称。

**示例**

1、创建测试表test。

CREATE TABLE test(id int);

2、 执行匿名块

BEGIN

for i in 0..10 LOOP

INSERT INTO test(id) values (i);

END LOOP;

END;

/

3、 验证结果

select \* from test;

当结果显示如下信息，则表示匿名块执行完成。

id

------------------

0

1

2

3

4

5

6

7

8

9

10

(11 rows)

**相关链接**

BEGIN（参考：开发者指南->SQL语法参考->SQL语法->BEGIN章节)

### BEGIN

**功能描述**

BEGIN可以用于开始一个匿名块，也可以用于开始一个事务。本节描述用BEGIN开始匿名块的语法，以BEGIN开始事务的语法见章节“START TRANSACTION”。

**语法格式**

开启匿名块：

[DECLARE [declare\_statements]]

BEGIN

execution\_statements

END;

/

**参数说明**

* declare\_statements：声明变量，包括变量名和变量类型，如“sales\_cnt int”。
* execution\_statements：匿名块中要执行的语句。
* 取值范围：DML操作(数据操纵操作：select、insert、delete、update)或系统表中已注册的函数名称。

**注意事项**

无。

**示例**

1、创建测试表test。

CREATE TABLE test(id int);

2、 执行匿名块

BEGIN

for i in 0..10 LOOP

INSERT INTO test(id) values (i);

END LOOP;

END;

/

3、 验证结果

select \* from test;

当结果显示如下信息，则表示匿名块执行完成。

id

------------------

0

1

2

3

4

5

6

7

8

9

10

(11 rows)

4、 删除测试表

DROP TABLE test;

### CALL

**功能描述**

使用CALL命令可以调用已定义的函数和存储过程。

**注意事项**

* 函数或存储过程的所有者、被授予了函数或存储过程EXECUTE权限的用户或被授予EXECUTE ANY FUNCTION权限的用户有权调用函数或存储过程，系统管理员默认拥有此权限。
* 在MySQL兼容模式下（DBCOMPATIBILITY='B')，针对无参的存储过程可不使用括号。

**语法格式**

CALL [schema.|package.] {func\_name| procedure\_name} ( param\_expr );

**参数说明**

* **schema**
* 函数或存储过程所在的模式名称。
* **package**
* 函数或存储过程所在的package名称。
* **func\_name**
* 所调用函数或存储过程的名称。
* 取值范围：已存在的函数名称。
* **param\_expr**
* 参数列表可以用符号“:=”或者“=>”将参数名和参数值隔开，这种方法的好处是参数可以以任意顺序排列。若参数列表中仅出现参数值，则参数值的排列顺序必须和函数或存储过程定义时的相同。
* 取值范围：已存在的函数参数名称或存储过程参数名称。
* 参数可以包含入参（参数名和类型之间指定“IN”关键字）和出参（参数名和类型之间指定“OUT”关键字），使用CALL命令调用函数或存储过程时，对于非重载的函数，参数列表必须包含出参，出参可以传入一个变量或者任一常量，详见[示例](#Xed8e958a524e65d3446ffef3d4b2602d05b1c14)。对于重载的package函数，参数列表里可以忽略出参，忽略出参时可能会导致函数找不到。包含出参时，出参只能是常量。

**示例**

* **示例1:**

1、创建一个函数func\_add\_sql，计算两个整数的和，并返回结果。

vastbase=# CREATE FUNCTION func\_add\_sql(num1 integer, num2 integer) RETURN integer

AS

BEGIN

RETURN num1 + num2;

END;

/

2、按参数值传递。

CALL func\_add\_sql(1, 3);

结果显示如下：

func\_add\_sql

--------------

4

(1 row)

3、使用命名标记法传参。

CALL func\_add\_sql(num1 => 1,num2 => 3);

结果显示如下：

func\_add\_sql

--------------

4

(1 row)

4、删除函数。

DROP FUNCTION func\_add\_sql;

* **示例2:**

1、创建带出参的函数。

CREATE FUNCTION func\_increment\_sql(num1 IN integer, num2 IN integer, res OUT integer)

RETURN integer

AS

BEGIN

res := num1 + num2;

END;

/

2、出参传入常量。

CALL func\_increment\_sql(1,2,1);

结果显示如下：

res

-----

3

(1 row)

3、删除函数。

DROP FUNCTION func\_increment\_sql;

* **示例3:**

1、创建并进入兼容MySQL的库db\_mysql。

CRETE DATABASE db\_mysql DBCOMPATIBILITY='B';

\c db\_mysql

2、创建无参存储过程。

CREATE PROCEDURE ptest()

AS

BEGIN

raise notice 'test';

END;

/

3、不加括号调用存储过程。

call ptest;

结果显示如下：

ptest

-------

(1 row)

### CHECKPOINT

**功能描述**

检查点（CHECKPOINT）是一个事务日志中的点，所有数据文件都在该点被更新以反映日志中的信息，所有数据文件都将被刷新到磁盘。

设置事务日志检查点。预写式日志（WAL）缺省时在事务日志中每隔一段时间放置一个检查点。可以使用vb\_guc命令设置相关运行时参数（checkpoint\_segments、checkpoint\_timeout和incremental\_checkpoint\_timeout）来调整这个原子化检查点的间隔。

**注意事项**

* 只有系统管理员和运维管理员可以调用CHECKPOINT。
* CHECKPOINT强制立即进行检查，而不是等到下一次调度时的检查点。

**语法格式**

CHECKPOINT;

**参数说明**

无。

**示例**

--设置检查点。

vastbase=# CHECKPOINT;

### CLEAN CONNECTION

**功能描述**

用来清理数据库连接。允许在节点上清理指定数据库的指定用户的相关连接。

**注意事项**

* Vastbase下不支持指定节点，仅支持TO ALL。
* 该功能仅在force模式下，可以清理正在使用的正常连接。

**语法格式**

CLEAN CONNECTION  
 TO { COORDINATOR ( nodename [, ... ] ) | NODE ( nodename [, ... ] )| ALL [ CHECK ] [ FORCE ] }  
 [ FOR DATABASE dbname ]  
 [ TO USER username ];

**参数说明**

* **CHECK**
* 仅在节点列表为TO ALL时可以指定。如果指定该参数，会在清理连接之前检查数据库是否被其他会话连接访问。此参数主要用于DROP DATABASE之前的连接访问检查，如果发现有其他会话连接，则将报错并停止删除数据库。
* **FORCE**
* 仅在节点列表为TO ALL时可以指定，如果指定该参数，所有和指定dbname和username相关的线程都会收到SIGTERM信号，然后被强制关闭。
* **COORDINATOR ( nodename [, ... ] ) | NODE ( nodename [, ... ] ) | ALL**
* 删除指定节点上的连接。有三种场景：
* 删除指定CN上的连接，Vastbase不支持。
* 删除指定DN上的连接，Vastbase不支持。
* 删除所有节点上的连接(TO ALL)，Vastbase仅支持该场景。
* **dbname**
* 删除指定数据库上的连接。如果不指定，则删除所有数据库的连接。
* 取值范围：已存在数据库名。
* **username**
* 删除指定用户上的连接。如果不指定，则删除所有用户的连接。
* 取值范围：已存在的用户。

**示例**

--创建jack用户。  
CREATE USER jack PASSWORD 'Bigdata123@';  
  
--删除用户jack在数据库template1上的所有连接。  
CLEAN CONNECTION TO ALL FOR DATABASE template1 TO USER jack;  
  
--删除用户jack的所有连接。  
CLEAN CONNECTION TO ALL TO USER jack;  
  
--删除在数据库gaussdb上的所有连接。  
CLEAN CONNECTION TO ALL FORCE FOR DATABASE gaussdb;  
  
--删除用户jack。  
DROP USER jack;

### CLOSE

**功能描述**

CLOSE释放和一个游标关联的所有资源。

**注意事项**

* 不允许对一个已关闭的游标再做任何操作。
* 一个不再使用的游标应该尽早关闭。
* 当创建游标的事务用COMMIT或ROLLBACK终止之后，每个不可保持的已打开游标都隐含关闭。
* 当创建游标的事务通过ROLLBACK退出之后，每个可以保持的游标都将隐含关闭。
* 当创建游标的事务成功提交，可保持的游标将保持打开，直到执行一个明确的CLOSE或者客户端断开。
* Vastbase没有明确打开游标的OPEN语句，因为一个游标在使用CURSOR命令定义的时候就打开了。可以通过查询系统视图pg\_cursors看到所有可用的游标。

**语法格式**

CLOSE { cursor\_name | ALL } ;

**参数说明**

* **cursor\_name**
* 一个待关闭的游标名称。
* **ALL**
* 关闭所有已打开的游标。

**示例**

参考：SQL语法参考->SQL语法->FETCH的示例。

**相关链接**

参考：SQL语法参考->SQL语法中的FETCH、MOVE章节。

### CLUSTER

**功能描述**

根据一个索引对表进行聚簇排序。

CLUSTER指定Vastbase通过索引名指定的索引聚簇由表名指定的表。表名上必须已经定义该索引。

当对一个表聚集后，该表将基于索引信息进行物理存储。聚集是一次性操作：当表被更新之后，更改的内容不会被聚集。也就是说，系统不会试图按照索引顺序对新的存储内容及更新记录进行重新聚集。

在对一个表聚簇之后，Vastbase会记录在哪个索引上建立了聚集。CLUSTER table\_name的聚集形式在之前的同一个索引的表上重新聚集。用户也可以用ALTER TABLE的CLUSTER或SET WITHOUT CLUSTER形式来设置索引来用于后续的聚集操作或清除任何之前的设置。

不含参数的CLUSTER会将当前用户所拥有的数据库中的先前做过聚簇的所有表重新处理，或者系统管理员调用的这些表。

在对一个表进行聚簇的时候，会在其上请求一个ACCESS EXCLUSIVE锁。这样就避免了在CLUSTER完成之前对此表执行其它的操作(包括读写)。

**注意事项**

* 只有行存B-tree索引支持CLUSTER操作。
* 如果用户只是随机访问表中的行，那么表中数据的实际存储顺序是无关紧要的。但是，如果对某些数据的访问多于其它数据，而且有一个索引将这些数据分组，那么将使用CLUSTER中会有所帮助。如果从一个表中请求一定索引范围的值，或者是一个索引值对应多行，CLUSTER也会有助于应用，因为如果索引标识出第一匹配行所在的存储页，所有其它行也可能已经在同一个存储页里了，这样便节省了磁盘访问的时间，加速了查询。
* 在聚簇过程中，系统先创建一个按照索引顺序建立的表的临时拷贝。同时也建立表上的每个索引的临时拷贝。因此，需要磁盘上有足够的剩余空间， 至少是表大小和索引大小的和。
* 因为CLUSTER记忆聚集信息，可以在第一次的时候手工对表进行聚簇，然后设置一个类似VACUUM的时间，这样就可以周期地自动对表进行聚簇操作。
* 因为优化器记录着有关表的排序的统计，所以建议在新近聚簇的表上运行ANALYZE。否则，优化器可能会选择很差劲的查询规划。
* CLUSTER不允许在事务中执行。
* 如果没有打开xc\_maintenance\_mode参数，那么CLUSTER操作将跳过所有系统表。

**语法格式**

* 对一个表进行聚簇排序。

CLUSTER [ VERBOSE ] table\_name [ USING index\_name ];

* 对一个分区进行聚簇排序。

CLUSTER [ VERBOSE ] table\_name PARTITION ( partition\_name ) [ USING index\_name ];

* 对已做过聚簇的表重新进行聚簇。

CLUSTER [ VERBOSE ];

**参数说明**

* **VERBOSE**
* 启用显示进度信息。
* **table\_name**
* 表名称。
* 取值范围：已存在的表名称。
* **index\_name**
* 索引名称。
* 取值范围：已存在的索引名称。
* **partition\_name**
* 分区名称。
* 取值范围：已存在的分区名称。

**示例**

-- 创建一个分区表。  
vastbase=# CREATE TABLE tpcds.inventory\_p1  
(  
 INV\_DATE\_SK INTEGER NOT NULL,  
 INV\_ITEM\_SK INTEGER NOT NULL,  
 INV\_WAREHOUSE\_SK INTEGER NOT NULL,  
 INV\_QUANTITY\_ON\_HAND INTEGER  
)  
PARTITION BY RANGE(INV\_DATE\_SK)  
(  
 PARTITION P1 VALUES LESS THAN(2451179),  
 PARTITION P2 VALUES LESS THAN(2451544),  
 PARTITION P3 VALUES LESS THAN(2451910),  
 PARTITION P4 VALUES LESS THAN(2452275),  
 PARTITION P5 VALUES LESS THAN(2452640),  
 PARTITION P6 VALUES LESS THAN(2453005),  
 PARTITION P7 VALUES LESS THAN(MAXVALUE)  
);  
  
-- 创建索引ds\_inventory\_p1\_index1。  
vastbase=# CREATE INDEX ds\_inventory\_p1\_index1 ON tpcds.inventory\_p1 (INV\_ITEM\_SK) LOCAL;  
  
-- 对表tpcds.inventory\_p1进行聚集。  
vastbase=# CLUSTER tpcds.inventory\_p1 USING ds\_inventory\_p1\_index1;  
  
-- 对分区p3进行聚集。  
vastbase=# CLUSTER tpcds.inventory\_p1 PARTITION (p3) USING ds\_inventory\_p1\_index1;  
  
-- 对数据库中可以进行聚集的表进聚集。  
vastbase=# CLUSTER;  
  
--删除索引。  
vastbase=# DROP INDEX tpcds.ds\_inventory\_p1\_index1;  
  
--删除分区表。  
vastbase=# DROP TABLE tpcds.inventory\_p1;

### COMMENT

**功能描述**

定义或修改一个对象的注释。

**语法格式**

COMMENT ON  
{  
 AGGREGATE agg\_name (agg\_type [, ...] ) |  
 CAST (source\_type AS target\_type) |  
 COLLATION object\_name |  
 COLUMN { table\_name.column\_name | view\_name.column\_name } |  
 CONSTRAINT constraint\_name ON table\_name |  
 CONVERSION object\_name |  
 DATABASE object\_name |  
 DOMAIN object\_name |  
 EXTENSION object\_name |  
 FOREIGN DATA WRAPPER object\_name |  
 FOREIGN TABLE object\_name |  
 FUNCTION function\_name ( [ {[ argname ] [ argmode ] argtype} [, ...] ] ) |  
 INDEX object\_name |  
 LARGE OBJECT large\_object\_oid |  
 OPERATOR operator\_name (left\_type, right\_type) |  
 OPERATOR CLASS object\_name USING index\_method |  
 OPERATOR FAMILY object\_name USING index\_method |  
 [ PROCEDURAL ] LANGUAGE object\_name |  
 ROLE object\_name |  
 SCHEMA object\_name |  
 SEQUENCE object\_name |  
 SERVER object\_name |  
 TABLE object\_name |  
 TABLESPACE object\_name |  
 TEXT SEARCH CONFIGURATION object\_name |  
 TEXT SEARCH DICTIONARY object\_name |  
 TEXT SEARCH PARSER object\_name |  
 TEXT SEARCH TEMPLATE object\_name |  
 TYPE object\_name |  
 VIEW object\_name |  
 TRIGGER trigger\_name ON table\_name  
}  
 IS 'text';

**参数说明**

* **agg\_name**
* 聚集函数的名称。
* **agg\_type**
* 聚集函数参数的类型。
* **source\_type**
* 类型转换的源数据类型。
* **target\_type**
* 类型转换的目标数据类型。
* **object\_name**
* 对象名。
* **table\_name.column\_name**
* **view\_name.column\_name**
* 定义/修改注释的列名称。前缀可加表名称或者视图名称。
* **constraint\_name**
* 定义/修改注释的表约束的名称。
* **table\_name**
* 表的名称。
* **function\_name**
* 定义/修改注释的函数名称。
* **argname,argmode,argtype**
* 函数参数的模式、名称、类型。
* **large\_object\_oid**
* 定义/修改注释的大对象的OID值。
* **operator\_name**
* 操作符名称。
* **left\_type,right\_type**
* 操作参数的数据类型（可以用模式修饰）。当前置或者后置操作符不存在时，可以增加NONE选项。
* **trigger\_name**
* 触发器名称。
* **text**
* 注释。

**注意事项**

* 每个对象只存储一条注释，因此要修改一个注释，对同一个对象发出一条新的COMMENT命令即可。要删除注释，在文本字符串的位置写上NULL即可。当删除对象时，注释自动被删除掉。
* 目前注释浏览没有安全机制：任何连接到某数据库上的用户都可以看到所有该数据库对象的注释。共享对象（比如数据库、角色、表空间）的注释是全局存储的，连接到任何数据库的任何用户都可以看到它们。因此，不要在注释里存放与安全有关的敏感信息。
* 对大多数对象，只有对象的所有者或者被授予了对象COMMENT权限的用户可以设置注释，系统管理员默认拥有该权限。
* 角色没有所有者，所以COMMENT ON ROLE命令仅可以由系统管理员对系统管理员角色执行，有CREATEROLE权限的角色也可以为非系统管理员角色设置注释。系统管理员可以对所有对象进行注释。

**示例**

1、创建测试表。

CREATE TABLE customer\_demographics\_t2   
(   
CD\_DEMO\_SK INTEGER NOT NULL,   
CD\_GENDER CHAR(1) ,   
CD\_MARITAL\_STATUS CHAR(1) ,   
CD\_EDUCATION\_STATUS CHAR(20) ,   
CD\_PURCHASE\_ESTIMATE INTEGER ,   
CD\_CREDIT\_RATING CHAR(10) ,   
CD\_DEP\_COUNT INTEGER ,   
CD\_DEP\_EMPLOYED\_COUNT INTEGER ,   
CD\_DEP\_COLLEGE\_COUNT INTEGER   
)   
WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE) ;

2、为customer\_demographics\_t2.cd\_demo\_sk列加注释。

COMMENT ON COLUMN customer\_demographics\_t2.cd\_demo\_sk IS 'Primary key of customer demographics table.';

3、创建一个视图。

CREATE VIEW customer\_details\_view\_v2 AS   
SELECT \*   
FROM customer\_demographics\_t2;

4、为customer\_details\_view\_v2视图加注释。

COMMENT ON VIEW customer\_details\_view\_v2 IS 'View of customer detail';

5、删除view和customer\_demographics\_t2。

DROP VIEW customer\_details\_view\_v2;   
DROP TABLE customer\_demographics\_t2;

### COMMIT | END

**功能描述**

通过COMMIT或者END可完成提交事务的功能，即提交事务的所有操作。

**语法格式**

{ COMMIT | END } [ WORK | TRANSACTION ] ;

**参数说明**

* COMMIT | END：提交当前事务，让所有当前事务的更改为其他事务可见。
* WORK | TRANSACTION：可选关键字，除了增加可读性没有其他任何作用。

**注意事项**

无。

**示例**

1、创建表。

CREATE TABLE customer\_demographics\_t2   
 (   
 CD\_DEMO\_SK INTEGER NOT NULL,   
 CD\_GENDER CHAR(1),   
 CD\_MARITAL\_STATUS CHAR(1),   
 CD\_EDUCATION\_STATUS CHAR(20),   
 CD\_PURCHASE\_ESTIMATE INTEGER,   
 CD\_CREDIT\_RATING CHAR(10),   
 CD\_DEP\_COUNT INTEGER,   
 CD\_DEP\_EMPLOYED\_COUNT INTEGER,   
 CD\_DEP\_COLLEGE\_COUNT INTEGER   
 )   
 WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE)   
 ;

2、 开启事务。

START TRANSACTION;

3、插入数据。

INSERT INTO customer\_demographics\_t2 VALUES(1,'M', 'U', 'DOCTOR DEGREE', 1200, 'GOOD', 1, 0, 0);   
INSERT INTO customer\_demographics\_t2 VALUES(2,'F', 'U', 'MASTER DEGREE', 300, 'BAD', 1, 0, 0);

4、 提交事务，让所有更改永久化。

COMMIT;

5、 新建连接查询数据。

SELECT \* FROM customer\_demographics\_t2;

当结果显示如下信息时，则表示事务提交成功。

cd\_demo\_sk | cd\_gender | cd\_marital\_status | cd\_education\_status | cd\_purchase\_estimate | cd\_credi  
t\_rating | cd\_dep\_count | cd\_dep\_employed\_count | cd\_dep\_college\_count   
------------+-----------+-------------------+----------------------+----------------------+---------  
---------+--------------+-----------------------+----------------------  
 1 | M | U | DOCTOR DEGREE | 1200 | GOOD   
 | 1 | 0 | 0  
 2 | F | U | MASTER DEGREE | 300 | BAD   
 | 1 | 0 | 0  
(2 rows)

6、 删除表customer\_demographics\_t2。

DROP TABLE customer\_demographics\_t2;

### COMMIT PREPARED

**功能描述**

提交一个早先为两阶段提交准备好的事务。

**注意事项**

* 该功能仅在维护模式（GUC参数xc\_maintenance\_mode为on时）下可用。该模式谨慎打开，一般供维护人员排查问题使用，一般用户不应使用该模式。
* 命令执行者必须是该事务的创建者或系统管理员，且创建和提交操作只能在同一个会话中。
* 事务功能由数据库自动维护，不应显式使用事务功能。

**语法格式**

COMMIT PREPARED transaction\_id ;  
COMMIT PREPARED transaction\_id WITH CSN;

**参数说明**

* **transaction\_id**
* 待提交事务的标识符。它不能和任何当前预备事务已经使用了的标识符同名。
* **CSN（commit sequence number）**
* 待提交事务的序列号。它是一个64位递增无符号数。

**示例**

--提交标识符为的trans\_test的事务。  
vastbase=# COMMIT PREPARED 'trans\_test';

**相关链接**

参考：SQL语法参考->SQL语法中的PREPARE TRANSACTION、ROLLBACK PREPARED章节。

### COPY

**功能描述**

通过COPY命令实现在表和文件之间拷贝数据。

COPY FROM从一个文件拷贝数据到一个表，COPY TO把一个表的数据拷贝到一个文件。

**注意事项**

* 当参数enable\_copy\_server\_files关闭时，只允许初始用户执行COPY FROM FILENAME或COPY TO FILENAME命令，当参数enable\_copy\_server\_files打开，允许具有SYSADMIN权限的用户执行，但默认禁止对数据库配置文件、密钥文件、证书文件和审计日志执行COPY FROM FILENAME或COPY TO FILENAME，以防止用户越权查看或修改敏感文件。
* COPY只能用于表，不能用于视图。
* COPY TO需要读取的表的select权限，copy from需要插入的表的insert权限。
* 如果声明了一个字段列表，COPY将只在文件和表之间拷贝已声明字段的数据。如果表中有任何不在字段列表里的字段，COPY FROM将为那些字段插入缺省值。
* 如果声明了数据源文件，服务器必须可以访问该文件；如果指定了STDIN，数据将在客户前端和服务器之间流动，输入时，表的列与列之间使用TAB键分隔，在新的一行中以反斜杠和句点（.）表示输入结束。
* 如果数据文件的任意行包含比预期多或者少的字段，COPY FROM将抛出一个错误。
* 数据的结束可以用一个只包含反斜杠和句点（.）的行表示。如果从文件中读取数据，数据结束的标记是不必要的；如果在客户端应用之间拷贝数据，必须要有结束标记。
* COPY FROM中\N为空字符串，如果要输入实际数据值\N ，使用\N。
* COPY FROM不支持在导入过程中对数据做预处理（比如说表达式运算、填充指定默认值等）。如果需要在导入过程中对数据做预处理，用户需先把数据导入到临时表中，然后执行SQL语句通过运算插入到表中，但此方法会导致I/O膨胀，降低导入性能。
* COPY FROM在遇到数据格式错误时会回滚事务，但没有足够的错误信息，不方便用户从大量的原始数据中定位错误数据。
* COPY FROM/TO适合低并发，本地小数据量导入导出。
* 目标表存在trigger，支持COPY操作。

**语法格式**

* 从一个文件拷贝数据到一个表。

COPY table\_name [ ( column\_name [, ...] ) ]   
 FROM { 'filename' | STDIN }  
 [ [ USING ] DELIMITERS 'delimiters' ]  
 [ WITHOUT ESCAPING ]  
 [ LOG ERRORS ]  
 [ REJECT LIMIT 'limit' ]  
 [ WITH ( option [, ...] ) ]  
 | copy\_option  
 | TRANSFORM ( { column\_name [ data\_type ] [ AS transform\_expr ] } [, ...] )  
 | FIXED FORMATTER ( { column\_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy\_option [ ...] ] ];

fig: **说明：**

语法中的FIXED FORMATTER ( { column\_name( offset, length ) } [, ...] )以及 [ ( option [, ...] ) | copy\_option [ ...] ] 可以任意排列组合。

* 把一个表的数据拷贝到一个文件。

COPY table\_name [ ( column\_name [, ...] ) ]  
 TO { 'filename' | STDOUT }  
 [ [ USING ] DELIMITERS 'delimiters' ]  
 [ WITHOUT ESCAPING ]  
 [ WITH ( option [, ...] ) ]  
 | copy\_option  
 | FIXED FORMATTER ( { column\_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy\_option [ ...] ] ];  
  
COPY query  
 TO { 'filename' | STDOUT }  
 [ WITHOUT ESCAPING ]  
 [ WITH ( option [, ...] ) ]  
 | copy\_option  
 | FIXED FORMATTER ( { column\_name( offset, length ) } [, ...] ) [ ( option [, ...] ) | copy\_option [ ...] ] ];

fig: **说明：**

1. COPY TO语法形式约束如下：  
   (query)与[USING] DELIMITER不兼容，即若COPY TO的数据来自于一个query的查询结果，那么COPY TO语法不能再指定[USING] DELIMITERS语法子句。
2. 对于FIXED FORMATTTER语法后面跟随的copy\_option是以空格进行分隔的。
3. copy\_option是指COPY原生的参数形式，而option是兼容外表导入的参数形式。
4. 语法中的FIXED FORMATTER ( { column\_name( offset, length ) } [, ...] )以及 [ ( option [, ...] ) | copy\_option [ ...] ] 可以任意排列组合。

其中可选参数option子句语法为：

FORMAT 'format\_name'  
| OIDS [ boolean ]  
| DELIMITER 'delimiter\_character'  
| NULL 'null\_string'  
| HEADER [ boolean ]  
| FILEHEADER 'header\_file\_string'  
| FREEZE [ boolean ]  
| QUOTE 'quote\_coolean ]  
| FORCE\_QUOTE { ( column\_name [, ...] ) | \* }  
| FORCE\_NOT\_NULL ( column\_name [, ...] )  
| ENCODING 'encoding\_name'  
| IGNORE\_EXTRA\_DATA [ boolean ]  
| FILL\_MISSING\_FIELDS [ boolean ]  
| COMPATIBLE\_ILLEGAL\_CHARS [ boolean ]  
| DATE\_FORMAT 'date\_format\_string'  
| TIME\_FORMAT 'time\_format\_string'  
| TIMESTAMP\_FORMAT 'timestamp\_format\_string'  
| SMALLDATETIME\_FORMAT 'smalldatetime\_format\_string'

其中可选参数copy\_option子句语法为：

OIDS   
| NULL 'null\_string'   
| HEADER   
| FILEHEADER 'header\_file\_string'   
| FREEZE   
| FORCE\_NOT\_NULL column\_name [, ...]  
| FORCE\_QUOTE { column\_name [, ...] | \* }  
| BINARY   
| CSV   
| QUOTE [ AS ] 'quote\_character'   
| ESCAPE [ AS ] 'escape\_character'   
| EOL 'newline\_character'  
| ENCODING 'encoding\_name'   
| IGNORE\_EXTRA\_DATA   
| FILL\_MISSING\_FIELDS  
| COMPATIBLE\_ILLEGAL\_CHARS   
| DATE\_FORMAT 'date\_format\_string'   
| TIME\_FORMAT 'time\_format\_string'   
| TIMESTAMP\_FORMAT 'timestamp\_format\_string'   
| SMALLDATETIME\_FORMAT 'smalldatetime\_format\_string'

**参数说明**

* **query**
* 其结果将被拷贝。
* 取值范围：一个必须用圆括弧包围的SELECT或VALUES命令。
* **table\_name**
* 表的名称（可以有模式修饰）。
* 取值范围：已存在的表名。
* **column\_name**
* 可选的待拷贝字段列表。
* 取值范围：如果没有声明字段列表，将使用所有字段。
* **STDIN**
* 声明输入是来自标准输入。
* **STDOUT**
* 声明输出打印到标准输出。
* **FIXED**
* 打开字段固定长度模式。在字段固定长度模式下，不能声明DELIMITER、NULL、CSV选项。指定FIXED类型后，不能再通过option或copy\_option指定BINARY、CSV、TEXT等类型。
* fig: **说明：**   
  定长格式定义如下：
* 1、每条记录的每个字段长度相同。
* 2、长度不足的字段以空格填充，数字类型字段左对齐，字符字段右对齐。
* 3、字段和字段之间没有分隔符。
* **[USING] DELIMITER 'delimiters'**
* 在文件中分隔各个字段的字符串，分隔符最大长度不超过10个字节。
* 取值范围：不允许包含.abcdefghijklmnopqrstuvwxyz0123456789中的任何一个字符。
* 缺省值：在文本模式下，缺省是水平制表符，在CSV模式下是一个逗号。
* **WITHOUT ESCAPING**
* 在TEXT格式中，不对''和后面的字符进行转义。
* 取值范围：仅支持TEXT格式。
* **LOG ERRORS**
* 若指定，则开启对于COPY FROM语句中数据类型错误的容错机制。
* 取值范围：仅支持导入（即COPY FROM）时指定。
* fig: **说明：**   
  此容错选项的使用限制如下：
* 此容错机制仅捕捉COPY FROM过程中数据库主节点上数据解析过程中相关的数据类型错误（DATA\_EXCEPTION）。
* COPY已有的容错选项（如IGNORE\_EXTRA\_DATA）开启时，对应类型的错误会按照已有的方式处理而不会报出异常，因此错误表也不会有相应数据。
* **LOG ERRORS DATA**
* LOG ERRORS DATA和LOG ERRORS的区别:
* 1、LOG ERRORS DATA会填充容错表的rawrecord字段。
* 2、只有supper权限的用户才能使用LOG ERRORS DATA参数选项。
* fig: **注意：**
* 使用**“LOG ERRORS DATA”**时，若错误内容过于复杂可能存在写入容错表失败的风险，导致任务失败。
* **REJECT LIMIT** **'imit'**
* 与LOG ERROR选项共同使用，对COPY FROM的容错机制设置数值上限，一旦此COPY FROM语句错误数据超过选项指定条数，则会按照原有机制报错。
* 取值范围：正整数（1-INTMAX），'unlimited'（无最大值限制）
* 缺省值：若未指定LOG ERRORS，则会报错；若指定LOG ERRORS，则默认为0。

fig: **说明：**   
如上述LOG ERRORS中描述的容错机制，REJECT LIMIT的计数也是按照执行COPY FROM的数据库主节点上遇到的解析错误数量计算，而不是数据库节点的错误数量。

* **FORMATTER**
* 在固定长度模式中，定义每一个字段在数据文件中的位置。按照column(offset,length)格式定义每一列在数据文件中的位置。
* 取值范围：
* offset取值不能小于0，以字节为单位。
* length取值不能小于0，以字节为单位。
* 所有列的总长度和不能大于1GB。
* 文件中没有出现的列默认以空值代替。
* **OPTION { option\_name ' value ' }**
* 用于指定兼容外表的各类参数。
* FORMAT
  + 数据源文件的格式。
  + 取值范围：CSV、TEXT、FIXED、BINARY。
* CSV格式的文件，可以有效处理数据列中的换行符，但对一些特殊字符处理有欠缺。
* TEXT格式的文件，可以有效处理一些特殊字符，但无法正确处理数据列中的换行符。
* FIXED格式的文件，适用于每条数据的数据列都比较固定的数据，长度不足的列会添加空格补齐，过长的列则会自动截断。
* BINARY形式的选项会使得所有的数据被存储/读作二进制格式而不是文本。 这比TEXT和CSV格式的要快一些，但是一个BINARY格式文件可移植性比较差。
  + 缺省值：TEXT
* DELIMITER
  + 指定数据文件行数据的字段分隔符。
  + fig: **说明：**
* 分隔符不能是\r和\n。
* 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。
* TEXT格式数据的分隔符不能包含：小写字母、数字和特殊字符.\。
* 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。
* 分隔符推荐使用多字符和不可见字符。多字符例如'$^&'；不可见字符例如0x07、0x08、0x1b等。
  + 取值范围：支持多字符分隔符，但分隔符不能超过10个字节。
  + 缺省值：
* TEXT格式的默认分隔符是水平制表符（tab）。
* CSV格式的默认分隔符为“,”。
* FIXED格式没有分隔符。
* NULL
  + 用来指定数据文件中空值的表示。
  + 取值范围：
* null值不能是\r和\n，最大为100个字符。
* null值不能和分隔符、quote参数相同。
  + 缺省值：
* CSV格式下默认值是一个没有引号的空字符串。
* 在TEXT格式下默认值是\N。
* HEADER
  + 指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV、FIXED格式的文件中。
  + 在导入数据时，如果header选项为on，则数据文本第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。
  + 在导出数据时，如果header选项为on，则需要指定fileheader。如果header为off，则导出数据文件不包含标题行。
  + 取值范围：true/on、false/off。
  + 缺省值：false
* QUOTE
  + CSV格式文件下的引号字符。
  + 缺省值：双引号
  + fig: **说明：**
* quote参数不能和分隔符、null参数相同。
* quote参数只能是单字节的字符。
* 推荐不可见字符作为quote，例如0x07、0x08、0x1b等。
* ESCAPE
* CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。
* 缺省值：双引号。当与quote值相同时，会被替换为'\0'。
* EOL 'newline\_character'
* 指定导入导出数据文件换行符样式。
* 取值范围：支持多字符换行符，但换行符不能超过10个字节。常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。
  + fig: **说明：**
* EOL参数只能用于TEXT格式的导入导出，不支持CSV格式和FIXED格式导入。为了兼容原有EOL参数，仍然支持导出CSV格式和FIXED格式时指定EOL参数为0x0D或0x0D0A。
* EOL参数不能和分隔符、null参数相同。
* EOL参数不能包含：.abcdefghijklmnopqrstuvwxyz0123456789。
* FORCE\_QUOTE { ( column\_name [, ...] ) | \* }
* 在CSV COPY TO模式下，强制在每个声明的字段周围对所有非NULL值都使用引号包围。NULL输出不会被引号包围。
* 取值范围：已存在的字段。
* FORCE\_NOT\_NULL ( column\_name [, ...] )
* 在CSV COPY FROM模式下，指定的字段输入不能为空。
* 取值范围：已存在的字段。
* ENCODING
* 指定数据文件的编码格式名称，缺省为当前数据库编码格式。
* IGNORE\_EXTRA\_DATA
* 若数据源文件比外表定义列数多，是否会忽略对多出的列。该参数只在数据导入过程中使用。
* 取值范围：true/on、false/off。
* 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。
* 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。
  + extra data after last expected column
* 缺省值：false。
* fig: **须知：**
* 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。
* COMPATIBLE\_ILLEGAL\_CHARS
* 导入非法字符容错参数。此语法仅对COPY FROM导入有效。
* 取值范围：true/on、false/off。
* 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。
* 参数为false/off，导入时遇到非法字符进行报错，中断导入。
* 缺省值：false/off
* fig: **说明：**   
  导入非法字符容错规则如下：  
  （1）对于'\0'，容错后转换为空格；  
  （2）对于其他非法字符，容错后转换为问号；
* （3）若compatible\_illegal\_chars为true/on标识导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如“illegal chars conversion may confuse COPY escape 0x20”等报错信息提示用户修改可能引起混淆的参数以避免导入错误。
* FILL\_MISSING\_FIELDS
* 当数据加载时，若数据源文件中一行的最后一个字段缺失的处理方式。
* 取值范围：true/on、false/off。
* 缺省值：false/off
* DATE\_FORMAT
* 导入对于DATE类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法DATE格式。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* fig: **说明：**
* 对于DATE类型内建为TIMESTAMP类型的数据库，在导入的时候，若需指定格式，可以参考下面的timestamp\_format参数。
* TIME\_FORMAT
* 导入对于TIME类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法TIME格式，不支持时区。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* TIMESTAMP\_FORMAT
* 导入对于TIMESTAMP类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法TIMESTAMP格式，不支持时区。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* SMALLDATETIME\_FORMAT
* 导入对于SMALLDATETIME类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法SMALLDATETIME格式。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* **COPY\_OPTION { option\_name ' value ' }**
* 用于指定COPY原生的各类参数。
* NULL null\_string
  + 用来指定数据文件中空值的表示。

fig: **须知：**   
在使用COPY FROM的时候，任何匹配这个字符串的字符串将被存储为NULL值，所以应该确保指定的字符串和COPY TO相同。

* + 取值范围：
* null值不能是\r和\n，最大为100个字符。
* null值不能和分隔符、quote参数相同。
  + 缺省值：
* 在TEXT格式下默认值是\N。
* CSV格式下默认值是一个没有引号的空字符串。
* HEADER
  + 指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV、FIXED格式的文件中。
  + 在导入数据时，如果header选项为on，则数据文本第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。
  + 在导出数据时，如果header选项为on，则需要指定fileheader。如果header为off，则导出数据文件不包含标题行。
* FILEHEADER
  + 导出数据时用于定义标题行的文件，一般用来描述每一列的数据信息。
  + fig: **须知：**
* 仅在header为on或true的情况下有效。
* fileheader指定的是绝对路径。
* 该文件只能包含一行标题信息，并以换行符结尾，多余的行将被丢弃（标题信息不能包含换行符）。
* 该文件包括换行符在内长度不超过1M。
* FREEZE
* 将COPY加载的数据行设置为已经被frozen，就像这些数据行执行过VACUUM FREEZE。
* 这是一个初始数据加载的性能选项。仅当以下三个条件同时满足时，数据行会被frozen：
* 在同一事务中create或truncate这张表之后执行COPY。
* 当前事务中没有打开的游标。
* 当前事务中没有原有的快照。

fig: **说明：**

COPY完成后，所有其他会话将会立刻看到这些数据。但是这违反了MVCC可见性的一般原则，用户应当了解这样会导致潜在的风险。

* FORCE NOT NULL column\_name [, ...]
* 在CSV COPY FROM模式下，指定的字段不为空。若输入为空，则将视为长度为0的字符串。
* 取值范围：已存在的字段。
* FORCE QUOTE { column\_name [, ...] | \* }
* 在CSV COPY TO模式下，强制在每个声明的字段周围对所有非NULL值都使用引号包围。NULL输出不会被引号包围。
* 取值范围：已存在的字段。
* BINARY
* 使用二进制格式存储和读取，而不是以文本的方式。在二进制模式下，不能声明DELIMITER、NULL、CSV选项。指定BINARY类型后，不能再通过option或copy\_option指定CSV、FIXED、TEXT等类型。
* CSV
* 打开逗号分隔变量（CSV）模式。指定CSV类型后，不能再通过option或copy\_option指定BINARY、FIXED、TEXT等类型。
* QUOTE [AS] 'quote\_character'
* CSV格式文件下的引号字符。
* 缺省值：双引号。
* fig: **说明：**
* quote参数不能和分隔符、null参数相同。
* quote参数只能是单字节的字符。
* 推荐不可见字符作为quote，例如0x07、0x08、0x1b等。
* ESCAPE [AS] 'escape\_character'
* CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。
* 默认值为双引号。当与quote值相同时，会被替换为'\0'。
* EOL 'newline\_character'
* 指定导入导出数据文件换行符样式。
* 取值范围：支持多字符换行符，但换行符不能超过10个字节。常见的换行符，如\r、\n、\r\n（设成0x0D、0x0A、0x0D0A效果是相同的），其他字符或字符串，如$、#。
* fig: **说明：**
* EOL参数只能用于TEXT格式的导入导出，不支持CSV格式和FIXED格式。为了兼容原有EOL参数，仍然支持导出CSV格式和FIXED格式时指定EOL参数为0x0D或0x0D0A。
* EOL参数不能和分隔符、null参数相同。
* EOL参数不能包含：.abcdefghijklmnopqrstuvwxyz0123456789。
* ENCODING 'encoding\_name'
* 指定文件编码格式名称。
* 取值范围：有效的编码格式。
* 缺省值：当前编码格式。
* IGNORE\_EXTRA\_DATA
* 指定当数据源文件比外表定义列数多时，忽略行尾多出来的列。该参数只在数据导入过程中使用。
* 若不使用该参数，在数据源文件比外表定义列数多，会显示如下错误信息。

extra data after last expected column

* COMPATIBLE\_ILLEGAL\_CHARS
* 指定导入时对非法字符进行容错处理，非法字符转换后入库。不报错，不中断导入。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 若不使用该参数，导入时遇到非法字符进行报错，中断导入。
* fig: **说明：**   
  导入非法字符容错规则如下：
* （1）对于'\0'，容错后转换为空格；
* （2）对于其他非法字符，容错后转换为问号；
* （3）若compatible\_illegal\_chars为true/on标识，导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如“illegal chars conversion may confuse COPY escape 0x20”等报错信息提示用户修改可能引起混淆的参数以避免导入错误。
* FILL\_MISSING\_FIELDS
* 当数据加载时，若数据源文件中一行的最后一个字段缺失的处理方式。
* 取值范围：true/on、false/off。
* 缺省值：false/off。
* fig: **须知：**
* 目前COPY指定此Option实际不会生效，即不会有相应的容错处理效果（不生效）。需要额外注意的是，打开此选项会导致解析器在数据库主节点数据解析阶段（即COPY错误表容错的涵盖范围）忽略此数据问题，而到数据库节点重新报错，从而使得COPY错误表（打开LOG ERRORS REJECT LIMIT）在此选项打开的情况下无法成功捕获这类少列的数据异常。因此请不要指定此选项。
* DATE\_FORMAT 'date\_format\_string'
* 导入对于DATE类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法DATE格式。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* fig: **说明：**
* 对于DATE类型内建为TIMESTAMP类型的数据库，在导入的时候，若需指定格式，可以参考下面的timestamp\_format参数。
* TIME\_FORMAT 'time\_format\_string'
* 导入对于TIME类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法TIME格式，不支持时区。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* TIMESTAMP\_FORMAT 'timestamp\_format\_string'
* 导入对于TIMESTAMP类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法TIMESTAMP格式，不支持时区。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* SMALLDATETIME\_FORMAT 'smalldatetime\_format\_string'
* 导入对于SMALLDATETIME类型指定格式。此参数不支持BINARY格式，会报“cannot specify bulkload compatibility options in BINARY mode”错误信息。此参数仅对COPY FROM导入有效。
* 取值范围：合法SMALLDATETIME格式。可参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符章节。
* TRANSFORM ( { column\_name [ data\_type ] [ AS transform\_expr ] } [, ...] )
* 指定表中各个列的转换表达式；其中data\_type指定该列在表达式参数中的数据类型；transform\_expr为目标表达式，返回与表中目标列数据类型一致的结果值，表达式可参考：SQL语法参考->SQL基本要素->表达式章节。
* COPY FROM能够识别的特殊反斜杠序列如下所示。
* **\b**：反斜杠 （ASCII 8）
* **\f**：换页（ASCII 12）
* **\n**：换行符 （ASCII 10）
* **\r**：回车符 （ASCII 13）
* **\t**：水平制表符 （ASCII 9）
* **\v**：垂直制表符 （ASCII 11）
* **\digits**：反斜杠后面跟着一到三个八进制数，表示ASCII值为该数的字符。
* **\xdigits**：反斜杠x后面跟着一个或两个十六进制位声明指定数值编码的字符。

**示例**

--将ship\_mode中的数据拷贝到/home/omm/ds\_ship\_mode.dat文件中。  
vastbase=# COPY ship\_mode TO '/home/omm/ds\_ship\_mode.dat';  
  
--将ship\_mode 输出到stdout。  
vastbase=# COPY ship\_mode TO stdout;  
  
--创建ship\_mode\_t1表。  
vastbase=# CREATE TABLE ship\_mode\_t1  
(  
 SM\_SHIP\_MODE\_SK INTEGER NOT NULL,  
 SM\_SHIP\_MODE\_ID CHAR(16) NOT NULL,  
 SM\_TYPE CHAR(30) ,  
 SM\_CODE CHAR(10) ,  
 SM\_CARRIER CHAR(20) ,  
 SM\_CONTRACT CHAR(20)  
)  
WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE)  
;  
  
--从stdin拷贝数据到表ship\_mode\_t1。  
vastbase=# COPY ship\_mode\_t1 FROM stdin;  
  
--从/home/omm/ds\_ship\_mode.dat文件拷贝数据到表ship\_mode\_t1。  
vastbase=# COPY ship\_mode\_t1 FROM '/home/omm/ds\_ship\_mode.dat';  
  
--从/home/omm/ds\_ship\_mode.dat文件拷贝数据到表ship\_mode\_t1，应用TRANSFORM表达式转换，取SM\_TYPE列左边10个字符插入到表中。  
vastbase=# COPY ship\_mode\_t1 FROM '/home/omm/ds\_ship\_mode.dat' TRANSFORM (SM\_TYPE AS LEFT(SM\_TYPE, 10));  
  
--从/home/omm/ds\_ship\_mode.dat文件拷贝数据到表ship\_mode\_t1，使用参数如下：导入格式为TEXT（format 'text'），分隔符为'\t'（delimiter E'\t'），忽略多余列（ignore\_extra\_data 'true'），不指定转义（noescaping 'true'）。  
vastbase=# COPY ship\_mode\_t1 FROM '/home/omm/ds\_ship\_mode.dat' WITH(format 'text', delimiter E'\t', ignore\_extra\_data 'true', noescaping 'true');  
  
--从/home/omm/ds\_ship\_mode.dat文件拷贝数据到表ship\_mode\_t1，使用参数如下：导入格式为FIXED（FIXED），指定定长格式（FORMATTER(SM\_SHIP\_MODE\_SK(0, 2), SM\_SHIP\_MODE\_ID(2,16), SM\_TYPE(18,30), SM\_CODE(50,10), SM\_CARRIER(61,20), SM\_CONTRACT(82,20))），忽略多余列（ignore\_extra\_data），有数据头（header）。  
vastbase=# COPY ship\_mode\_t1 FROM '/home/omm/ds\_ship\_mode.dat' FIXED FORMATTER(SM\_SHIP\_MODE\_SK(0, 2), SM\_SHIP\_MODE\_ID(2,16), SM\_TYPE(18,30), SM\_CODE(50,10), SM\_CARRIER(61,20), SM\_CONTRACT(82,20)) header ignore\_extra\_data;  
  
--删除ship\_mode\_t1。  
vastbase=# DROP TABLE ship\_mode\_t1;

### CREATE AGGREGATE

**功能描述**

定义一个新的聚合函数。

**语法格式**

CREATE AGGREGATE name ( input\_data\_type [ , ... ] ) (  
 SFUNC = sfunc,  
 STYPE = state\_data\_type  
 [ , FINALFUNC = ffunc ]  
 [ , INITCOND = initial\_condition ]  
 [ , SORTOP = sort\_operator ]  
)  
  
or the old syntax  
  
CREATE AGGREGATE name (  
 BASETYPE = base\_type,  
 SFUNC = sfunc,  
 STYPE = state\_data\_type  
 [ , FINALFUNC = ffunc ]  
 [ , INITCOND = initial\_condition ]  
 [ , SORTOP = sort\_operator ]  
)

**参数说明**

* **name**
* 要创建的聚合函数名(可以有模式修饰) 。
* **input\_data\_type**
* 该聚合函数要处理的输入数据类型。要创建一个零参数聚合函数，可以使用代替输入数据类型列表。（count()就是这种聚合函数的一个实例。）
* **base\_type**
* 在以前的CREATE AGGREGATE语法中，输入数据类型是通过basetype参数指定的，而不是写在聚合的名称之后。 需要注意的是这种以前语法仅允许一个输入参数。 要创建一个零参数聚合函数，可以将basetype指定为“ANY”(而不是\*)。
* **sfunc**
* 将在每一个输入行上调用的状态转换函数的名称。 对于有N个参数的聚合函数，sfunc必须有 +1 个参数，其中的第一个参数类型为state\_data\_type，其余的匹配已声明的输入数据类型。 函数必须返回一个state\_data\_type类型的值。 这个函数接受当前状态值和当前输入数据，并返回下个状态值。
* **state\_data\_type**
* 聚合的状态值的数据类型。
* **ffunc**
* 在转换完所有输入行后调用的最终处理函数，它计算聚合的结果。 此函数必须接受一个类型为state\_data\_type的参数。 聚合的输出数据 类型被定义为此函数的返回类型。 如果没有声明ffunc则使用聚合结果的状态值作为聚合的结果，且输出类型为state\_data\_type。
* **initial\_condition**
* 状态值的初始设置(值)。 它必须是一个state\_data\_type类型可以接受的文本常量值。 如果没有声明，状态值初始为 NULL 。
* **sort\_operator**
* 用于MIN或MAX类型聚合的排序操作符。 这个只是一个操作符名 (可以有模式修饰)。这个操作符假设接受和聚合一样的输入数据类型。

**示例**

CREATE AGGREGATE sum (complex)  
(  
 sfunc = complex\_add,  
 stype = complex,  
 initcond = '(0,0)'  
);  
  
SELECT sum(a) FROM test\_complex;  
  
 sum  
-----------  
 (34,53.9)

### CREATE AUDIT POLICY

**功能描述**

创建统一审计策略。

**注意事项**

只有poladmin、sysadmin或初始用户能进行此操作。

需要开启安全策略开关，即设置GUC参数enable\_security\_policy=on，审计策略才可以生效。

**语法格式**

CREATE AUDIT POLICY [ IF NOT EXISTS ] policy\_name { { privilege\_audit\_clause | access\_audit\_clause } [ filter\_group\_clause ] [ ENABLE | DISABLE ] };

* privilege\_audit\_clause：

PRIVILEGES { DDL | ALL } [ ON LABEL ( resource\_label\_name [, ... ] ) ]

* access\_audit\_clause：

ACCESS { DML | ALL } [ ON LABEL ( resource\_label\_name [, ... ] ) ]

* filter\_group\_clause：

FILTER ON { ( FILTER\_TYPE ( filter\_value [, ... ] ) ) [, ... ] }

**参数说明**

* **policy\_name**
* 审计策略名称，需要唯一，不可重复；
* 取值范围：字符串，要符合标识符的命名规范。
* **DDL**
* 指的是针对数据库执行如下操作时进行审计，目前支持：CREATE、ALTER、DROP、ANALYZE、COMMENT、GRANT、REVOKE、SET、SHOW、LOGIN\_ANY、LOGIN\_FAILURE、LOGIN\_SUCCESS、LOGOUT。
* **ALL**
* 指的是上述DDL支持的所有对数据库的操作。
* **resource\_label\_name**
* 资源标签名称。
* **DML**
* 指的是针对数据库执行如下操作时进行审计，目前支持：SELECT、COPY、DEALLOCATE、DELETE、EXECUTE、INSERT、PREPARE、REINDEX、TRUNCATE、UPDATE。
* **FILTER\_TYPE**
* 描述策略过滤的条件类型，包括IP | APP | ROLES。
* **filter\_value**
* 指具体过滤信息内容。
* **ENABLE|DISABLE**
* 可以打开或关闭统一审计策略。若不指定ENABLE|DISABLE，语句默认为ENABLE。

**示例**

--创建dev\_audit和bob\_audit用户。  
vastbase=# CREATE USER dev\_audit PASSWORD 'dev@1234';  
CREATE USER bob\_audit password 'bob@1234';  
  
--创建一个表tb\_for\_audit  
vastbase=# CREATE TABLE tb\_for\_audit(col1 text, col2 text, col3 text);  
  
--创建资源标签  
vastbase=# CREATE RESOURCE LABEL adt\_lb0 add TABLE(tb\_for\_audit);  
  
--对数据库执行create操作创建审计策略  
vastbase=# CREATE AUDIT POLICY adt1 PRIVILEGES CREATE;  
  
--对数据库执行select操作创建审计策略  
vastbase=# CREATE AUDIT POLICY adt2 ACCESS SELECT;  
  
--仅审计记录用户dev\_audit和bob\_audit在执行针对adt\_lb0资源进行的create操作数据库创建审计策略  
vastbase=# CREATE AUDIT POLICY adt3 PRIVILEGES CREATE ON LABEL(adt\_lb0) FILTER ON ROLES(dev\_audit, bob\_audit);  
  
--仅审计记录用户dev\_audit和bob\_audit,客户端工具为psql和gsql，IP地址为'10.20.30.40', '127.0.0.0/24'，在执行针对adt\_lb0资源进行的select、insert、delete操作数据库创建审计策略。  
vastbase=# CREATE AUDIT POLICY adt4 ACCESS SELECT ON LABEL(adt\_lb0), INSERT ON LABEL(adt\_lb0), DELETE FILTER ON ROLES(dev\_audit, bob\_audit), APP(psql, gsql), IP('10.20.30.40', '127.0.0.0/24');

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER AUDIT POLICY、DROP AUDIT POLICY章节。

### CREATE CAST

**功能描述**

定义一个用户自定义的转换。

**语法格式**

CREATE CAST (source\_type AS target\_type)  
 WITH FUNCTION function\_name (argument\_type [, ...])  
 [ AS ASSIGNMENT | AS IMPLICIT ]  
  
CREATE CAST (source\_type AS target\_type)  
 WITHOUT FUNCTION  
 [ AS ASSIGNMENT | AS IMPLICIT ]  
  
CREATE CAST (source\_type AS target\_type)  
 WITH INOUT  
 [ AS ASSIGNMENT | AS IMPLICIT ]

**参数说明**

* **source\_type**
* 转换的源数据类型。
* **target\_type**
* 转换的目标数据类型。
* **function\_name(argument\_type [, ...])**
* 用于执行转换的函数。这个函数名可以是用模式名修饰的。如果它没有用模式名修饰，那么该函数将从模式搜索路径中找出来。函数的结果数据类型必须匹配转换的目标类型。 它的参数在下面讨论。
* **WITHOUT FUNCTION**
* 表明源类型是对目标类型是二进制可强制转换的，所以没有函数需要执行此转换。
* **WITH INOUT**
* 表明转换是I/O转换，通过调用源数据类型的输出函数来执行，并将结果传给目标数据类型的输入函数。
* **AS ASSIGNMENT**
* 表示转换可以在赋值模式下隐含调用。
* **AS IMPLICIT**
* 表示转换可以在任何环境里隐含调用。
* 转换实现函数可以有一到三个参数。第一个参数的类型必须与转换的源类型相同的，或可以从转换的源类型二进制可强制转换的。第二个参数，如果存在，必须是integer类型；它接收这些与目标类型相关联的类型修饰符，或者若什么都没有则是-1。第三个参数，如果存在，必须是boolean类型；若转换是一个显式类型转换则会收到true，否则是false。
* 一个转换函数的返回类型必须是与转换的目标类型相同或者对转换的目标类型二进制可强制转换 。
* 通常，一个转换必须有不同的源和目标数据类型。然而，若有多于一个参数的转换实现函数，则允许声明一个有相同的源和目标类型的转换。这用于表示系统目录中的特定类型的长度强制函数。命名的函数用于强制一个该类型的值为第二个参数给出的类型修饰符值。
* 如果一个类型转换的源类型和目标类型不同，并且接收多于一个参数，它就表示从一种类型转换成另外一种类型只用一个步骤，并且同时实施长度转换。如果没有这样的项可用， 那么转换成一个使用了类型修饰词的类型将涉及两个步骤，一个是在数据类型之间转换， 另外一个是施加修饰词指定的转换。
* 对域类型的转换目前没有作用。转换一般是针对域相关的所属数据类型。

**示例**

为了从类型bigint到类型int4创建一个指派映射要通过使用函数int4(bigint):

CREATE CAST (bigint AS int4) WITH FUNCTION int4(bigint) AS ASSIGNMENT;

（这个转换在系统中已经预先定义了。）

**兼容性**

CREATE CAST指令符合SQL标准，除了SQL没有为二进制可强制转换类型或者实现函数的额外参数来实现功能。

### CREATE CLIENT MASTER KEY

**功能描述**

创建一个客户端主密钥对象，该对象可用于加密Column Encryption Key对象。

**注意事项**

本语法属于全密态数据库特有语法。

当使用vql连接数据库服务器时，需使用‘-C’参数，打开全密态数据库的开关，才能使用本语法。

由本语法创建的CMK对象中，仅存储从独立的密钥管理工具/服务/组件中读取密钥的方法，而不存储密钥本身。

**语法格式**

CREATE CLIENT MASTER KEY client\_master\_key\_name WITH （KEY\_STORE = key\_store\_name, KEY\_PATH = "key\_path\_value", ALGORITHM = algorithm\_type）

**参数说明**

* **client\_master\_key\_name**
* 该参数作为密钥对象名，在同一命名空间下，需满足命名唯一性约束。
* 取值范围：字符串，需符合标识符的命名规范。
* **KEY\_STORE**
* 指定管理CMK的密钥工具或组件；取值：目前仅支持localkms。
* **KEY\_PATH**
* KEY\_STORE负责管理多个CMK密钥，KEY\_PATH选项用于在KEY\_STORE中唯一标识CMK。取值类似：“key\_path\_value”。
* **ALGORITHM**
* 由本语法创建的用于加密COLUMN ENCRYPTION KEY，该参数用于指定加密算法的类型。取值范围：RSA\_2048、RSA\_3072和SM2。
* fig: **说明：**   
  **密钥存储路径：**默认情况下，localkms将在$LOCALKMS\_FILE\_PATH路径下生成/读取/删除密钥文件，用户可手动配置该环境变量。但是，用户也可以不用单独配置该环境变量，在尝试获取$LOCALKMS\_FILE\_PATH失败时，localkms会尝试获取$GAUSSHOME/etc/localkms/路径，如果该路径存在，则将其作为密钥存储路径。  
  **密钥相关文件名：**使用CREATE CMK语法时，localkms将会创建四个与存储密钥相关的文件。示例：当KEY\_PATH = "key\_path\_value", 四个文件的名称分别为key\_path\_value.pub、key\_path\_value.pub.rand、 key\_path\_value.priv、 key\_path\_value.priv.rand。  
  所以，为了能够成功创建密钥相关文件，在密钥存储路径下，应该保证没有已存在的与密钥相关文件名同名的文件。

**示例**

使用普通账户alice，连接全密态数据库。

[cmd] vql -U alice -h $host -p $port -d $database -C -r

使用本语法创建客户端加密主密钥(CMK)对象

vastbase=> CREATE CLIENT MASTER KEY a\_cmk WITH (KEY\_STORE = localkms, KEY\_PATH = "key\_path\_value", ALGORITHM = RSA\_2048);  
vastbase=> CREATE CLIENT MASTER KEY another\_cmk WITH (KEY\_STORE = localkms, KEY\_PATH = "another\_path\_value", ALGORITHM = SM2);

### CREATE COLUMN ENCRYPTION KEY

**功能描述**

创建一个列加密密钥，该密钥可用于加密表中指定列。

**注意事项**

本语法属于全密态数据库特有语法。

当使用vsql连接数据库服务器时，需使用‘-C’参数，打开全密态数据库的开关，才能使用本语法。

由该语法创建CEK对象可用于列级加密。在定义表中列字段时，可指定一个CEK对象，用于加密该列。

**语法格式**

CREATE COLUMN ENCRYPTION KEY column\_encryption\_key\_name WITH VALUES(CLIENT\_MASTER\_KEY = client\_master\_key\_name, ALGORITHM = algorithm\_type, ENCRYPTED\_VALUE = encrypted\_value);

**参数说明**

* **column\_encryption\_key\_name**
* 该参数作为密钥对象名，在同一命名空间下，需满足命名唯一性约束。
* 取值范围：字符串，要符合标识符的命名规范。
* **CLIENT\_MASTER\_KEY**
* 指定用于加密本CEK的CMK，取值为：CMK对象名，该CMK对象由CREATE CLIENT MASTER KEY语法创建。
* **ALGORITHM**
* 指定该CEK将用于何种加密算法，取值范围为：AEAD\_AES\_256\_CBC\_HMAC\_SHA256、AEAD\_AES\_128\_CBC\_HMAC\_SHA256和SM4\_SM3；
* **ENCRYPTED\_VALUE（可选项）**
* 该值为用户指定的密钥口令，密钥口令长度范围为28 ~ 256位，28位派生出来的密钥安全强度满足AES128，若用户需要用AES256，密钥口令的长度需要39位，如果不指定，则会自动生成256字符的密钥。
* fig: **须知：**   
  国密算法约束：由于SM2、SM3、SM4等算法属于中国国家密码标准算法，为规避法律风险，需配套使用。即如果将CEK用于SM4\_SM3算法，则仅能使用SM2算法来对该CEK进行加密。

**示例**

--创建列加密密钥(CEK)  
vastbase=> CREATE COLUMN ENCRYPTION KEY a\_cek WITH VALUES (CLIENT\_MASTER\_KEY = a\_cmk, ALGORITHM = AEAD\_AES\_256\_CBC\_HMAC\_SHA256);  
CREATE COLUMN ENCRYPTION KEY  
vastbase=> CREATE COLUMN ENCRYPTION KEY another\_cek WITH VALUES (CLIENT\_MASTER\_KEY = a\_cmk, ALGORITHM = SM4\_SM3);  
CREATE COLUMN ENCRYPTION KEY

### CREATE DATA SOURCE

**功能描述**

创建一个新的外部数据源对象，该对象用于定义Vastbase要连接的目标库信息。

**注意事项**

* Data Source名称在数据库中需唯一，遵循标识符命名规范，长度限制为63字节，过长则会被截断。
* 只有系统管理员或初始用户才有权限创建Data Source对象。且创建该对象的用户为其默认属主。
* 当在OPTIONS中出现password选项时，需要保证Vastbase每个节点的$GAUSSHOME/bin目录下存在datasource.key.cipher和datasource.key.rand文件，如果不存在这两个文件，请使用vb\_guc工具生成并发布到Vastbase每个节点的$GAUSSHOME/bin目录下。

**语法格式**

CREATE DATA SOURCE src\_name  
 [TYPE 'type\_str']  
 [VERSION {'version\_str' | NULL}]  
 [OPTIONS (optname 'optvalue' [, ...])];

**参数说明**

* **src\_name**
* 新建Data Source对象的名称，需在数据库内部唯一。
* 取值范围：字符串，要符标识符的命名规范。
* **TYPE**
* 新建Data Source对象的类型，可缺省。
* 取值范围：空串或非空字符串。
* **VERSION**
* 新建Data Source对象的版本号，可缺省或NULL值。
* 取值范围：空串或非空字符串或NULL。
* **OPTIONS**
* Data Source对象的选项字段，创建时可省略，如若指定，其关键字如下：
* optname
  + 选项名称。
  + 取值范围：dsn、 username、 password、 encoding。不区分大小写。
* dsn对应odbc配置文件中的DSN。
* username/password对应连接目标库的用户名和密码。
* Vastbase在后台会对用户输入的username/password加密以保证安全性。该加密所需密钥文件需要使用vb\_guc工具生成并发布到Vastbase每个节点的$GAUSSHOME/bin目录下。username/password不应当包含'encryptOpt'前缀，否则会被认为是加密后的密文。
* encoding表示与目标库交互的字符串编码方式（含发送的SQL语句和返回的字符类型数据），此处创建对象时不检查encoding取值的合法性，能否正确编解码取决于用户提供的编码方式是否在数据库本身支持的字符编码范围内。
* optvalue
  + 选项值。
  + 取值范围：空或者非空字符串。

**示例**

--创建一个空的Data Source对象，不含任何信息。  
vastbase=# CREATE DATA SOURCE ds\_test1;  
  
--创建一个Data Source对象，含TYPE信息，VERSION为NULL。  
vastbase=# CREATE DATA SOURCE ds\_test2 TYPE 'MPPDB' VERSION NULL;  
  
--创建一个Data Source对象，仅含OPTIONS。  
vastbase=# CREATE DATA SOURCE ds\_test3 OPTIONS (dsn 'Vastbase', encoding 'utf8');  
  
--创建一个Data Source对象，含TYPE, VERSION, OPTIONS。  
vastbase=# CREATE DATA SOURCE ds\_test4 TYPE 'unknown' VERSION '11.2.3' OPTIONS (dsn 'Vastbase', username 'userid', password 'pwd@123456', encoding '');  
  
--删除Data Source对象。  
vastbase=# DROP DATA SOURCE ds\_test1;  
vastbase=# DROP DATA SOURCE ds\_test2;  
vastbase=# DROP DATA SOURCE ds\_test3;  
vastbase=# DROP DATA SOURCE ds\_test4;

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER DATA SOURCE、DROP DATA SOURCE章节。

### CREATE DATABASE

**功能描述**

创建一个新的数据库。缺省情况下新数据库将通过复制标准系统数据库template0来创建，且仅支持使用template0来创建。

**注意事项**

* 只有拥有CREATEDB权限的用户才可以创建新数据库，系统管理员默认拥有此权限。
* 不能在事务块中执行创建数据库语句。
* 在创建数据库过程中，出现类似“Permission denied”的错误提示，可能是由于文件系统上数据目录的权限不足。出现类似“No space left on device”的错误提示，可能是由于磁盘满引起的。
* 事务中不支持创建database。
* 当新建数据库Encoding、LC-Collate 或LC\_Ctype与模板数据库（SQL\_ASCII）不匹配（为'GBK' /'UTF8'/'LATIN1'）时，必须指定template [=] template0。

**语法格式**

CREATE DATABASE database\_name  
 [ [ WITH ] { [ OWNER [=] user\_name ] |  
 [ TEMPLATE [=] template ] |  
 [ ENCODING [=] encoding ] |  
 [ LC\_COLLATE [=] lc\_collate ] |  
 [ LC\_CTYPE [=] lc\_ctype ] |  
 [ DBCOMPATIBILITY [=] compatibilty\_type ] |  
 [ TABLESPACE [=] tablespace\_name ] |  
 [ CONNECTION LIMIT [=] connlimit ]}[...] ];

**参数说明**

* **database\_name**
* 数据库名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **OWNER [ = ] user\_name**
* 数据库所有者。缺省时，新数据库的所有者是当前用户。
* 取值范围：已存在的用户名。
* **TEMPLATE [ = ] template**
* 模板名。即从哪个模板创建新数据库。Vastbase采用从模板数据库复制的方式来创建新的数据库。初始时，Vastbase包含两个模板数据库template0、template1，以及一个默认的用户数据库postgres。
* 取值范围：仅template0。
* **ENCODING [ = ] encoding**
* 指定数据库使用的字符编码，可以是字符串（如'SQL\_ASCII'）、整数编号。
* 不指定时，默认使用模版数据库的编码。模板数据库template0和template1的编码默认与操作系统环境相关。template1不允许修改字符编码，因此若要变更编码，请使用template0创建数据库。
* 常用取值：GBK、UTF8、Latin1。
* **表 1** Vastbase字符集

| **名称** | **描述** | **语言** | **是否服务器端？** | **ICU?** | **字节/字符** | **别名** |
| --- | --- | --- | --- | --- | --- | --- |
| BIG5 | Big Five | 繁体中文 | 否 | 否 | 1-2 | WIN950，Windows950 |
| EUC\_CN | 扩展UNIX编码-中国 | 简体中文 | 是 | 是 | 1-3 | - |
| EUC\_JP | 扩展UNIX编码-日本 | 日文 | 是 | 是 | 1-3 | - |
| EUC\_JIS\_2004 | 扩展UNIX编码-日本, JIS X 0213 | 日文 | 是 | 否 | 1-3 | - |
| EUC\_KR | 扩展UNIX编码-韩国 | 韩文 | 是 | 是 | 1-3 | - |
| EUC\_TW | 扩展UNIX编码-台湾 | 繁体中文，台湾话 | 是 | 是 | 1-3 | - |
| GB18030 | 国家标准 | 中文 | 是 | 否 | 1-4 | - |
| GBK | 扩展国家标准 | 简体中文 | 是 | 否 | 1-2 | WIN936，Windows936 |
| ISO\_8859\_5 | ISO 8859-5, ECMA 113 | 拉丁语/西里尔语 | 是 | 是 | 1 | - |
| ISO\_8859\_6 | ISO 8859-6, ECMA 114 | 拉丁语/阿拉伯语 | 是 | 是 | 1 | - |
| ISO\_8859\_7 | ISO 8859-7, ECMA 118 | 拉丁语/希腊语 | 是 | 是 | 1 | - |
| ISO\_8859\_8 | ISO 8859-8, ECMA 121 | 拉丁语/希伯来语 | 是 | 是 | 1 | - |
| JOHAB | JOHAB | 韩语 | 否 | 否 | 1-3 | - |
| KOI8R | KOI8-R | 西里尔语（俄语） | 是 | 是 | 1 | KOI8 |
| KOI8U | KOI8-U | 西里尔语（乌克兰语） | 是 | 是 | 1 | - |
| LATIN1 | ISO 8859-1, ECMA 94 | 西欧 | 是 | 是 | 1 | ISO88591 |
| LATIN2 | ISO 8859-2, ECMA 94 | 中欧 | 是 | 是 | 1 | ISO88592 |
| LATIN3 | ISO 8859-3, ECMA 94 | 南欧 | 是 | 是 | 1 | ISO88593 |
| LATIN4 | ISO 8859-4, ECMA 94 | 北欧 | 是 | 是 | 1 | ISO88594 |
| LATIN5 | ISO 8859-9, ECMA 128 | 土耳其语 | 是 | 是 | 1 | ISO88599 |
| LATIN6 | ISO 8859-10, ECMA 144 | 日耳曼语 | 是 | 是 | 1 | ISO885910 |
| LATIN7 | ISO 8859-13 | 波罗的海 | 是 | 是 | 1 | ISO885913 |
| LATIN8 | ISO 8859-14 | 凯尔特语 | 是 | 是 | 1 | ISO885914 |
| LATIN9 | ISO 8859-15 | 带欧罗巴和口音的LATIN1 | 是 | 是 | 1 | ISO885915 |
| LATIN10 | ISO 8859-16, ASRO SR 14111 | 罗马尼亚语 | 是 | 否 | 1 | ISO885916 |
| MULE\_INTERNAL | Mule内部编码 | 多语种编辑器 | 是 | 否 | 1-4 | - |
| SJIS | Shift JIS | 日语 | 否 | 否 | 1-2 | Mskanji，ShiftJIS，WIN932，Windows932 |
| SHIFT\_JIS\_2004 | Shift JIS, JIS X 0213 | 日语 | 否 | 否 | 1-2 | - |
| SQL\_ASCII | 未指定（见文本） | **任意** | 是 | 否 | 1 | - |
| UHC | 统一韩语编码 | 韩语 | 否 | 否 | 1-2 | WIN949，Windows949 |
| UTF8 | Unicode, 8-bit | **所有** | 是 | 是 | 1-4 | Unicode |
| WIN866 | Windows CP866 | 西里尔语 | 是 | 是 | 1 | ALT |
| WIN874 | Windows CP874 | 泰语 | 是 | 否 | 1 | - |
| WIN1250 | Windows CP1250 | 中欧 | 是 | 是 | 1 | - |
| WIN1251 | Windows CP1251 | 西里尔语 | 是 | 是 | 1 | WIN |
| WIN1252 | Windows CP1252 | 西欧 | 是 | 是 | 1 | - |
| WIN1253 | Windows CP1253 | 希腊语 | 是 | 是 | 1 | - |
| WIN1254 | Windows CP1254 | 土耳其语 | 是 | 是 | 1 | - |
| WIN1255 | Windows CP1255 | 希伯来语 | 是 | 是 | 1 | - |
| WIN1256 | Windows CP1256 | 阿拉伯语 | 是 | 是 | 1 | - |
| WIN1257 | Windows CP1257 | 波罗的海 | 是 | 是 | 1 | - |
| WIN1258 | Windows CP1258 | 越南语 | 是 | 是 | 1 | ABC, TCVN，TCVN5712，VSCII |

fig: **注意：**   
需要注意并非所有的客户端API都支持上面列出的字符集。  
SQL\_ASCII设置与其他设置表现得相当不同。如果服务器字符集是SQL\_ASCII，服务器把字节值0-127根据 ASCII标准解释，而字节值128-255则当作无法解析的字符。如果设置为SQL\_ASCII，就不会有编码转换。因此，这个设置基本不是用来声明所使用的指定编码， 因为这个声明会忽略编码。在大多数情况下，如果你使用了任何非ASCII数据，那么使用 SQL\_ASCII设置都是不明智的，因为Vastbase将无法帮助你转换或者校验非ASCII字符。

* 指定新的数据库字符集编码必须与所选择的本地环境中（LC\_COLLATE和LC\_CTYPE）的设置兼容。
* 当指定的字符编码集为GBK时，部分中文生僻字无法直接作为对象名。这是因为GBK第二个字节的编码范围在0x40-0x7E之间时，字节编码与ASCII字符@A-Z[]^*`a-z{|}重叠。其中@[]^*'{|}是数据库中的操作符，直接作为对象名时，会语法报错。例如“侤”字，GBK16进制编码为0x8240，第二个字节为0x40，与ASCII“@”符号编码相同，因此无法直接作为对象名使用。如果确实要使用，可以在创建和访问对象时，通过增加双引号来规避这个问题。
* 若客户端编码为A，服务器端编码为B，则需要满足数据库中存在编码格式A与B的转换，例如：若服务器端编码为gb18030，由于当前数据库不支持gb18030与gbk的相互转换，所以此时设置客户端编码格式为gbk时，会报错“Conversion between GB18030 and GBK is not supported.”。数据库能够支持的所有的编码格式转换详见系统表pg\_conversion。
* **LC\_COLLATE [ = ] lc\_collate**
* 指定新数据库使用的字符集。例如，通过lc\_collate = 'zh\_CN.gbk'设定该参数。
* 该参数的使用会影响到对字符串的排序顺序（如使用ORDER BY执行，以及在文本列上使用索引的顺序）。默认是使用模板数据库的排序顺序。
* 取值范围：有效的排序类型。
* **LC\_CTYPE [ = ] lc\_ctype**
* 指定新数据库使用的字符分类。例如，通过lc\_ctype = 'zh\_CN.gbk'设定该参数。该参数的使用会影响到字符的分类，如大写、小写和数字。默认是使用模板数据库的字符分类。
* 取值范围：有效的字符分类。
* **DBCOMPATIBILITY [ = ] compatibility\_type**
* 指定兼容的数据库的类型，默认兼容O。
* 取值范围：A、B、C、PG。分别表示兼容O、MY、TD和POSTGRES。
* fig: **说明：**
* A兼容性下，数据库将空字符串作为NULL处理，数据类型DATE会被替换为TIMESTAMP(0) WITHOUT TIME ZONE。
* 将字符串转换成整数类型时，如果输入不合法，B兼容性会将输入转换为0，而其它兼容性则会报错。
* PG兼容性下，CHAR和VARCHAR以字符为计数单位，其它兼容性以字节为计数单位。例如，对于UTF-8字符集，CHAR(3)在PG兼容性下能存放3个中文字符，而在其它兼容性下只能存放1个中文字符。
* **TABLESPACE [ = ] tablespace\_name**
* 指定数据库对应的表空间。
* 取值范围：已存在表空间名。
* **CONNECTION LIMIT [ = ] connlimit**
* 数据库可以接受的并发连接数。
* fig: **须知：**
* 系统管理员不受此参数的限制。
* connlimit数据库主节点单独统计，Vastbase整体的连接数 = connlimit \* 当前正常数据库主节点个数。

取值范围：>=-1的整数。默认值为-1，表示没有限制。

有关字符编码的一些限制：

* 若区域设置为C（或POSIX），则允许所有的编码类型，但是对于其他的区域设置，字符编码必须和区域设置相同。
* 若字符编码方式是SQL\_ASCII，并且修改者为管理员用户时，则字符编码可以和区域设置不相同。
* 编码和区域设置必须匹配模板数据库，除了将template0当作模板。 因为其他数据库可能会包含不匹配指定编码的数据，或者可能包含排序顺序受LC\_COLLATE和LC\_CTYPE影响的索引。复制这些数据会导致在新数据库中的索引失效。template0是不包含任何会受到影响的数据或者索引。

**示例**

--创建jim和tom用户。  
vastbase=# CREATE USER jim PASSWORD 'Aa@12345';  
vastbase=# CREATE USER tom PASSWORD 'Aa@12345';  
  
--创建一个"UTF-8"编码的数据库music  
vastbase=# CREATE DATABASE music ENCODING 'UTF-8' template = template0;  
  
--创建数据库music2，并指定所有者为jim。  
vastbase=# CREATE DATABASE music2 OWNER jim;  
  
--用模板template0创建数据库music3，并指定所有者为jim。  
vastbase=# CREATE DATABASE music3 OWNER jim TEMPLATE template0;  
  
--设置music数据库的连接数为10。  
vastbase=# ALTER DATABASE music CONNECTION LIMIT= 10;  
  
--将music名称改为music4。  
vastbase=# ALTER DATABASE music RENAME TO music4;  
  
--将数据库music2的所属者改为tom。  
vastbase=# ALTER DATABASE music2 OWNER TO tom;  
  
--设置music3的表空间为PG\_DEFAULT。  
vastbase=# ALTER DATABASE music3 SET TABLESPACE PG\_DEFAULT;  
  
--关闭在数据库music3上缺省的索引扫描。  
vastbase=# ALTER DATABASE music3 SET enable\_indexscan TO off;  
  
--重置enable\_indexscan参数。  
vastbase=# ALTER DATABASE music3 RESET enable\_indexscan;  
  
--删除数据库。  
vastbase=# DROP DATABASE music2;  
vastbase=# DROP DATABASE music3;  
vastbase=# DROP DATABASE music4;  
  
--删除jim和tom用户。  
vastbase=# DROP USER jim;  
vastbase=# DROP USER tom;  
  
--创建兼容TD(Teradata)格式的数据库。  
vastbase=# CREATE DATABASE td\_compatible\_db DBCOMPATIBILITY 'C';  
  
--创建兼容A(Oracle)格式的数据库。  
vastbase=# CREATE DATABASE ora\_compatible\_db DBCOMPATIBILITY 'A';  
  
--删除兼容TD、A格式的数据库。  
vastbase=# DROP DATABASE td\_compatible\_db;  
vastbase=# DROP DATABASE ora\_compatible\_db;

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER DATABASE、DROP ALTER DATABASE章节。

### CREATE DATABASE LINK

**功能描述**

定义一个新的数据库链接。当用户需要跨越本地数据库，访问远程数据库的数据时，可以通过DBLINK像访问本地数据库一样访问远程数据库表中的数据。

**注意事项**

* 编码和区域设置必须匹配模板数据库，除了将template0当作模板。 因为其他数据库可能会包含不匹配指定编码的数据，或者可能包含排序顺序受LC\_COLLATE和LC\_CTYPE影响的索引。复制这些数据会导致在新数据库中的索引失效。template0是不包含任何会受到影响的数据或者索引。
* 不同的用户可以创建同名的private dblink，且用户只能访问自己创建的DBLINK。
* 不同的用户可以创建同名的private dblink，且用户只能访问自己创建的DBLINK。
* 同一用户可以创建同名的private dblink和，在访问时，优先访问private dblink。
* 为支持不同用户创建同名dblink，需禁用GRANT FOREIGN SERVER，不再允许将server权限给予其他用户，以防一个用户能访问多个同名server，造成混乱。
* 创建private dblink或public dblink时，必须提前创建jdbc\_fdw，并具备DATABASE LINK相关权限。

**语法格式**

CREATE [PUBLIC] DATABASE LINK dblink\_name

CONNECT TO username IDENTIFIED BY 'password' [USING fdw\_name] (

jdbc\_fdw\_option

);

其中jdbc\_fdw\_option 可以包括：

url 'value',

jarfile 'value',

[ querytimeout 'value'] [, ...]

**参数说明**

* **dblink\_name**

连接名称，可自定义。

* **fdw\_name**

指定fdw名，fdw是foreign data wrapper的简称，为外部封装数据，由用户自定义创建。

fdw\_name为fdw的名称，如示例中jdbc\_fdw等。

* **url 'value'**

指定连接库的URL地址。

value值为连接库的url地址。

* **jarfile 'value'**

指定驱动包的获取路径。

value值为驱动包所在的路径。

* **querytimeout 'value'**

设置查询超时时长。

value值为超时时长。

**示例**

1、在Oracle数据库创建表。

create table emp\_fdw(empno int primary key,ename varchar(30));

insert into emp\_fdw values(1,'foo');

insert into emp\_fdw values(2,'bar');

2、在Vastbase数据库需要使用jdbc\_fdw的database下创建插件。

create extension jdbc\_fdw;

3、修改vastbase配置参数。修改postgresql.conf参数，将jdbc\_fdw配置到shared\_preload\_libraries参数中。修改后需要重启数据库实例。

shared\_preload\_libraries = 'jdbc\_fdw'

4、创建dblink。

CREATE PUBLIC DATABASE LINK dblink\_104 CONNECT TO lst IDENTIFIED BY 'test' USING jdbc\_fdw(

url 'jdbc:oracle:thin:@//172.16.103.104:1521/orcl',

jarfile '/home/vastbase\_lst/ojdbc7.jar'

);

3、创建用户。

CREATE USER lst PASSWORD 'Bigdata@123';

4、为用户授权。

grant usage on foreign data wrapper jdbc\_fdw to lst;

grant all on database vastbase to lst;

5、切换到用户lst，查询数据。

\c - lst

SELECT \* from emp\_fdw@dblink\_104;

**相关链接**

无。

### CREATE DIRECTORY

**功能描述**

使用CREATE DIRECTORY语句创建一个目录对象，该目录对象定义了服务器文件系统上目录的别名，用于存放用户使用的数据文件。

**注意事项**

* 当enable\_access\_server\_directory=off时，只允许初始用户创建directory对象；当enable\_access\_server\_directory=on时，具有SYSADMIN权限的用户和继承了内置角色gs\_role\_directory\_create权限的用户可以创建directory对象。
* 创建用户默认拥有此路径的READ和WRITE操作权限。
* 目录的默认owner为创建directory的用户。
* 以下路径禁止创建：
* 路径含特殊字符。
* 路径是相对路径。
* 路径是符号连接。
* 创建目录时会进行以下合法性校验：
* 创建时会检查添加路径是否为操作系统实际存在路径，如不存在会提示用户使用风险。
* 创建时会校验数据库初始化（vastbase）用户对于添加路径的权限(即操作系统目录权限，读/写/执行 - R/W/X)，如果权限不全，会提示用户使用风险。
* 在Vastbase环境下用户指定的路径需要用户保证各节点上路径的一致性，否则在不同节点上执行会产生找不到路径的问题。

**语法格式**

CREATE [OR REPLACE] DIRECTORY directory\_name  
AS 'path\_name';

**参数说明**

* **directory\_name**
* 目录名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **path\_name**
* 操作系统的路径。
* 取值范围： 有效的操作系统路径。

**示例**

--创建目录。  
vastbase=# CREATE OR REPLACE DIRECTORY dir as '/tmp/';

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER DIRECTORY、DROP DIRECTORY章节。

### CREATE EXTENSION

**功能描述**

安装一个扩展。

**注意事项**

* CREATE EXTENSION命令安装一个新的扩展到一个数据库中，必须保证没有同名的扩展已经被安装。
* 安装一个扩展意味着执行一个扩展的脚本文件，这个脚本会创建一个新的SQL实体，例如函数、数据类型、操作符、和索引支持的方法。
* 安装扩展需要有和创建他的组件对象相同的权限。对于大多数扩展这意味着需要超户或者数据库所有者的权限，对于后续的权限检查和该扩展脚本所创建的实体，运行CREATE EXTENSION命令的角色将变为扩展的所有者。

**语法格式**

CREATE EXTENSION [ IF NOT EXISTS ] extension\_name  
[ WITH ] [ SCHEMA schema\_name ]  
[ VERSION version ]  
[ FROM old\_version ]

**参数说明**

* **IF NOT EXISTS**
* 如果系统已经存在一个同名的扩展，不会报错。这种情况下会给出一个提示。请注意该参数不保证系统存在的扩展和现在脚本创建的扩展相同。
* **extension\_name**
* 将被安装扩展的名字。
* **schema\_name**
* 扩展的实例被安装在该模式下，扩展的内容可以被重新安装。指定的模式必须已经存在，如果没有指定，扩展的控制文件也不指定一个模式，这样将使用默认模式。
* fig: **注意：**   
  扩展不认为它在任何模式里面：扩展在一个数据库范围内的名字是不受限制的，但是一个扩展的实例是属于一个模式的。
* **version**
* 安装扩展的版本，可以写为一个标识符或者字符串.默认的版本在扩展的控制文件中指定。
* **old\_version**
* 当你想升级安装“old style” 模块中没有的内容时,你必须指定FROM old\_version。这个选项使CREATE EXTENSION 运行一个安装脚本将新的内容安装到扩展中，而不是创建一个新的实体.注意SCHEMA指定了包括这些已存在实体的模式。

**示例**

在当前数据库安装hstore扩展：

CREATE EXTENSION hstore;

### CREATE FOREIGN TABLE

**功能描述**

创建外表。

**注意事项**

外表中暂不支持使用系统列（如tableoid、ctid等），其中Private和Shares模式的外表，需要初始用户和运维模式下（operation\_mode）的运维管理员权限。

**语法格式**

CREATE FOREIGN TABLE [ IF NOT EXISTS ] table\_name ( [  
 column\_name type\_name [ OPTIONS ( option 'value' [, ... ] ) ] [ COLLATE collation ] [ column\_constraint [ ... ] ]  
 [, ... ]  
] )  
 SERVER server\_name  
[ OPTIONS ( option 'value' [, ... ] ) ]  
  
这里column\_constraint 可以是:  
[ CONSTRAINT constraint\_name ]  
{ NOT NULL |  
 NULL |  
 DEFAULT default\_expr }

**参数说明**

* **IF NOT EXISTS**
* 如果已经存在相同名称的表，不会抛出一个错误，而会发出一个通知，告知表关系已存在。
* **table\_name**
* 外表的表名。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 外表中的字段名。
* 取值范围：字符串，要符合标识符的命名规范。
* **type\_name**
* 字段的数据类型。
* **SERVER server\_name**
* 外表的server名称。默认值为mot\_server。
* **OPTIONS ( option 'value' [, ... ] )**
* 选项与新外部表或外部表中的字段有关。允许的选项名称和值，是由每一个外部数据封装器指定的。 也是通过外部数据封装器的验证函数来验证。重复的选项名称是不被允许的（尽管表选项和表字段选项可以有相同的名字）。
* oracle\_fdw支持的options包括：
* **table**

oracle server侧的表名。需要同oracle系统表中记录的表名完全一致，通常是由大写字符组成。

* **schema**

表所对应的schema（或owner）。需要同oracle系统表中记录的表名完全一致，通常是由大写字符组成。

* mysql\_fdw支持的options包括：
* **dbname**

MySQL的database名称。

* **table\_name**

MySQL侧的表名。

* postgres\_fdw支持的options包括：
* **schema\_name**

远端server的schema名称。如果不指定的话，将使用外表自身的schema名称作为远端的schema名称。

* **table\_name**

远端server的表名。如果不指定的话，将使用外表自身的表名作为远端的表名。

* **column\_name**

远端server的表的列名。如果不指定的话，将使用外表自身的列名作为远端的的表的列名。

* file\_fdw支持的options包括：
* filename

指定要读取的文件，必需的参数，且必须是一个绝对路径名。

* format

远端server的文件格式，支持text/csv/binary/fixed四种格式，和COPY语句的FORMAT选项相同。

* header

指定的文件是否有标题行，与COPY语句的HEADER选项相同。

- delimiter

指定文件的分隔符，与COPY的DELIMITER选项相同。

- quote

指定文件的引用字符，与COPY的QUOTE选项相同。

- escape

指定文件的转义字符，与COPY的ESCAPE选项相同。

- null

指定文件的null字符串，与COPY的NULL选项相同。

- encoding

指定文件的编码，与COPY的ENCODING选项相同。

- force\_not\_null

这是一个布尔选项。如果为真，则声明字段的值不应该匹 配空字符串（也就是，文件级别null选项）。与COPY的 FORCE\_NOT\_NULL选项里的字段相同。

>![](public\_sys-resources/icon-note.gif) \*\*说明：\*\*   
 >file\_fdw更多使用请参见[file\_fdw](file\_fdw.md)。

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER FOREIGN TABLE、DROP FOREIGN TABLE章节。

### CREATE FUNCTION

**功能描述**

创建一个函数。

**注意事项**

* 如果创建函数时参数或返回值带有精度，不进行精度检测。
* 创建函数时，函数定义中对表对象的操作建议都显式指定模式，否则可能会导致函数执行异常。
* 在创建函数时，函数内部通过SET语句设置current\_schema和search\_path无效。执行完函数search\_path和current\_schema与执行函数前的search\_path和current\_schema保持一致。
* 如果函数参数中带有出参，SELECT调用函数必须缺省出参，CALL调用函数必须指定出参，对于调用重载的带有PACKAGE属性的函数，CALL调用函数可以缺省出参，具体信息参见：SQL语法参考->SQL语法中CALL的示例。
* 兼容Postgresql风格的函数或者带有PACKAGE属性的函数支持重载。在指定REPLACE的时候，如果参数个数、类型、返回值有变化，不会替换原有函数，而是会建立新的函数。
* SELECT调用可以指定不同参数来进行同名函数调用。由于语法不支持调用不带有PACKAGE属性的同名函数。
* 在创建function时，不能在avg函数外面嵌套其他agg函数或者其他系统函数。
* 新创建的函数默认会给PUBLIC授予执行权限（详见SQL语法参考->SQL语法->GRANT章节）。用户可以选择收回PUBLIC默认执行权限，然后根据需要将执行权限授予其他用户，为了避免出现新函数能被所有人访问的时间窗口，应在一个事务中创建函数并且设置函数执行权限。
* 在函数内部调用其它无参数的函数时，可以省略括号，直接使用函数名进行调用。
* 兼容Oracle风格的函数支持参数注释的查看与导出、导入。
* 兼容Oracle风格的函数支持介于IS/AS与plsql\_body之间的注释的查看与导出、导入。

**语法格式**

* 兼容PostgreSQL风格的创建自定义函数语法。

CREATE [ OR REPLACE ] FUNCTION function\_name  
 ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ]  
 } [, ...] ] )  
 [ RETURNS rettype [ DETERMINISTIC ]  
 | RETURNS TABLE ( { column\_name column\_type } [, ...] )]  
 LANGUAGE lang\_name  
 [  
 {IMMUTABLE | STABLE | VOLATILE}  
 | {SHIPPABLE | NOT SHIPPABLE}  
 | [ NOT ] LEAKPROOF  
 | WINDOW  
 | {CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT}  
 | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AU  
 THID DEFINER | AUTHID CURRENT\_USER}  
 | {FENCED | NOT FENCED}  
 | {PACKAGE}  
 | COST execution\_cost  
 | ROWS result\_rows  
 | SET configuration\_parameter { {TO | =} value | FROM CURRENT }  
 ] [...]  
 {  
 AS 'definition'  
 | AS 'obj\_file', 'link\_symbol'  
 }

* 兼容Oracle风格的创建自定义函数的语法。

CREATE [ OR REPLACE ] FUNCTION function\_name  
 ( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ] }  
 [, ...] ] )  
 RETURN rettype [ DETERMINISTIC ]  
 [  
 {IMMUTABLE | STABLE | VOLATILE }  
 | {SHIPPABLE | NOT SHIPPABLE}  
 | {PACKAGE}  
 | [ NOT ] LEAKPROOF  
 | {CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }  
 | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | |  
 AUTHID DEFINER | AUTHID CURRENT\_USER}  
 | COST execution\_cost  
 | ROWS result\_rows  
 | SET configuration\_parameter { {TO | =} value | FROM CURRENT }  
 ][...]  
 {  
 IS | AS  
 } plsql\_body  
 /

**参数说明**

* **function\_name**
* 要创建的函数名称（可以用模式修饰）。
* 取值范围：字符串，要符合标识符的命名规范。且最多为63个字符。若超过63个字符，数据库会截断并保留前63个字符当做函数名称。
* **argname**
* 函数参数的名称。
* 取值范围：字符串，要符合标识符的命名规范。且最多为63个字符。若超过63个字符，数据库会截断并保留前63个字符当做函数参数名称。
* **argmode**
* 函数参数的模式。
* 取值范围：IN、OUT、INOUT或VARIADIC。缺省值是IN。并且OUT和INOUT模式的参数不能用在RETURNS TABLE的函数定义中。
* fig: **说明：**   
  VARIADIC用于声明数组类型的参数。
* **argtype**
* 函数参数的类型。可以使用%TYPE或%ROWTYPE间接引用变量或表的类型，详细可参考：PL/SQL->存储过程->基本语句->定义变量章节。
* **expression**
* 参数的默认表达式。
* **rettype**
* 函数返回值的数据类型。
* 如果存在OUT或INOUT参数，可以省略RETURNS子句。如果存在，该子句必须和输出参数所表示的结果类型一致：如果有多个输出参数，则为RECORD，否则与单个输出参数的类型相同。
* SETOF修饰词表示该函数将返回一个集合，而不是单独一项。
* 与argtype相同，同样可以使用%TYPE或%ROWTYPE间接引用类型。
* **column\_name**
* 字段名称。
* **column\_type**
* 字段类型。
* **definition**
* 一个定义函数的字符串常量，含义取决于语言。它可以是一个内部函数名称、一个指向某个目标文件的路径、一个SQL查询、一个过程语言文本。
* **DETERMINISTIC**
* SQL语法兼容接口，未实现功能，不推荐使用。
* **LANGUAGE lang\_name**
* 用以实现函数的语言的名称。可以是SQL、internal或者是用户定义的过程语言名称。为了保证向下兼容，该名称可以用单引号包围。若采用单引号，则引号内必须为大写。
* **WINDOW**
* 表示该函数是窗口函数。替换函数定义时不能改变WINDOW属性。
* fig: **须知：**   
  自定义窗口函数只支持LANGUAGE是internal，并且引用的内部函数必须是窗口函数。
* **IMMUTABLE**
* 表示该函数在给出同样的参数值时总是返回同样的结果。
* **STABLE**
* 表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。
* **VOLATILE**
* 表示该函数值可以在一次表扫描内改变，因此不会做任何优化。
* **SHIPPABLE**|**NOT SHIPPABLE**
* 表示该函数是否可以下推执行。预留接口，不推荐使用。
* **FENCED**|**NOT FENCED**
* 声明用户定义的C函数是在保护模式还是非保护模式下执行。预留接口，不推荐使用。
* **PACKAGE**
* 表示该函数是否支持重载。PostgreSQL风格的函数本身就支持重载，此参数主要是针对其它风格的函数。
* 不允许package函数和非package函数重载或者替换。
* package函数不支持VARIADIC类型的参数。
* 不允许修改函数的package属性。
* **LEAKPROOF**
* 指出该函数的参数只包括返回值。LEAKPROOF只能由系统管理员设置。
* **CALLED ON NULL INPUT**
* 表明该函数的某些参数是NULL的时候可以按照正常的方式调用。该参数可以省略。
* **RETURNS NULL ON NULL INPUT**
* **STRICT**
* STRICT用于指定如果函数的某个参数是NULL，此函数总是返回NULL。如果声明了这个参数，当有NULL值参数时该函数不会被执行；而只是自动返回一个NULL结果。
* RETURNS NULL ON NULL INPUT和STRICT的功能相同。
* **EXTERNAL**
* 目的是和SQL兼容，是可选的，这个特性适合于所有函数，而不仅是外部函数。
* **SECURITY INVOKER**
* **AUTHID CURRENT\_USER**
* 表明该函数将带着调用它的用户的权限执行。该参数可以省略。
* SECURITY INVOKER和AUTHID CURRENT\_USER的功能相同。
* **SECURITY DEFINER**
* **AUTHID DEFINER**
* 声明该函数将以创建它的用户的权限执行。
* AUTHID DEFINER和SECURITY DEFINER的功能相同。
* **COST execution\_cost**
* 用来估计函数的执行成本。
* execution\_cost以cpu\_operator\_cost为单位。
* 取值范围：正数
* **ROWS result\_rows**
* 估计函数返回的行数。用于函数返回的是一个集合。
* 取值范围：正数，默认值是1000行。
* **configuration\_parameter**
* **value**
  + 把指定的数据库会话参数值设置为给定的值。如果value是DEFAULT或者RESET，则在新的会话中使用系统的缺省设置。OFF关闭设置。
  + 取值范围：字符串
* DEFAULT
* OFF
* RESET
  + 指定默认值。
* **from current**
  + 取当前会话中的值设置为configuration\_parameter的值。
* **plsql\_body**
* PL/SQL存储过程体。
* fig: **须知：**   
  当在函数体中创建用户时，日志中会记录密码的明文。因此不建议用户在函数体中创建用户。

**示例**

--定义函数为SQL查询。  
vastbase=# CREATE FUNCTION func\_add\_sql(integer, integer) RETURNS integer  
 AS 'select $1 + $2;'  
 LANGUAGE SQL  
 IMMUTABLE  
 RETURNS NULL ON NULL INPUT;  
  
--利用参数名用 PL/pgSQL 自增一个整数。  
vastbase=# CREATE OR REPLACE FUNCTION func\_increment\_plsql(i integer) RETURNS integer AS $$  
 BEGIN  
 RETURN i + 1;  
 END;  
$$ LANGUAGE plpgsql;  
  
--返回RECORD类型  
vastbase=# CREATE OR REPLACE FUNCTION func\_increment\_sql(i int, out result\_1 bigint, out result\_2 bigint)  
returns SETOF RECORD  
as $$  
begin  
 result\_1 = i + 1;  
 result\_2 = i \* 10;  
return next;  
end;  
$$language plpgsql;  
  
--返回一个包含多个输出参数的记录。  
vastbase=# CREATE FUNCTION func\_dup\_sql(in int, out f1 int, out f2 text)  
 AS $$ SELECT $1, CAST($1 AS text) || ' is text' $$  
 LANGUAGE SQL;  
  
vastbase=# SELECT \* FROM func\_dup\_sql(42);  
  
--计算两个整数的和，并返回结果。如果输入为null，则返回null。  
vastbase=# CREATE FUNCTION func\_add\_sql2(num1 integer, num2 integer) RETURN integer  
AS  
BEGIN   
RETURN num1 + num2;  
END;  
/  
--修改函数func\_add\_sql2的执行规则为IMMUTABLE，即参数不变时返回相同结果。  
vastbase=# ALTER FUNCTION func\_add\_sql2(INTEGER, INTEGER) IMMUTABLE;  
  
--将函数func\_add\_sql2的名称修改为add\_two\_number。  
vastbase=# ALTER FUNCTION func\_add\_sql2(INTEGER, INTEGER) RENAME TO add\_two\_number;  
  
--将函数add\_two\_number的属者改为omm。  
vastbase=# ALTER FUNCTION add\_two\_number(INTEGER, INTEGER) OWNER TO omm;  
  
--删除函数。  
vastbase=# DROP FUNCTION add\_two\_number;  
vastbase=# DROP FUNCTION func\_increment\_sql;  
vastbase=# DROP FUNCTION func\_dup\_sql;  
vastbase=# DROP FUNCTION func\_increment\_plsql;  
vastbase=# DROP FUNCTION func\_add\_sql;

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER FUNCTION、DROP FUNCTION章节。

### CREATE GROUP

**功能描述**

创建一个新用户组。

**注意事项**

CREATE GROUP是CREATE ROLE的别名，非SQL标准语法，不推荐使用，建议用户直接使用CREATE ROLE替代。

**语法格式**

CREATE GROUP group\_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' [ EXPIRED ] | DISABLE };

其中可选项option子句语法为：

{SYSADMIN | NOSYSADMIN}  
 | {MONADMIN | NOMONADMIN}  
 | {OPRADMIN | NOOPRADMIN}  
 | {POLADMIN | NOPOLADMIN}  
 | {AUDITADMIN | NOAUDITADMIN}  
 | {CREATEDB | NOCREATEDB}  
 | {USEFT | NOUSEFT}  
 | {CREATEROLE | NOCREATEROLE}  
 | {INHERIT | NOINHERIT}  
 | {LOGIN | NOLOGIN}  
 | {REPLICATION | NOREPLICATION}  
 | {INDEPENDENT | NOINDEPENDENT}  
 | {VCADMIN | NOVCADMIN}  
 | {PERSISTENCE | NOPERSISTENCE}  
 | CONNECTION LIMIT connlimit  
 | VALID BEGIN 'timestamp'  
 | VALID UNTIL 'timestamp'  
 | RESOURCE POOL 'respool'  
 | PERM SPACE 'spacelimit'  
 | TEMP SPACE 'tmpspacelimit'  
 | SPILL SPACE 'spillspacelimit'  
 | IN ROLE role\_name [, ...]  
 | IN GROUP role\_name [, ...]  
 | ROLE role\_name [, ...]  
 | ADMIN rol e\_name [, ...]  
 | USER role\_name [, ...]  
 | SYSID uid  
 | DEFAULT TABLESPACE tablespace\_name  
 | PROFILE DEFAULT  
 | PROFILE profile\_name  
 | PGUSER

**参数说明**

请参考：SQL语法参考->SQL语法->CREATE ROLE章节中的参数说明。

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER GROUP、DROP GROUP、CREATE ROLE章节。

### CREATE INCREMENTAL MATERIALIZED VIEW

**功能描述**

CREATE INCREMENTAL MATERIALIZED VIEW会创建一个增量物化视图，并且后续可以使用REFRESH MATERIALIZED VIEW（全量刷新）和REFRESH INCREMENTAL MATERIALIZED VIEW（增量刷新）刷新物化视图的数据。

CREATE INCREMENTAL MATERIALIZED VIEW类似于CREATE TABLE AS，不过它会记住被用来初始化该视图的查询， 因此它可以在后续中进行数据刷新。一个物化视图有很多和表相同的属性，但是不支持临时物化视图。

**注意事项**

* 增量物化视图不可以在临时表或全局临时表上创建。
* 增量物化视图仅支持简单过滤查询和基表UNION ALL查询。
* 创建增量物化视图不可指定分布列。
* 创建增量物化视图后，基表中的绝大多数DDL操作不再支持。
* 不支持对增量物化视图进行IUD操作。
* 增量物化视图创建后，当基表数据发生变化时，需要使用刷新（REFRESH）命令保持物化视图与基表同步。

**语法格式**

CREATE INCREMENTAL MATERIALIZED VIEW mv\_name  
 [ (column\_name [, ...] ) ]  
 [ TABLESPACE tablespace\_name ]  
 AS query;

**参数说明**

* **mv\_name**
* 要创建的物化视图的名称（可以被模式限定）。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 新物化视图中的一个列名。物化视图支持指定列，指定列需要和后面的查询语句结果的列数量保持一致；如果没有提供列名，会从查询的输出列名中获取列名。
* 取值范围：字符串，要符合标识符的命名规范。
* **TABLESPACE tablespace\_name**
* 指定新建物化视图所属表空间。如果没有声明，将使用默认表空间。
* **AS query**
* 一个SELECT或者TABLE命令。这个查询将在一个安全受限的操作中运行。

**示例**

--创建一个普通表  
vastbase=# CREATE TABLE my\_table (c1 int, c2 int);  
--创建增量物化视图  
vastbase=# CREATE INCREMENTAL MATERIALIZED VIEW my\_imv AS SELECT \* FROM my\_table;  
--基表写入数据  
vastbase=# INSERT INTO my\_table VALUES(1,1),(2,2);  
--对增量物化视图my\_imv进行增量刷新  
vastbase=# REFRESH INCREMENTAL MATERIALIZED VIEW my\_imv;

**相关链接**

参考：SQL语法参考->SQL语法中的ALTER MATERIALIZED VIEW、CREATE MATERIALIZED VIEW、CREATE TABLE、DROP MATERIALIZED VIEW、REFRESH INCREMENTAL MATERIALIZED VIEW、REFRESH MATERIALIZED VIEW章节。

### CREATE INDEX

**功能描述**

在指定的表上创建索引。索引可以用来提高数据库查询性能，但是不恰当的使用将导致数据库性能下降。建议仅在匹配如下某条原则时创建索引：

* 经常执行查询的字段。
* 在连接条件上创建索引，对于存在多字段连接的查询，建议在这些字段上建立组合索引。例如，select \* from t1 join t2 on t1.a=t2.a and t1.b=t2.b，可以在t1表上的a、b字段上建立组合索引。
* where子句的过滤条件字段上（尤其是范围条件）。
* 经常出现在order by、group by和distinct后的字段。

在分区表上创建索引与在普通表上创建索引的语法不太一样，使用时请注意，如分区表上不支持并行创建索引，不支持创建部分索引。

**注意事项**

* 索引自身也占用存储空间、消耗计算资源，创建过多的索引将对数据库性能造成负面影响（尤其影响数据导入的性能，建议在数据导入后再建索引）。因此，仅在必要时创建索引。
* 索引定义里的所有函数和操作符都必须是immutable类型的，即它们的结果必须只能依赖于它们的输入参数，而不受任何外部的影响（如另外一个表的内容或者当前时间）。这个限制可以确保该索引的行为是定义良好的。要在一个索引上或WHERE中使用用户定义函数，请把它标记为immutable类型函数。
* 分区表索引分为LOCAL索引与GLOBAL索引，LOCAL索引与某个具体分区绑定，而GLOBAL索引则对应整个分区表。
* 列存表支持的PSORT和B-tree索引都不支持创建表达式索引、部分索引，PSORT不支持创建唯一索引，B-tree支持创建唯一索引。
* 列存表支持的GIN索引支持创建表达式索引，但表达式不能包含空分词、空列和多列，不支持创建部分索引和唯一索引。
* HASH索引目前仅限于行存表索引、临时表索引和分区表LOCAL索引，且不支持创建多字段索引。
* 被授予CREATE ANY INDEX权限的用户，可以在public模式和用户模式下创建索引。
* 建议仅在匹配如下条件之一时创建索引：
* 经常执行查询的字段。
* 在连接条件上创建索引，对于存在多字段连接的查询，建议在这些字段上建立组合索引。例如，select \* from t1 join t2 on t1.a=t2.a and t1.b=t2.b，可以在t1表上的a、b字段上建立组合索引。
* where子句的过滤条件字段上（尤其是范围条件）。
* 在经常出现在order by、group by和distinct后的字段。
* 约束限制：
* 分区表上不支持创建部分索引。
* 分区表创建GLOBAL索引时，存在以下约束条件：
* 不支持表达式索引、部分索引
* 不支持列存表
* 仅支持B-tree索引
* 在相同属性列上，分区LOCAL索引与GLOBAL索引不能共存。
* GLOBAL索引，最大支持31列。
* 如果alter语句不带有UPDATE GLOBAL INDEX，那么原有的GLOBAL索引将失效，查询时将使用其他索引进行查询；如果alter语句带有UPDATE GLOBAL INDEX，原有的GLOBAL索引仍然有效，并且索引功能正确。

**语法格式**

* 在表上创建索引。

CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [IF NOT EXISTS] [ [schema\_name.]index\_name ] ON table\_name [ USING method ]  
 ({ { column\_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS { FIRST | LAST } ] }[, ...] )  
 [ INCLUDE ( column\_name [, ...] )]   
 [ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ TABLESPACE tablespace\_name ]  
 [ WHERE predicate ];

* 在分区表上在线创建全局索引和在线创建一级分区索引。

CREATE [ UNIQUE ] INDEX [ CONCURRENTLY] [ [schema\_name.]index\_name ] ON table\_name [ USING method ]  
 ( {{ column\_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS LAST ] }[, ...] )  
 [ LOCAL [ ( { PARTITION index\_partition\_name [ TABLESPACE index\_partition\_tablespace ] } [, ...] ) ] | GLOBAL ]  
 [ WITH ( { storage\_parameter = value } [, ...] ) ]  
 [ TABLESPACE tablespace\_name ];

**参数说明**

* UNIQUE：创建唯一性索引，每次添加数据时检测表中是否有重复值。如果插入或更新的值会引起重复的记录时，将导致一个错误。目前只有B-tree及UBTree索引支持唯一索引。
* CONCURRENTLY：以不阻塞DML的方式创建索引（加ShareUpdateExclusiveLock锁）。创建索引时，一般会阻塞其他语句对该索引所依赖表的访问。指定此关键字，可以实现创建过程中不阻塞DML。
* 此选项只能指定一个索引的名称。
* 普通CREATE INDEX命令可以在事务内执行，但是CREATE INDEX CONCURRENTLY不可以在事务内执行。
* 列存表和临时表不支持CONCURRENTLY方式创建索引。
* 支持在行存一级分区表上在线创建全局索引和本地分区索引。
* 创建索引时指定此关键字，需要执行先后两次对该表的全表扫描来完成build，第一次扫描的时候创建索引，不阻塞读写操作；第二次扫描的时候合并更新第一次扫描到目前为止发生的变更。
* 由于需要执行两次对表的扫描和build，而且必须等待现有的所有可能对该表执行修改的事务结束。这意味着该索引的创建比正常耗时更长，同时因此带来的CPU和I/O消耗对其他业务也会造成影响。
* 如果在索引构建时发生失败，那会留下一个“不可用”的索引。这个索引会被查询忽略，但它仍消耗更新开销。这种情况推荐的恢复方法是删除该索引并尝试再次CONCURRENTLY建索引。
* 由于在第二次扫描之后，索引构建必须等待任何持有早于第二次扫描拿的快照的事务终止，而且建索引时加的ShareUpdateExclusiveLock锁（4级）会和大于等于4级的锁冲突，在创建这类索引时，容易引发卡住（hang）或者死锁问题。例如：
* 两个会话对同一个表创建CONCURRENTLY索引，会引起死锁问题；
* 两个会话，一个对表创建CONCURRENTLY索引，一个drop table，会引起死锁问题；
* 三个会话，会话1先对表a加锁，不提交，会话2接着对表b创建CONCURRENTLY索引，会话3接着对表a执行写入操作，在会话1事务未提交之前，会话2会一直被阻塞；
* 将事务隔离级别设置成可重复读（默认为读已提交），起两个会话，会话1起事务对表a执行写入操作，不提交，会话2对表b创建CONCURRENTLY索引，在会话1事务未提交之前，会话2会一直被阻塞。
* IF NOT EXISTS：用于判断是否存在同名索引。
* schema\_name：模式的名称。
* 取值范围：已存在模式名。
* index\_name：要创建的索引名，索引的模式与表相同。
* 取值范围：字符串，要符合标识符的命名规范。
* table\_name：需要为其创建索引的表的名称，可以用模式修饰。
* 取值范围：已存在的表名。
* USING method：指定创建索引的方法。
* 取值范围：
* btree：B-tree索引使用一种类似于B+树的结构来存储数据的键值，通过这种结构能够快速的查找索引。btree适合支持比较查询以及查询范围。
* hash：Hash索引使用Hash函数对索引的关键字进行散列。只能处理简单等值比较，比较适合在索引值较长的情况下使用。
* gin：GIN索引是倒排索引，可以处理包含多个键的值（比如数组）。
* gist：Gist索引适用于几何和地理等多维数据类型和集合数据类型。目前支持的数据类型有box、point、poly、circle、tsvector、tsquery、range。
* Psort：Psort索引是针对列存表进行局部排序的索引。
* ubtree：仅供ustore表使用的多版本B-tree索引，索引页面上包含事务信息，能并自主回收页面。
* 列存表对GIN索引支持仅限于对于tsvector类型的支持，即创建列存GIN索引入参需要为to\_tsvector函数（的返回值)。此方法为GIN索引比较普遍的使用方式。
* 行存表（ASTORE存储引擎）支持的索引类型：btree（行存表缺省值）、hash、gin、gist。行存表（USTORE存储引擎）支持的索引类型：ubtree。列存表支持的索引类型：Psort（列存表缺省值）、btree、gin。全局临时表不支持GIN索引和Gist索引。
* column\_name：表中需要创建索引的列的名称（字段名）。如果索引方式支持多字段索引，可以声明多个字段。全局索引最多可以声明31个字段，其他索引最多可以声明32个字段。
* expression：创建一个基于该表的一个或多个字段的表达式索引，通常必须写在圆括弧中。如果表达式有函数调用的形式，圆括弧可以省略。表达式索引可用于获取对基本数据的某种变形的快速访问。比如，一个在upper(col)上的函数索引将允许WHERE upper(col) = 'JIM'子句使用索引。在创建表达式索引时，如果表达式中包含IS NULL子句，则这种索引是无效的。此时，建议用户尝试创建一个部分索引。
* COLLATE collation：COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。如果没有指定，则使用默认的排序规则。排序规则可以使用“select \* from pg\_collation”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。
* opclass：操作符类的名称。对于索引的每一列可以指定一个操作符类，操作符类标识了索引那一列的使用的操作符。例如一个B-tree索引在一个四字节整数上可以使用int4\_ops；这个操作符类包括四字节整数的比较函数。实际上对于列上的数据类型默认的操作符类是足够用的。操作符类主要用于一些有多种排序的数据。例如，用户想按照绝对值或者实数部分排序一个复数。能通过定义两个操作符类然后当建立索引时选择合适的类。
* ASC：指定按升序排序（默认）。
* DESC：指定按降序排序。
* NULLS FIRST：指定空值在排序中排在非空值之前，当指定DESC排序时，本选项为默认的。
* NULLS LAST：指定空值在排序中排在非空值之后，未指定DESC排序时，本选项为默认的。
* LOCAL：指定创建的分区索引为LOCAL索引。
* GLOBAL：指定创建的分区索引为GLOBAL索引，当不指定LOCAL、GLOBAL关键字时，默认创建GLOBAL索引。
* INCLUDE ( column\_name [, …] )：可选的 INCLUDE 子句指定将一些非键列（non-key columns）包含在索引中。
* 非键列不能用于作为索引扫描的加速搜索条件，同时在检查索引的唯一性约束时会忽略它们。
* 仅索引扫描 (Index Only Scan) 可以直接返回非键列中的内容，而不必去访问索引所对应的堆表。
* 将非键列添加为 INCLUDE 列需要保守一些，尤其是对于宽列。如果索引元组超过索引类型允许的最大大小，数据将插入失败。需要注意的是，任何情况下为索引添加非键列都会增加索引的空间占用，从而可能减慢搜索速度。
* 目前只有ubtree索引访问方式支持该特性。非键列会被保存在与堆元组对应的索引叶子元组中，不会包含在索引上层页面的元组中。
* WITH ( {storage\_parameter = value} [, … ] )：指定索引方法的存储参数。取值范围：
* FILLFACTOR：一个索引的填充因子（fillfactor）是一个介于10和100之间的百分数。  
  取值范围：10~100
* FASTUPDATE：GIN索引是否使用快速更新。  
  取值范围：ON，OFF  
  默认值：ON
* GIN\_PENDING\_LIST\_LIMIT：当GIN索引启用fastupdate时，设置该索引pending list容量的最大值。  
  取值范围：64~INT\_MAX，单位KB。  
  默认值：gin\_pending\_list\_limit的默认取决于GUC中gin\_pending\_list\_limit的值（默认为4MB）
* INDEXSPLIT：UBTREE索引选择采取哪种分裂策略。其中DEFAULT策略指的是与BTREE相同的分裂策略。INSERTPT策略能在某些场景下显著降低索引空间占用。  
  取值范围：INSERTPT，DEAFAULT  
  默认值：INSERTPT

只有GIN索引支持FASTUPDATE、GIN\_PENDING\_LIST\_LIMIT参数。GIN和Psort之外的索引都支持FILLFACTOR参数。只有UBTREE索引支持INDEXSPLIT参数。

* TABLESPACE tablespace\_name：指定索引的表空间，如果没有声明则使用默认的表空间。
* 取值范围：已存在的表空间名。
* WHERE predicate：创建一个部分索引。部分索引是一个只包含表的一部分记录的索引，通常是该表中比其他部分数据更有用的部分。例如，有一个表，表里包含已记账和未记账的订单，未记账的订单只占表的一小部分而且这部分是最常用的部分，此时就可以通过只在未记账部分创建一个索引来改善性能。另外一个可能的用途是使用带有UNIQUE的WHERE强制一个表的某个子集的唯一性。
* 取值范围：predicate表达式只能引用表的字段，它可以使用所有字段，而不仅是被索引的字段。目前，子查询和聚集表达式不能出现在WHERE子句里。不建议使用int等数值类型作为predicate，因为int等数值类型可以隐式转换为bool值（非0值隐式转换为true，0转换为false），可能导致非预期的结果。
* PARTITION index\_partition\_name：索引分区的名称。
* 取值范围：字符串，要符合标识符的命名规范。
* SUBPARTITION index\_subpartition\_name：索引二级分区的名称。
* 取值范围：字符串，要符合标识符的命名规范
* TABLESPACE index\_partition\_tablespace：索引分区的表空间。
* 取值范围：如果没有声明，将使用分区表索引的表空间index\_tablespace。
* COMPRESSTYPE：索引参数，设置索引压缩算法。1代表pglz算法，2代表zstd算法，默认不压缩。（仅支持B-TREE索引）
* 取值范围：0~2，默认值为0
* COMPRESS\_LEVEL：索引参数，设置索引压缩算法等级，仅当COMPRESSTYPE为2时生效。压缩等级越高，索引的压缩效果越好，索引的访问速度越慢。（仅支持B-TREE索引）
* 取值范围：-31~31，默认值为0
* COMPRESS\_CHUNK\_SIZE：索引参数，设置索引压缩chunk块大小。chunk数据块越聚集，预期能达到的压缩效果越好；chunk数据块越离散，索引的访问速度越受影响。（仅支持B-TREE索引）
* 取值范围：与页面大小有关。在页面大小为8k场景，取值范围为：512、1024、2048、4096。
* 默认值：4096
* COMPRESS\_PREALLOC\_CHUNKS：索引参数，设置索引压缩chunk块预分配数量。预分配数量越大，索引的压缩率相对越差，离散度越小，访问性能越好。（仅支持B-TREE索引）
* 取值范围：0~7，默认值为0
* 当COMPRESS\_CHUNK\_SIZE为512和1024时，支持预分配设置最大为7。
* 当COMPRESS\_CHUNK\_SIZE为2048时，支持预分配设置最大为3。
* 当COMPRESS\_CHUNK\_SIZE为4096时，支持预分配设置最大为1。
* COMPRESS\_BYTE\_CONVERT：索引参数，设置索引压缩字节转换预处理。在一些场景下可以提升压缩效果，同时会导致一定性能劣化。
* 取值范围：布尔值，默认关闭。
* COMPRESS\_DIFF\_CONVERT：索引参数，设置索引压缩字节差分预处理。只能与compress\_byte\_convert一起使用。在一些场景下可以提升压缩效果，同时会导致一定性能劣化。
* 取值范围：布尔值，默认关闭。

**示例**

* 创建表tpcds.ship\_mode\_t1。

CREATE schema tpcds;  
CREATE TABLE tpcds.ship\_mode\_t1  
 (  
 SM\_SHIP\_MODE\_SK INTEGER NOT NULL,  
 SM\_SHIP\_MODE\_ID CHAR(16) NOT NULL,  
 SM\_TYPE CHAR(30) ,  
 SM\_CODE CHAR(10) ,  
 SM\_CARRIER CHAR(20) ,  
 SM\_CONTRACT CHAR(20)  
 )   
 ;

* 在表tpcds.ship\_mode\_t1上的SM\_SHIP\_MODE\_SK字段上创建普通的唯一索引。

CREATE UNIQUE INDEX ds\_ship\_mode\_t1\_index1 ON tpcds.ship\_mode\_t1(SM\_SHIP\_MODE\_SK);

* 在表tpcds.ship\_mode\_t1上的SM\_SHIP\_MODE\_SK字段上创建指定B-tree索引。

CREATE INDEX ds\_ship\_mode\_t1\_index4 ON tpcds.ship\_mode\_t1 USING btree(SM\_SHIP\_MODE\_SK);

* 在表tpcds.ship\_mode\_t1上SM\_CODE字段上创建表达式索引。

CREATE INDEX ds\_ship\_mode\_t1\_index2 ON tpcds.ship\_mode\_t1(SUBSTR(SM\_CODE,1 ,4));

* 在表tpcds.ship\_mode\_t1上的SM\_SHIP\_MODE\_SK字段上创建SM\_SHIP\_MODE\_SK大于10的部分索引。

CREATE UNIQUE INDEX ds\_ship\_mode\_t1\_index3 ON tpcds.ship\_mode\_t1(SM\_SHIP\_MODE\_SK) WHERE SM\_SHIP\_MODE\_SK>10;

* 重命名一个现有的索引。

ALTER INDEX tpcds.ds\_ship\_mode\_t1\_index1 RENAME TO ds\_ship\_mode\_t1\_index5;

* 设置索引不可用。

ALTER INDEX tpcds.ds\_ship\_mode\_t1\_index2 UNUSABLE;

* 重建索引。

ALTER INDEX tpcds.ds\_ship\_mode\_t1\_index2 REBUILD;

* 删除一个现有的索引。

DROP INDEX tpcds.ds\_ship\_mode\_t1\_index2;

* 删除表。

DROP TABLE tpcds.ship\_mode\_t1;

* 创建表空间。

CREATE TABLESPACE example1 RELATIVE LOCATION 'tablespace1/tablespace\_1';  
CREATE TABLESPACE example2 RELATIVE LOCATION 'tablespace2/tablespace\_2';  
CREATE TABLESPACE example3 RELATIVE LOCATION 'tablespace3/tablespace\_3';  
CREATE TABLESPACE example4 RELATIVE LOCATION 'tablespace4/tablespace\_4';

* 创建表tpcds.customer\_address\_p1。

CREATE TABLE tpcds.customer\_address\_p1  
 (  
 CA\_ADDRESS\_SK INTEGER NOT NULL,  
 CA\_ADDRESS\_ID CHAR(16) NOT NULL,  
 CA\_STREET\_NUMBER CHAR(10) ,  
 CA\_STREET\_NAME VARCHAR(60) ,  
 CA\_STREET\_TYPE CHAR(15) ,  
 CA\_SUITE\_NUMBER CHAR(10) ,  
 CA\_CITY VARCHAR(60) ,  
 CA\_COUNTY VARCHAR(30) ,  
 CA\_STATE CHAR(2) ,  
 CA\_ZIP CHAR(10) ,  
 CA\_COUNTRY VARCHAR(20) ,  
 CA\_GMT\_OFFSET DECIMAL(5,2) ,  
 CA\_LOCATION\_TYPE CHAR(20)  
 )  
 TABLESPACE example1  
 PARTITION BY RANGE(CA\_ADDRESS\_SK)  
 (   
 PARTITION p1 VALUES LESS THAN (3000),  
 PARTITION p2 VALUES LESS THAN (5000) TABLESPACE example1,  
 PARTITION p3 VALUES LESS THAN (MAXVALUE) TABLESPACE example2  
 )  
 ENABLE ROW MOVEMENT;

* 在线在CA\_STREET\_NAME字段创建全局索引。

CREATE INDEX CONCURRENTLY idx\_global\_name ON tpcds.customer\_address\_p1(CA\_STREET\_NAME);

* 在线在CA\_CITY字段创建一级分区索引。

CREATE INDEX CONCURRENTLY idx\_local\_city ON tpcds.customer\_address\_p1(CA\_CITY);

* 创建分区表索引ds\_customer\_address\_p1\_index1，不指定索引分区的名称。

CREATE INDEX ds\_customer\_address\_p1\_index1 ON tpcds.customer\_address\_p1(CA\_ADDRESS\_SK) LOCAL;

* 创建分区表索引ds\_customer\_address\_p1\_index2，并指定索引分区的名称。

CREATE INDEX ds\_customer\_address\_p1\_index2 ON tpcds.customer\_address\_p1(CA\_ADDRESS\_SK) LOCAL  
 (  
 PARTITION CA\_ADDRESS\_SK\_index1,  
 PARTITION CA\_ADDRESS\_SK\_index2 TABLESPACE example3,  
 PARTITION CA\_ADDRESS\_SK\_index3 TABLESPACE example4  
 )   
 TABLESPACE example2;

* 创建GLOBAL分区索引。

CREATE INDEX ds\_customer\_address\_p1\_index3 ON tpcds.customer\_address\_p1(CA\_ADDRESS\_ID) GLOBAL;

* 不指定关键字，默认创建GLOBAL分区索引。

CREATE INDEX ds\_customer\_address\_p1\_index4 ON tpcds.customer\_address\_p1(CA\_ADDRESS\_ID);

* 修改分区表索引CA\_ADDRESS\_SK\_index2的表空间为example1。

ALTER INDEX tpcds.ds\_customer\_address\_p1\_index2 MOVE PARTITION CA\_ADDRESS\_SK\_index2 TABLESPACE example1;

* 修改分区表索引CA\_ADDRESS\_SK\_index3的表空间为example2。

ALTER INDEX tpcds.ds\_customer\_address\_p1\_index2 MOVE PARTITION CA\_ADDRESS\_SK\_index3 TABLESPACE example2;

* 重命名分区表索引。

ALTER INDEX tpcds.ds\_customer\_address\_p1\_index2 RENAME PARTITION CA\_ADDRESS\_SK\_index1 TO CA\_ADDRESS\_SK\_index4;

* 删除索引和分区表。

DROP INDEX tpcds.ds\_customer\_address\_p1\_index1;  
DROP INDEX tpcds.ds\_customer\_address\_p1\_index2;  
DROP TABLE tpcds.customer\_address\_p1;

* 删除表空间。

DROP TABLESPACE example1;  
DROP TABLESPACE example2;  
DROP TABLESPACE example3;  
DROP TABLESPACE example4;

* 创建列存表以及列存表GIN索引。

CREATE table cgin\_create\_test(a int, b text) with (orientation = column);  
CREATE index cgin\_test on cgin\_create\_test using gin(to\_tsvector('ngram', b));

**优化建议**

create index

### CREATE MASKING POLICY

**功能描述**

创建脱敏策略。

**注意事项**

只有poladmin、sysadmin或初始用户能执行此操作。

需要开启安全策略开关，即设置GUC参数enable\_security\_policy=on，脱敏策略才可以生效。

**语法格式**

CREATE MASKING POLICY policy\_name masking\_clause[, ...]\* policy\_filter [ENABLE | DISABLE];

* masking\_clause：

masking\_function ON LABEL(label\_name[, ...]\*)

* masking\_function：
* maskall不是预置函数，硬编码在代码中，不支持df展示。
* 预置时脱敏方式如下：

maskall | randommasking | creditcardmasking | basicemailmasking | fullemailmasking | shufflemasking | alldigitsmasking | regexpmasking

* policy\_filter:

FILTER ON FILTER\_TYPE(filter\_value [,...]\*)[,...]\*

* FILTER\_TYPE:

IP | APP | ROLES

**参数说明**

* **policy\_name**
* 审计策略名称，需要唯一，不可重复。
* 取值范围：字符串，要符合标识符的命名规范。
* **label\_name**
* 资源标签名称。
* **masking\_clause**
* 指出使用何种脱敏函数对被label\_name标签标记的数据库资源进行脱敏，支持用schema.function的方式指定脱敏函数。
* **policy\_filter**
* 指出该脱敏策略对何种身份的用户生效，若为空表示对所用用户生效。
* **FILTER\_TYPE**
* 描述策略过滤的条件类型，包括IP | APP | ROLES。
* **filter\_value**
* 指具体过滤信息内容，例如具体的IP、具体的APP名称、具体的用户名。
* **ENABLE|DISABLE**
* 可以打开或关闭脱敏策略。若不指定ENABLE|DISABLE，语句默认为ENABLE。

**示例**

--创建dev\_mask和bob\_mask用户。  
vastbase=# CREATE USER dev\_mask PASSWORD 'dev@1234';  
vastbase=# CREATE USER bob\_mask PASSWORD 'bob@1234';  
  
--创建一个表tb\_for\_masking  
vastbase=# CREATE TABLE tb\_for\_masking(col1 text, col2 text, col3 text);  
  
--创建资源标签标记敏感列col1  
vastbase=# CREATE RESOURCE LABEL mask\_lb1 ADD COLUMN(tb\_for\_masking.col1);  
  
--创建资源标签标记敏感列col2  
vastbase=# CREATE RESOURCE LABEL mask\_lb2 ADD COLUMN(tb\_for\_masking.col2);  
  
--对访问敏感列col1的操作创建脱敏策略  
vastbase=# CREATE MASKING POLICY maskpol1 maskall ON LABEL(mask\_lb1);  
  
--创建仅对用户dev\_mask和bob\_mask,客户端工具为psql和vsql，IP地址为'10.20.30.40', '127.0.0.0/24'场景下生效的脱敏策略。  
vastbase=# CREATE MASKING POLICY maskpol2 randommasking ON LABEL(mask\_lb2) FILTER ON ROLES(dev\_mask, bob\_mask), APP(psql, vsql), IP('10.20.30.40', '127.0.0.0/24');

**相关链接**

参考SQL语法参考->SQL语法->ALTER MASKING POLICY、DROP MASKING POLICY章节。

### CREATE MATERIALIZED VIEW

CREATE MATERIALIZED VIEW会创建一个全量物化视图，并且后续可以使用REFRESH MATERIALIZED VIEW（全量刷新）刷新物化视图的数据。

CREATE MATERIALIZED VIEW类似于CREATE TABLE AS，不过它会记住被用来初始化该视图的查询， 因此它可以在后续中进行数据刷新。一个物化视图有很多和表相同的属性，但是不支持临时物化视图。

**注意事项**

* 全量物化视图不可以在临时表或全局临时表上创建。
* 全量物化视图不支持nodegroup。
* 创建全量物化视图后，基表中的绝大多数DDL操作不再支持。
* 不支持对全量物化视图进行IUD操作。
* 全量物化视图创建后，当基表数据发生变化时，需要使用刷新（REFRESH）命令保持物化视图与基表同步。
* Ustore引擎不支持物化创建、使用视图。

**语法格式**

CREATE MATERIALIZED VIEW mv\_name  
 [ (column\_name [, ...] ) ]  
 [ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ TABLESPACE tablespace\_name ]  
 AS query  
 [ WITH [ NO ] DATA ];

**参数说明**

* **mv\_name**
* 要创建的物化视图的名称（可以被模式限定）。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 新物化视图中的一个列名。物化视图支持指定列，指定列需要和后面的查询语句结果的列数量保持一致；如果没有提供列名，会从查询的输出列名中获取列名。
* 取值范围：字符串，要符合标识符的命名规范。
* **WITH ( storage\_parameter [= value] [, ... ] )**
* 这个子句为表或索引指定一个可选的存储参数。详见：SQL语法参考->SQL语法->CREATE TABLE章节。
* **TABLESPACE tablespace\_name**
* 指定新建物化视图所属表空间。如果没有声明，将使用默认表空间。
* **AS query**
* 一个SELECT、TABLE 或者VALUES命令。这个查询将在一个安全受限的操作中运行。

**示例**

--创建一个普通表  
vastbase=# CREATE TABLE my\_table (c1 int, c2 int);  
--创建全量物化视图  
vastbase=# CREATE MATERIALIZED VIEW my\_mv AS SELECT \* FROM my\_table;  
--基表写入数据  
vastbase=# INSERT INTO my\_table VALUES(1,1),(2,2);  
--对全量物化视图my\_mv进行全量刷新  
vastbase=# REFRESH MATERIALIZED VIEW my\_mv;

**相关链接**

SQL语法参考->SQL语法->ALTER MATERIALIZED VIEW、CREATE INCREMENTAL MATERIALIZED VIEW、CREATE TABLE、DROP MATERIALIZED VIEW、REFRESH INCREMENTAL MATERIALIZED VIEW、REFRESH MATERIALIZED VIEW章节。

### CREATE MODEL

**功能描述**

训练机器学习模型并保存模型。

**注意事项**

* 模型名称具有唯一性约束，注意命名格式。
* AI训练时长波动较大，在部分情况下训练运行时间较长，设置的GUC参数statement\_timeout时长过短会导致训练中断。建议statement\_timeout设置为0，不对语句执行时长进行限制。

**语法格式**

CREATE MODEL model\_name USING algorithm\_name   
[FEATURES { {expression [ [ AS ] output\_name ]} [, ...] }]  
[TARGET { {expression [ [ AS ] output\_name ]} [, ...] }]  
FROM { table\_name | select\_query }  
WITH hyperparameter\_name = { hyperparameter\_value | DEFAULT } [, ...] }

**参数说明**

* **model\_name**
* 对训练模型进行命名，模型名称具有唯一性约束。
* 取值范围：字符串，需要符合标识符的命名规范。
* **architecture\_name**
* 训练模型的算法类型。
* 取值范围：字符型，当前支持：logistic\_regression、linear\_regression、svm\_classification、kmeans。
* **attribute\_list**
* 枚举训练模型的输入列名。
* 取值范围：字符型，需要符合数据属性名的命名规范。
* **attribute\_name**
* 在监督学习任务重训练模型的目标列名(可进行简单的表达式处理)。
* 取值范围：字符型，需要符合数据属性名的命名规范。
* **subquery**
* 数据源。
* 取值范围：字符串，符合数据库SQL语法。
* **hyper\_parameter\_name**
* 机器学习模型的超参名称。
* 取值范围：字符串，针对不同算法范围不同，详情请参考：AI特性->DB4AI：数据库驱动AI->原生DB4AI引擎章节中的表2。
* **hp\_value**
* 超参数值。
* 取值范围：字符串，针对不同算法范围不同，详情请参考：AI特性->DB4AI：数据库驱动AI->原生DB4AI引擎章节中的表3。

**示例**

1、创建测试表houses，并插入测试数据。

CREATE TABLE houses(id int,size int,lot varchar,price int);  
INSERT INTO houses values('1','2','3','4');  
INSERT INTO houses values('5','6','7','8');

2、创建模型。

CREATE MODEL price\_model USING logistic\_regression  
 FEATURES size  
 TARGET price  
 FROM houses  
 WITH learning\_rate=0.88, max\_iterations=default;

**相关链接**

SQL语法参考->SQL语法->DROP MODEL、PREDICT BY章节。

### CREATE OPERATOR

**功能描述**

定义一个新操作符。

**注意事项**

CREATE OPERATOR定义一个新的 name操作符。 定义该操作符的用户将成为其所有者。如果给出了一个模式名， 那么该操作符将在指定的模式中创建。否则它会在当前模式中创建。

操作符 name 是一个由下列字符组成的字符串：

+ - \* / < > = ~ ! @ # % ^ & | ` ?

选择名字的时候有几个限制：

* --和/\*不能在操作符名的任何地方出现， 因为它们会被认为是一个注释的开始。
* 一个多字符的操作符不能以+或-结尾， 除非该名字还包含至少下面字符之一：
* ~ ! @ # % ^ & | ` ?
* => 作为一个操作符名的使用已经废弃了。

操作符!=在输入时映射成<>， 因此这两个名称总是等价的。

至少需要定义一个LEFTARG和RIGHTARG。对于双目操作符来说， 两者都需要定义。对右目操作符来说，只需要定义LEFTARG， 而对于左目操作符来说，只需要定义RIGHTARG。

同样，function\_name 过程必须已经用CREATE FUNCTION定义过， 而且必须定义为接受正确数量的指定类型参数(一个或是两个)。

其它子句声明可选的操作符优化子句。他们的含义在PostgreSQL 9.3.1 中文手册第 35.13 节里定义。详见：<http://postgres.cn/docs/9.3/xoper-optimization.html>。

要想能够创建一个操作符，你必须在参数类型和返回类型上有USAGE权限， 还要在底层函数上有EXECUTE权限。如果指定了交换或者负操作符， 你必须拥有这些操作符。

**语法格式**

CREATE OPERATOR name (  
 PROCEDURE = function\_name  
 [, LEFTARG = left\_type ] [, RIGHTARG = right\_type ]  
 [, COMMUTATOR = com\_op ] [, NEGATOR = neg\_op ]  
 [, RESTRICT = res\_proc ] [, JOIN = join\_proc ]  
 [, HASHES ] [, MERGES ]  
)

**参数说明**

* **name**
* 要定义的操作符。可用的字符见上文。其名字可以用模式修饰， 比如CREATE OPERATOR myschema.+ (...)。如果没有模式， 则在当前模式中创建操作符。同一个模式中的两个操作符可以有一样的名字， 只要他们操作不同的数据类型。这是一个重载过程。
* **function\_name**
* 用于实现该操作符的函数。
* **left\_type**
* 操作符左边的参数数据类型，如果存在的话。如果是左目操作符，这个参数可以省略。
* **right\_type**
* 操作符右边的参数数据类型，如果存在的话。如果是右目操作符，这个参数可以省略。
* **com\_op**
* 该操作符对应的交换操作符。
* **neg\_op**
* 该操作符对应的负操作符。
* **res\_proc**
* 此操作符约束选择性评估函数。
* **join\_proc**
* 此操作符连接选择性评估函数。
* **HASHES**
* 表明此操作符支持 Hash 连接。
* **MERGES**
* 表明此操作符可以支持一个融合连接。
* 使用OPERATOR()语法在com\_op 或者其它可选参数里给出一个模式修饰的操作符名，比如：

COMMUTATOR = OPERATOR(myschema.===) ,

**示例**

下面命令定义一个新操作符：面积相等，用于box数据类型。

CREATE OPERATOR === (  
 LEFTARG = box,  
 RIGHTARG = box,  
 PROCEDURE = area\_equal\_procedure,  
 COMMUTATOR = ===,  
 NEGATOR = !==,  
 RESTRICT = area\_restriction\_procedure,  
 JOIN = area\_join\_procedure,  
 HASHES, MERGES  
);

### CREATE PACKAGE

**功能描述**

创建一个新的PACKAGE。

**注意事项**

* package只支持集中式，无法在分布式中使用。
* 在package specification中声明过的函数或者存储过程，必须在package body中找到定义。
* 在实例化中，无法调用带有commit/rollback的存储过程。
* 不能在Trigger中调用package函数。
* 不能在外部SQL中直接使用package当中的变量。
* 不允许在package外部调用package的私有变量和存储过程。
* 不支持其它存储过程不支持的用法，例如，在function中不允许调用commit/rollback，则package的function中同样无法调用commit/rollback。
* 不支持schema与package同名。
* 只支持A风格的存储过程和函数定义。
* 不支持package内有同名变量，包括包内同名参数。
* package的全局变量为session级，不同session之间package的变量不共享。
* package中调用自治事务的函数，不允许使用package中的cursor变量，以及递归的使用package中cursor变量的函数。
* package中不支持声明ref cursor变量。
* package默认为SECURITY INVOKER权限，如果想将默认行为改为SECURITY DEFINER权限，需要设置GUC参数behavior\_compat\_options='plsql\_security\_definer'。
* 被授予CREATE ANY PACKAGE权限的用户，可以在public模式和用户模式下创建PACKAGE。
* 如果需要创建带有特殊字符的package名，特殊字符中不能含有空格，并且最好设置GUC参数behavior\_compat\_options='skip\_insert\_gs\_source',否则可能引起报错。

**语法格式**

* CREATE PACKAGE SPECIFICATION语法格式。

CREATE [ OR REPLACE ] PACKAGE [ schema. ]package\_name  
 [ invoker\_rights\_clause ] { IS | AS } item\_list\_1 END ;

invoker\_rights\_clause可以被声明为AUTHID DEFINER或者AUTHID CURRENT\_USER，分别为定义者权限和调用者权限。  
item\_list\_1可以为声明的变量或者存储过程以及函数。

PACKAGE SPECIFICATION（包规格）声明了包内的公有变量、函数、异常等，可以被外部函数或者存储过程调用。在PACKAGE SPECIFICATION中只能声明存储过程、函数，不能定义存储过程或者函数。

* CREATE PACKAGE BODY语法格式。

CREATE [ OR REPLACE ] PACKAGE BODY [ schema. ]package\_name  
 { IS | AS } declare\_section [ initialize\_section ] END package\_name;

PACKAGE BODY（包体内）定义了包的私有变量、函数等。如果变量或者函数没有在PACKAGE SPECIFICATION中声明过，那么这个变量或者函数则为私有变量或者函数。

PACKAGE BODY也可以声明实例化部分，用来初始化package，详见示例。

**示例**

* CREATE PACKAGE SPECIFICATION示例

CREATE OR REPLACE PACKAGE emp\_bonus IS  
var1 int:=1;--公有变量  
var2 int:=2;  
PROCEDURE testpro1(var3 int);--公有存储过程，可以被外部调用  
END ;  
/

* CREATE PACKAGE BODY示例

drop table if exists test1;  
create or replace package body emp\_bonus is  
var3 int:=3;  
var4 int:=4;  
procedure testpro1(var3 int)  
is  
begin  
create table if not exists test1(col1 int);  
insert into test1 values(var1);  
insert into test1 values(var4);  
end;  
begin --实例化开始  
var4:=9;  
testpro1(var4);  
end;  
end;  
/

* 调用PACKAGE示例

call emp\_bonus.testpro1(1); --使用call调用package存储过程  
select emp\_bonus.testpro1(1); --使用select调用package存储过程  
--匿名块里调用package存储过程  
begin  
emp\_bonus.testpro1(1);  
end;  
/

### CREATE PROCEDURE

**功能描述**

创建一个新的存储过程。

**注意事项**

* 如果创建存储过程时参数或返回值带有精度，不进行精度检测。
* 创建存储过程时，存储过程定义中对表对象的操作建议都显示指定模式，否则可能会导致存储过程执行异常。
* 在创建存储过程时，存储过程内部通过SET语句设置current\_schema和search\_path无效。执行完函数search\_path和current\_schema与执行函数前的search\_path和current\_schema保持一致。
* 如果存储过程参数中带有出参，SELECT调用存储过程必须缺省出参，CALL调用存储过程调用非重载函数时必须指定出参，对于重载的package函数，out参数可以缺省，具体信息参见“SQL语法参考->SQL语法->CALL”章节中的示例。
* 存储过程指定package属性时支持重载。
* 在创建procedure时，不能在avg函数外面嵌套其他agg函数，或者其他系统函数。
* 在存储过程内部调用其它无参数的存储过程时，可以省略括号，直接使用存储过程名进行调用。
* 在存储过程内部调用其他有出参的函数，如果在赋值表达式中调用时，被调函数的出参可以省略，给出了也会被忽略。
* 存储过程支持参数注释的查看与导出、导入。
* 存储过程支持介于IS/AS与plsql\_body之间的注释的查看与导出、导入。
* 存储过程默认为SECURITY INVOKER权限，如果想将默认行为改为SECURITY DEFINER权限，需要设置GUC参数behavior\_compat\_options='plsql\_security\_definer'。
* 被授予CREATE ANY FUNCTION权限的用户，可以在用户模式下创建/替换存储过程。
* out/inout参数必须传入变量，不能够传入常量。
* 集中式环境下，想要调用in参数相同，out参数不同的存储过程，需要设置GUC参数behavior\_compat\_options='proc\_outparam\_override',并且打开参数后，无论使用select还是call调用存储过程，都必须加上out参数。打开参数后，不支持使用perform调用存储过程或函数。
* 不支持在函数或存储过程中执行analyze操作。

**语法格式**

CREATE [ OR REPLACE ] PROCEDURE procedure\_name  
 [ ( {[ argname ] [ argmode ] argtype [ { DEFAULT | := | = } expression ]}[,...]) ]  
 [  
 { IMMUTABLE | STABLE | VOLATILE }  
 | { SHIPPABLE | NOT SHIPPABLE }  
 | {PACKAGE}  
 | [ NOT ] LEAKPROOF  
 | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }  
 | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER | AUTHID CURRENT\_USER}  
 | COST execution\_cost  
 | SET configuration\_parameter { TO value | = value | FROM CURRENT }  
 ][ ... ]  
 { IS | AS }   
plsql\_body   
/

**参数说明**

* **OR REPLACE**
* 当存在同名的存储过程时，替换原来的定义。
* **procedure\_name**
* 创建的存储过程名称，可以带有模式名。
* 取值范围：字符串，要符合标识符的命名规范。
* **argmode**
* 参数的模式。
* fig: **须知：**   
  VARIADIC用于声明数组类型的参数。
* 取值范围： IN、OUT、INOUT或VARIADIC。缺省值是IN。只有OUT模式的参数能跟在VARIADIC参数之后。
* **argname**
* 参数的名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **argtype**
* 参数的数据类型。可以使用%TYPE或%ROWTYPE间接引用变量或表的类型，详细可参考：PL/SQL->存储过程->基本语句->定义变量章节。
* 取值范围：可用的数据类型。
* **configuration\_parameter**
* **value**
  + 把指定的配置参数设置为给定的值。如果value是DEFAULT，则在新的会话中使用系统的缺省设置。OFF关闭设置。
  + 取值范围：字符串
* DEFAULT
* OFF
* 指定默认值。
* **from current**
  + 取当前会话中的值设置为configuration\_parameter的值。
* **IMMUTABLE、STABLE**等
* 行为约束可选项。各参数的功能与CREATE FUNCTION类似，详细说明见：SQL语法参考->SQL语法->CREATE FUNCTION章节。
* **plsql\_body**
* PL/SQL存储过程体。
* fig: **须知：**   
  当在存储过程体中进行创建用户等涉及用户密码相关操作时，系统表及csv日志中会记录密码的明文。因此不建议用户在存储过程体中进行涉及用户密码的相关操作。

fig: **说明：**   
argname和argmode的顺序没有严格要求，推荐按照argname、argmode、argtype的顺序使用。

**相关链接**

参考：SQL语法参考->SQL语法->DROP PROCEDURE章节。

### CREATE PUBLICATION

**功能描述**

向当前数据库添加一个新的发布，发布的名称必须与当前数据库中任何现有发布的名称不同。发布本质上是通过逻辑复制将一组表的数据变更进行复制。

**注意事项**

* 如果既没有指定FOR TABLE，也没有指定FOR ALL TABLES， 那么这个发布就是以一组空表开始的，可以在后续添加表。
* 创建发布不会开始复制。它只为未来的订阅者定义一个分组和过滤逻辑。 要创建一个发布，调用者必须拥有当前数据库的CREATE权限。（当然，系统管理员不需要这个检查。）
* 要将表添加到发布中，调用者必须拥有该表的所有权。FOR ALL TABLES子句要求调用者是具有SYSADMIN权限用户。
* 添加到发布UPDATE或DELETE操作的发布的表必须已经定义了REPLICA IDENTITY，否则将在这些表上禁止这些操作。
* COPY ... FROM命令是作为INSERT操作发布的。不发布TRUNCATE和DDL操作。

**语法格式**

CREATE PUBLICATION name   
 [ FOR TABLE table\_name [, ...]   
 | FOR ALL TABLES ]   
 [ WITH ( publication\_parameter [=value] [, ... ] ) ];

**参数说明**

* **name**
* 新发布的名称。
* **FOR TABLE**
* 指定要添加到发布的表的列表。 只有持久基表才能成为发布的一部分，临时表、非日志表、外表、MOT表、物化视图、常规视图不能被发布。
* **FOR ALL TABLES**
* 将发布标记为复制数据库中所有表的更改，包括在将来创建的表。
* **WITH ( publication\_parameter [= value] [, ... ] )**
* 该子句指定发布的可选参数。支持下列参数：
* **publish (string)**
  + 这个参数决定了哪些DML操作可以发布给订阅者。该值是一个用逗号分隔的操作列表，允许的操作是insert、update和delete，不指定则默认发布所有的动作。该选项的默认值是'insert, update, delete'。

**示例**

--创建一个发布，发布两个表中所有更改。  
CREATE PUBLICATION mypublication FOR TABLE users, departments;  
  
--创建一个发布，发布所有表中的所有更改。  
CREATE PUBLICATION alltables FOR ALL TABLES;  
  
--创建一个发布，只发布一个表中的INSERT操作。  
CREATE PUBLICATION insert\_only FOR TABLE mydata WITH (publish = 'insert');  
  
--修改发布的动作。  
ALTER PUBLICATION insert\_only SET (publish='insert,update,delete');  
  
--删除发布。  
DROP PUBLICATION insert\_only;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER PUBLICATION、DROP PUBLICATION章节。

### CREATE RESOURCE LABEL

**功能描述**

创建资源标签。

**注意事项**

只有poladmin、sysadmin或初始用户能正常执行此操作。

**语法格式**

CREATE RESOURCE LABEL [IF NOT EXISTS] label\_name ADD label\_item\_list[, ...]\*;

* label\_item\_list：

resource\_type(resource\_path[, ...]\*)

* resource\_type：

TABLE | COLUMN | SCHEMA | VIEW | FUNCTION

**参数说明**

* **label\_name**
* 资源标签名称，创建时要求不能与已有标签重名。
* 取值范围：字符串，要符合标识符的命名规范。
* **resource\_type**
* 指的是要标记的数据库资源类型。
* **resource\_path**
* 指的是描述具体的数据库资源的路径。

**示例**

--创建一个表tb\_for\_label  
vastbase=# CREATE TABLE tb\_for\_label(col1 text, col2 text, col3 text);  
  
--创建一个模式schema\_for\_label  
vastbase=# CREATE SCHEMA schema\_for\_label;  
  
--创建一个视图view\_for\_label  
vastbase=# CREATE VIEW view\_for\_label AS SELECT 1;  
  
--创建一个函数func\_for\_label  
vastbase=# CREATE FUNCTION func\_for\_label RETURNS TEXT AS $$ SELECT col1 FROM tb\_for\_label; $$ LANGUAGE SQL;  
  
--基于表创建资源标签  
vastbase=# CREATE RESOURCE LABEL IF NOT EXISTS table\_label add TABLE(public.tb\_for\_label);  
  
--基于列创建资源标签  
vastbase=# CREATE RESOURCE LABEL IF NOT EXISTS column\_label add COLUMN(public.tb\_for\_label.col1);  
  
--基于模式创建资源标签  
vastbase=# CREATE RESOURCE LABEL IF NOT EXISTS schema\_label add SCHEMA(schema\_for\_label);  
  
--基于视图创建资源标签  
vastbase=# CREATE RESOURCE LABEL IF NOT EXISTS view\_label add VIEW(view\_for\_label);  
  
--基于函数创建资源标签  
vastbase=# CREATE RESOURCE LABEL IF NOT EXISTS func\_label add FUNCTION(func\_for\_label);

**相关链接**

参考：SQL语法参考->SQL语法->ALTER RESOURCE LABEL、DROP RESOURCE LABEL章节。

### CREATE ROLE

**功能描述**

创建角色。角色是拥有数据库对象和权限的实体。在不同的环境中角色可以认为是一个用户，一个组或者兼顾两者。

**语法格式**

CREATE ROLE role\_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' | DISABLE };

其中角色信息设置子句option语法为：

{SYSADMIN | NOSYSADMIN}  
 | {MONADMIN | NOMONADMIN}  
 | {OPRADMIN | NOOPRADMIN}  
 | {POLADMIN | NOPOLADMIN}  
 | {AUDITADMIN | NOAUDITADMIN}  
 | {SSOADMIN | NOSSOADMIN}  
 | {CREATEDB | NOCREATEDB}  
 | {USEFT | NOUSEFT}  
 | {CREATEROLE | NOCREATEROLE}  
 | {INHERIT | NOINHERIT}  
 | {LOGIN | NOLOGIN}  
 | {REPLICATION | NOREPLICATION}  
 | {INDEPENDENT | NOINDEPENDENT}  
 | {VCADMIN | NOVCADMIN}  
 | {PERSISTENCE | NOPERSISTENCE}  
 | CONNECTION LIMIT connlimit  
 | VALID BEGIN 'timestamp'  
 | VALID UNTIL 'timestamp'  
 | RESOURCE POOL 'respool'  
 | PERM SPACE 'spacelimit'  
 | TEMP SPACE 'tmpspacelimit'  
 | SPILL SPACE 'spillspacelimit'  
 | IN ROLE role\_name [, ...]  
 | IN GROUP role\_name [, ...]  
 | ROLE role\_name [, ...]  
 | ADMIN rol e\_name [, ...]  
 | USER role\_name [, ...]  
 | SYSID uid  
 | DEFAULT TABLESPACE tablespace\_name  
 | PROFILE DEFAULT  
 | PROFILE profile\_name  
 | PGUSER

**参数说明**

* role\_name：角色名称
* 取值范围：字符串，要符合标识符的命名规范，且最多为63个字符。若超过63个字符，数据库会截断并保留前63个字符当做角色名称。在创建角色时，数据库的时候会给出提示信息。
* 标识符需要为字母、下划线、数字（0-9）或美元符号（$），且必须以字母（a-z）或下划线（\_)开头。
* password：登录密码
* 密码规则如下：
* 密码默认不少于8个字符。
* 不能与用户名及用户名倒序相同。
* 至少包含大写字母（A-Z）、小写字母（a-z）、数字（0-9）、非字母数字字符（限定为~!@#$%^&\*()-\_=+|[{}];:,<.>/?）四类字符中的三类字符。
* 密码也可以是符合格式要求的密文字符串，这种情况主要用于用户数据导入场景，不推荐用户直接使用。如果直接使用密文密码，用户需要知道密文密码对应的明文，并且保证明文密码复杂度，数据库不会校验密文密码复杂度，直接使用密文密码的安全性由用户保证。
* 创建角色时，应当使用双引号或单引号将用户密码括起来。
* 取值范围：不为空的字符串。
* EXPIRED：在创建用户时可指定EXPIRED参数，即创建密码失效用户，该用户不允许执行简单查询和扩展查询。只有在修改自身密码后才可正常执行语句。
* DISABLE：默认情况下，用户可以更改自己的密码，除非密码被禁用。要禁用用户的密码，请指定DISABLE。禁用某个用户的密码后，将从系统中删除该密码，此类用户只能通过外部认证来连接数据库，例如：kerberos认证。只有管理员才能启用或禁用密码。普通用户不能禁用初始用户的密码。要启用密码，请运行ALTER USER并指定密码。
* ENCRYPTED | UNENCRYPTED：控制密码存储在系统表里的口令是否加密。按照产品安全要求，密码必须加密存储，所以，UNENCRYPTED在Vastbase中禁止使用。因为系统无法对指定的加密口令字符串进行解密，所以如果目前的口令字符串已经是用SHA256加密的格式，则会继续照此存放，而不管是否声明了ENCRYPTED或UNENCRYPTED。这样就允许在dump/restore的时候重新加载加密的口令。
* SYSADMIN | NOSYSADMIN：决定一个新角色是否为“系统管理员”，具有SYSADMIN属性的角色拥有系统最高权限。
* 缺省为NOSYSADMIN。
* MONADMIN | NOMONADMIN：定义角色是否是监控管理员。
* 缺省为NOMONADMIN。
* OPRADMIN | NOOPRADMIN：定义角色是否是运维管理员。
* 缺省为NOOPRADMIN。
* POLADMIN | NOPOLADMIN：定义角色是否是安全策略管理员。
* 缺省为NOPOLADMIN。
* SSOADMIN | NOSSOADMIN：定义角色是否有安全管理属性。
* 缺省为NOSSOADMIN。
* AUDITADMIN | NOAUDITADMIN：定义角色是否有审计管理属性。
* 缺省为NOAUDITADMIN。
* CREATEDB | NOCREATEDB：决定一个新角色是否能创建数据库。新角色没有创建数据库的权限。
* 缺省为NOCREATEDB。
* USEFT | NOUSEFT：该参数为保留参数，暂未启用。
* CREATEROLE | NOCREATEROLE：决定一个角色是否可以创建新角色（也就是执行CREATE ROLE和CREATE USER）。 一个拥有CREATEROLE权限的角色也可以修改和删除其他角色。
* 缺省为NOCREATEROLE。
* INHERIT | NOINHERIT：这些子句决定一个角色是否“继承”它所在组的角色的权限。不推荐使用。
* LOGIN | NOLOGIN：具有LOGIN属性的角色才可以登录数据库。一个拥有LOGIN属性的角色可以认为是一个用户。
* 缺省为NOLOGIN。
* REPLICATION | NOREPLICATION：定义角色是否允许流复制或设置系统为备份模式。REPLICATION属性是特定的角色，仅用于复制。
* 缺省为NOREPLICATION。
* INDEPENDENT | NOINDEPENDENT：定义私有、独立的角色。具有INDEPENDENT属性的角色，管理员对其进行的控制、访问的权限被分离，具体规则如下：
* 未经INDEPENDENT角色授权，系统管理员无权对其表对象进行增、删、查、改、拷贝、授权操作。
* 若将私有用户表的相关权限授予其他非私有用户，系统管理员也会获得同样的权限。
* 未经INDEPENDENT角色授权，系统管理员和拥有CREATEROLE属性的安全管理员无权修改INDEPENDENT角色的继承关系。
* 系统管理员无权修改INDEPENDENT角色的表对象的属主。
* 系统管理员和拥有CREATEROLE属性的安全管理员无权去除INDEPENDENT角色的INDEPENDENT属性。
* 系统管理员和拥有CREATEROLE属性的安全管理员无权修改INDEPENDENT角色的数据库口令，INDEPENDENT角色需管理好自身口令，口令丢失无法重置。
* 管理员属性用户不允许定义修改为INDEPENDENT属性。
* VCADMIN | NOVCADMIN：该版本没有实际意义。
* PERSISTENCE | NOPERSISTENCE：定义永久用户。仅允许初始用户创建、修改和删除具有PERSISTENCE属性的永久用户。
* CONNECTION LIMIT：声明该角色可以使用的并发连接数量。
* 系统管理员不受此参数的限制。
* connlimit数据库主节点单独统计，Vastbase整体的连接数 = connlimit \* 当前正常数据库主节点个数。
* 取值范围：整数，>=-1，缺省值为100，-1表示没有限制。
* VALID BEGIN：设置角色生效的时间戳。如果省略了该子句，角色无有效开始时间限制。
* VALID UNTIL：设置角色失效的时间戳。如果省略了该子句，角色无有效结束时间限制。
* RESOURCE POOL：设置角色使用的resource pool名称，该名称属于系统表：pg\_resource\_pool。
* PERM SPACE：设置用户使用空间的大小。
* TEMP SPACE：设置用户临时表存储空间限额。
* SPILL SPACE：设置用户算子落盘空间限额。
* IN ROLE：新角色立即拥有IN ROLE子句中列出的一个或多个现有角色拥有的权限。不推荐使用。
* IN GROUP：IN GROUP是IN ROLE过时的拼法。不推荐使用。
* ROLE：ROLE子句列出一个或多个现有的角色，它们将自动添加为这个新角色的成员，拥有新角色所有的权限。
* ADMIN：ADMIN子句类似ROLE子句，不同的是ADMIN后的角色可以把新角色的权限赋给其他角色。
* USER：USER子句是ROLE子句过时的拼法。
* SYSID：SYSID子句将被忽略，无实际意义。
* DEFAULT TABLESPACE：DEFAULT TABLESPACE子句将被忽略，无实际意义。
* PROFILE：PROFILE子句将被忽略，无实际意义。
* PGUSER：当前版本该属性没有实际意义，仅为了语法的前向兼容而保留。

**注意事项**

* 在数据库中添加一个新角色，角色无登录权限。
* 创建角色的用户必须具备CREATE ROLE的权限或者是系统管理员。
* 只允许创建相同属性的管理员用户。

**示例**

* 创建一个角色，名为manager，密码为Bigdata@123。

CREATE ROLE manager IDENTIFIED BY 'Bigdata@123';

* 创建一个角色，从2015年1月1日开始生效，到2026年1月1日失效。

CREATE ROLE miriam WITH LOGIN PASSWORD 'Bigdata@123' VALID BEGIN '2015-01-01' VALID UNTIL '2026-01-01';

* 修改角色manager的密码为abcd@123。

ALTER ROLE manager IDENTIFIED BY 'abcd@123' REPLACE 'Bigdata@123';

* 修改角色manager为系统管理员。

ALTER ROLE manager SYSADMIN;

* 删除角色manager。

DROP ROLE manager;

* 删除角色miriam。

DROP ROLE miriam;

### CREATE ROW LEVEL SECURITY POLICY

**功能描述**

对表创建行访问控制策略。

当对表创建了行访问控制策略，只有打开该表的行访问控制开关（ALTER TABLE ... ENABLE ROW LEVEL SECURITY），策略才能生效。否则不生效。

当前行访问控制影响数据表的读取操作（SELECT、UPDATE、DELETE），暂不影响数据表的写入操作（INSERT、MERGE INTO）。表所有者或系统管理员可以在USING子句中创建表达式，在客户端执行数据表读取操作时，数据库后台在查询重写阶段会将满足条件的表达式拼接并应用到执行计划中。针对数据表的每一条元组，当USING表达式返回TRUE时，元组对当前用户可见，当USING表达式返回FALSE或NULL时，元组对当前用户不可见。

行访问控制策略名称是针对表的，同一个数据表上不能有同名的行访问控制策略；对不同的数据表，可以有同名的行访问控制策略。

行访问控制策略可以应用到指定的操作（SELECT、UPDATE、DELETE、ALL），ALL表示会影响SELECT、UPDATE、DELETE三种操作；定义行访问控制策略时，若未指定受影响的相关操作，默认为ALL。

行访问控制策略可以应用到指定的用户（角色），也可应用到全部用户（PUBLIC）；定义行访问控制策略时，若未指定受影响的用户，默认为PUBLIC。

**注意事项**

* 支持对行存表、行存分区表、列存表、列存分区表、unlogged表、hash表定义行访问控制策略。
* 不支持外表、本地临时表定义行访问控制策略。
* 不支持对视图定义行访问控制策略。
* 同一张表上可以创建多个行访问控制策略，一张表最多创建100个行访问控制策略。
* 系统管理员不受行访问控制影响，可以查看表的全量数据。
* 通过SQL语句、视图、函数、存储过程查询包含行访问控制策略的表，都会受影响。

**语法格式**

CREATE [ ROW LEVEL SECURITY ] POLICY policy\_name ON table\_name  
 [ AS { PERMISSIVE | RESTRICTIVE } ]  
 [ FOR { ALL | SELECT | UPDATE | DELETE } ]  
 [ TO { role\_name | PUBLIC | CURRENT\_USER | SESSION\_USER } [, ...] ]  
 USING ( using\_expression )

**参数说明**

* **policy\_name**
* 行访问控制策略名称，同一个数据表上行访问控制策略名称不能相同。
* **table\_name**
* 行访问控制策略的表名。
* **PERMISSIVE | RESTRICTIVE**
* PERMISSIVE指定行访问控制策略为宽容性策略，宽容性策略的条件用OR表达式拼接。
* RESTRICTIVE指定行访问控制策略为限制性策略，限制性策略的条件用AND表达式拼接。拼接方式如下：

(using\_expression\_permissive\_1 OR using\_expression\_permissive\_2 ...) AND (using\_expression\_restrictive\_1 AND using\_expression\_restrictive\_2 ...)

* 缺省值为PERMISSIVE。
* **command**
* 当前行访问控制影响的SQL操作，可指定操作包括：ALL、SELECT、UPDATE、DELETE。当未指定时，ALL为默认值，涵盖SELECT、UPDATE、DELETE操作。
* 当command为SELECT时，SELECT类操作受行访问控制的影响，只能查看到满足条件(using\_expression返回值为TRUE)的元组数据，受影响的操作包括SELECT、UPDATE ... RETURNING、DELETE ... RETURNING。
* 当command为UPDATE时，UPDATE类操作受行访问控制的影响，只能更新满足条件(using\_expression返回值为TRUE)的元组数据，受影响的操作包括UPDATE、 UPDATE ... RETURNING、 SELECT ... FOR UPDATE/SHARE。
* 当command为DELETE时，DELETE类操作受行访问控制的影响，只能删除满足条件(using\_expression返回值为TRUE)的元组数据，受影响的操作包括DELETE、 DELETE ... RETURNING。
* 行访问控制策略与适配的SQL语法关系参加下表：
* **表 1** ROW LEVEL SECURITY策略与适配SQL语法关系

| **Command** | **SELECT/ALL policy** | **UPDATE/ALL policy** | **DELETE/ALL policy** |
| --- | --- | --- | --- |
| SELECT | Existing row | No | No |
| SELECT FOR UPDATE/SHARE | Existing row | Existing row | No |
| UPDATE | No | Existing row | No |
| UPDATE RETURNING | Existing row | Existing row | No |
| DELETE | No | No | Existing row |
| DELETE RETURNING | Existing row | No | Existing row |

* **role\_name**
* 行访问控制影响的数据库用户。
* 当未指定时，PUBLIC为默认值，PUBLIC表示影响所有数据库用户，可以指定多个受影响的数据库用户。
* fig: **须知：**   
  系统管理员不受行访问控制特性影响。
* **using\_expression**
* 行访问控制的表达式（返回boolean值）。
* 条件表达式中不能包含AGG函数和窗口（WINDOW）函数。在查询重写阶段，如果数据表的行访问控制开关打开，满足条件的表达式会添加到计划树中。针对数据表的每条元组，会进行表达式计算，只有表达式返回值为TRUE时，行数据对用户才可见（SELECT、UPDATE、DELETE）；当表达式返回FALSE时，该元组对当前用户不可见，用户无法通过SELECT语句查看此元组，无法通过UPDATE语句更新此元组，无法通过DELETE语句删除此元组。

**示例**

--创建用户alice  
vastbase=# CREATE USER alice PASSWORD 'xxxxxxxxx';  
  
--创建用户bob  
vastbase=# CREATE USER bob PASSWORD 'xxxxxxxxx';  
  
--创建数据表all\_data  
vastbase=# CREATE TABLE all\_data(id int, role varchar(100), data varchar(100));  
  
--向数据表插入数据  
vastbase=# INSERT INTO all\_data VALUES(1, 'alice', 'alice data');  
vastbase=# INSERT INTO all\_data VALUES(2, 'bob', 'bob data');  
vastbase=# INSERT INTO all\_data VALUES(3, 'peter', 'peter data');  
  
--将表all\_data的读取权限赋予alice和bob用户  
vastbase=# GRANT SELECT ON all\_data TO alice, bob;  
  
--打开行访问控制策略开关  
vastbase=# ALTER TABLE all\_data ENABLE ROW LEVEL SECURITY;  
  
--创建行访问控制策略，当前用户只能查看用户自身的数据  
vastbase=# CREATE ROW LEVEL SECURITY POLICY all\_data\_rls ON all\_data USING(role = CURRENT\_USER);  
  
--查看表all\_data相关信息  
vastbase=# d+ all\_data  
 Table "public.all\_data"  
 Column | Type | Modifiers | Storage | Stats target | Description  
--------+------------------------+-----------+----------+--------------+-------------  
 id | integer | | plain | |  
 role | character varying(100) | | extended | |  
 data | character varying(100) | | extended | |  
Row Level Security Policies:  
 POLICY "all\_data\_rls"  
 USING (((role)::name = "current\_user"()))  
Has OIDs: no  
Options: orientation=row, compression=no, enable\_rowsecurity=true  
  
--当前用户执行SELECT操作  
vastbase=# SELECT \* FROM all\_data;  
 id | role | data  
----+-------+------------  
 1 | alice | alice data  
 2 | bob | bob data  
 3 | peter | peter data  
(3 rows)  
  
vastbase=# EXPLAIN(COSTS OFF) SELECT \* FROM all\_data;  
 QUERY PLAN  
----------------------  
 Seq Scan on all\_data  
(1 row)  
  
--切换至alice用户执行SELECT操作  
vastbase=# SELECT \* FROM all\_data;  
 id | role | data  
----+-------+------------  
 1 | alice | alice data  
(1 row)  
  
vastbase=# EXPLAIN(COSTS OFF) SELECT \* FROM all\_data;  
 QUERY PLAN  
----------------------------------------------------------------  
 Seq Scan on all\_data  
 Filter: ((role)::name = 'alice'::name)  
 Notice: This query is influenced by row level security feature  
(3 rows)

**相关链接**

参考：SQL语法参考->SQL语法->DROP ROW LEVEL SECURITY POLICY、ALTER ROW LEVEL SECURITY POLICY章节。

### CREATE RULE

**功能描述**

定义一个新的重写规则。

**注意事项**

* 为了在表上定义或修改规则，你必须是该表的拥有者。
* 如果在同一个表定义了多个相同类型的规则，则按规则的名称字母顺序触发它们。
* 在视图上用于INSERT、UPDATE、DELETE的规则中可以添加RETURNING子句基于视图的字段返回。如果规则被INSERT RETURNING、UPDATE RETURNING、DELETE RETURNING命令触发，这些子句将用来计算输出结果。如果规则被不带RETURNING的命令触发，那么规则的RETURNING子句将被忽略。目前仅允许无条件的INSTEAD规则包含RETURNING子句，而且在同一个事件内的所有规则中最多只能有一个RETURNING子句。这样就确保只有一个RETURNING子句可以用于计算结果。如果在任何有效规则中都不存在RETURNING子句，该视图上的RETURNING查询将被拒绝。
* 不建议在rule内使用列存表，尤其是一些写操作。因为列存表与行存表的架构实现、事务处理等存在很大差异，因此rule的表现也会有很多与行存表不同的地方。

**语法格式**

CREATE [ OR REPLACE ] RULE name AS ON event  
 TO table\_name [ WHERE condition ]  
 DO [ ALSO | INSTEAD ] { NOTHING | command | ( command ; command ... ) }

其中event包含以下几种：

SELECT  
INSERT  
DELETE  
UPDATE

**参数说明**

* name
* 创建的规则名。它必须在同一个表上的所有规则名字中唯一。
* 取值范围：符合标识符命名规范的字符串，且最大长度不超过63个字符。
* table\_name
* 规则作用的表或者视图的名字（可以有模式修饰）。
* condition
* 返回boolean的SQL条件表达式，决定是否实际执行规则。表达式除了引用NEW和OLD之外不能引用任何表， 并且不能有聚合函数。
* INSTEAD
* INSTEAD指示使用该命令替换初始事件。
* ALSO
* ALSO指示该命令应该在初始事件执行之后执行。如果既没有声明ALSO也没有声明INSTEAD， 那么ALSO为缺省值。
* command
* 组成规则动作的命令。有效的命令是SELECT、 INSERT、UPDATE、 DELETE语句之一。

**示例**

CREATE RULE "\_RETURN" AS  
 ON SELECT TO t1  
 DO INSTEAD  
 SELECT \* FROM t2;

### CREATE SCHEMA

**功能描述**

创建模式。

访问命名对象时可以使用模式名作为前缀进行访问，如果无模式名前缀，则访问当前模式下的命名对象。创建命名对象时也可用模式名作为前缀修饰。

另外，CREATE SCHEMA可以包括在新模式中创建对象的子命令，这些子命令和那些在创建完模式后发出的命令没有任何区别。如果使用了AUTHORIZATION子句，则所有创建的对象都将被该用户所拥有。

**注意事项**

* 只要用户对当前数据库有CREATE权限，就可以创建模式。
* 系统管理员在普通用户同名schema下创建的对象，所有者为schema的同名用户（非系统管理员）。

**语法格式**

* 根据指定的名称创建模式。

CREATE SCHEMA schema\_name   
 [ AUTHORIZATION user\_name ] [WITH BLOCKCHAIN] [ schema\_element [ ... ] ];

* 根据用户名创建模式。

CREATE SCHEMA AUTHORIZATION user\_name [ schema\_element [ ... ] ];

**参数说明**

* **schema\_name**
* 模式名称。
* fig: **须知：**   
  模式名不能和当前数据库里其他的模式重名。  
  模式的名称不可以“pg\_”开头。
* 取值范围：字符串，要符合标识符的命名规范。
* **AUTHORIZATION user\_name**
* 指定模式的所有者。当不指定schema\_name时，把user\_name当作模式名，此时user\_name只能是角色名。
* 取值范围：已存在的用户名/角色名。
* **WITH BLOCKCHAIN**
* 指定模式的防篡改属性，防篡改模式下的行存普通用户表将自动扩展为防篡改用户表。
* **schema\_element**
* 在模式里创建对象的SQL语句。目前仅支持CREATE TABLE、CREATE VIEW、CREATE INDEX、CREATE PARTITION、CREATE SEQUENCE、CREATE TRIGGER、GRANT子句。
* 子命令所创建的对象都被AUTHORIZATION子句指定的用户所拥有。

fig: **说明：**   
如果当前搜索路径上的模式中存在同名对象时，需要明确指定引用对象所在的模式。可以通过命令SHOW SEARCH\_PATH来查看当前搜索路径上的模式。

**示例**

--创建一个角色role1。  
vastbase=# CREATE ROLE role1 IDENTIFIED BY 'xxxxxxxxx';  
  
-- 为用户role1创建一个同名schema，子命令创建的表films和winners的拥有者为role1。  
vastbase=# CREATE SCHEMA AUTHORIZATION role1  
 CREATE TABLE films (title text, release date, awards text[])   
 CREATE VIEW winners AS   
 SELECT title, release FROM films WHERE awards IS NOT NULL;  
  
--删除schema。  
vastbase=# DROP SCHEMA role1 CASCADE;  
--删除用户。  
vastbase=# DROP USER role1 CASCADE;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SCHEMA、DROP SCHEMA章节。

### CREATE SECURITY LABEL

**功能描述**

定义一个安全标签。

**注意事项**

无

**语法格式**

CREATE SECURITY LABEL label\_name 'label\_context';

**参数说明**

* **label\_name**

安全标签名称，创建时要求不能与已有标签重名。

取值范围：字符串，要符合标识符的命名规范。

* **abel\_context**

标签上下文。

**示例**

1、开启线程池，修改$PGDATA/postgresql.conf文件，并重启数据库。

enable\_thread\_pool = on

2、新建函数。

create or replace function stat\_parallel(out rec boolean, inout statements text) returns record

as $$

begin

begin

rec := true;

execute statements;

exception when others then rec := false;

end;

end;

$$ LANGUAGE plpgsql;

3、连接到vastbase的数据库连接池2，执行以下操作：

\parallel on 2

select stat\_parallel('create user user1 with password "user1@abcd";');

结果显示为：

stat\_parallel

------------------------------------------------------------------

(t,"create user user1 with password ""user1@abcd"";")

(1 row)

4、执行以下操作：

select stat\_parallel('create user user1 with password "user1@abcd";');

\parallel off

显示结果为：

stat\_parallel

-------------------------------------------------------------------------

(f,"create user user1 with password ""user1@abcd"";")

(1 row)

5、创建安全标签，并为安全标签指定属性。

create security label l1 'L3:G1,G2';

security label on role user1 is 'l1';

**相关链接**

无

### CREATE SEQUENCE

**功能描述**

CREATE SEQUENCE用于向当前数据库里增加一个新的序列。序列的Owner为创建此序列的用户。

**注意事项**

* Sequence是一个存放等差数列的特殊表。这个表没有实际意义，通常用于为行或者表生成唯一的标识符。
* 如果给出一个模式名，则该序列就在给定的模式中创建，否则会在当前模式中创建。序列名必须和同一个模式中的其他序列、表、索引、视图或外表的名称不同。
* 创建序列后，在表中使用序列的nextval()函数和generate\_series(1,N)函数对表插入数据，请保证nextval的可调用次数大于等于N+1次，否则会因为generate\_series()函数会调用N+1次而导致报错。
* Sequence默认最大值为263-1，如果使用了Large标识则最大值可以支持到2127-1。
* 被授予CREATE ANY SEQUENCE权限的用户，可以在public模式和用户模式下创建序列。

**语法格式**

CREATE [ LARGE ] SEQUENCE name [ INCREMENT [ BY ] increment ]  
 [ MINVALUE minvalue | NO MINVALUE | NOMINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE | NOMAXVALUE]   
 [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE | NOCYCLE ]   
 [ OWNED BY { table\_name.column\_name | NONE } ];

**参数说明**

* **name**
* 将要创建的序列名称。
* 取值范围: 仅可以使用小写字母（a~z）、 大写字母（A~Z）、数字和特殊字符“#”，“\_”，“$”的组合。
* **increment**
* 指定序列的步长。一个正数将生成一个递增的序列，一个负数将生成一个递减的序列。
* 缺省值为1。
* **MINVALUE minvalue | NO MINVALUE| NOMINVALUE**
* 执行序列的最小值。如果没有声明minvalue或者声明了NO MINVALUE，则递增序列的缺省值为1，递减序列的缺省值为-263-1。NOMINVALUE等价于NO MINVALUE。
* **MAXVALUE maxvalue | NO MAXVALUE| NOMAXVALUE**
* 执行序列的最大值。如果没有声明maxvalue或者声明了NO MAXVALUE，则递增序列的缺省值为263-1，递减序列的缺省值为-1。NOMAXVALUE等价于NO MAXVALUE。
* **start**
* 指定序列的起始值。缺省值：对于递增序列为minvalue，递减序列为maxvalue。
* **cache**
* 为了快速访问，而在内存中预先存储序列号的个数。
* 缺省值为1，表示一次只能生成一个值，也就是没有缓存。
* fig: **说明：**   
  不建议同时定义cache和maxvalue或minvalue。因为定义cache后不能保证序列的连续性，可能会产生空洞，造成序列号段浪费。
* **CYCLE**
* 用于使序列达到maxvalue或者minvalue后可循环并继续下去。
* 如果声明了NO CYCLE，则在序列达到其最大值后任何对nextval的调用都会返回一个错误。
* NOCYCLE的作用等价于NO CYCLE。
* 缺省值为NO CYCLE。
* 若定义序列为CYCLE，则不能保证序列的唯一性。
* **OWNED BY**
* 将序列和一个表的指定字段进行关联。这样，在删除那个字段或其所在表的时候会自动删除已关联的序列。关联的表和序列的所有者必须是同一个用户，并且在同一个模式中。需要注意的是，通过指定OWNED BY，仅仅是建立了表的对应列和sequence之间关联关系，并不会在插入数据时在该列上产生自增序列。
* 缺省值为OWNED BY NONE，表示不存在这样的关联。
* fig: **须知：**   
  通过OWNED BY创建的Sequence不建议用于其他表，如果希望多个表共享Sequence，该Sequence不应该从属于特定表。

**示例**

创建一个名为serial的递增序列，从101开始：

vastbase=# CREATE SEQUENCE serial  
 START 101  
 CACHE 20;

从序列中选出下一个数字：

vastbase=# SELECT nextval('serial');  
 nextval   
 ---------  
 101

从序列中选出下一个数字：

vastbase=# SELECT nextval('serial');  
 nextval   
 ---------  
 102

创建与表关联的序列：

vastbase=# CREATE TABLE customer\_address  
(  
 ca\_address\_sk integer not null,  
 ca\_address\_id char(16) not null,  
 ca\_street\_number char(10) ,  
 ca\_street\_name varchar(60) ,  
 ca\_street\_type char(15) ,  
 ca\_suite\_number char(10) ,  
 ca\_city varchar(60) ,  
 ca\_county varchar(30) ,  
 ca\_state char(2) ,  
 ca\_zip char(10) ,  
 ca\_country varchar(20) ,  
 ca\_gmt\_offset decimal(5,2) ,  
 ca\_location\_type char(20)   
);  
  
vastbase=# CREATE SEQUENCE serial1  
 START 101  
 CACHE 20  
OWNED BY customer\_address.ca\_address\_sk;  
--删除表和序列  
vastbase=# DROP TABLE customer\_address;  
vastbase=# DROP SEQUENCE serial cascade;  
vastbase=# DROP SEQUENCE serial1 cascade;

**相关链接**

参考：SQL语法参考->SQL语法->DROP SEQUENCE、ALTER SEQUENCE章节。

### CREATE SERVER

**功能描述**

定义一个新的外部服务器。

**语法格式**

CREATE SERVER server\_name  
 FOREIGN DATA WRAPPER fdw\_name  
 OPTIONS ( { option\_name ' value ' } [, ...] ) ;

**参数说明**

* **server\_name**
* server的名称。
* 取值范围：长度必须小于等于63。
* **fdw\_name**
* 指定外部数据封装器的名称。
* 取值范围：dist\_fdw，hdfs\_fdw，log\_fdw，file\_fdw，mot\_fdw，oracle\_fdw，mysql\_fdw，postgres\_fdw。
* **OPTIONS ( { option\_name ' value ' } [, ...] )**
* 这个子句为服务器指定选项。这些选项通常定义该服务器的连接细节，但是实际的名称和值取决于该服务器的外部数据包装器。
* oracle\_fdw支持的options包括：
* **dbserver**

远端Oracle数据库的连接字符串。

* **isolation\_level** （默认值为serializable）
  + - oracle数据库的事务隔离级别。
    - 取值范围：serializable、 read\_committed 、 read\_only
* mysql\_fdw支持的options包括：
* **host** （默认值为 127.0.0.1）
  + - MySQL Server/MariaDB的地址。
* **port** （默认值为 3306）
  + - MySQL Server/MariaDB侦听的端口号。
* postgres\_fdw支持的options同libpq支持的连接参数一致，如下表所示：

| **字符串** | **描述** |
| --- | --- |
| host | 要链接的主机名。如果主机名以斜杠开头，则它声明使用Unix域套接字通讯而不是TCP/IP通讯；该值就是套接字文件所存储的目录。如果没有声明host，那么默认是与位于/tmp目录（或者安装数据库的时候声明的套接字目录）里面的Unix-域套接字链接。在没有Unix域套接字的机器上，默认与localhost链接。 |
| hostaddr | 与之链接的主机的IP地址，是标准的IPv4地址格式，比如，172.28.40.9。如果机器支持IPv6，那么也可以使用IPv6的地址。如果声明了一个非空的字符串，那么使用TCP/IP通讯机制。使用hostaddr取代host可以让应用避免一次主机名查找，这一点对于那些有时间约束的应用来说可能是非常重要的。不过，GSSAPI或SSPI认证方法要求主机名（host）。因此，应用下面的规则：如果声明了不带hostaddr的host那么就强制进行主机名查找。如果声明中没有host，hostaddr的值给出服务器网络地址；如果认证方法要求主机名，那么链接尝试将失败。如果同时声明了host和hostaddr，那么hostaddr的值作为服务器网络地址。host的值将被忽略，除非认证方法需要它，在这种情况下它将被用作主机名。须知：要注意如果host不是网络地址hostaddr处的服务器名，那么认证很有可能失败。如果主机名（host）和主机地址都没有，那么libpq将使用一个本地的Unix域套接字进行链接；或者是在没有Unix域套接字的机器上，它将尝试与localhost链接。 |
| port | 主机服务器的端口号，或者在Unix域套接字链接时的套接字扩展文件名。 |
| user | 要链接的用户名，缺省是与运行该应用的用户操作系统名同名的用户。 |
| dbname | 数据库名，缺省和用户名相同。 |
| password | 如果服务器要求口令认证，所用的口令。 |
| connect\_timeout | 链接的最大等待时间，以秒计（用十进制整数字符串书写），0或者不声明表示无穷。不建议把链接超时的值设置得小于2秒。 |
| client\_encoding | 为这个链接设置client\_encoding配置参数。除了对应的服务器选项接受的值，你可以使用auto从客户端中的当前环境中确定正确的编码（Unix系统上是LC\_CTYPE环境变量）。 |
| options | 添加命令行选项以在运行时发送到服务器。 |
| application\_name | 为application\_name配置参数指定一个值，表明当前用户身份。 |
| keepalives | 控制客户端侧的TCP保持激活是否使用。缺省值是1，意思为打开，但是如果不想要保持激活，你可以更改为0，意思为关闭。通过Unix域套接字做的链接忽略这个参数。 |
| keepalives\_idle | 在TCP应该发送一个保持激活的信息给服务器之后，控制不活动的秒数。0值表示使用系统缺省。通过Unix域套接字做的链接或者如果禁用了保持激活则忽略这个参数。 |
| keepalives\_interval | 在TCP保持激活信息没有被应该传播的服务器承认之后，控制秒数。0值表示使用系统缺省。通过Unix域套接字做的链接或者如果禁用了保持激活则忽略这个参数。 |
| keepalives\_count | 添加命令行选项以在运行时发送到服务器。例如，设置为-c comm\_debug\_mode=off，则设置guc参数comm\_debug\_mode参数的会话的值为off。 |

需要注意的是，以下几个options不支持设置：

* **user**和**password**
  + - 用户名和密码将在创建user mapping时指定。
* **client\_encoding**
  + - 将自动获取本地server的编码方式并设置该值。
* **application\_name**
  + - 总是设置成postgres\_fdw。
* 用于指定外部服务器的各类参数，详细的参数说明如下所示。
* encrypt
  + - 是否对数据进行加密，该参数仅支持type为OBS时设置。默认值为on。
    - 取值范围：

on表示对数据进行加密，使用HTTPS协议通信。

off表示不对数据进行加密，使用HTTP协议通信。

* access\_key
  + - OBS访问协议对应的AK值（OBS云服务界面由用户获取），创建外表时AK值会加密保存到数据库的元数据表中。该参数仅支持type为OBS时设置。
* secret\_access\_key
  + - OBS访问协议对应的SK值（OBS云服务界面由用户获取），创建外表时SK值会加密保存到数据库的元数据表中。该参数仅支持type为OBS时设置。

除了libpq支持的连接参数外，还额外提供3个options：

* **use\_remote\_estimate**
* 控制postgres\_fdw是否发出EXPLAIN命令以获取运行消耗估算。默认值为false。
* **fdw\_startup\_cost**
* 执行一个外表扫描时的启动耗时估算。这个值通常包含建立连接、远端对请求的分析和生成计划的耗时。默认值为100。
* **fdw\_typle\_cost**
* 在远端服务器上对每一个元组进行扫描时的额外消耗。这个值通常表示数据在server间传输的额外消耗。默认值为0.01。

**示例**

创建server。

CREATE FOREIGN DATA WRAPPER extstats\_dummy\_fdw;  
CREATE SERVER extstats\_dummy\_srv FOREIGN DATA WRAPPER extstats\_dummy\_fdw;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SERVER、DROP SERVER章节。

### CREATE SUBSCRIPTION

**功能描述**

为当前数据库添加一个新的订阅。订阅表示到发布者的复制连接。

订阅名称必须与数据库中任何现有的订阅不同。 因此，此命令不仅在本地系统表中添加定义，还会在发布端创建复制槽。 在运行此命令的事务提交时，将启动逻辑复制线程以复制新订阅的数据。

**注意事项**

创建复制槽时（默认行为），CREATE SUBSCRIPTION不能在事务块内部执行。

**语法格式**

CREATE SUBSCRIPTION subscription\_name  
 CONNECTION 'conninfo'  
 PUBLICATION publication\_name [, ...]  
 [ WITH ( subscription\_parameter [= value] [, ... ] ) ]

**参数说明**

* **subscription\_name**
* 新订阅的名称。
* **CONNECTION 'conninfo'**
* 连接发布端的字符串。
* 如'host=1.1.1.1,2.2.2.2 port=10000,20000 dbname=postgres user=repusr1 password=password\_123'。
* **Host**

发布端IP地址，可以同时指定发布端主机和备机的IP地址，如果同时指定了多个IP，以英文逗号分隔。

* + **port**

发布端端口，此处的端口不能使用主端口，而应该使用主端口+1端口，否则会与线程池冲突。可以同时指定发布端主机和备机的端口，如果同时指定了多个端口，以英文逗号分隔。

fig: **注意:**   
host和port的数量要一致，并且要一一对应。

* **dbname**

发布所在的数据库。

* **user和password**

用于连接发布端且具有系统管理员权限(SYSADMIN)或者运维管理员权限(OPRADMIN)的用户名和密码。password需要加密，创建订阅前需要在订阅端执行vb\_guc generate -S xxxxxx -D $GAUSSHOME/bin -o subscription。

* **PUBLICATION publication\_name**
* 要订阅的发布端的发布名称，一个订阅可以对应多个发布。
* **WITH ( subscription\_parameter [= value] [, ... ] )**
* 该子句指定订阅的可选参数。支持的参数有：
* **copy\_data (boolean)**

指定在复制启动后是否应复制正在订阅的发布中的现有数据。默认值是true。

* **enabled (boolean)**

指定订阅是否应该主动复制，或者是否应该只是设置，但尚未启动。默认值是true。

* **slot\_name (string)**

要使用的复制插槽的名称。默认使用订阅名称作为复制槽的名称。

如果创建订阅时设置enable为false，则slot\_name将被强制设置为NONE，即空值，即使用户指定了slot\_name的值，表示复制槽不存在。

* **synchronous\_commit (enum)**

该参数的值会覆盖synchronous\_commit设置。 默认值是off。

对于逻辑复制使用off是安全的，如果订阅端由于缺少同步而丢失事务，数据将从发布者再次发送。进行同步逻辑复制时，一个不同的设置可能是合适的。逻辑复制线程向发布端报告写入和刷新的位置，当使用同步复制时，发布端将等待实际刷新。这意味着，当订阅用于同步复制时，将订阅者的synchronous\_commit设置为off可能会增加发布端服务器上COMMIT的延迟。在这种情况下，将synchronous\_commit设置为local或更高是有利的。

* **binary (boolean)**

该参数指定是否需要该订阅对应的发布端以二进制格式发送数据，为true表示需要以二进制发送，为false表示不以二进制格式而知以默认的文本格式发送。默认值false。

**示例**

1、对用于订阅的password进行加密。

vb\_guc generate -S "Bigdata@123" -D $GAUSSHOME/bin -o subscription

2、创建订阅。

* 创建一个到远程服务器的订阅，复制发布mypublication和insert\_only中的表，并在提交时立即开始复制。

CREATE SUBSCRIPTION mysub  
 CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb password=xxxx'  
 PUBLICATION mypublication, insert\_only;

* 创建一个到远程服务器的订阅，复制insert\_only发布中的表， 并且不开始复制直到稍后启用复制。

CREATE SUBSCRIPTION mysub  
 CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb password=xxxx '  
 PUBLICATION insert\_only  
 WITH (enabled = false);

3、修改订阅的连接信息。

ALTER SUBSCRIPTION mysub CONNECTION 'host=192.168.1.51 port=5432 user=foo dbname=foodb password=xxxx';

4、激活订阅。

ALTER SUBSCRIPTION mysub SET(enabled=true);

5、删除订阅。

DROP SUBSCRIPTION mysub;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SUBSCRIPTION、DROP SUBSCRIPTION章节。

### CREATE SYNONYM

**功能描述**

创建一个同义词对象。同义词是数据库对象的别名，用于记录与其他数据库对象名间的映射关系，用户可以使用同义词访问关联的数据库对象。

**注意事项**

* 定义同义词的用户成为其所有者。
* 若指定模式名称，则同义词在指定模式中创建。否则，在当前模式创建。
* 支持通过同义词访问的数据库对象包括：表、视图、函数和存储过程。
* 使用同义词时，用户需要具有对关联对象的相应权限。
* 支持使用同义词的DML语句包括：SELECT、INSERT、UPDATE、DELETE、EXPLAIN、CALL。
* 不建议对临时表创建同义词。如果需要创建的话，需要指定同义词的目标临时表的模式名，负责无法正常使用同义词，并且在当前会话结束前执行DROP SYNONYM命令。
* 删除原对象后，与之关联同义词不会被级联删除，继续访问该同义词会报错，并提示已失效。

**语法格式**

CREATE [ OR REPLACE ] SYNONYM synonym\_name   
 FOR object\_name;

**参数说明**

* **synonym**
* 创建的同义词名字，可以带模式名。
* 取值范围：字符串，要符合标识符的命名规范。
* **object\_name**
* 关联的对象名字，可以带模式名。
* 取值范围：字符串，要符合标识符的命名规范。  
  object\_name可以是不存在的对象名称。

**示例**

1. 创建模式ot。

\create schema ot;

1. 创建表ot.t1及其同义词t1。

CREATE TABLE ot.t1(id int, name varchar2(10));  
CREATE OR REPLACE SYNONYM t1 FOR ot.t1;

1. 使用同义词t1。

SELECT \* FROM t1;  
 INSERT INTO t1 VALUES (1, 'ada'), (2, 'bob');  
UPDATE t1 SET t1.name = 'cici' WHERE t1.id = 2;

1. 创建同义词v1及其关联视图ot.v\_t1。

CREATE SYNONYM v1 FOR ot.v\_t1;

CREATE VIEW ot.v\_t1 AS SELECT \* FROM ot.t1;

1. 使用同义词v1。

SELECT \* FROM v1;

6、创建重载函数ot.add及其同义词add。

CREATE OR REPLACE FUNCTION ot.add(a integer, b integer) RETURNS integer AS  
$$  
SELECT $1 + $2  
$$  
LANGUAGE sql;  
  
CREATE OR REPLACE FUNCTION ot.add(a decimal(5,2), b decimal(5,2)) RETURNS decimal(5,2) AS  
$$  
SELECT $1 + $2  
$$  
LANGUAGE sql;  
  
CREATE OR REPLACE SYNONYM add FOR ot.add;

7、使用同义词add。

SELECT add(1,2);

SELECT add(1.2,2.3);

8、创建存储过程ot.register及其同义词register。

CREATE PROCEDURE ot.register(n\_id integer, n\_name varchar2(10))  
SECURITY INVOKER  
AS  
BEGIN  
 INSERT INTO ot.t1 VALUES(n\_id, n\_name);  
END;  
/  
  
CREATE OR REPLACE SYNONYM register FOR ot.register;

9、使用同义词register，调用存储过程。

CALL register(3,'mia');

10、创建序列，并为序列创建同义词。

create table tab\_1100628(id int);

create sequence sequ1\_1100628 increment 1 maxvalue 200 start with 1 owned by tab\_1100628.id;

create synonym syn1\_1100628 for sequ1\_1100628;

11、创建调用同义词的函数。

create function fun\_1100628(n int)return int

as

a1 int;

begin

for i in 1..n loop

insert into tab\_1100628 value(syn1\_1100628.nextval);

end loop;

select count(\*) into a1 from tab\_1100628;

return a1;

end;

/

12、调用函数。

select fun\_1100628(3);

返回结果为：

id

----

2

3

4

(3 rows)

13、删除同义词。

DROP SYNONYM t1;  
DROP SYNONYM IF EXISTS v1;

DROP SYNONYM IF EXISTS add;

DROP SYNONYM register;

DROP SYNONYM syn1\_1100628;

DROP SCHEMA ot CASCADE;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SYNONYM、DROP SYNONYM章节。

### CREATE TABLE

**功能描述**

在当前数据库中创建一个新的空白表，该表由命令执行者所有。

**注意事项**

* 列存表支持的数据类型请参考列存表支持的数据类型。
* 列存表不支持数组。
* 列存表不支持生成列。
* 列存表不支持创建全局临时表。
* 创建列存表的数量建议不超过1000个。
* 表中的主键约束和唯一约束必须包含分布列。
* 如果在建表过程中数据库系统发生故障，系统恢复后可能无法自动清除之前已创建的、大小为0的磁盘文件。此种情况出现概率小，不影响数据库系统的正常运行。
* 列存表的表级约束只支持PARTIAL CLUSTER KEY，不支持主外键等表级约束。
* 列存表的字段约束只支持NULL、NOT NULL、DEFAULT常量值、UNIQUE和PRIMARY KEY。
* 列存表支持delta表，受参数enable\_delta\_store 控制是否开启，受参数deltarow\_threshold控制进入delta表的阀值。
* 使用JDBC时，支持通过PrepareStatement对DEFAUTL值进行参数化设置。
* 每张表的列数最大为1600，具体取决于列的类型，所有列的大小加起来不能超过8192 byte，text、varchar、char等长度可变的类型除外。
* 被授予CREATE ANY TABLE权限的用户，可以在public模式和用户模式下创建表。如果想要创建包含serial类型列的表，还需要授予CREATE ANY SEQUENCE创建序列的权限。
* 表名长度最多支持128字符。

**语法格式**

CREATE [ [ GLOBAL | LOCAL ] [ TEMPORARY | TEMP ] | UNLOGGED ] TABLE [ IF NOT EXISTS ] table\_name   
 ({ column\_name data\_type [ compress\_mode ] [ COLLATE collation ] [ column\_constraint [ ... ] ]  
 | table\_constraint  
 | LIKE source\_table [ like\_option [...] ] }  
 [, ... ])  
 [ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]  
 [ COMPRESS | NOCOMPRESS ]  
 [ TABLESPACE tablespace\_name ];

* 其中列约束column\_constraint为：

[ CONSTRAINT constraint\_name ]  
 { NOT NULL |  
 NULL |  
 CHECK ( expression ) |  
 DEFAULT default\_expr |  
 UNIQUE index\_parameters |  
 ENCRYPTED WITH ( COLUMN\_ENCRYPTION\_KEY = column\_encryption\_key, ENCRYPTION\_TYPE = encryption\_type\_value ) |  
 PRIMARY KEY index\_parameters |  
 REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]  
 [ ON DELETE action ] [ ON UPDATE action ] }  
 [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* 其中列的压缩可选项compress\_mode为：

{ DELTA | PREFIX | DICTIONARY | NUMSTR | NOCOMPRESS }

* 其中表约束table\_constraint为：

[ CONSTRAINT constraint\_name ]  
 { CHECK ( expression ) |  
 UNIQUE ( column\_name [, ... ] ) index\_parameters |  
 PRIMARY KEY ( column\_name [, ... ] ) index\_parameters |  
 FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ (refcolumn [, ... ] ) ]  
 [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] |  
 PARTIAL CLUSTER KEY ( column\_name [, ... ] ) }  
 [ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* 其中like选项like\_option为：

{ INCLUDING | EXCLUDING } { DEFAULTS | GENERATED | CONSTRAINTS | INDEXES | STORAGE | COMMENTS | PARTITION | RELOPTIONS | ALL }

* 其中索引参数index\_parameters为：

[ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ USING INDEX TABLESPACE tablespace\_name ]

* 创建虚拟列分区表语法如下：

CREATE TABLE [IF NOT EXISTS ] partition\_table\_name  
 ([{column\_name data\_type [ COLLATE\_COLLATION ] [column\_constraint [...]] | table\_constraint   
 | LIKE source\_table [ like\_option [...]]}  
 [,...]  
 [column\_name2 data\_type GENERATED ALWAYS AS (column\_expression) STORED] ])  
 [ WITH ( { storage\_parameter = value} [,...] ) ]  
 [ COMPRESS | NOCOMPRESS ]  
 [ TABLESPACE tablespace\_name ]  
 [ TO { GROUP groupname | NODE ( nodename [,...] ) } ]  
 PARTITION BY {  
 {VALUES (column\_name [,...] ) }   
 | {RANGE ( column\_name2 [,...] ) [INTERVAL ('interval\_expr') [STORE IN (tablespace\_name [,...])]]}  
 | { LIST ( column\_name2 [,...])}   
 | { HASH (column\_name2 [,...])}  
 }  
 [ SUBPARTITON BY [RANGE | LIST | HASH] (column\_name2 [,...])]  
 [{ SUBPARTITION TEMPLATE ( subpartition\_desc [,...])} | hash\_subpartitions\_by\_quantity]  
 table\_partitioning\_clauses  
 [ { ENABLE | DISABLE} ROW MOVEMENT];



column\_name2可以使用新增的虚拟列作为分区键

* 创建继承表语法如下：

CREATE TABLE table\_name[INHERITS(parent\_table)[,...]]



* 该功能仅对普通行存表支持，禁止分区表、ustore表、列存表、行压缩表、全局临时表或外部表使用继承功能。
* 不能对有继承关系的父表或者子表的约束做启用或禁用操作。
* 子表不能修改从父表继承的属性和约束。
* 子表存在的情况下，删除父表会报错，但是删除父表时使用cascade则可删除成功。
* 对父表的数据查询、数据修改或者模式修改的命令（SELECT、UPDATE、DELETE、大部分ALTER TABLE的变体，但是INSERT或者ALTER TABLE ...RENAME不在此范围内）默认会将子表包含在内，支持使用ONLY选项来排除子表。
* 父表上所有检查约束和非空约束都将自动被后代继承，其他类型的约束，比如：唯一约束，主键和外键约束则不会被继承。

**参数说明**

* UNLOGGED：如果指定此关键字，则创建的表为非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是非日志表在冲突、执行操作系统重启、强制重启、切断电源操作或异常关机后会被自动截断，会造成数据丢失的风险。非日志表中的内容也不会被复制到备服务器中。在非日志表中创建的索引也不会被自动记录。
* 使用场景：非日志表不能保证数据的安全性，用户应该在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。
* 故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。
* GLOBAL | LOCAL：创建临时表时可以在TEMP或TEMPORARY前指定GLOBAL或LOCAL关键字。如果指定GLOBAL关键字，Vastbase会创建全局临时表，否则Vastbase会创建本地临时表。
* TEMPORARY | TEMP：如果指定TEMP或TEMPORARY关键字，则创建的表为临时表。临时表分为全局临时表和本地临时表两种类型。创建临时表时如果指定GLOBAL关键字则为全局临时表，否则为本地临时表。
* 全局临时表的元数据对所有会话可见，会话结束后元数据继续存在。会话与会话之间的用户数据、索引和统计信息相互隔离，每个会话只能看到和更改自己提交的数据。全局临时表有两种模式：一种是基于会话级别的(ON COMMIT PRESERVE ROWS), 当会话结束时自动清空用户数据；一种是基于事务级别的(ON COMMIT DELETE ROWS), 当执行commit或rollback时自动清空用户数据。建表时如果没有指定ON COMMIT选项，则缺省为会话级别。与本地临时表不同，全局临时表建表时可以指定非pg\_temp\_开头的schema。
* 本地临时表只在当前会话可见，本会话结束后会自动删除。因此，在除当前会话连接的数据库节点故障时，仍然可以在当前会话上创建和使用临时表。由于临时表只在当前会话创建，对于涉及对临时表操作的DDL语句，会产生DDL失败的报错。因此，建议DDL语句中不要对临时表进行操作。TEMP和TEMPORARY等价。
* 本地临时表通过每个会话独立的以pg\_temp开头的schema来保证只对当前会话可见，因此，不建议用户在日常操作中手动删除以pg\_temp、pg\_toast\_temp开头的schema。
* 如果建表时不指定TEMPORARY/TEMP关键字，而指定表的schema为当前会话的pg\_temp\_开头的schema，则此表会被创建为临时表。
* ALTER/DROP全局临时表和索引，如果其它会话正在使用它，禁止操作（ALTER INDEX index\_name REBUILD除外）。
* 全局临时表的DDL只会影响当前会话的用户数据和索引。例如truncate、reindex、analyze只对当前会话有效。
* 全局临时表功能可以通过设置GUC参数max\_active\_global\_temporary\_table控制是否启用。如果max\_active\_global\_temporary\_table=0，关闭全局临时表功能。
* 临时表只对当前会话可见，因此不支持与parallel on并行执行一起使用。
* parallel on临时表不支持主备切换。
* IF NOT EXISTS：如果已经存在相同名称的表，不会报出错误，而会发出通知，告知通知此表已存在。
* table\_name：要创建的表名。



物化视图的一些处理逻辑会通过表名的前缀来识别是不是物化视图日志表和物化视图关联表，因此，用户不要创建表名以mlog或matviewmap为前缀的表，否则会影响此表的一些功能。

* column\_name：新表中要创建的字段名。
* data\_type：字段的数据类型。
* compress\_mode：表字段的压缩选项。该选项指定表字段优先使用的压缩算法。行存表不支持压缩。
* 取值范围：DELTA、PREFIX、DICTIONARY、NUMSTR、NOCOMPRESS
* COLLATE collation：COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。如果没有指定，则使用默认的排序规则。排序规则可以使用“select \* from pg\_collation;”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。
* LIKE source\_table [ like\_option … ]：LIKE子句声明一个表，新表自动从这个表中继承所有字段名及其数据类型和非空约束。新表与源表之间在创建动作完毕之后是完全无关的。在源表做的任何修改都不会传播到新表中，并且也不可能在扫描源表的时候包含新表的数据。被复制的列和约束并不使用相同的名称进行融合。如果明确的指定了相同的名称或者在另外一个LIKE子句中，将会报错。
* 源表上的字段缺省表达式只有在指定INCLUDING DEFAULTS时，才会复制到新表中。缺省是不包含缺省表达式的，即新表中的所有字段的缺省值都是NULL。
* 源表上的CHECK约束仅在指定INCLUDING CONSTRAINTS时，会复制到新表中，而其他类型的约束永远不会复制到新表中。非空约束总是复制到新表中。此规则同时适用于表约束和列约束。
* 如果指定了INCLUDING INDEXES，则源表上的索引也将在新表上创建，默认不建立索引。
* 如果指定了INCLUDING STORAGE，则复制列的STORAGE设置会复制到新表中，默认情况下不包含STORAGE设置。
* 如果指定了INCLUDING COMMENTS，则源表列、约束和索引的注释会复制到新表中。默认情况下，不复制源表的注释。
* 如果指定了INCLUDING PARTITION，则源表的分区定义会复制到新表中，同时新表将不能再使用PARTITION BY子句。默认情况下，不拷贝源表的分区定义。如果源表上带有索引，可以使用INCLUDING PARTITION INCLUDING INDEXES语法实现。如果对分区表只使用INCLUDING INDEXES，目标表定义将是普通表，但是索引是分区索引，最后结果会报错，因为普通表不支持分区索引。
* 如果指定了INCLUDING RELOPTIONS，则源表的存储参数（即源表的WITH子句）会复制到新表中。默认情况下，不复制源表的存储参数。
* INCLUDING ALL包含了INCLUDING DEFAULTS、INCLUDING CONSTRAINTS、INCLUDING INDEXES、INCLUDING STORAGE、INCLUDING COMMENTS、INCLUDING PARTITION和INCLUDING RELOPTIONS的内容。

**须知：**

* 如果源表包含serial、bigserial、smallserial、largeserial类型，或者源表字段的默认值是sequence，且sequence属于源表（通过CREATE SEQUENCE … OWNED BY创建），这些Sequence不会关联到新表中，新表中会重新创建属于自己的sequence。这和之前版本的处理逻辑不同。如果用户希望源表和新表共享Sequence，需要首先创建一个共享的Sequence（避免使用OWNED BY），并配置为源表字段默认值，这样创建的新表会和源表共享该Sequence。
* 不建议将其他表私有的Sequence配置为源表字段的默认值，尤其是其他表只分布在特定的NodeGroup上，这可能导致CREATE TABLE … LIKE执行失败。另外，如果源表配置其他表私有的Sequence，当该表删除时Sequence也会连带删除，这样源表的Sequence将不可用。如果用户希望多个表共享Sequence，建议创建共享的Sequence。
* 对于分区表EXCLUDING，需要配合INCLUDING ALL使用，如INCLUDING ALL EXCLUDING DEFAULTS，除源分区表的DEFAULTS，其它全包含。
* WITH ( { storage\_parameter = value } [, … ] )：这个子句为表或索引指定一个可选的存储参数。
* 使用任意精度类型Numeric定义列时，建议指定精度p以及刻度s。在不指定精度和刻度时，会按输入的显示出来。
* 参数的详细描述如下所示。
* FILLFACTOR：一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是最佳选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数对于列存表没有意义。

取值范围：10~100

* ORIENTATION：指定表数据的存储方式，即行存方式、列存方式，该参数设置成功后就不再支持修改。

取值范围：

* ROW，表示表的数据将以行式存储。
  + - 行存储适合于OLTP业务，适用于点查询或者增删操作较多的场景。
* COLUMN，表示表的数据将以列式存储。
  + - 列存储适合于数据仓库业务，此类型的表上会做大量的汇聚计算，且涉及的列操作较少。

默认值：若指定表空间为普通表空间，默认值为ROW。

* STORAGE\_TYPE：指定存储引擎类型，该参数设置成功后就不再支持修改。

取值范围：USTORE，表示表支持Inplace-Update存储引擎。ASTORE表、示表支持Append-Only存储引擎。

默认值：不指定表时，默认是Append-Only存储。

* INIT\_TD：创建Ustore表时，指定初始化的TD个数，该参数只在创建Ustore表时才能设置生效。
* 取值范围：2~128
* 默认值：4。
* COMPRESSION：指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比也越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。行存表默认增加COMPRESSION=NO字段。
* 取值范围：列存表的有效值为YES/NO/LOW/MIDDLE/HIGH 。
* 默认值：LOW。
* COMPRESSLEVEL：指定表数据同一压缩级别下的不同压缩水平，它决定了同一压缩级别下表数据的压缩比以及压缩时间。对同一压缩级别进行了更加详细的划分，为用户选择压缩比和压缩时间提供了更多的空间。总体来讲，此值越大，表示同一压缩级别下压缩比越大，压缩时间越长；反之亦然。
* 取值范围：0~3
* 默认值：0。
* COMPRESSTYPE：行存表参数，设置行存表压缩算法。1代表pglz算法，2代表zstd算法，默认不压缩。（仅支持ASTORE下的普通表）
* 取值范围：0~2
* 默认值：0。
* COMPRESS\_LEVEL：行存表参数，设置行存表压缩算法等级，仅当COMPRESSTYPE为2时生效。压缩等级越高，表的压缩效果越好，表的访问速度越慢。（仅支持ASTORE下的普通表）
* 取值范围：-31~31
* 默认值：0。
* COMPRESS\_CHUNK\_SIZE：行存表参数，设置行存表压缩chunk块大小。chunk数据块越小，预期能达到的压缩效果越好，同时数据越离散，影响表的访问速度。（仅支持ASTORE下的普通表）
* 取值范围：与页面大小有关。在页面大小为8k场景，取值范围为：512、1024、2048、4096。
* 默认值：4096
* COMPRESS\_PREALLOC\_CHUNKS：行存表参数，设置行存表压缩chunk块预分配数量。预分配数量越大，表的压缩率相对越差，离散度越小，访问性能越好。（仅支持ASTORE下的普通表）。
* 取值范围：0~7
* 默认值：0
* 当COMPRESS\_CHUNK\_SIZE为512和1024时，支持预分配设置最大为7。
* 当COMPRESS\_CHUNK\_SIZE为2048时，支持预分配设置最大为3。
* 当COMPRESS\_CHUNK\_SIZE为4096时，支持预分配设置最大为1。
* COMPRESS\_BYTE\_CONVERT：行存表参数，设置行存表压缩字节转换预处理。在一些场景下可以提升压缩效果，同时会导致一定性能劣化。
* 取值范围：布尔值
* 默认值：关闭。
* COMPRESS\_DIFF\_CONVERT：行存表参数，设置行存表压缩字节差分预处理。只能与compress\_byte\_convert一起使用。在一些场景下可以提升压缩效果，同时会导致一定性能劣化。
* 取值范围：布尔值
* 默认值：关闭。
* MAX\_BATCHROW：指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。
* 取值范围：10000~60000
* 默认值：60000。
* PARTIAL\_CLUSTER\_ROWS：指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。
* 取值范围：大于等于MAX\_BATCHROW，建议取值为MAX\_BATCHROW的整数倍。
* DELTAROW\_THRESHOLD：指定列存表导入时小于多少行的数据进入delta表，只在GUC参数enable\_delta\_store开启时生效。该参数只对列存表有效。
* 取值范围：0～9999
* 默认值：100
* VERSION：指定ORC存储格式的版本。
* 取值范围：0.12，目前支持ORC 0.12格式，后续会随着ORC格式的发展，支持更多格式。
* 默认值：0.12
* segment
* 使用段页式的方式存储。本参数仅支持行存表。不支持列存表、临时表、unlog表。不支持ustore存储引擎。
* 取值范围：on/off
* 默认值：off
* dek\_cipher
* 透明数据加密密钥的密文。当开启enable\_tde选项时会自动申请创建，用户不可单独指定。通过密钥轮转功能可以对密钥进行更新。
* 取值范围：字符串。
* 默认值：不开启加密时默认为空。
* hasuids
* 参数开启：更新表元组时，为元组分配表级唯一标识id。
* 取值范围：on/off。
* 默认值：off。
* ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP }：ON COMMIT选项决定在事务中执行创建临时表操作，当事务提交时，此临时表的后续操作。有以下三个选项，当前支持PRESERVE ROWS和DELETE ROWS选项。
* PRESERVE ROWS（缺省值）：提交时不对临时表做任何操作，临时表及其表数据保持不变。
* DELETE ROWS：提交时删除临时表中数据。
* DROP：提交时删除此临时表。只支持本地临时表，不支持全局临时表。
* COMPRESS | NOCOMPRESS：创建新表时，需要在CREATE TABLE语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。行存表不支持压缩。
* 缺省值：NOCOMPRESS，即不对元组数据进行压缩。
* TABLESPACE tablespace\_name：创建新表时指定此关键字，表示新表将要在指定表空间内创建。如果没有声明，将使用默认表空间。
* CONSTRAINT constraint\_name：列约束或表约束的名称。可选的约束子句用于声明约束，新行或者更新的行必须满足这些约束才能成功插入或更新。定义约束有两种方法：
* 列约束：作为一个列定义的一部分，仅影响该列。
* 表约束：不和某个列绑在一起，可以作用于多个列。
* NOT NULL：字段值不允许为NULL。
* NULL：字段值允许为NULL ，这是缺省值。这个子句只是为和非标准SQL数据库兼容。不建议使用。
* CHECK ( expression )：CHECK约束声明一个布尔表达式，每次要插入的新行或者要更新的行的新值必须使表达式结果为真或未知才能成功，否则会抛出一个异常并且不会修改数据库。声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。
* expression表达式中，如果存在“<>NULL”或“！=NULL”，这种写法是无效的，需要写成“is NOT NULL”。
* DEFAULT default\_expr：DEFAULT子句给字段指定缺省值。该数值可以是任何不含变量的表达式(不允许使用子查询和对本表中的其他字段的交叉引用)。缺省表达式的数据类型必须和字段类型匹配。缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。
* UNIQUE ( column\_name [, … ] ) index\_parameters：UNIQUE约束表示表里的一个字段或多个字段的组合必须在全表范围内唯一。对于唯一约束，NULL被认为是互不相等的。
* PRIMARY KEY ( column\_name [, … ] ) index\_parameters：主键约束声明表中的一个或者多个字段只能包含唯一的非NULL值。一个表只能声明一个主键。
* REFERENCES reftable [ ( refcolum ) ] [ MATCH matchtype ] [ ON DELETE action ][ ON UPDATE action ] (column constraint)
* FOREIGN KEY ( column\_name [, … ] ) REFERENCES reftable [ ( refcolumn [, … ] ) ] [ MATCH matchtype ] [ ON DELETE action ]\ [ ON UPDATE action ] (table constraint)：外键约束要求新表中一列或多列构成的组应该只包含、匹配被参考表中被参考字段值。若省略refcolum，则将使用reftable的主键。被参考列应该是被参考表中的唯一字段或主键。外键约束不能被定义在临时表和永久表之间。：参考字段与被参考字段之间存在三种类型匹配，分别是：
* MATCH FULL：不允许一个多字段外键的字段为NULL，除非全部外键字段都是NULL。
* MATCH SIMPLE（缺省）：允许任意外键字段为NULL。
* MATCH PARTIAL：目前暂不支持。
* 另外，当被参考表中的数据发生改变时，某些操作也会在新表对应字段的数据上执行。ON DELETE子句声明当被参考表中的被参考行被删除时要执行的操作。ON UPDATE子句声明当被参考表中的被参考字段数据更新时要执行的操作。对于ON DELETE子句、ON UPDATE子句的可能动作：
* NO ACTION（缺省）：删除或更新时，创建一个表明违反外键约束的错误。若约束可推迟，且若仍存在任何引用行，那这个错误将会在检查约束的时候产生。
* RESTRICT：删除或更新时，创建一个表明违反外键约束的错误。与NO ACTION相同，只是动作不可推迟。
* CASCADE：删除新表中任何引用了被删除行的行，或更新新表中引用行的字段值为被参考字段的新值。
* SET NULL：设置引用字段为NULL。
* SET DEFAULT：设置引用字段为它们的缺省值。
* DEFERRABLE | NOT DEFERRABLE：这两个关键字设置该约束是否可推迟。一个不可推迟的约束将在每条命令之后马上检查。可推迟约束可以推迟到事务结尾使用SET CONSTRAINTS命令检查。缺省是NOT DEFERRABLE。目前，UNIQUE约束、主键约束、外键约束可以接受这个子句。所有其他约束类型都是不可推迟的。

Ustore表不支持DEFERRABLE以及INITIALLY DEFERRED关键字。

* PARTIAL CLUSTER KEY：局部聚簇存储，列存表导入数据时按照指定的列(单列或多列)，进行局部排序。
* INITIALLY IMMEDIATE | INITIALLY DEFERRED：如果约束是可推迟的，则这个子句声明检查约束的缺省时间。
* 如果约束是INITIALLY IMMEDIATE（缺省），则在每条语句执行之后就立即检查它；
* 如果约束是INITIALLY DEFERRED ，则只有在事务结尾才检查它。
* 约束检查的时间可以用SET CONSTRAINTS命令修改。
* USING INDEX TABLESPACE tablespace\_name：为UNIQUE或PRIMARY KEY约束相关的索引声明一个表空间。如果没有提供这个子句，这个索引将在default\_tablespace中创建，如果default\_tablespace为空，将使用数据库的缺省表空间。
* ENCRYPTION\_TYPE = encryption\_type\_value：为ENCRYPTED WITH约束中的加密类型，encryption\_type\_value的值为[ DETERMINISTIC | RANDOMIZED ]

**示例**

**示例1：**创建简单的表。

CREATE TABLE warehouse\_t1   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例2：**创建表，并指定W\_STATE字段的缺省值为GA。

CREATE TABLE warehouse\_t2   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) DEFAULT 'GA',   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例3：**创建表，并在事务结束时检查W\_WAREHOUSE\_NAME字段是否有重复。

CREATE TABLE warehouse\_t3   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) UNIQUE DEFERRABLE,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例4：**创建一个带有70%填充因子的表。

CREATE TABLE warehouse\_t4   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 UNIQUE(W\_WAREHOUSE\_NAME) WITH(fillfactor=70)   
);

或者用下面的语法。

CREATE TABLE warehouse\_t5   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) UNIQUE,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) WITH(fillfactor=70);

**示例5：**创建表，并指定该表数据不写入预写日志。

CREATE UNLOGGED TABLE warehouse\_t6   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例6：**创建表临时表。

CREATE TEMPORARY TABLE warehouse\_t7   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例7：**事务中创建表临时表，并指定提交事务时删除该临时表数据。

CREATE TEMPORARY TABLE warehouse\_t8   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) ON COMMIT DELETE ROWS;

**示例8：**创建表时，不希望因为表已存在而报错。

CREATE TABLE IF NOT EXISTS warehouse\_t9   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例9：**创建带有自增列的表。

create table test\_b  
(  
id serial PRIMARY KEY,  
name character varying(64)   
);

**示例10：**创建普通表空间。

CREATE TABLESPACE DS\_TABLESPACE1 RELATIVE LOCATION 'tablespace/tablespace\_1';

**示例11：**创建表时，指定表空间。

CREATE TABLE warehouse\_t10   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) TABLESPACE DS\_TABLESPACE1;

**示例12：**创建表时，单独指定W\_WAREHOUSE\_NAME的索引表空间。

CREATE TABLE warehouse\_t11  
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) UNIQUE USING INDEX TABLESPACE DS\_TABLESPACE1,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例13：**创建一个有主键约束的表。

CREATE TABLE warehouse\_t12   
(   
 W\_WAREHOUSE\_SK INTEGER PRIMARY KEY,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

或是用下面的语法，效果完全一样。

CREATE TABLE warehouse\_t13  
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 PRIMARY KEY(W\_WAREHOUSE\_SK)   
);

或是用下面的语法，指定约束的名称。

CREATE TABLE warehouse\_t14   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 CONSTRAINT W\_CSTR\_KEY1 PRIMARY KEY(W\_WAREHOUSE\_SK)   
);

**示例14：**创建一个有复合主键约束的表。

CREATE TABLE warehouse\_t15  
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 CONSTRAINT W\_CSTR\_KEY2 PRIMARY KEY(W\_WAREHOUSE\_SK, W\_WAREHOUSE\_ID)   
);

**示例15：**创建列存表。

CREATE TABLE warehouse\_t16   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) WITH (ORIENTATION = COLUMN);

**示例16：**创建局部聚簇存储的列存表。

CREATE TABLE warehouse\_t17   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 PARTIAL CLUSTER KEY(W\_WAREHOUSE\_SK, W\_WAREHOUSE\_ID)   
) WITH (ORIENTATION = COLUMN);

**示例17：**定义一个带压缩的列存表。

CREATE TABLE warehouse\_t18   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) WITH (ORIENTATION = COLUMN, COMPRESSION=HIGH);

**示例18：**定义一个带压缩的表。

CREATE TABLE warehouse\_t19   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
) COMPRESS;

**示例19：**定义一个检查列约束。

CREATE TABLE warehouse\_t20   
(   
 W\_WAREHOUSE\_SK INTEGER PRIMARY KEY CHECK (W\_WAREHOUSE\_SK > 0),   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) CHECK (W\_WAREHOUSE\_NAME IS NOT NULL),   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

CREATE TABLE warehouse\_t21  
(   
 W\_WAREHOUSE\_SK INTEGER PRIMARY KEY,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) CHECK (W\_WAREHOUSE\_NAME IS NOT NULL),   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,  
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 CONSTRAINT W\_CONSTR\_KEY2 CHECK(W\_WAREHOUSE\_SK > 0 AND W\_WAREHOUSE\_NAME IS NOT NULL)   
);

**示例20：**定义一个表，表中每一个行存在数据库节点中。

CREATE TABLE warehouse\_t22   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2)   
);

**示例21：**定义一个表，使用HASH分布。

CREATE TABLE warehouse\_t23   
(   
 W\_WAREHOUSE\_SK INTEGER NOT NULL,   
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,   
 W\_WAREHOUSE\_NAME VARCHAR(20) ,   
 W\_WAREHOUSE\_SQ\_FT INTEGER ,   
 W\_STREET\_NUMBER CHAR(10) ,   
 W\_STREET\_NAME VARCHAR(60) ,   
 W\_STREET\_TYPE CHAR(15) ,   
 W\_SUITE\_NUMBER CHAR(10) ,   
 W\_CITY VARCHAR(60) ,   
 W\_COUNTY VARCHAR(30) ,   
 W\_STATE CHAR(2) ,   
 W\_ZIP CHAR(10) ,   
 W\_COUNTRY VARCHAR(20) ,   
 W\_GMT\_OFFSET DECIMAL(5,2),   
 CONSTRAINT W\_CONSTR\_KEY3 UNIQUE(W\_WAREHOUSE\_SK)   
);

**示例22：**使用ALTER TABLE修改表属性。

* 向warehouse\_t20表中增加一个varchar列。

ALTER TABLE warehouse\_t20 ADD W\_GOODS\_CATEGORY varchar(30);

* 给tpcds.warehouse\_t20表增加一个检查约束。

ALTER TABLE warehouse\_t20 ADD CONSTRAINT W\_CONSTR\_KEY4 CHECK (W\_STATE IS NOT NULL);

* 在一个操作中改变两个现存字段的类型。

ALTER TABLE warehouse\_t20   
ALTER COLUMN W\_GOODS\_CATEGORY TYPE varchar(80),   
ALTER COLUMN W\_STREET\_NAME TYPE varchar(100);

* 此语句与上面语句等效。

ALTER TABLE warehouse\_t20 MODIFY (W\_GOODS\_CATEGORY varchar(30), W\_STREET\_NAME varchar(60));

* 给一个已存在字段添加非空约束。

ALTER TABLE warehouse\_t20 ALTER COLUMN W\_GOODS\_CATEGORY SET NOT NULL;

* 移除已存在字段的非空约束。

ALTER TABLE warehouse\_t20 ALTER COLUMN W\_GOODS\_CATEGORY DROP NOT NULL;

* 如果列存表中还未指定局部聚簇，向在一个列存表中添加局部聚簇列。

ALTER TABLE warehouse\_t18 ADD PARTIAL CLUSTER KEY(W\_WAREHOUSE\_SK);

* 删除一个列存表中的局部聚簇列。

ALTER TABLE warehouse\_t18 DROP CONSTRAINT warehouse\_t18\_cluster;

* 将表移动到另一个表空间。

ALTER TABLE warehouse\_t20 SET TABLESPACE PG\_DEFAULT;

* 创建模式joe。

CREATE SCHEMA joe;

* 将表移动到另一个模式中。

ALTER TABLE warehouse\_t20 SET SCHEMA joe;

* 重命名已存在的表。

ALTER TABLE joe.warehouse\_t20 RENAME TO warehouse\_t23;

* 从warehouse\_t23表中删除一个字段。

ALTER TABLE joe.warehouse\_t23 DROP COLUMN W\_STREET\_NAME;

**示例23：**创建虚拟列表，虚拟列作为一级分区键。

CREATE TABLE t\_virtual  
 (  
 object\_id number,  
 object\_name varchar2(100),  
 created date,  
 create\_year int GENERATED ALWAYS AS (to\_number(to\_char(created,'MM')))STORED  
 )  
 partition by list(create\_year)  
 (  
 partition P1 VALUES (1),  
 partition P2 VALUES (2),  
 partition P3 VALUES (3),  
 partition P4 VALUES (4),  
 partition P5 VALUES (5)  
 );

**示例21：**创建继承表。

1、创建父表和子表。

create table parent(id int);

create table child1() inherits(parent);

create table child2() inherits(parent);

2、插入数据。

insert into parent values(1);

insert into child1 values(2);

insert into child2 values(3);

3、查询父表。

select \* from parent;

返回结果为：

id

---

1

2

3

(3 rows)

4、使用ONLY选项查询父表。

select \* from only parent;

返回结果为：

id

---

1

(1 rows)

### CREATE TABLE AS

**功能描述**

根据查询结果创建表。

CREATE TABLE AS创建一个表并且用来自SELECT命令的结果填充该表。该表的字段和SELECT输出字段的名称及数据类型相关。不过用户可以通过明确地给出一个字段名称列表来覆盖SELECT输出字段的名称。

CREATE TABLE AS对源表进行一次查询，然后将数据写入新表中，而查询视图结果会根据源表的变化而有所改变。相比之下，每次做查询的时候，视图都重新计算定义它的SELECT语句。

**注意事项**

* 分区表不能采用此方式进行创建。
* 如果在建表过程中数据库系统发生故障，系统恢复后可能无法自动清除之前已创建的、大小非0的磁盘文件。此种情况出现概率小，不影响数据库系统的正常运行。

**语法格式**

CREATE [ [ GLOBAL | LOCAL ] [ TEMPORARY | TEMP ] | UNLOGGED ] TABLE table\_name  
 [ (column\_name [, ...] ) ]  
 [ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]  
 [ COMPRESS | NOCOMPRESS ]  
 [ TABLESPACE tablespace\_name ]  
 AS query  
 [ WITH [ NO ] DATA ];

**参数说明**

* **UNLOGGED**
* 指定表为非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是，它也是不安全的，非日志表在冲突或异常关机后会被自动删截。非日志表中的内容也不会被复制到备用服务器中。在该类表中创建的索引也不会被自动记录。
* 使用场景：非日志表不能保证数据的安全性，用户应该在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。
* 故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。
* **GLOBAL | LOCAL**
* 创建临时表时可以在TEMP或TEMPORARY前指定GLOBAL或LOCAL关键字。如果指定GLOBAL关键字，Vastbase会创建全局临时表，否则Vastbase会创建本地临时表。
* **TEMPORARY | TEMP**
* 如果指定TEMP或TEMPORARY关键字，则创建的表为临时表。临时表分为全局临时表和本地临时表两种类型。创建临时表时如果指定GLOBAL关键字则为全局临时表，否则为本地临时表。
* 全局临时表的元数据对所有会话可见，会话结束后元数据继续存在。会话与会话之间的用户数据、索引和统计信息相互隔离，每个会话只能看到和更改自己提交的数据。全局临时表有两种模式：一种是基于会话级别的（ON COMMIT PRESERVE ROWS），当会话结束时自动清空用户数据；一种是基于事务级别的（ON COMMIT DELETE ROWS），当执行commit或rollback时自动清空用户数据。建表时如果没有指定ON COMMIT选项，则缺省为会话级别。与本地临时表不同，全局临时表建表时可以指定非pg\_temp\_开头的schema。
* 本地临时表只在当前会话可见，本会话结束后会自动删除。因此，在除当前会话连接的数据库节点故障时，仍然可以在当前会话上创建和使用临时表。由于临时表只在当前会话创建，对于涉及对临时表操作的DDL语句，会产生DDL失败的报错。因此，建议DDL语句中不要对临时表进行操作。TEMP和TEMPORARY等价。
* fig: **须知：**
* 本地临时表通过每个会话独立的以pg\_temp开头的schema来保证只对当前会话可见，因此，不建议用户在日常操作中手动删除以pg\_temp、pg\_toast\_temp开头的schema。
* 如果建表时不指定TEMPORARY/TEMP关键字，而指定表的schema为当前会话的pg\_temp\_开头的schema，则此表会被创建为临时表。
* ALTER/DROP全局临时表和索引，如果其它会话正在使用它，禁止操作。
* 全局临时表的DDL只会影响当前会话的用户数据和索引。例如truncate、reindex、analyze只对当前会话有效。
* **table\_name**
* 要创建的表名。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 新表中要创建的字段名。
* 取值范围：字符串，要符合标识符的命名规范。
* **WITH ( storage\_parameter [= value] [, ... ] )**
* 这个子句为表或索引指定一个可选的存储参数。参数的详细说明如下所示。
* FILLFACTOR
  + 一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是最佳选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数只对行存表有效。
  + 取值范围：10~100
* ORIENTATION
  + 取值范围：
  + COLUMN：表的数据将以列式存储。
  + ROW（缺省值）：表的数据将以行式存储。
* COMPRESSION
  + 指定表数据的压缩级别，它决定了表数据的压缩比以及压缩时间。一般来讲，压缩级别越高，压缩比也越大，压缩时间也越长；反之亦然。实际压缩比取决于加载的表数据的分布特征。
  + 取值范围：
  + 列存表的有效值为YES/NO/LOW/MIDDLE/HIGH，默认值为LOW。
  + 行存表不支持压缩。
* MAX\_BATCHROW
  + 指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。
  + 取值范围：10000~60000
* **ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP }**
* ON COMMIT选项决定在事务中执行创建临时表操作，当事务提交时，此临时表的后续操作。有以下三个选项，当前仅支持PRESERVE ROWS和DELETE ROWS选项。
* PRESERVE ROWS（缺省值）：提交时不对临时表执行任何操作，临时表及其表数据保持不变。
* DELETE ROWS：提交时删除临时表中数据。
* DROP：提交时删除此临时表。只支持删除本地临时表，不支持删除全局临时表。
* **COMPRESS / NOCOMPRESS**
* 创建一个新表时，需要在创建表语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。行存表不支持压缩。
* 缺省值：NOCOMPRESS，即不对元组数据进行压缩。
* **TABLESPACE tablespace\_name**
* 指定新表将要在tablespace\_name表空间内创建。如果没有声明，将使用默认表空间。
* **AS query**
* 一个SELECT VALUES命令或者一个运行预备好的SELECT或VALUES查询的EXECUTE命令。创建表时，是否也插入查询到的数据。默认是要数据，选择“NO”参数时，则不要数据。

**示例**

--创建一个表tpcds.store\_returns表。  
vastbase=# CREATE TABLE tpcds.store\_returns  
(  
 W\_WAREHOUSE\_SK INTEGER NOT NULL,  
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,  
 sr\_item\_sk VARCHAR(20) ,  
 W\_WAREHOUSE\_SQ\_FT INTEGER   
);  
--创建一个表tpcds.store\_returns\_t1并插入tpcds.store\_returns表中sr\_item\_sk字段中大于16的数值。  
vastbase=# CREATE TABLE tpcds.store\_returns\_t1 AS SELECT \* FROM tpcds.store\_returns WHERE sr\_item\_sk > '4795';  
  
--使用tpcds.store\_returns拷贝一个新表tpcds.store\_returns\_t2。  
vastbase=# CREATE TABLE tpcds.store\_returns\_t2 AS table tpcds.store\_returns;  
  
--删除表。  
vastbase=# DROP TABLE tpcds.store\_returns\_t1 ;  
vastbase=# DROP TABLE tpcds.store\_returns\_t2 ;  
vastbase=# DROP TABLE tpcds.store\_returns;

**相关链接**

参考：SQL语法参考->SQL语法->CREATE TABLE、SELECT章节。

### CREATE TABLE PARTITION

**功能描述**

创建分区表。分区表是把逻辑上的一张表根据某种方案分成几张物理块进行存储，这张逻辑上的表称之为分区表，物理块称之为分区。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。

常见的分区方案有范围分区（Range Partitioning）、间隔分区（Interval Partitioning）、哈希分区（Hash Partitioning）、列表分区（List Partitioning）、数值分区（Value Partition）等。目前行存表支持范围分区、间隔分区、哈希分区、列表分区，列存表仅支持范围分区。

范围分区是根据表的一列或者多列，将要插入表的记录分为若干个范围，这些范围在不同的分区里没有重叠。为每个范围创建一个分区，用来存储相应的数据。

范围分区的分区策略是指记录插入分区的方式。目前范围分区仅支持范围分区策略。

范围分区策略：根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息。这是最常用的分区策略。

间隔分区是一种特殊的范围分区，相比范围分区，新增间隔值定义，当插入记录找不到匹配的分区时，可以根据间隔值自动创建分区。

间隔分区只支持基于表的一列分区，并且该列只支持TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE数据类型。

间隔分区策略：根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则根据分区键值和表定义信息自动创建一个分区，然后将记录插入新分区中，新创建的分区数据范围等于间隔值。

哈希分区是根据表的一列，为每个分区指定模数和余数，将要插入表的记录划分到对应的分区中，每个分区所持有的行都需要满足条件：分区键的值除以为其指定的模数将产生为其指定的余数。

哈希分区策略：根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则返回报错和提示信息。

列表分区是根据表的一列，将要插入表的记录通过每一个分区中出现的键值划分到对应的分区中，这些键值在不同的分区里没有重叠。为每组键值创建一个分区，用来存储相应的数据。

列表分区策略：根据分区键值将记录映射到已创建的某个分区上，如果可以映射到已创建的某一分区上，则把记录插入到对应的分区上，否则给出报错和提示信息。

分区可以提供若干好处：

* 某些类型的查询性能可以得到极大提升。特别是表中访问率较高的行位于一个单独分区或少数几个分区上的情况下。分区可以减少数据的搜索空间，提高数据访问效率。
* 当查询或更新一个分区的大部分记录时，连续扫描那个分区而不是访问整个表可以获得巨大的性能提升。
* 如果需要大量加载或者删除的记录位于单独的分区上，则可以通过直接读取或删除那个分区以获得巨大的性能提升，同时还可以避免由于大量DELETE导致的VACUUM超载（仅范围分区）。

**注意事项**

* 唯一约束和主键约束的约束键包含所有分区键将为约束创建LOCAL索引，否则创建GLOBAL索引。
* 目前哈希分区和列表分区仅支持单列构建分区键，暂不支持多列构建分区键。
* 只需要有间隔分区表的INSERT权限，往该表INSERT数据时就可以自动创建分区。
* 对于分区表PARTITION FOR (values)语法，values只能是常量。
* 对于分区表PARTITION FOR (values)语法，values在需要数据类型转换时，建议使用强制类型转换，以防隐式类型转换结果与预期不符。
* 分区数最大值为1048575个，一般情况下业务不可能创建这么多分区，这样会导致内存不足。应参照参数local\_syscache\_threshold的值合理创建分区，分区表使用内存大致为（分区数 \* 3 / 1024）MB。理论上分区占用内存不允许大于local\_syscache\_threshold的值，同时还需要预留部分空间以供其他功能使用。
* 指定分区语句目前不能走全局索引扫描。

**语法格式**

CREATE TABLE [ IF NOT EXISTS ] partition\_table\_name  
( [   
 { column\_name data\_type [ COLLATE collation ] [ column\_constraint [ ... ] ]  
 | table\_constraint  
 | LIKE source\_table [ like\_option [...] ] }[, ... ]  
] )  
 [ WITH ( {storage\_parameter = value} [, ... ] ) ]  
 [ COMPRESS | NOCOMPRESS ]  
 [ TABLESPACE tablespace\_name ]  
 PARTITION BY {   
 {RANGE (partition\_key) [ INTERVAL ('interval\_expr') [ STORE IN (tablespace\_name [, ... ] ) ] ] ( partition\_less\_than\_item [, ... ] )} |  
 {RANGE (partition\_key) [ INTERVAL ('interval\_expr') [ STORE IN (tablespace\_name [, ... ] ) ] ] ( partition\_start\_end\_item [, ... ] )} |  
 {LIST | HASH (partition\_key) (PARTITION partition\_name [VALUES (list\_values\_clause)] opt\_table\_space )}  
 } [ { ENABLE | DISABLE } ROW MOVEMENT ];

* 列约束column\_constraint：

[ CONSTRAINT constraint\_name ]  
{ NOT NULL |  
 NULL |   
 CHECK ( expression ) |   
 DEFAULT default\_e xpr |   
 GENERATED ALWAYS AS ( generation\_expr ) STORED |  
 UNIQUE index\_parameters |   
 PRIMARY KEY index\_parameters |  
 REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]  
 [ ON DELETE action ] [ ON UPDATE action ] }  
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* 表约束table\_constraint：

[ CONSTRAINT constraint\_name ]  
{ CHECK ( expression ) |   
 UNIQUE ( column\_name [, ... ] ) index\_parameters |   
 PRIMARY KEY ( column\_name [, ... ] ) index\_parameters |  
 FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]  
 [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] }  
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* like选项like\_option：

{ INCLUDING | EXCLUDING } { DEFAULTS | GENERATED | CONSTRAINTS | INDEXES | STORAGE | COMMENTS | RELOPTIONS| ALL }

* 索引存储参数index\_parameters：

[ WITH ( {storage\_parameter = value} [, ... ] ) ]  
[ USING INDEX TABLESPACE tablespace\_name ]

* partition\_less\_than\_item：

PARTITION partition\_name VALUES LESS THAN ( { partition\_value | MAXVALUE } ) [TABLESPACE tablespace\_name]

* partition\_start\_end\_item：

PARTITION partition\_name {  
 {START(partition\_value) END (partition\_value) EVERY (interval\_value)} |  
 {START(partition\_value) END ({partition\_value | MAXVALUE})} |  
 {START(partition\_value)} |  
 {END({partition\_value | MAXVALUE})}  
 } [TABLESPACE tablespace\_name]

**参数说明**

* **IF NOT EXISTS**
* 如果已经存在相同名称的表，不会抛出一个错误，而会发出一个通知，告知表关系已存在。
* **partition\_table\_name**
* 分区表的名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 新表中要创建的字段名。
* 取值范围：字符串，要符合标识符的命名规范。
* **data\_type**
* 字段的数据类型。
* **COLLATE collation**
* COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。如果没有指定，则使用默认的排序规则。排序规则可以使用“select \* from pg\_collation;”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。
* **CONSTRAINT constraint\_name**
* 列约束或表约束的名称。可选的约束子句用于声明约束，新行或者更新的行必须满足这些约束才能成功插入或更新。
* 定义约束有两种方法：
* 列约束：作为一个列定义的一部分，仅影响该列。
* 表约束：不和某个列绑在一起，可以作用于多个列。
* **LIKE source\_table [ like\_option ... ]**
* LIKE子句声明一个表，新表自动从这个表里面继承所有字段名及其数据类型和非空约束。
* 和INHERITS不同，新表与原来的表之间在创建动作完毕之后是完全无关的。在源表做的任何修改都不会传播到新表中，并且也不可能在扫描源表的时候包含新表的数据。
* 字段缺省表达式只有在声明了INCLUDING DEFAULTS之后才会包含进来。缺省是不包含缺省表达式的，即新表中所有字段的缺省值都是NULL。
* 如果指定了INCLUDING GENERATED，则源表列的生成表达式会复制到新表中。默认不复制生成表达式。
* 非空约束将总是复制到新表中，CHECK约束则仅在指定了INCLUDING CONSTRAINTS的时候才复制，而其他类型的约束则永远也不会被复制。此规则同时适用于表约束和列约束。
* 和INHERITS不同，被复制的列和约束并不使用相同的名称进行融合。如果明确的指定了相同的名称或者在另外一个LIKE子句中，将会报错。
* 如果指定了INCLUDING INDEXES，则源表上的索引也将在新表上创建，默认不建立索引。
* 如果指定了INCLUDING STORAGE，则拷贝列的STORAGE设置也将被拷贝，默认情况下不包含STORAGE设置。
* 如果指定了INCLUDING COMMENTS，则源表列、约束和索引的注释也会被拷贝过来。默认情况下，不拷贝源表的注释。
* 如果指定了INCLUDING RELOPTIONS，则源表的存储参数（即源表的WITH子句）也将拷贝至新表。默认情况下，不拷贝源表的存储参数。
* INCLUDING ALL包含了INCLUDING DEFAULTS、INCLUDING CONSTRAINTS、INCLUDING INDEXES、INCLUDING STORAGE、INCLUDING COMMENTS、INCLUDING PARTITION和INCLUDING RELOPTIONS的内容。
* **WITH ( storage\_parameter [= value] [, ... ] )**
* 这个子句为表或索引指定一个可选的存储参数。参数的详细描述如下所示：
* FILLFACTOR

一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是最佳选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数对于列存表没有意义。

取值范围：10~100

* ORIENTATION

决定了表的数据的存储方式。

取值范围：

* COLUMN：表的数据将以列式存储。
* ROW（缺省值）：表的数据将以行式存储。

fig: **须知：**   
orientation不支持修改。

* STORAGE\_TYPE

指定存储引擎类型，该参数设置成功后就不再支持修改。

取值范围：

* USTORE，表示表支持Inplace-Update存储引擎。特别需要注意，使用USTORE表，必须要开启track\_counts和track\_activities参数，否则会引起空间膨胀。
* ASTORE，表示表支持Append-Only存储引擎。

默认值：不指定表时，默认是Append-Only存储。

* COMPRESSION
* 列存表的有效值为LOW/MIDDLE/HIGH/YES/NO，压缩级别依次升高，默认值为LOW。
* 行存表不支持压缩。
* MAX\_BATCHROW

指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。

取值范围：10000~60000，默认60000。

* PARTIAL\_CLUSTER\_ROWS

指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。

取值范围：大于等于MAX\_BATCHROW，建议取值为MAX\_BATCHROW的整数倍数。

* DELTAROW\_THRESHOLD

预留参数。该参数只对列存表有效。

取值范围：0～9999

* segment

使用段页式的方式存储。本参数仅支持行存表。不支持列存表、临时表、unlog表。不支持ustore存储引擎。

取值范围：on/off

默认值：off

* **COMPRESS / NOCOMPRESS**
* 创建一个新表时，需要在创建表语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。行存表不支持压缩。
* 缺省值为NOCOMPRESS，即不对元组数据进行压缩。
* **TABLESPACE tablespace\_name**
* 指定新表将要在tablespace\_name表空间内创建。如果没有声明，将使用默认表空间。
* **PARTITION BY RANGE(partition\_key)**
* 创建范围分区。partition\_key为分区键的名称。
* 1、对于从句是VALUES LESS THAN的语法格式：
* fig: **须知：**   
  对于从句是VALUE LESS THAN的语法格式，范围分区策略的分区键最多支持4列。
* 该情形下，分区键支持的数据类型为：SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE PRECISION、CHARACTER VARYING(n)、VARCHAR(n)、CHARACTER(n)、CHAR(n)、CHARACTER、CHAR、TEXT、NVARCHAR、NVARCHAR2、NAME、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。
* 2、对于从句是START END的语法格式：
* fig: **须知：**   
  对于从句是START END的语法格式，范围分区策略的分区键仅支持1列。
* 该情形下，分区键支持的数据类型为：SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE PRECISION、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。
* 3、对于指定了INTERVAL子句的语法格式：
* fig: **须知：**   
  对于指定了INTERVAL子句的语法格式，范围分区策略的分区键仅支持1列。
* 该情形下，分区键支持的数据类型为：TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。
* **PARTITION partition\_name VALUES LESS THAN ( { partition\_value | MAXVALUE } )**
* 指定各分区的信息。partition\_name为范围分区的名称。partition\_value为范围分区的上边界，取值依赖于partition\_key的类型。MAXVALUE表示分区的上边界，它通常用于设置最后一个范围分区的上边界。
* fig: **须知：**
* 每个分区都需要指定一个上边界。
* 分区上边界的类型应当和分区键的类型一致。
* 分区列表是按照分区上边界升序排列的，值较小的分区位于值较大的分区之前。
* **PARTITION partition\_name {START (partition\_value) END (partition\_value) EVERY (interval\_value)} | {START (partition\_value) END (partition\_value|MAXVALUE)} | {START(partition\_value)} | {END (partition\_value | MAXVALUE)}**
* 指定各分区的信息，各参数意义如下：
* partition\_name：范围分区的名称或名称前缀，除以下情形外（假定其中的partition\_name是p1），均为分区的名称。
* 若该定义是START+END+EVERY从句，则语义上定义的分区的名称依次为p1\_1, p1\_2, ...。例如对于定义“PARTITION p1 START(1) END(4) EVERY(1)”，则生成的分区是：[1, 2), [2, 3) 和 [3, 4)，名称依次为p1\_1, p1\_2和p1\_3，即此处的p1是名称前缀。
* 若该定义是第一个分区定义，且该定义有START值，则范围（MINVALUE, START）将自动作为第一个实际分区，其名称为p1\_0，然后该定义语义描述的分区名称依次为p1\_1, p1\_2, ...。例如对于完整定义“PARTITION p1 START(1), PARTITION p2 START(2)”，则生成的分区是：(MINVALUE, 1), [1, 2) 和 [2, MAXVALUE)，其名称依次为p1\_0, p1\_1和p2，即此处p1是名称前缀，p2是分区名称。这里MINVALUE表示最小值。
* partition\_value：范围分区的端点值（起始或终点），取值依赖于partition\_key的类型，不可是MAXVALUE。
* interval\_value：对[START，END) 表示的范围进行切分，interval\_value是指定切分后每个分区的宽度，不可是MAXVALUE；如果（END-START）值不能整除以EVERY值，则仅最后一个分区的宽度小于EVERY值。
* MAXVALUE：表示最大值，它通常用于设置最后一个范围分区的上边界。
* fig: **须知：**

1、在创建分区表若第一个分区定义含START值，则范围（MINVALUE，START）将自动作为 >实际的第一个分区。

2、START END语法需要遵循以下限制：

* 每个partition\_start\_end\_item中的START值（如果有的话，下同）必须小于 >其END值。
* 相邻的两个partition\_start\_end\_item，第一个的END值必须等于第二个的 >START值；
* 每个partition\_start\_end\_item中的EVERY值必须是正向递增的，且必须小于（END-START）值；
* 每个分区包含起始值，不包含终点值，即形如：[起始值，终点值)，起始值是MINVALUE时则不包含；
* 一个partition\_start\_end\_item创建的每个分区所属的TABLESPACE一样；
* partition\_name作为分区名称前缀时，其长度不要超过57字节，超过时自动截断；
* 在创建、修改分区表时请注意分区表的分区总数不可超过最大限制（1048575）；

3、在创建分区表时START END与LESS THAN语法不可混合使用。

4、即使创建分区表时使用START END语法，备份（vb\_dump）出的SQL语句也是VALUES LESS THAN语法格式。

* **INTERVAL ('interval\_expr') [ STORE IN (tablespace\_name [, ... ] ) ]**
* 间隔分区定义信息。
* interval\_expr：自动创建分区的间隔，例如：1 day、1 month。
* STORE IN (tablespace\_name [, ... ] )：指定存放自动创建分区的表空间列表，如果有指定，则自动创建的分区从表空间列表中循环选择使用，否则使用分区表默认的表空间。
* fig: **须知：**   
  列存表不支持间隔分区。
* **PARTITION BY LIST(partition\_key)**
* 创建列表分区。partition\_key为分区键的名称。
* 对于partition\_key，列表分区策略的分区键仅支持1列。
* 对于从句是VALUES (list\_values\_clause)的语法格式，list\_values\_clause中包含了对应分区存在的键值，推荐每个分区的键值数量不超过127个。
* 分区键支持的数据类型为：INT1、INT2、INT4、INT8、NUMERIC、VARCHAR(n)、CHAR、BPCHAR、NVARCHAR、NVARCHAR2、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。分区个数不能超过 1048575 个。
* **PARTITION BY HASH(partition\_key)**
* 创建哈希分区。partition\_key为分区键的名称。
* 对于partition\_key，哈希分区策略的分区键仅支持1列。
* 分区键支持的数据类型为：INT1、INT2、INT4、INT8、NUMERIC、VARCHAR(n)、CHAR、BPCHAR、TEXT、NVARCHAR、NVARCHAR2、TIMESTAMP[(p)] [WITHOUT TIME ZONE]、TIMESTAMP[(p)] [WITH TIME ZONE]、DATE。分区个数不能超过1048575 个。
* **{ ENABLE | DISABLE } ROW MOVEMENT**
* 行迁移开关。
* 如果进行UPDATE操作时，更新了元组在分区键上的值，造成了该元组所在分区发生变化，就会根据该开关给出报错信息，或者进行元组在分区间的转移。
* 取值范围：
* ENABLE（缺省值）：行迁移开关打开。
* DISABLE：行迁移开关关闭。
* fig: **须知：**   
  列表/哈希分区表暂不支持ROW MOVEMENT。
* **NOT NULL**
* 字段值不允许为NULL。ENABLE用于语法兼容，可省略。
* **NULL**
* 字段值允许NULL ，这是缺省。
* 这个子句只是为和非标准SQL数据库兼容。不建议使用。
* **CHECK (condition) [ NO INHERIT ]**
* CHECK约束声明一个布尔表达式，每次要插入的新行或者要更新的行的新值必须使表达式结果为真或未知才能成功，否则会抛出一个异常并且不会修改数据库。
* 声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。
* 用NO INHERIT标记的约束将不会传递到子表中去。
* ENABLE用于语法兼容，可省略。
* **DEFAULT default\_expr**
* DEFAULT子句给字段指定缺省值。该数值可以是任何不含变量的表达式(不允许使用子查询和对本表中的其他字段的交叉引用)。缺省表达式的数据类型必须和字段类型匹配。
* 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。
* **GENERATED ALWAYS AS ( generation\_expr ) STORED**
* 该子句将字段创建为生成列，生成列的值在写入（插入或更新）数据时由generation\_expr计算得到，STORED表示像普通列一样存储生成列的值。
* fig: **说明：**
* 生成表达式不能以任何方式引用当前行以外的其他数据。生成表达式不能引用其他生成列，不能引用系统列。生成表达式不能返回结果集，不能使用子查询，不能使用聚集函数，不能使用窗口函数。生成表达式调用的函数只能是不可变（IMMUTABLE）函数。
* 不能为生成列指定默认值。
* 生成列不能作为分区键的一部分。
* 生成列不能和ON UPDATE约束字句的CASCADE,SET NULL,SET DEFAULT动作同时指定。生成列不能和ON DELETE约束字句的SET NULL、SET DEFAULT动作同时指定。
* 修改和删除生成列的方法和普通列相同。删除生成列依赖的普通列，生成列被自动删除。不能改变生成列所依赖的列的类型。
* 生成列不能被直接写入。在INSERT或UPDATE命令中, 不能为生成列指定值, 但是可以指定关键字DEFAULT。
* 生成列的权限控制和普通列一样。
* 列存表、内存表MOT不支持生成列。外表中仅postgres\_fdw支持生成列。
* **UNIQUE index\_parameters**
* **UNIQUE ( column\_name [, ... ] ) index\_parameters**
* UNIQUE约束表示表里的一个字段或多个字段的组合必须在全表范围内唯一。
* 对于唯一约束，NULL被认为是互不相等的。
* **PRIMARY KEY index\_parameters**
* **PRIMARY KEY ( column\_name [, ... ] ) index\_parameters**
* 主键约束声明表中的一个或者多个字段只能包含唯一的非NULL值。
* 一个表只能声明一个主键。
* **DEFERRABLE | NOT DEFERRABLE**
* 这两个关键字设置该约束是否可推迟。一个不可推迟的约束将在每条命令之后马上检查。可推迟约束可以推迟到事务结尾使用SET CONSTRAINTS命令检查。缺省是NOT DEFERRABLE。目前，UNIQUE约束、主键约束、外键约束可以接受这个子句。所有其他约束类型都是不可推迟的。
* **INITIALLY IMMEDIATE | INITIALLY DEFERRED**
* 如果约束是可推迟的，则这个子句声明检查约束的缺省时间。
* 如果约束是INITIALLY IMMEDIATE（缺省），则在每条语句执行之后就立即检查它；
* 如果约束是INITIALLY DEFERRED ，则只有在事务结尾才检查它。
* 约束检查的时间可以用SET CONSTRAINTS命令修改。
* **USING INDEX TABLESPACE tablespace\_name**
* 为UNIQUE或PRIMARY KEY约束相关的索引声明一个表空间。如果没有提供这个子句，这个索引将在default\_tablespace中创建，如果default\_tablespace为空，将使用数据库的缺省表空间。

**示例**

* 示例1：创建范围分区表tpcds.web\_returns\_p1，含有8个分区，分区键为integer类型。 分区的范围分别为：wr\_returned\_date\_sk< 2450815、2450815<= wr\_returned\_date\_sk< 2451179、2451179<=wr\_returned\_date\_sk< 2451544、2451544 <= wr\_returned\_date\_sk< 2451910、2451910 <= wr\_returned\_date\_sk< 2452275、2452275 <= wr\_returned\_date\_sk< 2452640、2452640 <= wr\_returned\_date\_sk< 2453005、wr\_returned\_date\_sk>=2453005。

--创建表tpcds.web\_returns。  
vastbase=# CREATE TABLE tpcds.web\_returns  
(  
 W\_WAREHOUSE\_SK INTEGER NOT NULL,  
 W\_WAREHOUSE\_ID CHAR(16) NOT NULL,  
 W\_WAREHOUSE\_NAME VARCHAR(20) ,  
 W\_WAREHOUSE\_SQ\_FT INTEGER ,  
 W\_STREET\_NUMBER CHAR(10) ,  
 W\_STREET\_NAME VARCHAR(60) ,  
 W\_STREET\_TYPE CHAR(15) ,  
 W\_SUITE\_NUMBER CHAR(10) ,  
 W\_CITY VARCHAR(60) ,  
 W\_COUNTY VARCHAR(30) ,  
 W\_STATE CHAR(2) ,  
 W\_ZIP CHAR(10) ,  
 W\_COUNTRY VARCHAR(20) ,  
 W\_GMT\_OFFSET DECIMAL(5,2)  
);  
 --创建分区表tpcds.web\_returns\_p1。  
vastbase=# CREATE TABLE tpcds.web\_returns\_p1  
(  
 WR\_RETURNED\_DATE\_SK INTEGER ,  
 WR\_RETURNED\_TIME\_SK INTEGER ,  
 WR\_ITEM\_SK INTEGER NOT NULL,  
 WR\_REFUNDED\_CUSTOMER\_SK INTEGER ,  
 WR\_REFUNDED\_CDEMO\_SK INTEGER ,  
 WR\_REFUNDED\_HDEMO\_SK INTEGER ,  
 WR\_REFUNDED\_ADDR\_SK INTEGER ,  
 WR\_RETURNING\_CUSTOMER\_SK INTEGER ,  
 WR\_RETURNING\_CDEMO\_SK INTEGER ,  
 WR\_RETURNING\_HDEMO\_SK INTEGER ,  
 WR\_RETURNING\_ADDR\_SK INTEGER ,  
 WR\_WEB\_PAGE\_SK INTEGER ,  
 WR\_REASON\_SK INTEGER ,  
 WR\_ORDER\_NUMBER BIGINT NOT NULL,  
 WR\_RETURN\_QUANTITY INTEGER ,  
 WR\_RETURN\_AMT DECIMAL(7,2) ,  
 WR\_RETURN\_TAX DECIMAL(7,2) ,  
 WR\_RETURN\_AMT\_INC\_TAX DECIMAL(7,2) ,  
 WR\_FEE DECIMAL(7,2) ,  
 WR\_RETURN\_SHIP\_COST DECIMAL(7,2) ,  
 WR\_REFUNDED\_CASH DECIMAL(7,2) ,  
 WR\_REVERSED\_CHARGE DECIMAL(7,2) ,  
 WR\_ACCOUNT\_CREDIT DECIMAL(7,2) ,  
 WR\_NET\_LOSS DECIMAL(7,2)  
)  
WITH (ORIENTATION = COLUMN,COMPRESSION=MIDDLE)  
PARTITION BY RANGE(WR\_RETURNED\_DATE\_SK)  
 (  
 PARTITION P1 VALUES LESS THAN(2450815),  
 PARTITION P2 VALUES LESS THAN(2451179),  
 PARTITION P3 VALUES LESS THAN(2451544),  
 PARTITION P4 VALUES LESS THAN(2451910),  
 PARTITION P5 VALUES LESS THAN(2452275),  
 PARTITION P6 VALUES LESS THAN(2452640),  
 PARTITION P7 VALUES LESS THAN(2453005),  
 PARTITION P8 VALUES LESS THAN(MAXVALUE)  
 );  
   
 --从示例数据表导入数据。  
vastbase=# INSERT INTO tpcds.web\_returns\_p1 SELECT \* FROM tpcds.web\_returns;  
   
 --删除分区P8。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 DROP PARTITION P8;  
   
 --增加分区WR\_RETURNED\_DATE\_SK介于2453005和2453105之间。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 ADD PARTITION P8 VALUES LESS THAN (2453105);  
   
 --增加分区WR\_RETURNED\_DATE\_SK介于2453105和MAXVALUE之间。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 ADD PARTITION P9 VALUES LESS THAN (MAXVALUE);  
   
 --删除分区P8。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 DROP PARTITION FOR (2453005);  
   
 --分区P7重命名为P10。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 RENAME PARTITION P7 TO P10;  
   
 --分区P6重命名为P11。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p1 RENAME PARTITION FOR (2452639) TO P11;  
   
 --查询分区P10的行数。  
vastbase=# SELECT count(\*) FROM tpcds.web\_returns\_p1 PARTITION (P10);  
 count   
--------  
 0  
(1 row)  
   
 --查询分区P1的行数。  
vastbase=# SELECT COUNT(\*) FROM tpcds.web\_returns\_p1 PARTITION FOR (2450815);  
 count   
--------  
 0  
(1 row)

* 示例2：创建范围分区表tpcds.web\_returns\_p2，含有8个分区，分区键类型为integer类型，其中第8个分区上边界为MAXVALUE。
* 八个分区的范围分别为： wr\_returned\_date\_sk< 2450815、2450815<= wr\_returned\_date\_sk< 2451179、2451179<=wr\_returned\_date\_sk< 2451544、2451544 <= wr\_returned\_date\_sk< 2451910、2451910 <= wr\_returned\_date\_sk< 2452275、2452275 <= wr\_returned\_date\_sk< 2452640、2452640 <= wr\_returned\_date\_sk< 2453005、wr\_returned\_date\_sk>=2453005。
* 分区表tpcds.web\_returns\_p2的表空间为example1；分区P1到P7没有声明表空间，使用采用分区表tpcds.web\_returns\_p2的表空间example1；指定分区P8的表空间为example2。
* 假定数据库节点的数据目录/pg\_location/mount1/path1，数据库节点的数据目录/pg\_location/mount2/path2，数据库节点的数据目录/pg\_location/mount3/path3，数据库节点的数据目录/pg\_location/mount4/path4是dwsadmin用户拥有读写权限的空目录。

vastbase=# CREATE TABLESPACE example1 RELATIVE LOCATION 'tablespace1/tablespace\_1';  
vastbase=# CREATE TABLESPACE example2 RELATIVE LOCATION 'tablespace2/tablespace\_2';  
vastbase=# CREATE TABLESPACE example3 RELATIVE LOCATION 'tablespace3/tablespace\_3';  
vastbase=# CREATE TABLESPACE example4 RELATIVE LOCATION 'tablespace4/tablespace\_4';  
   
vastbase=# CREATE TABLE tpcds.web\_returns\_p2  
(  
 WR\_RETURNED\_DATE\_SK INTEGER ,  
 WR\_RETURNED\_TIME\_SK INTEGER ,  
 WR\_ITEM\_SK INTEGER NOT NULL,  
 WR\_REFUNDED\_CUSTOMER\_SK INTEGER ,  
 WR\_REFUNDED\_CDEMO\_SK INTEGER ,  
 WR\_REFUNDED\_HDEMO\_SK INTEGER ,  
 WR\_REFUNDED\_ADDR\_SK INTEGER ,  
 WR\_RETURNING\_CUSTOMER\_SK INTEGER ,  
 WR\_RETURNING\_CDEMO\_SK INTEGER ,  
 WR\_RETURNING\_HDEMO\_SK INTEGER ,  
 WR\_RETURNING\_ADDR\_SK INTEGER ,  
 WR\_WEB\_PAGE\_SK INTEGER ,  
 WR\_REASON\_SK INTEGER ,  
 WR\_ORDER\_NUMBER BIGINT NOT NULL,  
 WR\_RETURN\_QUANTITY INTEGER ,  
 WR\_RETURN\_AMT DECIMAL(7,2) ,  
 WR\_RETURN\_TAX DECIMAL(7,2) ,  
 WR\_RETURN\_AMT\_INC\_TAX DECIMAL(7,2) ,  
 WR\_FEE DECIMAL(7,2) ,  
 WR\_RETURN\_SHIP\_COST DECIMAL(7,2) ,  
 WR\_REFUNDED\_CASH DECIMAL(7,2) ,  
 WR\_REVERSED\_CHARGE DECIMAL(7,2) ,  
 WR\_ACCOUNT\_CREDIT DECIMAL(7,2) ,  
 WR\_NET\_LOSS DECIMAL(7,2)  
 )  
TABLESPACE example1  
PARTITION BY RANGE(WR\_RETURNED\_DATE\_SK)  
 (  
 PARTITION P1 VALUES LESS THAN(2450815),  
 PARTITION P2 VALUES LESS THAN(2451179),  
 PARTITION P3 VALUES LESS THAN(2451544),  
 PARTITION P4 VALUES LESS THAN(2451910),  
 PARTITION P5 VALUES LESS THAN(2452275),  
 PARTITION P6 VALUES LESS THAN(2452640),  
 PARTITION P7 VALUES LESS THAN(2453005),  
 PARTITION P8 VALUES LESS THAN(MAXVALUE) TABLESPACE example2  
 )  
ENABLE ROW MOVEMENT;  
   
 --以like方式创建一个分区表。  
vastbase=# CREATE TABLE tpcds.web\_returns\_p3 (LIKE tpcds.web\_returns\_p2 INCLUDING PARTITION);  
   
 --修改分区P1的表空间为example2。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p2 MOVE PARTITION P1 TABLESPACE example2;  
   
 --修改分区P2的表空间为example3。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p2 MOVE PARTITION P2 TABLESPACE example3;  
   
 --以2453010为分割点切分P8。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p2 SPLIT PARTITION P8 AT (2453010) INTO  
 (  
 PARTITION P9,  
 PARTITION P10  
 );   
   
 --将P6，P7合并为一个分区。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p2 MERGE PARTITIONS P6, P7 INTO PARTITION P8;  
   
 --修改分区表迁移属性。  
vastbase=# ALTER TABLE tpcds.web\_returns\_p2 DISABLE ROW MOVEMENT;  
 --删除表和表空间。  
vastbase=# DROP TABLE tpcds.web\_returns\_p1;  
vastbase=# DROP TABLE tpcds.web\_returns\_p2;  
vastbase=# DROP TABLE tpcds.web\_returns\_p3;  
vastbase=# DROP TABLESPACE example1;  
vastbase=# DROP TABLESPACE example2;  
vastbase=# DROP TABLESPACE example3;  
vastbase=# DROP TABLESPACE example4;

* 示例3：START END语法创建、修改Range分区表。
* 假定/home/vastbase/startend\_tbs1、/home/vastbase/startend\_tbs2、/home/vastbase/startend\_tbs3、/home/vastbase/startend\_tbs4是vastbase用户拥有读写权限的空目录。

-- 创建表空间  
vastbase=# CREATE TABLESPACE startend\_tbs1 LOCATION '/home/vastbase/startend\_tbs1';  
vastbase=# CREATE TABLESPACE startend\_tbs2 LOCATION '/home/vastbase/startend\_tbs2';  
vastbase=# CREATE TABLESPACE startend\_tbs3 LOCATION '/home/vastbase/startend\_tbs3';  
vastbase=# CREATE TABLESPACE startend\_tbs4 LOCATION '/home/vastbase/startend\_tbs4';  
   
 -- 创建临时schema  
vastbase=# CREATE SCHEMA tpcds;  
vastbase=# SET CURRENT\_SCHEMA TO tpcds;  
   
 -- 创建分区表，分区键是integer类型  
vastbase=# CREATE TABLE tpcds.startend\_pt (c1 INT, c2 INT)   
TABLESPACE startend\_tbs1   
PARTITION BY RANGE (c2) (  
 PARTITION p1 START(1) END(1000) EVERY(200) TABLESPACE startend\_tbs2,  
 PARTITION p2 END(2000),  
 PARTITION p3 START(2000) END(2500) TABLESPACE startend\_tbs3,  
 PARTITION p4 START(2500),  
 PARTITION p5 START(3000) END(5000) EVERY(1000) TABLESPACE startend\_tbs4  
 )  
ENABLE ROW MOVEMENT;  
   
 -- 查看分区表信息  
vastbase=# SELECT relname, boundaries, spcname FROM pg\_partition p JOIN pg\_tablespace t ON p.reltablespace=t.oid and p.parentid='tpcds.startend\_pt'::regclass ORDER BY 1;  
 relname | boundaries | spcname  
 -------------+------------+---------------  
 p1\_0 | {1} | startend\_tbs2  
 p1\_1 | {201} | startend\_tbs2  
 p1\_2 | {401} | startend\_tbs2  
 p1\_3 | {601} | startend\_tbs2  
 p1\_4 | {801} | startend\_tbs2  
 p1\_5 | {1000} | startend\_tbs2  
 p2 | {2000} | startend\_tbs1  
 p3 | {2500} | startend\_tbs3  
 p4 | {3000} | startend\_tbs1  
 p5\_1 | {4000} | startend\_tbs4  
 p5\_2 | {5000} | startend\_tbs4  
 startend\_pt | | startend\_tbs1  
 (12 rows)  
   
 -- 导入数据，查看分区数据量  
vastbase=# INSERT INTO tpcds.startend\_pt VALUES (GENERATE\_SERIES(0, 4999), GENERATE\_SERIES(0, 4999));  
vastbase=# SELECT COUNT(\*) FROM tpcds.startend\_pt PARTITION FOR (0);  
 count  
-------  
 1  
(1 row)  
   
vastbase=# SELECT COUNT(\*) FROM tpcds.startend\_pt PARTITION (p3);  
 count  
-------  
 500  
(1 row)  
   
 -- 增加分区: [5000, 5300), [5300, 5600), [5600, 5900), [5900, 6000)  
vastbase=# ALTER TABLE tpcds.startend\_pt ADD PARTITION p6 START(5000) END(6000) EVERY(300) TABLESPACE startend\_tbs4;  
   
 -- 增加MAXVALUE分区: p7  
vastbase=# ALTER TABLE tpcds.startend\_pt ADD PARTITION p7 END(MAXVALUE);  
   
 -- 重命名分区p7为p8  
vastbase=# ALTER TABLE tpcds.startend\_pt RENAME PARTITION p7 TO p8;  
   
 -- 删除分区p8  
vastbase=# ALTER TABLE tpcds.startend\_pt DROP PARTITION p8;  
   
 -- 重命名5950所在的分区为：p71  
vastbase=# ALTER TABLE tpcds.startend\_pt RENAME PARTITION FOR(5950) TO p71;  
   
 -- 分裂4500所在的分区[4000, 5000)  
vastbase=# ALTER TABLE tpcds.startend\_pt SPLIT PARTITION FOR(4500) INTO(PARTITION q1 START(4000) END(5000) EVERY(250) TABLESPACE startend\_tbs3);  
   
 -- 修改分区p2的表空间为startend\_tbs4  
vastbase=# ALTER TABLE tpcds.startend\_pt MOVE PARTITION p2 TABLESPACE startend\_tbs4;  
   
 -- 查看分区情形  
vastbase=# SELECT relname, boundaries, spcname FROM pg\_partition p JOIN pg\_tablespace t ON p.reltablespace=t.oid and p.parentid='tpcds.startend\_pt'::regclass ORDER BY 1;  
 relname | boundaries | spcname  
 -------------+------------+---------------  
 p1\_0 | {1} | startend\_tbs2  
 p1\_1 | {201} | startend\_tbs2  
 p1\_2 | {401} | startend\_tbs2  
 p1\_3 | {601} | startend\_tbs2  
 p1\_4 | {801} | startend\_tbs2  
 p1\_5 | {1000} | startend\_tbs2  
 p2 | {2000} | startend\_tbs4  
 p3 | {2500} | startend\_tbs3  
 p4 | {3000} | startend\_tbs1  
 p5\_1 | {4000} | startend\_tbs4  
 p6\_1 | {5300} | startend\_tbs4  
 p6\_2 | {5600} | startend\_tbs4  
 p6\_3 | {5900} | startend\_tbs4  
 p71 | {6000} | startend\_tbs4  
 q1\_1 | {4250} | startend\_tbs3  
 q1\_2 | {4500} | startend\_tbs3  
 q1\_3 | {4750} | startend\_tbs3  
 q1\_4 | {5000} | startend\_tbs3  
 startend\_pt | | startend\_tbs1  
 (19 rows)  
   
 -- 删除表和表空间  
vastbase=# DROP SCHEMA tpcds CASCADE;  
vastbase=# DROP TABLESPACE startend\_tbs1;  
vastbase=# DROP TABLESPACE startend\_tbs2;  
vastbase=# DROP TABLESPACE startend\_tbs3;  
vastbase=# DROP TABLESPACE startend\_tbs4;

* 示例4：创建间隔分区表sales，初始包含2个分区，分区键为DATE类型。 分区的范围分别为：time\_id < '2019-02-01 00:00:00'、

'2019-02-01 00:00:00' <= time\_id < '2019-02-02 00:00:00' 。

--创建表sales  
vastbase=# CREATE TABLE sales  
(prod\_id NUMBER(6),  
cust\_id NUMBER,  
time\_id DATE,  
channel\_id CHAR(1),  
promo\_id NUMBER(6),  
quantity\_sold NUMBER(3),  
amount\_sold NUMBER(10,2)  
)  
PARTITION BY RANGE (time\_id)  
INTERVAL('1 day')  
( PARTITION p1 VALUES LESS THAN ('2019-02-01 00:00:00'),  
PARTITION p2 VALUES LESS THAN ('2019-02-02 00:00:00')  
);  
  
-- 数据插入分区p1  
vastbase=# INSERT INTO sales VALUES(1, 12, '2019-01-10 00:00:00', 'a', 1, 1, 1);  
  
-- 数据插入分区p2  
vastbase=# INSERT INTO sales VALUES(1, 12, '2019-02-01 00:00:00', 'a', 1, 1, 1);  
  
-- 查看分区信息  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'sales' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
---------+--------------+-------------------------  
p1 | r | {"2019-02-01 00:00:00"}  
p2 | r | {"2019-02-02 00:00:00"}  
(2 rows)  
  
-- 插入数据没有匹配的分区，新创建一个分区，并将数据插入该分区  
-- 新分区的范围为 '2019-02-05 00:00:00' <= time\_id < '2019-02-06 00:00:00'  
vastbase=# INSERT INTO sales VALUES(1, 12, '2019-02-05 00:00:00', 'a', 1, 1, 1);  
  
-- 插入数据没有匹配的分区，新创建一个分区，并将数据插入该分区  
-- 新分区的范围为 '2019-02-03 00:00:00' <= time\_id < '2019-02-04 00:00:00'  
vastbase=# INSERT INTO sales VALUES(1, 12, '2019-02-03 00:00:00', 'a', 1, 1, 1);  
  
-- 查看分区信息  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'sales' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
---------+--------------+-------------------------  
sys\_p1 | i | {"2019-02-06 00:00:00"}  
sys\_p2 | i | {"2019-02-04 00:00:00"}  
p1 | r | {"2019-02-01 00:00:00"}  
p2 | r | {"2019-02-02 00:00:00"}  
(4 rows)

* 示例5：创建LIST分区表test\_list，初始包含4个分区，分区键为INT类型。4个分区的范围分别为：2000、3000、4000、5000。

--创建表test\_list  
vastbase=# create table test\_list (col1 int, col2 int)  
partition by list(col1)  
(  
partition p1 values (2000),  
partition p2 values (3000),  
partition p3 values (4000),  
partition p4 values (5000)  
);  
  
-- 数据插入  
vastbase=# INSERT INTO test\_list VALUES(2000, 2000);  
INSERT 0 1  
vastbase=# INSERT INTO test\_list VALUES(3000, 3000);  
INSERT 0 1  
  
-- 查看分区信息  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'test\_list' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
--------+--------------+------------  
p1 | l | {2000}  
p2 | l | {3000}  
p3 | l | {4000}  
p4 | l | {5000}  
(4 rows)  
  
-- 插入数据没有匹配到分区，报错处理  
vastbase=# INSERT INTO test\_list VALUES(6000, 6000);  
ERROR: inserted partition key does not map to any table partition  
  
-- 添加分区  
vastbase=# alter table test\_list add partition p5 values (6000);  
ALTER TABLE  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'test\_list' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
---------+--------------+------------  
p5 | l | {6000}  
p4 | l | {5000}  
p1 | l | {2000}  
p2 | l | {3000}  
p3 | l | {4000}  
(5 rows)  
vastbase=# INSERT INTO test\_list VALUES(6000, 6000);  
INSERT 0 1  
  
-- 分区表和普通表交换数据  
vastbase=# create table t1 (col1 int, col2 int);  
CREATE TABLE  
vastbase=# select \* from test\_list partition (p1);  
col1 | col2  
------+------  
2000 | 2000  
(1 row)  
vastbase=# alter table test\_list exchange partition (p1) with table t1;  
ALTER TABLE  
vastbase=# select \* from test\_list partition (p1);  
col1 | col2  
------+------  
(0 rows)  
vastbase=# select \* from t1;  
col1 | col2  
------+------  
2000 | 2000  
(1 row)  
  
-- truncate分区  
vastbase=# select \* from test\_list partition (p2);  
col1 | col2  
------+------  
3000 | 3000  
(1 row)  
vastbase=# alter table test\_list truncate partition p2;  
ALTER TABLE  
vastbase=# select \* from test\_list partition (p2);  
col1 | col2  
------+------  
(0 rows)  
  
-- 删除分区  
vastbase=# alter table test\_list drop partition p5;  
ALTER TABLE  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'test\_list' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
---------+--------------+------------  
p4 | l | {5000}  
p1 | l | {2000}  
p2 | l | {3000}  
p3 | l | {4000}  
(4 rows)  
  
vastbase=# INSERT INTO test\_list VALUES(6000, 6000);  
ERROR: inserted partition key does not map to any table partition  
  
-- 删除分区表  
vastbase=# drop table test\_list;

* 示例6：创建HASH分区表test\_hash，初始包含2个分区，分区键为INT类型。

--创建表test\_hash  
vastbase=# create table test\_hash (col1 int, col2 int)  
partition by hash(col1)  
(  
partition p1,  
partition p2  
);  
  
-- 数据插入  
vastbase=# INSERT INTO test\_hash VALUES(1, 1);  
INSERT 0 1  
vastbase=# INSERT INTO test\_hash VALUES(2, 2);  
INSERT 0 1  
vastbase=# INSERT INTO test\_hash VALUES(3, 3);  
INSERT 0 1  
vastbase=# INSERT INTO test\_hash VALUES(4, 4);  
INSERT 0 1  
  
-- 查看分区信息  
vastbase=# SELECT t1.relname, partstrategy, boundaries FROM pg\_partition t1, pg\_class t2 WHERE t1.parentid = t2.oid AND t2.relname = 'test\_hash' AND t1.parttype = 'p';  
relname | partstrategy | boundaries  
---------+--------------+------------  
p1 | h | {0}  
p2 | h | {1}  
(2 rows)  
  
-- 查看数据  
vastbase=# select \* from test\_hash partition (p1);  
col1 | col2  
------+------  
3 | 3  
4 | 4  
(2 rows)  
  
vastbase=# select \* from test\_hash partition (p2);  
col1 | col2  
------+------  
1 | 1  
2 | 2  
(2 rows)  
  
-- 分区表和普通表交换数据  
vastbase=# create table t1 (col1 int, col2 int);  
CREATE TABLE  
vastbase=# alter table test\_hash exchange partition (p1) with table t1;  
ALTER TABLE  
vastbase=# select \* from test\_hash partition (p1);  
col1 | col2  
------+------  
(0 rows)  
vastbase=# select \* from t1;  
col1 | col2  
------+------  
3 | 3  
4 | 4  
(2 rows)  
  
-- truncate分区  
vastbase=# alter table test\_hash truncate partition p2;  
ALTER TABLE  
vastbase=# select \* from test\_hash partition (p2);  
col1 | col2  
------+------  
(0 rows)  
  
-- 删除分区表  
vastbase=# drop table test\_hash;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TABLE PARTITION、DROP TABLE章节。

### CREATE TABLE SUBPARTITION

**功能描述**

创建二级分区表。分区表是把逻辑上的一张表根据某种方案分成几张物理块进行存储，这张逻辑上的表称之为分区表，物理块称之为分区。分区表是一张逻辑表，不存储数据，数据实际是存储在分区上的。对于二级分区表，顶层节点表和一级分区都是逻辑表，不存储数据，只有二级分区（叶子节点）存储数据。

二级分区表的分区方案是由两个一级分区的分区方案组合而来的，一级分区的分区方案详见：SQL语法参考->SQL语法->CREATE TABLE PARTITION章节。

常见的二级分区表组合方案有Range-Range分区、Range-List分区、Range-Hash分区、List-Range分区、List-List分区、List-Hash分区、Hash-Range分区、Hash-List分区、Hash-Hash分区等。目前二级分区仅支持行存表。

**注意事项**

* 二级分区表有两个分区键，每个分区键只能支持1列。
* 唯一约束和主键约束的约束键包含所有分区键将为约束创建LOCAL索引，否则创建GLOBAL索引。
* 二级分区表的二级分区（叶子节点）个数不能超过1048575个，一级分区无限制，但一级分区下面至少有一个二级分区。
* 二级分区表只支持行存，不支持列存、段页式、hashbucket。
* 不支持Upsert、Merge into。
* 指定分区查询时，如select \* from tablename partition/subpartition (partitionname)，关键字partition和subpartition注意不要写错。如果写错，查询不会报错，这时查询会变为对表起别名进行查询。
* 不支持对二级分区 subpartition for (values)查询。如select \* from tablename subpartition for (values)。
* 不支持密态数据库、账本数据库和行级访问控制。
* 对于二级分区表PARTITION FOR (values)语法，values只能是常量。
* 对于分区表PARTITION/SUBPARTITION FOR (values)语法，values在需要数据类型转换时，建议使用强制类型转换，以防隐式类型转换结果与预期不符。
* 指定分区语句目前不能走全局索引扫描。

**语法格式**

CREATE TABLE [ IF NOT EXISTS ] subpartition\_table\_name  
(   
{ column\_name data\_type [ COLLATE collation ] [ column\_constraint [ ... ] ]  
| table\_constraint  
| LIKE source\_table [ like\_option [...] ] }[, ... ]  
)  
[ WITH ( {storage\_parameter = value} [, ... ] ) ]  
[ COMPRESS | NOCOMPRESS ]  
[ TABLESPACE tablespace\_name ]  
PARTITION BY {RANGE | LIST | HASH} (partition\_key) SUBPARTITION BY {RANGE | LIST | HASH} (subpartition\_key)  
(  
 PARTITION partition\_name1 [ VALUES LESS THAN (val1) | VALUES (val1[, …]) ] [ TABLESPACE tablespace ]  
 (  
 { SUBPARTITION subpartition\_name1 [ VALUES LESS THAN (val1\_1) | VALUES (val1\_1[, …])] [ TABLESPACE tablespace ] } [, ...]  
 )[, ...]  
)[ { ENABLE | DISABLE } ROW MOVEMENT ];

* 列约束column\_constraint：

[ CONSTRAINT constraint\_name ]  
 { NOT NULL |  
 NULL |   
 CHECK ( expression ) |   
 DEFAULT default\_e xpr |   
 GENERATED ALWAYS AS ( generation\_expr ) STORED |  
 UNIQUE index\_parameters |   
 PRIMARY KEY index\_parameters |  
 REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]  
 [ ON DELETE action ] [ ON UPDATE action ] }  
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* 表约束table\_constraint：

[ CONSTRAINT constraint\_name ]  
 { CHECK ( expression ) |   
 UNIQUE ( column\_name [, ... ] ) index\_parameters |   
 PRIMARY KEY ( column\_name [, ... ] ) index\_parameters |  
 FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]  
 [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] }  
[ DEFERRABLE | NOT DEFERRABLE | INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

* like选项like\_option：

{ INCLUDING | EXCLUDING } { DEFAULTS | GENERATED | CONSTRAINTS | INDEXES | STORAGE | COMMENTS | RELOPTIONS| ALL }

* 索引存储参数index\_parameters：

[ WITH ( {storage\_parameter = value} [, ... ] ) ]  
[ USING INDEX TABLESPACE tablespace\_name ]

**参数说明**

* **IF NOT EXISTS**
* 如果已经存在相同名称的表，不会抛出一个错误，而会发出一个通知，告知表关系已存在。
* **subpartition\_table\_name**
* 二级分区表的名称。
* 取值范围：字符串，要符合标识符的命名规范。
* **column\_name**
* 新表中要创建的字段名。
* 取值范围：字符串，要符合标识符的命名规范。
* **data\_type**
* 字段的数据类型。
* **COLLATE collation**
* COLLATE子句指定列的排序规则（该列必须是可排列的数据类型）。如果没有指定，则使用默认的排序规则。排序规则可以使用“select \* from pg\_collation;”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。
* **CONSTRAINT constraint\_name**
* 列约束或表约束的名称。可选的约束子句用于声明约束，新行或者更新的行必须满足这些约束才能成功插入或更新。
* 定义约束有两种方法：
* 列约束：作为一个列定义的一部分，仅影响该列。
* 表约束：不和某个列绑在一起，可以作用于多个列。
* **LIKE source\_table [ like\_option ... ]**
* 二级分区表暂不支持该功能。
* **WITH ( storage\_parameter [= value] [, ... ] )**
* 这个子句为表或索引指定一个可选的存储参数。参数的详细描述如下所示：
* FILLFACTOR
  + 一个表的填充因子（fillfactor）是一个介于10和100之间的百分数。100（完全填充）是默认值。如果指定了较小的填充因子，INSERT操作仅按照填充因子指定的百分率填充表页。每个页上的剩余空间将用于在该页上更新行，这就使得UPDATE有机会在同一页上放置同一条记录的新版本，这比把新版本放置在其他页上更有效。对于一个从不更新的表将填充因子设为100是最佳选择，但是对于频繁更新的表，选择较小的填充因子则更加合适。该参数对于列存表没有意义。
  + 取值范围：10~100
* ORIENTATION
  + 决定了表的数据的存储方式,orientation不支持修改。
  + 取值范围：
* COLUMN：表的数据将以列式存储。
* ROW（缺省值）：表的数据将以行式存储。
* COMPRESSION
* 列存表的有效值为LOW/MIDDLE/HIGH/YES/NO，压缩级别依次升高，默认值为LOW。
* 行存表不支持压缩。
* MAX\_BATCHROW
  + 指定了在数据加载过程中一个存储单元可以容纳记录的最大数目。该参数只对列存表有效。
  + 取值范围：10000~60000，默认60000。
* PARTIAL\_CLUSTER\_ROWS
  + 指定了在数据加载过程中进行将局部聚簇存储的记录数目。该参数只对列存表有效。
  + 取值范围：大于等于MAX\_BATCHROW，建议取值为MAX\_BATCHROW的整数倍数。
* DELTAROW\_THRESHOLD
  + 预留参数。该参数只对列存表有效。
  + 取值范围：0～9999
* **COMPRESS / NOCOMPRESS**
* 创建一个新表时，需要在创建表语句中指定关键字COMPRESS，这样，当对该表进行批量插入时就会触发压缩特性。该特性会在页范围内扫描所有元组数据，生成字典、压缩元组数据并进行存储。指定关键字NOCOMPRESS则不对表进行压缩。行存表不支持压缩。
* 缺省值为NOCOMPRESS，即不对元组数据进行压缩。
* **TABLESPACE tablespace\_name**
* 指定新表将要在tablespace\_name表空间内创建。如果没有声明，将使用默认表空间。
* **PARTITION BY {RANGE | LIST | HASH} (partition\_key)**
* 对于partition\_key，分区策略的分区键仅支持1列。
* 分区键支持的数据类型和一级分区表约束保持一致。
* **SUBPARTITION BY {RANGE | LIST | HASH} (subpartition\_key)**
* 对于subpartition\_key，分区策略的分区键仅支持1列。
* 分区键支持的数据类型和一级分区表约束保持一致。
* **{ ENABLE | DISABLE } ROW MOVEMENT**
* 行迁移开关。
* 如果进行UPDATE操作时，更新了元组在分区键上的值，造成了该元组所在分区发生变化，就会根据该开关给出报错信息，或者进行元组在分区间的转移。
* 取值范围：
* ENABLE（缺省值）：行迁移开关打开。
* DISABLE：行迁移开关关闭。
* **NOT NULL**
* 字段值不允许为NULL。ENABLE用于语法兼容，可省略。
* **NULL**
* 字段值允许NULL ，这是缺省。
* 这个子句只是为和非标准SQL数据库兼容。不建议使用。
* **CHECK (condition) [ NO INHERIT ]**
* CHECK约束声明一个布尔表达式，每次要插入的新行或者要更新的行的新值必须使表达式结果为真或未知才能成功，否则会抛出一个异常并且不会修改数据库。
* 声明为字段约束的检查约束应该只引用该字段的数值，而在表约束里出现的表达式可以引用多个字段。
* 用NO INHERIT标记的约束将不会传递到子表中去。
* ENABLE用于语法兼容，可省略。
* **DEFAULT default\_expr**
* DEFAULT子句给字段指定缺省值。该数值可以是任何不含变量的表达式(不允许使用子查询和对本表中的其他字段的交叉引用)。缺省表达式的数据类型必须和字段类型匹配。
* 缺省表达式将被用于任何未声明该字段数值的插入操作。如果没有指定缺省值则缺省值为NULL 。
* **GENERATED ALWAYS AS ( generation\_expr ) STORED**
* 该子句将字段创建为生成列，生成列的值在写入（插入或更新）数据时由generation\_expr计算得到，STORED表示像普通列一样存储生成列的值。
* fig: **说明：**
* 生成表达式不能以任何方式引用当前行以外的其他数据。生成表达式不能引用其他生成列，不能引用系统列。生成表达式不能返回结果集，不能使用子查询，不能使用聚集函数，不能使用窗口函数。生成表达式调用的函数只能是不可变（IMMUTABLE）函数。
* 不能为生成列指定默认值。
* 生成列不能作为分区键的一部分。
* 生成列不能和ON UPDATE约束字句的CASCADE、SET NULL、SET DEFAULT动作同时指定。生成列不能和ON DELETE约束字句的SET NULL、SET DEFAULT动作同时指定。
* 修改和删除生成列的方法和普通列相同。删除生成列依赖的普通列，生成列被自动删除。不能改变生成列所依赖的列的类型。
* 生成列不能被直接写入。在INSERT或UPDATE命令中, 不能为生成列指定值, 但是可以指定关键字DEFAULT。
* 生成列的权限控制和普通列一样。
* 不能为生成列指定默认值。
* 生成列不能作为分区键的一部分。
* 生成列不能和ON UPDATE约束字句的CASCADE、SET NULL、SET DEFAULT动作同时指定。生成列不能和ON DELETE约束字句的SET NULL、SET DEFAULT动作同时指定。
* 修改和删除生成列的方法和普通列相同。删除生成列依赖的普通列，生成列被自动删除。不能改变生成列所依赖的列的类型。
* 生成列不能被直接写入。在INSERT或UPDATE命令中, 不能为生成列指定值, 但是可以指定关键字DEFAULT。
* 生成列的权限控制和普通列一样。
* 列存表、内存表MOT不支持生成列。外表中仅postgres\_fdw支持生成列。
* **UNIQUE index\_parameters**
* **UNIQUE ( column\_name [, ... ] ) index\_parameters**
* UNIQUE约束表示表里的一个字段或多个字段的组合必须在全表范围内唯一。
* 对于唯一约束，NULL被认为是互不相等的。
* **PRIMARY KEY index\_parameters**
* **PRIMARY KEY ( column\_name [, ... ] ) index\_parameters**
* 主键约束声明表中的一个或者多个字段只能包含唯一的非NULL值。
* 一个表只能声明一个主键。
* **DEFERRABLE | NOT DEFERRABLE**
* 这两个关键字设置该约束是否可推迟。一个不可推迟的约束将在每条命令之后马上检查。可推迟约束可以推迟到事务结尾使用SET CONSTRAINTS命令检查。缺省是NOT DEFERRABLE。目前，UNIQUE约束、主键约束、外键约束可以接受这个子句。所有其他约束类型都是不可推迟的。
* **INITIALLY IMMEDIATE | INITIALLY DEFERRED**
* 如果约束是可推迟的，则这个子句声明检查约束的缺省时间。
* 如果约束是INITIALLY IMMEDIATE（缺省），则在每条语句执行之后就立即检查它；
* 如果约束是INITIALLY DEFERRED ，则只有在事务结尾才检查它。
* 约束检查的时间可以用SET CONSTRAINTS命令修改。
* **USING INDEX TABLESPACE tablespace\_name**
* 为UNIQUE或PRIMARY KEY约束相关的索引声明一个表空间。如果没有提供这个子句，这个索引将在default\_tablespace中创建，如果default\_tablespace为空，将使用数据库的缺省表空间。

**示例**

* 示例1：创建各种组合类型的二级分区表

CREATE TABLE list\_list  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
)  
PARTITION BY LIST (month\_code) SUBPARTITION BY LIST (dept\_code)  
(  
 PARTITION p\_201901 VALUES ( '201902' )  
 (  
 SUBPARTITION p\_201901\_a VALUES ( '1' ),  
 SUBPARTITION p\_201901\_b VALUES ( '2' )  
 ),  
 PARTITION p\_201902 VALUES ( '201903' )  
 (  
 SUBPARTITION p\_201902\_a VALUES ( '1' ),  
 SUBPARTITION p\_201902\_b VALUES ( '2' )  
 )  
 );  
  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201902', '2', '1', 1);  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
insert into list\_list values('201903', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
  
select \* from list\_list;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 201902 | 2 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 (6 rows)  
  
drop table list\_list;  
  
CREATE TABLE list\_hash  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
)  
PARTITION BY LIST (month\_code) SUBPARTITION BY HASH (dept\_code)  
(  
 PARTITION p\_201901 VALUES ( '201902' )  
 (  
 SUBPARTITION p\_201901\_a,  
 SUBPARTITION p\_201901\_b  
 ),  
 PARTITION p\_201902 VALUES ( '201903' )  
 (  
 SUBPARTITION p\_201902\_a,  
 SUBPARTITION p\_201902\_b  
 )  
);  
  
insert into list\_hash values('201902', '1', '1', 1);  
insert into list\_hash values('201902', '2', '1', 1);  
insert into list\_hash values('201902', '3', '1', 1);  
insert into list\_hash values('201903', '4', '1', 1);  
insert into list\_hash values('201903', '5', '1', 1);  
insert into list\_hash values('201903', '6', '1', 1);  
  
select \* from list\_hash;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201903 | 4 | 1 | 1  
 201903 | 5 | 1 | 1  
 201903 | 6 | 1 | 1  
 201902 | 2 | 1 | 1  
 201902 | 3 | 1 | 1  
 201902 | 1 | 1 | 1  
(6 rows)  
  
drop table list\_hash;  
  
CREATE TABLE list\_range  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
 )  
PARTITION BY LIST (month\_code) SUBPARTITION BY RANGE (dept\_code)  
(  
 PARTITION p\_201901 VALUES ( '201902' )  
 (  
 SUBPARTITION p\_201901\_a values less than ('4'),  
 SUBPARTITION p\_201901\_b values less than ('6')  
 ),  
 PARTITION p\_201902 VALUES ( '201903' )  
 (  
 SUBPARTITION p\_201902\_a values less than ('3'),  
 SUBPARTITION p\_201902\_b values less than ('6')  
 )  
);  
insert into list\_range values('201902', '1', '1', 1);  
insert into list\_range values('201902', '2', '1', 1);  
insert into list\_range values('201902', '3', '1', 1);  
insert into list\_range values('201903', '4', '1', 1);  
insert into list\_range values('201903', '5', '1', 1);  
insert into list\_range values('201903', '6', '1', 1);  
ERROR: inserted partition key does not map to any table partition  
select \* from list\_range;  
 month\_code | dept\_code | user\_no | sales\_amt  
------------+-----------+---------+-----------  
 201902 | 1 | 1 | 1  
 201902 | 2 | 1 | 1  
 201902 | 3 | 1 | 1  
 201903 | 4 | 1 | 1  
 201903 | 5 | 1 | 1  
(5 rows)  
  
drop table list\_range;  
  
  
CREATE TABLE range\_list  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
 )  
PARTITION BY RANGE (month\_code) SUBPARTITION BY LIST (dept\_code)  
(  
 PARTITION p\_201901 VALUES LESS THAN( '201903' )  
 (  
 SUBPARTITION p\_201901\_a values ('1'),  
 SUBPARTITION p\_201901\_b values ('2')  
 ),  
 PARTITION p\_201902 VALUES LESS THAN( '201904' )  
 (  
 SUBPARTITION p\_201902\_a values ('1'),  
 SUBPARTITION p\_201902\_b values ('2')  
 )  
);  
insert into range\_list values('201902', '1', '1', 1);  
insert into range\_list values('201902', '2', '1', 1);  
insert into range\_list values('201902', '1', '1', 1);  
insert into range\_list values('201903', '2', '1', 1);  
insert into range\_list values('201903', '1', '1', 1);  
insert into range\_list values('201903', '2', '1', 1);  
select \* from range\_list;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201902 | 2 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
(6 rows)  
  
drop table range\_list;  
CREATE TABLE range\_hash  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
)  
PARTITION BY RANGE (month\_code) SUBPARTITION BY HASH (dept\_code)  
(  
 PARTITION p\_201901 VALUES LESS THAN( '201903' )  
 (  
 SUBPARTITION p\_201901\_a,  
 SUBPARTITION p\_201901\_b  
 ),  
 PARTITION p\_201902 VALUES LESS THAN( '201904' )  
 (  
 SUBPARTITION p\_201902\_a,  
 SUBPARTITION p\_201902\_b  
 )  
);  
insert into range\_hash values('201902', '1', '1', 1);  
insert into range\_hash values('201902', '2', '1', 1);  
insert into range\_hash values('201902', '1', '1', 1);  
insert into range\_hash values('201903', '2', '1', 1);  
insert into range\_hash values('201903', '1', '1', 1);  
insert into range\_hash values('201903', '2', '1', 1);  
select \* from range\_hash;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201902 | 2 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 (6 rows)  
  
drop table range\_hash;  
CREATE TABLE range\_range  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
)  
PARTITION BY RANGE (month\_code) SUBPARTITION BY RANGE (dept\_code)  
(  
 PARTITION p\_201901 VALUES LESS THAN( '201903' )  
 (  
 SUBPARTITION p\_201901\_a VALUES LESS THAN( '2' ),  
 SUBPARTITION p\_201901\_b VALUES LESS THAN( '3' )  
 ),  
 PARTITION p\_201902 VALUES LESS THAN( '201904' )  
 (  
 SUBPARTITION p\_201902\_a VALUES LESS THAN( '2' ),  
 SUBPARTITION p\_201902\_b VALUES LESS THAN( '3' )  
 )  
);  
insert into range\_range values('201902', '1', '1', 1);  
insert into range\_range values('201902', '2', '1', 1);  
insert into range\_range values('201902', '1', '1', 1);  
insert into range\_range values('201903', '2', '1', 1);  
insert into range\_range values('201903', '1', '1', 1);  
insert into range\_range values('201903', '2', '1', 1);  
select \* from range\_range;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
(6 rows)  
  
drop table range\_range;  
CREATE TABLE hash\_list  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
 )  
PARTITION BY hash (month\_code) SUBPARTITION BY LIST (dept\_code)  
(  
 PARTITION p\_201901  
 (  
 SUBPARTITION p\_201901\_a VALUES ( '1' ),  
 SUBPARTITION p\_201901\_b VALUES ( '2' )  
 ),  
 PARTITION p\_201902  
 (  
 SUBPARTITION p\_201902\_a VALUES ( '1' ),  
 SUBPARTITION p\_201902\_b VALUES ( '2' )  
)  
);  
insert into hash\_list values('201901', '1', '1', 1);  
insert into hash\_list values('201901', '2', '1', 1);  
insert into hash\_list values('201901', '1', '1', 1);  
insert into hash\_list values('201903', '2', '1', 1);  
insert into hash\_list values('201903', '1', '1', 1);  
insert into hash\_list values('201903', '2', '1', 1);  
select \* from hash\_list;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 201901 | 2 | 1 | 1  
 201901 | 1 | 1 | 1  
 201901 | 1 | 1 | 1  
(6 rows)  
  
drop table hash\_list;  
CREATE TABLE hash\_hash  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
 )  
PARTITION BY hash (month\_code) SUBPARTITION BY hash (dept\_code)  
(  
 PARTITION p\_201901  
 (  
 SUBPARTITION p\_201901\_a,  
 SUBPARTITION p\_201901\_b  
 ),  
 PARTITION p\_201902  
 (  
 SUBPARTITION p\_201902\_a,  
 SUBPARTITION p\_201902\_b  
 )  
);  
insert into hash\_hash values('201901', '1', '1', 1);  
insert into hash\_hash values('201901', '2', '1', 1);  
insert into hash\_hash values('201901', '1', '1', 1);  
insert into hash\_hash values('201903', '2', '1', 1);  
insert into hash\_hash values('201903', '1', '1', 1);  
insert into hash\_hash values('201903', '2', '1', 1);  
select \* from hash\_hash;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 201901 | 2 | 1 | 1  
 201901 | 1 | 1 | 1  
 201901 | 1 | 1 | 1  
(6 rows)  
  
drop table hash\_hash;  
CREATE TABLE hash\_range  
(  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
)  
PARTITION BY hash (month\_code) SUBPARTITION BY range (dept\_code)  
(  
 PARTITION p\_201901  
 (  
 SUBPARTITION p\_201901\_a VALUES LESS THAN ( '2' ),  
 SUBPARTITION p\_201901\_b VALUES LESS THAN ( '3' )  
 ),  
 PARTITION p\_201902  
 (  
 SUBPARTITION p\_201902\_a VALUES LESS THAN ( '2' ),  
 SUBPARTITION p\_201902\_b VALUES LESS THAN ( '3' )  
 )  
);  
insert into hash\_range values('201901', '1', '1', 1);  
insert into hash\_range values('201901', '2', '1', 1);  
insert into hash\_range values('201901', '1', '1', 1);  
insert into hash\_range values('201903', '2', '1', 1);  
insert into hash\_range values('201903', '1', '1', 1);  
insert into hash\_range values('201903', '2', '1', 1);  
select \* from hash\_range;  
  
 month\_code | dept\_code | user\_no | sales\_amt  
------------+-----------+---------+-----------  
 201903 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 201901 | 1 | 1 | 1  
 201901 | 1 | 1 | 1  
 201901 | 2 | 1 | 1  
(6 rows)

* 示例2：对二级分区表进行truncate操作

CREATE TABLE list\_list  
 (  
 month\_code VARCHAR2 ( 30 ) NOT NULL ,  
 dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
 user\_no VARCHAR2 ( 30 ) NOT NULL ,  
 sales\_amt int  
 )  
 PARTITION BY LIST (month\_code) SUBPARTITION BY LIST (dept\_code)  
 (  
 PARTITION p\_201901 VALUES ( '201902' )  
 (  
 SUBPARTITION p\_201901\_a VALUES ( '1' ),  
 SUBPARTITION p\_201901\_b VALUES ( default )  
 ),  
 PARTITION p\_201902 VALUES ( '201903' )  
 (  
 SUBPARTITION p\_201902\_a VALUES ( '1' ),  
 SUBPARTITION p\_201902\_b VALUES ( '2' )  
 )  
 );  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201902', '2', '1', 1);  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
insert into list\_list values('201903', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
select \* from list\_list;  
   
 month\_code | dept\_code | user\_no | sales\_amt  
------------+-----------+---------+-----------  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 2 | 1 | 1  
 201903 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
 (6 rows)  
   
select \* from list\_list partition (p\_201901);  
 month\_code | dept\_code | user\_no | sales\_amt  
------------+-----------+---------+-----------  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
 201902 | 2 | 1 | 1  
(3 rows)  
   
alter table list\_list truncate partition p\_201901;  
select \* from list\_list partition (p\_201901);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list partition (p\_201902);  
  
 month\_code | dept\_code | user\_no | sales\_amt  
------------+-----------+---------+-----------  
 201903 | 1 | 1 | 1  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
(3 rows)  
   
alter table list\_list truncate partition p\_201902;  
select \* from list\_list partition (p\_201902);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list;  
month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201902', '2', '1', 1);  
insert into list\_list values('201902', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
insert into list\_list values('201903', '1', '1', 1);  
insert into list\_list values('201903', '2', '1', 1);  
select \* from list\_list subpartition (p\_201901\_a);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201902 | 1 | 1 | 1  
 201902 | 1 | 1 | 1  
(2 rows)  
   
alter table list\_list truncate subpartition p\_201901\_a;  
select \* from list\_list subpartition (p\_201901\_a);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list subpartition (p\_201901\_b);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201902 | 2 | 1 | 1  
(1 row)  
   
alter table list\_list truncate subpartition p\_201901\_b;  
select \* from list\_list subpartition (p\_201901\_b);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list subpartition (p\_201902\_a);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
 201903 | 1 | 1 | 1  
(1 row)  
   
alter table list\_list truncate subpartition p\_201902\_a;  
select \* from list\_list subpartition (p\_201902\_a);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list subpartition (p\_201902\_b);  
 month\_code | dept\_code | user\_no | sales\_amt   
-----------+-----------+---------+-----------  
 201903 | 2 | 1 | 1  
 201903 | 2 | 1 | 1  
(2 rows)  
   
alter table list\_list truncate subpartition p\_201902\_b;  
select \* from list\_list subpartition (p\_201902\_b);  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
select \* from list\_list;  
 month\_code | dept\_code | user\_no | sales\_amt   
------------+-----------+---------+-----------  
(0 rows)  
   
drop table list\_list;

### CREATE TABLESPACE

**功能描述**

在数据库中创建一个新的表空间。

**注意事项**

* 系统管理员或者继承了内置角色gs\_roles\_tablespace权限的用户可以创建表空间。
* 不允许在一个事务块内部执行CREATE TABLESPACE。
* 执行CREATE TABLESPACE失败，如果内部创建目录（文件）操作成功了就会产生残留的目录（文件），重新创建时需要用户手动清理表空间指定的目录下残留的内容。如果在创建过程中涉及到数据目录下的表空间软连接残留，需要先将软连接的残留文件删除，再重新执行OM相关操作。
* CREATE TABLESPACE不支持两阶段事务，如果部分节点执行失败，不支持回滚。
* 创建表空间前的准备工作参考下述参数说明。
* 在HCS等场景下一般不建议用户使用自定义的表空间。原因：用户自定义表空间通常配合主存（即默认表空间所在的存储设备，如磁盘）以外的其它存储介质使用，以隔离不同业务可以使用的IO资源，而在HCS等场景下，存储设备都是采用标准化的配置，无其它可用的存储介质，自定义表空间使用不当不利于系统长稳运行以及影响整体性能，因此建议使用默认表空间即可。

**语法格式**

CREATE TABLESPACE tablespace\_name  
 [ OWNER user\_name ] [RELATIVE] LOCATION 'directory' [ MAXSIZE 'space\_size' ]  
 [with\_option\_clause];

其中普通表空间的with\_option\_clause为：

WITH ( {filesystem= { 'general'| "general" | general} |  
 random\_page\_cost = { 'value ' | value } |  
 seq\_page\_cost = { 'value ' | value }}[,...])

**参数说明**

* **tablespace\_name**
* 要创建的表空间名称。
* 表空间名称不能和Vastbase中的其他表空间重名，且名称不能以“pg”开头，这样的名称留给系统表空间使用。
* 取值范围：字符串，要符合标识符的命名规范。
* **OWNER user\_name**
* 指定该表空间的所有者。缺省时，新表空间的所有者是当前用户。
* 只有系统管理员可以创建表空间，但是可以通过OWNER子句把表空间的所有权赋给其他非系统管理员。
* 取值范围：字符串，已存在的用户。
* **RELATIVE**
* 使用相对路径，LOCATION目录是相对于各个数据库节点数据目录下的。
* 目录层次：数据库节点的数据目录/pg\_location/相对路径
* 相对路径最多指定两层。
* **LOCATION directory**
* 用于表空间的目录，对于目录有如下要求：
* Vastbase系统用户必须对该目录拥有读写权限，并且目录为空。如果该目录不存在，将由系统自动创建。
* 目录必须是绝对路径，目录中不得含有特殊字符（如$、> 等）。
* 目录不允许指定在数据库数据目录下。
* 目录需为本地路径。
* 取值范围：字符串，有效的目录。
* **MAXSIZE 'space\_size'**
* 指定表空间在单个数据库节点上的最大值。
* 取值范围：字符串格式为正整数+单位，单位当前支持K/M/G/T/P。解析后的数值以K为单位，且范围不能够超过64比特表示的有符号整数，即1KB~9007199254740991KB。
* **random\_page\_cost**
* 指定随机读取page的开销。
* 取值范围：0~1.79769e+308。
* 默认值：使用GUC参数random\_page\_cost的值。
* **seq\_page\_cost**
* 指定顺序读取page的开销。
* 取值范围：0~1.79769e+308。
* 默认值：使用GUC参数seq\_page\_cost的值。

**示例**

--创建表空间。  
vastbase=# CREATE TABLESPACE ds\_location1 RELATIVE LOCATION 'tablespace/tablespace\_1';  
  
--创建用户joe。  
vastbase=# CREATE ROLE joe IDENTIFIED BY 'xxxxxxxxx';  
  
--创建用户jay。  
vastbase=# CREATE ROLE jay IDENTIFIED BY 'xxxxxxxxx';  
  
--创建表空间，且所有者指定为用户joe。  
vastbase=# CREATE TABLESPACE ds\_location2 OWNER joe RELATIVE LOCATION 'tablespace/tablespace\_1';  
  
--把表空间ds\_location1重命名为ds\_location3。  
vastbase=# ALTER TABLESPACE ds\_location1 RENAME TO ds\_location3;  
  
--改变表空间ds\_location2的所有者。  
vastbase=# ALTER TABLESPACE ds\_location2 OWNER TO jay;  
  
--删除表空间。  
vastbase=# DROP TABLESPACE ds\_location2;  
vastbase=# DROP TABLESPACE ds\_location3;  
  
--删除用户。  
vastbase=# DROP ROLE joe;  
vastbase=# DROP ROLE jay;

**相关链接**

参考：SQL语法参考->SQL语法->CREATE DATABASE、CREATE TABLE、CREATE INDEX、DROP TABLESPACE、ALTER TABLESPACE章节。

**优化建议**

* create tablespace
* 不建议在事务内部创建表空间。

### CREATE TEXT SEARCH CONFIGURATION

**功能描述**

创建新的文本搜索配置。一个文本搜索配置声明一个能将一个字符串划分成符号的文本搜索解析器，加上可以用于确定搜索对哪些标记感兴趣的字典。

**注意事项**

* 若仅声明分析器，那么新的文本搜索配置初始没有从符号类型到词典的映射， 因此会忽略所有的单词。后面必须调用ALTER TEXT SEARCH CONFIGURATION命令创建映射使配置生效。如果声明了COPY选项，那么会自动拷贝指定的文本搜索配置的解析器、映射、配置选项等信息。
* 若模式名称已给出，那么文本搜索配置会在声明的模式中创建。否则会在当前模式创建。
* 定义文本搜索配置的用户成为其所有者。
* PARSER和COPY选项是互相排斥的，因为当一个现有配置被复制，其分析器配置也被复制了。
* 若仅声明分析器，那么新的文本搜索配置初始没有从符号类型到词典的映射， 因此会忽略所有的单词。

**语法格式**

CREATE TEXT SEARCH CONFIGURATION name   
 ( PARSER = parser\_name | COPY = source\_config )  
 [ WITH ( {configuration\_option = value} [, ...] )];

**参数说明**

* **name**
* 要创建的文本搜索配置的名称。该名称可以有模式修饰。
* **parser\_name**
* 用于该配置的文本搜索分析器的名称。
* **source\_config**
* 要复制的现有文本搜索配置的名称。
* **configuration\_option**
* 文本搜索配置的配置参数，主要是针对parser\_name执行的解析器或者source\_config隐含的解析器而言的。
* 取值范围：目前共支持default、ngram两种类型的解析器，其中default类型的解析器没有对应的configuration\_option、ngram类型解析器对应的configuration\_option如[表1](#Xf01527fb0a345a19b76e1d182d0050337a9b4b9)所示。
* **表 1** ngram类型解析器对应的配置参数

| **解析器** | **配置参数** | **参数描述** | **取值范围** |
| --- | --- | --- | --- |
| ngram | gram\_size | 分词长度。 | 正整数，1~4  默认值：2 |
| punctuation\_ignore | 是否忽略标点符号。 | true（默认值）：忽略标点符号。  false：不忽略标点符号。 |
| grapsymbol\_ignore | 是否忽略图形化字符。 | true：忽略图形化字符。  false（默认值）：不忽略图形化字符。 |

**示例**

1. 创建文本搜索配置。

vastbase=# CREATE TEXT SEARCH CONFIGURATION ngram2 (parser=ngram) WITH (gram\_size = 2, grapsymbol\_ignore = false);

1. 创建文本搜索配置。

vastbase=# CREATE TEXT SEARCH CONFIGURATION ngram3 (copy=ngram2) WITH (gram\_size = 2, grapsymbol\_ignore = false);

1. 添加类型映射。

vastbase=# ALTER TEXT SEARCH CONFIGURATION ngram2 ADD MAPPING FOR multisymbol WITH simple;

1. 创建用户joe。

vastbase=# CREATE USER joe IDENTIFIED BY 'xxxxxxxxx';

1. 修改文本搜索配置的所有者。

vastbase=# ALTER TEXT SEARCH CONFIGURATION ngram2 OWNER TO joe;

1. 修改文本搜索配置的schema。

vastbase=# ALTER TEXT SEARCH CONFIGURATION ngram2 SET SCHEMA joe;

1. 重命名文本搜索配置。

vastbase=# ALTER TEXT SEARCH CONFIGURATION joe.ngram2 RENAME TO ngram\_2;

1. 删除类型映射。

vastbase=# ALTER TEXT SEARCH CONFIGURATION joe.ngram\_2 DROP MAPPING IF EXISTS FOR multisymbol;

1. 删除文本搜索配置。

vastbase=# DROP TEXT SEARCH CONFIGURATION joe.ngram\_2;  
vastbase=# DROP TEXT SEARCH CONFIGURATION ngram3;

1. -删除Schema及用户joe。

vastbase=# DROP SCHEMA IF EXISTS joe CASCADE;  
vastbase=# DROP ROLE IF EXISTS joe;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TEXT SEARCH CONFIGURATION、DROP TEXT SEARCH CONFIGURATION章节。

### CREATE TEXT SEARCH DICTIONARY

**功能描述**

创建一个新的全文检索词典。词典是一种指定在全文检索时识别特定词并处理的方法。

词典的创建依赖于预定义模板（在系统表PG\_TS\_TEMPLATE中定义），支持创建五种类型的词典，分别是Simple、Ispell、Synonym、Thesaurus以及Snowball，每种类型的词典可以完成不同的任务。

**注意事项**

* 具有SYSADMIN权限的用户可以执行创建词典操作，创建该词典的用户自动成为其所有者。
* 临时模式（pg\_temp）下不允许创建词典。
* 创建或修改词典之后，任何对于用户自定义的词典定义文件的修改，将不会影响到数据库中的词典。如果需要在数据库中使用这些修改，需使用ALTER语句更新对应词典的定义文件。

**语法格式**

CREATE TEXT SEARCH DICTIONARY name (  
 TEMPLATE = template  
 [, option = value [, ... ]]  
);

**参数说明**

* **name**
* 要创建的词典的名称（可指定模式名，否则在当前模式下创建）。
* 取值范围：符合标识符命名规范的字符串，且最大长度不超过63个字符。
* **template**
* 模板名。
* 取值范围：系统表PG\_TS\_TEMPLATE中定义的模板：Simple/Synonym/Thesaurus/Ispell/Snowball。
* **option**
* 参数名。与template值对应，不同的词典模板具有不同的参数列表，且与指定顺序无关。
* Simple词典对应的option
* **STOPWORDS**

停用词表文件名，默认后缀名为stop。停用词文件格式为一组word列表，每行定义一个停用词。词典处理时，文件中的空行和空格会被忽略，并将stopword词组转换为小写形式。

* **ACCEPT**

是否将非停用词设置为已识别。默认值为true。

当Simple词典设置参数ACCEPT=true时，将不会传递任何token给后继词典，此时建议将其放置在词典列表的最后。反之，当ACCEPT=false时，建议将该Simple词典放置在列表中的至少一个词典之前。

* **FILEPATH**

词典文件所在目录。目录可以指定为本地目录和OBS目录（只能在安全模式下指定OBS目录，通过启动时添加securitymode选项进入安全模式）。其中，本地目录格式为“file://absolute\_path”，OBS目录格式为“obs://bucket\_name/path accesskey=ak secretkey=sk region=rg”。默认值为预定义词典文件所在目录。FILEPATH参数必须和STOPWORDS参数同时指定，不允许单独指定。

* Synonym词典对应的option
* **SYNONYM**
* 同义词词典的定义文件名，默认后缀名为syn。

文件格式为一组同义词列表，每行格式为“token synonym”，即token和其对应的synonym，中间以空格相连。

* **CASESENSITIVE**

设置是否大小写敏感，默认值为false，此时词典文件中的token和synonym均会转为小写形式处理。如果设置为true，则不会进行小写转换。

* **FILEPATH**

同义词词典文件所在目录。目录可以指定为本地目录和OBS目录两种形式（只能在安全模式下指定OBS目录，通过启动时添加securitymode选项进入安全模式）。其中，本地目录格式为“file://absolute\_path”，OBS目录格式为“obs://bucket\_name/path accesskey=ak secretkey=sk region=rg”。默认值为预定义词典文件所在目录。

* Thesaurus词典对应的option
* **DICTFILE**
* 词典定义文件名，默认后缀名为ths。

文件格式为一组同义词列表，每行格式为“sample words : indexed words”，中间冒号（:）作为短语和其替换词间的分隔符。TZ词典处理时，如果有多个匹配的sample words，将选择最长匹配输出。

* **DICTIONARY**

用于词规范化的子词典名，必须且仅能定义一个。该词典必须是已经存在的，在检查短语匹配之前使用，用于识别和规范输入文本。

如果子词典无法识别输入词，将会报错。此时，需要移除该词或者更新子词典使其识别。此外，可在indexed words的开头放上一个星号（\*）来跳过在其上应用子词典，但是所有sample words必须可以被子词典识别。

如果词典文件定义的sample words中，含有子词典中定义的停用词，需要用问号（？）替代停用词。假设a和the是子词典中所定义的停用词，如(? one ? two : swsw )。

上述同义词组定义会匹配“a one the two”以及“the one a two”，这两个短语均会被swsw替代输出。

* **FILEPATH**

词典定义文件所在目录。目录可以指定为本地目录和OBS目录两种形式（只能在安全模式下指定OBS目录，通过启动时添加securitymode选项进入安全模式）。其中，本地目录格式为“file://absolute\_path”，OBS目录格式为“obs://bucket\_name/path accesskey=ak secretkey=sk region=rg”。默认值为预定义词典文件所在目录。

* Ispell词典
* **DICTFILE**

词典定义文件名，默认后缀名为dict。

* **AFFFILE**

词缀文件名，默认后缀名为affix。

* **STOPWORDS**

停用词文件名，默认后缀名为stop，文件格式要求与Simple类型 词典的停用词文件相同。

* **FILEPATH**

词典文件所在目录。可以指定为本地目录和OBS目录两种形式 （只能在安全模式下指定OBS目录，通过启动时添加 securitymode选项进入安全模式）。其中，本地目录格式为 “file://absolute\_path”，OBS目录格式为“obs://bucket\_name/path accesskey=ak secretkey=sk region=rg”。默认值为预定义词典文件 所在目录。

* Snowball词典
* **LANGUAGE**

语言名，标识使用哪种语言的词干分析算法。算法按照对应语言 中的拼写规则，缩减输入词的常见变体形式为一个基础词或词 干。

* **STOPWORDS**

停用词表文件名，默认后缀名为stop，文件格式要求与Simple类 型词典的停用词文件相同。

* **FILEPATH**
* 词典定义文件所在目录。可以指定为本地目录或者OBS目录（只能在安全模式下指定OBS目录，通过启动时添加securitymode选项进入安全模式）。其中，本地目录格式为“file://absolute\_path”，OBS目录格式为“obs://bucket\_name/path accesskey=ak secretkey=sk region=rg”。默认值为预定义词典文件所在目录。FILEPATH参数必须和STOPWORDS参数同时指定，不允许单独指定。

fig: **说明：**   
词典定义文件的文件名仅支持小写字母、数据、下划线混合。

* **value**
* 参数值。如果不是简单的标识符或数字，则参数值必须加单引号（标示符和数字同样可以加上单引号）。

**示例**

参考：SQL语法参考->全文检索->词典->配置示例章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TEXT SEARCH DICTIONARY、CREATE TEXT SEARCH DICTIONARY章节。

### CREATE TRIGGER

**功能描述**

创建一个触发器。 触发器将与指定的表或视图关联，并在特定条件下执行指定的函数。

**注意事项**

* 当前仅支持在普通行存表上创建触发器，不支持在列存表、临时表、unlogged表等类型表上创建触发器。
* 如果为同一事件定义了多个相同类型的触发器，则按触发器的名称字母顺序触发它们。
* 触发器常用于多表间数据关联同步场景，对SQL执行性能影响较大，不建议在大数据量同步及对性能要求高的场景中使用。

**语法格式**

CREATE [ CONSTRAINT ] TRIGGER trigger\_name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }  
 ON table\_name  
 [ FROM referenced\_table\_name ]  
 { NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } }  
 [ FOR [ EACH ] { ROW | STATEMENT } ]  
 [ WHEN ( condition ) ]  
 EXECUTE PROCEDURE function\_name ( arguments );

其中event包含以下几种：

INSERT  
 UPDATE [ OF column\_name [, ... ] ]  
 DELETE  
 TRUNCATE

**参数说明**

* **CONSTRAINT**
* 可选项，指定此参数将创建约束触发器，即触发器作为约束来使用。除了可以使用SET CONSTRAINTS调整触发器触发的时间之外，这与常规触发器相同。 约束触发器必须是AFTER ROW触发器。
* **trigger\_name**
* 触发器名称，该名称不能限定模式，因为触发器自动继承其所在表的模式，且同一个表的触发器不能重名。 对于约束触发器，使用SET CONSTRAINTS修改触发器行为时也使用此名称。
* 取值范围：符合标识符命名规范的字符串，且最大长度不超过63个字符。
* **BEFORE**
* 触发器函数是在触发事件发生前执行。
* **AFTER**
* 触发器函数是在触发事件发生后执行，约束触发器只能指定为AFTER。
* **INSTEAD OF**
* 触发器函数直接替代触发事件。
* **event**
* 启动触发器的事件，取值范围包括：INSERT、UPDATE、DELETE或TRUNCATE，也可以通过OR同时指定多个触发事件。
* 对于UPDATE事件类型，可以使用下面语法指定列：

UPDATE OF column\_name1 [, column\_name2 ... ]

表示当这些列作为UPDATE语句的目标列时，才会启动触发器，但是INSTEAD OF UPDATE类型不支持指定列信息。

* **table\_name**
* 需要创建触发器的表名称。
* 取值范围：数据库中已经存在的表名称。
* **referenced\_table\_name**
* 约束引用的另一个表的名称。 只能为约束触发器指定，常见于外键约束。
* 取值范围：数据库中已经存在的表名称。
* **DEFERRABLE | NOT DEFERRABLE**
* 约束触发器的启动时机，仅作用于约束触发器。这两个关键字设置该约束是否可推迟。
* 详细介绍请参见：SQL语法参考->SQL语法->CREATE TABLE章节。
* **INITIALLY IMMEDIATE** **| INITIALLY DEFERRED**
* 如果约束是可推迟的，则这个子句声明检查约束的缺省时间，仅作用于约束触发器。
* 详细介绍请参见：SQL语法参考->SQL语法->CREATE TABLE章节。
* **FOR EACH ROW | FOR EACH STATEMENT**
* 触发器的触发频率。
* FOR EACH ROW是指该触发器是受触发事件影响的每一行触发一次。
* FOR EACH STATEMENT是指该触发器是每个SQL语句只触发一次。
* 未指定时默认值为FOR EACH STATEMENT。约束触发器只能指定为FOR EACH ROW。
* **condition**
* 决定是否实际执行触发器函数的条件表达式。当指定WHEN时，只有在条件返回true时才会调用该函数。
* 在FOR EACH ROW触发器中，WHEN条件可以通过分别写入OLD.column\_name或NEW.column\_name来引用旧行或新行值的列。 当然，INSERT触发器不能引用OLD和DELETE触发器不能引用NEW。
* INSTEAD OF触发器不支持WHEN条件。
* WHEN表达式不能包含子查询。
* 对于约束触发器，WHEN条件的评估不会延迟，而是在执行更新操作后立即发生。 如果条件返回值不为true，则触发器不会排队等待延迟执行。
* **function\_name**
* 用户定义的函数，必须声明为不带参数并返回类型为触发器，在触发器触发时执行。
* **arguments**
* 执行触发器时要提供给函数的可选的以逗号分隔的参数列表。参数是文字字符串常量，简单的名称和数字常量也可以写在这里，但它们都将被转换为字符串。 请检查触发器函数的实现语言的描述，以了解如何在函数内访问这些参数。
* fig: **说明：**   
  关于触发器种类：
* INSTEAD OF的触发器必须标记为FOR EACH ROW，并且只能在视图上定义。
* BEFORE和AFTER触发器作用在视图上时，只能标记为FOR EACH STATEMENT。
* TRUNCATE类型触发器仅限FOR EACH STATEMENT。

**表 1** 表和视图上支持的触发器种类：

| **触发时机** | **触发事件** | **行级** | **语句级** |
| --- | --- | --- | --- |
| BEFORE | INSERT/UPDATE/DELETE | 表 | 表和视图 |
| TRUNCATE | 不支持 | 表 |
| AFTER | INSERT/UPDATE/DELETE | 表 | 表和视图 |
| TRUNCATE | 不支持 | 表 |
| INSTEAD OF | INSERT/UPDATE/DELETE | 视图 | 不支持 |
| TRUNCATE | 不支持 | 不支持 |



**表 2** PLPGSQL类型触发器函数特殊变量：

| **变量名** | **变量含义** |
| --- | --- |
| NEW | INSERT及UPDATE操作涉及tuple信息中的新值，对DELETE为空。 |
| OLD | UPDATE及DELETE操作涉及tuple信息中的旧值，对INSERT为空。 |
| TG\_NAME | 触发器名称。 |
| TG\_WHEN | 触发器触发时机（BEFORE/AFTER/INSTEAD OF）。 |
| TG\_LEVEL | 触发频率（ROW/STATEMENT）。 |
| TG\_OP | 触发操作（INSERT/UPDATE/DELETE/TRUNCATE）。 |
| TG\_RELID | 触发器所在表OID。 |
| TG\_RELNAME | 触发器所在表名（已废弃，现用TG\_TABLE\_NAME替代）。 |
| TG\_TABLE\_NAME | 触发器所在表名。 |
| TG\_TABLE\_SCHEMA | 触发器所在表的SCHEMA信息。 |
| TG\_NARGS | 触发器函数参数个数。 |
| TG\_ARGV[] | 触发器函数参数列表。 |

**示例**

--创建源表及触发表  
vastbase=# CREATE TABLE test\_trigger\_src\_tbl(id1 INT, id2 INT, id3 INT);  
vastbase=# CREATE TABLE test\_trigger\_des\_tbl(id1 INT, id2 INT, id3 INT);  
  
--创建触发器函数  
vastbase=# CREATE OR REPLACE FUNCTION tri\_insert\_func() RETURNS TRIGGER AS  
 $$  
 DECLARE  
 BEGIN  
 INSERT INTO test\_trigger\_des\_tbl VALUES(NEW.id1, NEW.id2, NEW.id3);  
 RETURN NEW;  
 END  
 $$ LANGUAGE PLPGSQL;  
  
vastbase=# CREATE OR REPLACE FUNCTION tri\_update\_func() RETURNS TRIGGER AS  
 $$  
 DECLARE  
 BEGIN  
 UPDATE test\_trigger\_des\_tbl SET id3 = NEW.id3 WHERE id1=OLD.id1;  
 RETURN OLD;  
 END  
 $$ LANGUAGE PLPGSQL;  
  
vastbase=# CREATE OR REPLACE FUNCTION TRI\_DELETE\_FUNC() RETURNS TRIGGER AS  
 $$  
 DECLARE  
 BEGIN  
 DELETE FROM test\_trigger\_des\_tbl WHERE id1=OLD.id1;  
 RETURN OLD;  
 END  
 $$ LANGUAGE PLPGSQL;  
  
--创建INSERT触发器  
vastbase=# CREATE TRIGGER insert\_trigger  
 BEFORE INSERT ON test\_trigger\_src\_tbl  
 FOR EACH ROW  
 EXECUTE PROCEDURE tri\_insert\_func();  
  
--创建UPDATE触发器  
vastbase=# CREATE TRIGGER update\_trigger  
 AFTER UPDATE ON test\_trigger\_src\_tbl   
 FOR EACH ROW  
 EXECUTE PROCEDURE tri\_update\_func();  
  
--创建DELETE触发器  
vastbase=# CREATE TRIGGER delete\_trigger  
 BEFORE DELETE ON test\_trigger\_src\_tbl  
 FOR EACH ROW  
 EXECUTE PROCEDURE tri\_delete\_func();  
  
--执行INSERT触发事件并检查触发结果  
vastbase=# INSERT INTO test\_trigger\_src\_tbl VALUES(100,200,300);  
vastbase=# SELECT \* FROM test\_trigger\_src\_tbl;  
vastbase=# SELECT \* FROM test\_trigger\_des\_tbl; //查看触发操作是否生效。  
  
--执行UPDATE触发事件并检查触发结果  
vastbase=# UPDATE test\_trigger\_src\_tbl SET id3=400 WHERE id1=100;  
vastbase=# SELECT \* FROM test\_trigger\_src\_tbl;  
vastbase=# SELECT \* FROM test\_trigger\_des\_tbl; //查看触发操作是否生效  
  
--执行DELETE触发事件并检查触发结果  
vastbase=# DELETE FROM test\_trigger\_src\_tbl WHERE id1=100;  
vastbase=# SELECT \* FROM test\_trigger\_src\_tbl;  
vastbase=# SELECT \* FROM test\_trigger\_des\_tbl; //查看触发操作是否生效  
  
--修改触发器  
vastbase=# ALTER TRIGGER delete\_trigger ON test\_trigger\_src\_tbl RENAME TO delete\_trigger\_renamed;  
  
--禁用insert\_trigger触发器  
vastbase=# ALTER TABLE test\_trigger\_src\_tbl DISABLE TRIGGER insert\_trigger;   
  
--禁用当前表上所有触发器  
vastbase=# ALTER TABLE test\_trigger\_src\_tbl DISABLE TRIGGER ALL;   
  
--删除触发器  
vastbase=# DROP TRIGGER insert\_trigger ON test\_trigger\_src\_tbl;  
vastbase=# DROP TRIGGER update\_trigger ON test\_trigger\_src\_tbl;  
vastbase=# DROP TRIGGER delete\_trigger\_renamed ON test\_trigger\_src\_tbl;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TRIGGER、DROP TRIGGER、ALTER TABLE章节。

### CREATE TYPE

**功能描述**

在当前数据库中定义一种新的数据类型。定义数据类型的用户将成为该数据类型的拥有者。类型只适用于行存表

有五种形式的CREATE TYPE，分别为：复合类型、基本类型、shell类型、枚举类型和集合类型。

* 复合类型
* 复合类型由一个属性名和数据类型的列表指定。如果属性的数据类型是可排序的，也可以指定该属性的排序规则。复合类型本质上和表的行类型相同，但是如果只想定义一种类型，使用CREATE TYPE避免了创建一个实际的表。单独的复合类型也是很有用的，例如可以作为函数的参数或者返回类型。
* 为了能够创建复合类型，必须拥有在其所有属性类型上的USAGE特权。
* 基本类型
* 用户可以自定义一种新的基本类型（标量类型）。通常来说这些函数必须是底层语言所编写。
* shell类型
* shell类型是一种用于后面要定义的类型的占位符，通过发出一个不带除类型名之外其他参数的CREATE TYPE命令可以创建这种类型。在创建基本类型时，需要shell类型作为一种向前引用。
* 枚举类型
* 由若干个标签构成的列表，每一个标签值都是一个非空字符串，且字符串长度必须不超过63个字节。
* 集合类型
* 类似数组，但是没有长度限制，主要在存储过程中使用。
* 被授予CREATE ANY TYPE权限的用户，可以在public模式和用户模式下创建类型。

**注意事项**

如果给定一个模式名，那么该类型将被创建在指定的模式中。否则它会被创建在当前模式中。类型名称必须与同一个模式中任何现有的类型或者域相区别（因为表具有相关的数据类型，类型名称也必须与同一个模式中任何现有表的名称不同）。

**语法格式**

CREATE TYPE name AS  
 ( [ attribute\_name data\_type [ COLLATE collation ] [, ... ] ] )  
  
CREATE TYPE name (  
 INPUT = input\_function,  
 OUTPUT = output\_function  
 [ , RECEIVE = receive\_function ]  
 [ , SEND = send\_function ]  
 [ , TYPMOD\_IN =  
type\_modifier\_input\_function ]  
 [ , TYPMOD\_OUT =  
type\_modifier\_output\_function ]  
 [ , ANALYZE = analyze\_function ]  
 [ , INTERNALLENGTH = { internallength |  
VARIABLE } ]  
 [ , PASSEDBYVALUE ]  
 [ , ALIGNMENT = alignment ]  
 [ , STORAGE = storage ]  
 [ , LIKE = like\_type ]  
 [ , CATEGORY = category ]  
 [ , PREFERRED = preferred ]  
 [ , DEFAULT = default ]  
 [ , ELEMENT = element ]  
 [ , DELIMITER = delimiter ]  
 [ , COLLATABLE = collatable ]  
)  
  
CREATE TYPE name  
  
CREATE TYPE name AS ENUM  
 ( [ 'label' [, ... ] ] )  
   
CREATE TYPE name AS TABLE OF data\_type

**参数说明**

复合类型

* **name**
* 要创建的类型的名称（可以被模式限定）。
* **attribute\_name**
* 复合类型的一个属性（列）的名称。
* **data\_type**
* 要成为复合类型的一个列的现有数据类型的名称。可以使用%ROWTYPE间接引用表的类型，或者使用%TYPE间接引用表或复合类型中某一列的类型。
* **collation**
* 要关联到复合类型的一列的现有排序规则的名称。排序规则可以使用“select \* from pg\_collation”命令从pg\_collation系统表中查询，默认的排序规则为查询结果中以default开始的行。

基本类型

自定义基本类型时，参数可以以任意顺序出现，input\_function和output\_function为必选参数，其它为可选参数。

* **input\_function**
* 将数据从类型的外部文本形式转换为内部形式的函数名。
* 输入函数可以被声明为有一个cstring类型的参数，或者有三个类型分别为cstring、 oid、integer的参数。
* cstring参数是以C字符串存在的输入文本。
* oid参数是该类型自身的OID（对于数组类型则是其元素类型的OID）。
* integer参数是目标列的typmod（如果知道，不知道则将传递 -1）。
* 输入函数必须返回一个该数据类型本身的值。通常，一个输入函数应该被声明为STRICT。 如果不是这样，在读到一个NULL输入值时，调用输入函数时第一个参数会是NULL。在这种情况下，该函数必须仍然返回NULL，除非调用函数发生了错误（这种情况主要是想支持域输入函数，域输入函数可能需要拒绝NULL输入）。
* fig: **说明：**   
  输入和输出函数能被声明为具有新类型的结果或参数是因为：必须在创建新类型之前创建这两个函数。而新类型应该首先被定义为一种shell type，它是一种占位符类型，除了名称和拥有者之外它没有其他属性。这可以通过不带额外参数的命令CREATE TYPE name做到。然后用C写的I/O函数可以被定义为引用这种shell type。最后，用带有完整定义的CREATE TYPE把该shell type替换为一个完全的、合法的类型定义，之后新类型就可以正常使用了。
* **output\_function**
* 将数据从类型的内部形式转换为外部文本形式的函数名。
* 输出函数必须被声明为有一个新数据类型的参数。输出函数必须返回类型cstring。对于NULL值不会调用输出函数。
* **receive\_function**
* 可选参数。将数据从类型的外部二进制形式转换成内部形式的函数名。
* 如果没有该函数，该类型不能参与到二进制输入中。二进制表达转换成内部形式代价更低，然而却更容易移植（例如，标准的整数数据类型使用网络字节序作为外部二进制表达，而内部表达是机器本地的字节序）。receive\_function应该执行足够的检查以确保该值是有效的。
* 接收函数可以被声明为有一个internal类型的参数，或者有三个类型分别为internal、oid、integer的参数。
* internal参数是一个指向StringInfo缓冲区的指针，其中保存着接收到的字节串。
* oid和integer参数和文本输入函数的相同。
* 接收函数必须返回一个该数据类型本身的值。通常，一个接收函数应该被声明为STRICT。如果不是这样，在读到一个NULL输入值时调用接收函数时第一个参数会是NULL。在这种情况下，该函数必须仍然返回NULL，除非接收函数发生了错误（这种情况主要是想支持域接收函数，域接收函数可能需要拒绝NULL输入）。
* **send\_function**
* 可选参数。将数据从类型的内部形式转换为外部二进制形式的函数名。
* 如果没有该函数，该类型将不能参与到二进制输出中。发送函数必须被声明为有一个新数据类型的参数。发送函数必须返回类型bytea。对于NULL值不会调用发送函数。
* **type\_modifier\_input\_function**
* 可选参数。将类型的修饰符数组转换为内部形式的函数名。
* **type\_modifier\_output\_function**
* 可选参数。将类型的修饰符的内部形式转换为外部文本形式的函数名。
* fig: **说明：**   
  如果该类型支持修饰符（附加在类型声明上的可选约束，例如，char(5)或numeric(30,2)），则需要可选的type\_modifier\_input\_function以及type\_modifier\_output\_function。Vastbase允许用户定义的类型有一个或者多个简单常量或者标识符作为修饰符。不过，为了存储在系统目录中，该信息必须能被打包到一个非负整数值中。所声明的修饰符会被以cstring数组的形式传递给type\_modifier\_input\_function。 type\_modifier\_input\_function必须检查该值的合法性（如果值错误就抛出一个错误），如果值正确，要返回一个非负integer值，该值将被存储在“typmod”列中。如果类型没有 type\_modifier\_input\_function则类型修饰符将被拒绝。type\_modifier\_output\_function把内部的整数typmod值转换回正确的形式用于用户显示。type\_modifier\_output\_function必须返回一个cstring值，该值就是追加到类型名称后的字符串。例如，numeric的函数可能会返回(30,2)。如果默认的显示格式就是只把存储的typmod整数值放在圆括号内，则允许省略type\_modifier\_output\_function。
* **analyze\_function**
* 可选参数。为该数据类型执行统计分析的函数名的可选参数。
* 默认情况下，如果该类型有一个默认的B-tree操作符类，ANALYZE将尝试用类型的“equals”和“less-than”操作符来收集统计信息。这种行为对于非标量类型并不合适，因此可以通过指定一个自定义分析函数来覆盖这种行为。分析函数必须被声明为有一个类型为internal的参数，并且返回一个boolean结果。
* **internallength**
* 可选参数。一个数字常量，用于指定新类型的内部表达的字节长度。默认为变长。
* 虽然只有I/O函数和其他为该类型创建的函数才知道新类型的内部表达的细节， 但是内部表达的一些属性必须被向Vastbase声明。其中最重要的是internallength。基本数据类型可以是定长的（这种情况下internallength是一个正整数）或者是变长的（把internallength设置为VARIABLE，在内部通过把typlen设置为-1表示）。所有变长类型的内部表达都必须以一个4字节整数开始，internallength定义了总长度。
* **PASSEDBYVALUE**
* 可选参数。表示这种数据类型的值需要被传值而不是传引用。传值的类型必须是定长的，并且它们的内部表达不能超过Datum类型（某些机器上是4字节，其他机器上是8字节）的尺寸。
* **alignment**
* 可选参数。该参数指定数据类型的存储对齐需求。如果被指定，必须是char、int2、int4或者double。默认是int4。
* 允许的值等同于以1、2、4或8字节边界对齐。要注意变长类型的alignment参数必须至少为4，因为它们需要包含一个int4作为它们的第一个组成部分。
* **storage**
* 可选参数。该数据类型的存储策略。
* 如果被指定，必须是plain、external、extended或者main。 默认是plain。
* plain指定该类型的数据将总是被存储在线内并且不会被压缩。（对定长类型只允许plain）
* extended 指定系统将首先尝试压缩一个长的数据值，并且将在数据仍然太长的情况下把值移出主表行。
* external允许值被移出主表， 但是系统将不会尝试对它进行压缩。
* main允许压缩，但是不鼓励把值移出主表（如果没有其他办法让行的大小变得合适，具有这种存储策略的数据项仍将被移出主表，但比起extended以及external项来，这种存储策略的数据项会被优先考虑保留在主表中）。
  + 除plain之外所有的storage值都暗示该数据类型的函数能处理被TOAST过的值。指定的值仅仅是决定一种可TOAST数据类型的列的默认TOAST存储策略，用户可以使用ALTER TABLE SET STORAGE为列选取其他策略。
* **like\_type**
* 可选参数。与新类型具有相同表达的现有数据类型的名称。会从这个类型中复制internallength、 passedbyvalue、 alignment以及storage的值（ 除非在这个CREATE TYPE命令的其他地方用显式说明覆盖）。
* 当新类型的低层实现是以一种现有的类型为参考时，用这种方式指定表达特别有用。
* **category**
* 可选参数。这种类型的分类码（一个ASCII 字符）。 默认是“用户定义类型”的'U'。为了创建自定义分类， 也可以选择其他 ASCII字符。
* **preferred**
* 可选参数。如果这种类型是其类型分类中的优先类型则为TRUE，否则为FALSE。默认为假。在一个现有类型分类中创建一种新的优先类型要非常谨慎， 因为这可能会导致很大的改变。
* fig: **说明：**   
  category和preferred参数可以被用来帮助控制在混淆的情况下应用哪一种隐式造型。每一种数据类型都属于一个用单个ASCII 字符命名的分类，并且每一种类型可以是其所属分类中的“首选”。当有助于解决重载函数或操作符时，解析器将优先造型到首选类型（但是只能从同类的其他类型造型）。对于没有隐式转换到或来自任意其他类型的类型，让这些设置保持默认即可。不过，对于有隐式转换的相关类型的组，把它们都标记为属于同一个类别并且选择一种或两种“最常用”的类型作为该类别的首选通常是很有用的。在把一种用户定义的类型增加到一个现有的内建类别（例如，数字或者字符串类型）中时，category参数特别有用。不过，也可以创建新的全部是用户定义类型的类别。对这样的类别，可选择除大写字母之外的任何ASCII 字符。
* **default**
* 可选参数。数据类型的默认值。如果被省略，默认值是空。
* 如果用户希望该数据类型的列被默认为某种非空值，可以指定一个默认值。默认值可以用DEFAULT关键词指定（这样一个默认值可以被附加到一个特定列的显式DEFAULT子句覆盖）。
* **element**
* 可选参数。被创建的类型是一个数组，element指定了数组元素的类型。例如，要定义一个4字节整数的数组（int4）， 应指定ELEMENT = int4。
* **delimiter**
* 可选参数。指定这种类型组成的数组中分隔值的定界符。
* 可以把delimiter设置为一个特定字符，默认的定界符是逗号（,）。注意定界符是与数组元素类型相关的，而不是数组类型本身相关。
* **collatable**
* 可选参数。如果这个类型的操作可以使用排序规则信息，则为TRUE。默认为FALSE。
* 如果collatable为TRUE，这种类型的列定义和表达式可能通过使用COLLATE子句携带有排序规则信息。在该类型上操作的函数的实现负责真正利用这些信息，仅把类型标记为可排序的并不会让它们自动地去使用这类信息。
* **lable**
* 可选参数。与枚举类型的一个值相关的文本标签，其值为长度不超过63个字符的非空字符串。

fig: **说明：**   
在创建用户定义类型的时候， Vastbase会自动创建一个与之关联的数组类型，其名称由该元素类型的名称前缀一个下划线组成。

**示例**

--创建一种复合类型，建表并插入数据以及查询。  
vastbase=# CREATE TYPE compfoo AS (f1 int, f2 text);  
vastbase=# CREATE TABLE t1\_compfoo(a int, b compfoo);  
vastbase=# CREATE TABLE t2\_compfoo(a int, b compfoo);  
vastbase=# INSERT INTO t1\_compfoo values(1,(1,'demo'));  
vastbase=# INSERT INTO t2\_compfoo select \* from t1\_compfoo;  
vastbase=# SELECT (b).f1 FROM t1\_compfoo;  
vastbase=# SELECT \* FROM t1\_compfoo t1 join t2\_compfoo t2 on (t1.b).f1=(t1.b).f1;  
  
--创建一个集合类型  
vastbase=# CREATE TYPE compfoo\_table AS TABLE OF compfoo;  
  
--重命名数据类型。  
vastbase=# ALTER TYPE compfoo RENAME TO compfoo1;  
  
--要改变一个用户定义类型compfoo1的所有者为usr1。  
vastbase=# CREATE USER usr1 PASSWORD 'xxxxxxxxx';  
vastbase=# ALTER TYPE compfoo1 OWNER TO usr1;  
  
--把用户定义类型compfoo1的模式改变为usr1。  
vastbase=# ALTER TYPE compfoo1 SET SCHEMA usr1;  
  
--给一个数据类型增加一个新的属性。  
vastbase=# ALTER TYPE usr1.compfoo1 ADD ATTRIBUTE f3 int;  
  
--删除compfoo1类型。  
vastbase=# DROP TYPE usr1.compfoo1 cascade;  
  
--删除相关表和用户。  
vastbase=# DROP TABLE t1\_compfoo;  
vastbase=# DROP TABLE t2\_compfoo;  
vastbase=# DROP SCHEMA usr1;  
vastbase=# DROP USER usr1;  
  
--创建一个枚举类型。  
vastbase=# CREATE TYPE bugstatus AS ENUM ('create', 'modify', 'closed');  
  
--添加一个标签值。  
vastbase=# ALTER TYPE bugstatus ADD VALUE IF NOT EXISTS 'regress' BEFORE 'closed';  
  
--重命名一个标签值。  
vastbase=# ALTER TYPE bugstatus RENAME VALUE 'create' TO 'new';

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TYPE、DROP TYPE章节。

### CREATE USER

**功能描述**

创建一个用户。

**注意事项**

* 通过CREATE USER创建的用户，默认具有LOGIN权限。
* 通过CREATE USER创建用户的同时，系统会在执行该命令的数据库中，为该用户创建一个同名的SCHEMA。
* 系统管理员在普通用户同名schema下创建的对象，所有者为schema的同名用户（非系统管理员）。

**语法格式**

CREATE USER user\_name [ [ WITH ] option [ ... ] ] [ ENCRYPTED | UNENCRYPTED ] { PASSWORD | IDENTIFIED BY } { 'password' [ EXPIRED ] | DISABLE };

其中option子句用于设置权限及属性等信息。

{SYSADMIN | NOSYSADMIN}  
 | {MONADMIN | NOMONADMIN}  
 | {OPRADMIN | NOOPRADMIN}  
 | {POLADMIN | NOPOLADMIN}  
 | {AUDITADMIN | NOAUDITADMIN}  
 | {CREATEDB | NOCREATEDB}  
 | {USEFT | NOUSEFT}  
 | {CREATEROLE | NOCREATEROLE}  
 | {INHERIT | NOINHERIT}  
 | {LOGIN | NOLOGIN}  
 | {REPLICATION | NOREPLICATION}  
 | {INDEPENDENT | NOINDEPENDENT}  
 | {VCADMIN | NOVCADMIN}  
 | {PERSISTENCE | PERSISTENCE}  
 | CONNECTION LIMIT connlimit  
 | VALID BEGIN 'timestamp'  
 | VALID UNTIL 'timestamp'  
 | RESOURCE POOL 'respool'  
 | USER GROUP 'groupuser'  
 | PERM SPACE 'spacelimit'  
 | TEMP SPACE 'tmpspacelimit'  
 | SPILL SPACE 'spillspacelimit'  
 | NODE GROUP logic\_cluster\_name  
 | IN ROLE role\_name [, ...]  
 | IN GROUP role\_name [, ...]  
 | ROLE role\_name [, ...]  
 | ADMIN role\_name [, ...]  
 | USER role\_name [, ...]  
 | SYSID uid  
 | DEFAULT TABLESPACE tablespace\_name  
 | PROFILE DEFAULT  
 | PROFILE profile\_name  
 | PGUSER

**参数说明**

* **user\_name**
* 用户名称。
* 取值范围：字符串，要符合标识符的命名规范。且最大长度不超过63个字符。
* **password**
* 登录密码。
* 密码规则如下：
* 密码默认不少于8个字符。
* 不能与用户名及用户名倒序相同。
* 至少包含大写字母（A-Z）、小写字母（a-z）、数字（0-9）、非字母数字字符（限定为~!@#$%^&\*()-\_=+|[{}];:,<.>/?）四类字符中的三类字符。
* 密码也可以是符合格式要求的密文字符串，这种情况主要用于用户数据导入场景，不推荐用户直接使用。如果直接使用密文密码，用户需要知道密文密码对应的明文，并且保证明文密码复杂度，数据库不会校验密文密码复杂度，直接使用密文密码的安全性由用户保证。
* 创建用户时，应当使用双引号或单引号将用户密码括起来。
* 取值范围：字符串。
* **groupuser**
* 用户组名称。
* 取值范围：字符串。
* **logic\_cluster\_name**
* 节点组名称。
* 取值范围：字符串。

CREATE USER的其他参数值请参考：SQL语法参考->SQL语法->CREATE ROLE章节。

**示例**

--创建用户jim，登录密码为xxxxxxxxx。  
vastbase=# CREATE USER jim PASSWORD 'xxxxxxxxx';  
  
--下面语句与上面的等价。  
vastbase=# CREATE USER kim IDENTIFIED BY 'xxxxxxxxx';  
  
--如果创建有“创建数据库”权限的用户，则需要加CREATEDB关键字。  
vastbase=# CREATE USER dim CREATEDB PASSWORD 'xxxxxxxxx';  
  
--将用户jim的登录密码由xxxxxxxxx修改为Abcd@123。  
vastbase=# ALTER USER jim IDENTIFIED BY 'Abcd@123' REPLACE 'xxxxxxxxx';  
  
--为用户jim追加CREATEROLE权限。  
vastbase=# ALTER USER jim CREATEROLE;  
  
--将enable\_seqscan的值设置为on， 设置成功后，在下一会话中生效。  
vastbase=# ALTER USER jim SET enable\_seqscan TO on;  
  
--重置jim的enable\_seqscan参数。  
vastbase=# ALTER USER jim RESET enable\_seqscan;  
  
--锁定jim帐户。  
vastbase=# ALTER USER jim ACCOUNT LOCK;  
  
--删除用户。  
vastbase=# DROP USER kim CASCADE;  
vastbase=# DROP USER jim CASCADE;  
vastbase=# DROP USER dim CASCADE;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER USER、CREATE ROLE、DROP USER章节。

### CREATE USER MAPPING

**功能描述**

定义一个用户到一个外部服务器的新映射。

**注意事项**

当在OPTIONS中出现password选项时，需要保证Vastbase每个节点的$GAUSSHOME/bin目录下存在usermapping.key.cipher和usermapping.key.rand文件，如果不存在这两个文件，请使用vb\_guc工具生成并发布到Vastbase每个节点的$GAUSSHOME/bin目录下。

**语法格式**

CREATE USER MAPPING FOR { user\_name | USER | CURRENT\_USER | PUBLIC }  
 SERVER server\_name  
 [ OPTIONS ( option 'value' [ , ... ] ) ]

**参数说明**

* **user\_name**
* 要映射到外部服务器的一个现有用户的名称。
* CURRENT\_USER和USER匹配当前用户的名称。 当PUBLIC被指定时，一个公共映射会被创建，当没有特定用户的映射可用时将会使用它。
* **server\_name**
* 将为其创建用户映射的现有服务器的名称。
* **OPTIONS ( { option\_name ' value ' } [, ...] )**
* 这个子句指定用户映射的选项。这些选项通常定义该映射实际的用户名和口令。选项名必须唯一。允许的选项名和值与该服务器的外部数据包装器有关。
* fig: **说明：**
* 用户的口令会加密后保存到系统表PG\_USER\_MAPPING中，加密时需要使用usermapping.key.cipher和usermapping.key.rand作为加密密码文件和加密因子。首次使用前需要通过如下命令创建这两个文件，并将这两个文件放入各节点目录$GAUSSHOME/bin，且确保具有读权限。

vb\_guc generate -o usermapping -S default -D $GAUSSHOME/bin

* 其中-S参数指定default时会随机生成密码，用户也可为-S参数指定密码，此密码用于保证生成密码文件的安全性和唯一性，用户无需保存或记忆。其他参数请参考“工具参考中vb\_guc工具说明”。
* oracle\_fdw支持的options包括：
* user

oracle server的用户名。

* password
  + - oracle用户对应的密码。
* mysql\_fdw支持的options包括：
* username
  + - MySQL Server/MariaDB的用户名。
* password
  + - MySQL Server/MariaDB用户对应的密码。
* postgres\_fdw支持的options包括：
* user
  + - 远端Vastbase的用户名。
* password
  + - 远端Vastbase用户对应的密码。
    - fig: **说明：**   
      Vastbase在后台会对用户输入的password加密以保证安全性。该加密所需密钥文件需要使用vb\_guc工具生成，并被发布到Vastbase每个节点的$GAUSSHOME/bin目录下。password不应当包含'encryptOpt'前缀，否则会被认为是加密后的密文。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER USER MAPPING、DROP USER MAPPING章节。

### CREATE VIEW

**功能描述**

创建一个视图。视图与基本表不同，是一个虚拟的表。数据库中仅存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。若基本表中的数据发生变化，从视图中查询出的数据也随之改变。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据及变化。

**注意事项**

被授予CREATE ANY TABLE权限的用户，可以在public模式和用户模式下创建视图。

**语法格式**

CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW view\_name [ ( column\_name [, ...] ) ]  
 [ WITH ( {view\_option\_name [= view\_option\_value]} [, ... ] ) ]  
 AS query;

fig: **说明：**   
创建视图时使用WITH(security\_barrier)可以创建一个相对安全的视图，避免攻击者利用低成本函数的RAISE语句打印出隐藏的基表数据。

**参数说明**

* **OR REPLACE**
* 如果视图已存在，则重新定义。
* **TEMP | TEMPORARY**
* 创建临时视图。
* **view\_name**
* 要创建的视图名称。可以用模式修饰。
* 取值范围：字符串，符合标识符命名规范。
* **column\_name**
* 可选的名称列表，用作视图的字段名。如果没有给出，字段名取自查询中的字段名。
* 取值范围：字符串，符合标识符命名规范。
* **view\_option\_name [= view\_option\_value]**
* 该子句为视图指定一个可选的参数。
* 目前view\_option\_name支持的参数仅有security\_barrier，当VIEW试图提供行级安全时，应使用该参数。
* 取值范围：Boolean类型，TRUE、FALSE
* **query**
* 为视图提供行和列的SELECT或VALUES语句。

**示例**

--创建字段spcname为pg\_default组成的视图。  
vastbase=# CREATE VIEW myView AS  
 SELECT \* FROM pg\_tablespace WHERE spcname = 'pg\_default';  
  
--查看视图。  
vastbase=# SELECT \* FROM myView ;  
  
--删除视图myView。  
vastbase=# DROP VIEW myView;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER VIEW、DROP VIEW章节。

### CREATE WEAK PASSWORD DICTIONARY

**功能描述**

向gs\_global\_config表中插入一个或者多个弱口令。

**注意事项**

* 只有初始用户、系统管理员和安全管理员拥有权限执行本语法。
* 弱口令字典中的口令存放在gs\_global\_config系统表中。
* 弱口令字典默认为空，用户通过本语法可以新增一条或多条弱口令。
* 当用户尝试通过本语法插入gs\_global\_config表中已存在的弱口令时，会只在表中保留一条该弱口令。

**语法格式**

CREATE WEAK PASSWORD DICTIONARY   
 [WITH VALUES] ( {'weak\_password'} [, ...] );

**参数说明**

**weak\_password**

弱口令。

范围：字符串。

**示例**

--向gs\_global\_config系统表中插入单个弱口令。   
vastbase=# CREATE WEAK PASSWORD DICTIONARY WITH VALUES ('password1');   
  
--向gs\_global\_config系统表中插入多个弱口令。   
vastbase=# CREATE WEAK PASSWORD DICTIONARY WITH VALUES ('password2'),('password3');   
  
--清空gs\_global\_config系统表中所有弱口令。   
vastbase=# DROP WEAK PASSWORD DICTIONARY;   
  
--查看现有弱口令。   
vastbase=# SELECT \* FROM gs\_global\_config WHERE NAME LIKE 'weak\_password';

**相关链接**

参考：SQL语法参考->SQL语法->DROP WEAK PASSWORD DICTIONARY章节。

### CURSOR

**功能描述**

CURSOR命令定义一个游标，用于在一个大的查询里面检索少数几行数据。

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

**注意事项**

* 游标命令只能在事务块里使用。
* 通常游标和SELECT一样返回文本格式。因为数据在系统内部是用二进制格式存储的，系统必须对数据做一定转换以生成文本格式。一旦数据是以文本形式返回，客户端应用需要把它们转换成二进制进行操作。使用FETCH语句，游标可以返回文本或二进制格式。
* 应该小心使用二进制游标。文本格式一般都比对应的二进制格式占用的存储空间大。二进制游标返回内部二进制形态的数据，可能更易于操作。如果想以文本方式显示数据，则以文本方式检索会为用户节约很多客户端的工作。比如，如果查询从某个整数列返回1，在缺省的游标里将获得一个字符串1，但在二进制游标里将得到一个4字节的包含该数值内部形式的数值（大端顺序）。

**语法格式**

CURSOR cursor\_name  
 [ BINARY ] [ NO SCROLL ] [ { WITH | WITHOUT } HOLD ]  
 FOR query ;

**参数说明**

* **cursor\_name**
* 将要创建的游标名。
* 取值范围：遵循数据库对象命名规范。
* **BINARY**
* 指明游标以二进制而不是文本格式返回数据。
* **NO SCROLL**
* 声明游标检索数据行的方式。
* NO SCROLL：声明该游标不能用于以倒序的方式检索数据行。
* 未声明：根据执行计划的不同，自动判断该游标是否可以用于以倒序的方式检索数据行。
* **WITH HOLD | WITHOUT HOLD**
* 声明当创建游标的事务结束后，游标是否能继续使用。
* WITH HOLD：声明该游标在创建它的事务结束后仍可继续使用。
* WITHOUT HOLD：声明该游标在创建它的事务之外不能再继续使用，此游标将在事务结束时被自动关闭。
* 如果不指定WITH HOLD或WITHOUT HOLD，默认行为是WITHOUT HOLD。
* 跨节点事务不支持WITH HOLD（例如在多DBnode部署Vastbase中所创建的含有DDL的事务属于跨节点事务）。
* **query**
* 使用SELECT或VALUES子句指定游标返回的行。
* 取值范围：SELECT或VALUES子句。

**示例**

参考：SQL语法参考->SQL语法->FETCH章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->FETCH章节。

### DEALLOCATE

**功能描述**

DEALLOCATE用于删除前面编写的预备语句。如果用户没有明确删除一个预备语句，那么它将在会话结束的时候被删除。

PREPARE关键字总被忽略。

**注意事项**

无。

**语法格式**

DEALLOCATE [ PREPARE ] { name | ALL };

**参数说明**

* **name**
* 将要删除的预备语句。
* **ALL**
* 删除所有预备语句。

**示例**

无。

### DELETE

**功能描述**

DELETE从指定的表里删除满足WHERE子句的行。如果WHERE子句不存在，将删除表中所有行，结果只保留表结构。

**注意事项**

* 表的所有者、被授予了表DELETE权限的用户或被授予DELETE ANY TABLE权限的用户有权删除表中数据，系统管理员默认拥有此权限。同时也必须有USING子句引用的表以及condition上读取的表的SELECT权限。
* 对于列存表，暂时不支持RETURNING子句。

**语法格式**

[ WITH [ RECURSIVE ] with\_query [, ...] ]  
DELETE [/\*+ plan\_hint \*/] [FROM] [ ONLY ] table\_name [partition\_clause] [ \* ] [ [ AS ] alias ]  
 [ USING using\_list ]  
 [ WHERE condition | WHERE CURRENT OF cursor\_name ]  
 [ RETURNING { \* | { output\_expr [ [ AS ] output\_name ] } [, ...] } ];

**参数说明**

* **WITH [ RECURSIVE ] with\_query [, ...]**
* 用于声明一个或多个可以在主查询中通过名称引用的子查询，相当于临时表。
* 如果声明了RECURSIVE，那么允许SELECT子查询通过名称引用它自己。
* 其中with\_query的详细格式为：

with\_query\_name [ ( column\_name [, ...] ) ] AS [ [ NOT ] MATERIALIZED]  
 ( {select | values | insert | update | delete} )

* with\_query\_name指定子查询生成的结果集名称，在查询中可使用该名称访问子查询的结果集。  
  column\_name指定子查询结果集中显示的列名。  
  - 每个子查询可以是SELECT、VALUES、INSERT、UPDATE或DELETE语句。  
  - 用户可以使用MATERIALIZED / NOT MATERIALIZED对CTE进行修饰。  
    
  如果声明为MATERIALIZED，WITH查询将被物化，生成一个子查询结果集的拷贝，在引用处直接查询该拷贝，因此WITH子查询无法和主干SELECT语句进行联合优化（如谓词下推、等价类传递等），对于此类场景可以使用NOT MATERIALIZED进行修饰，如果WITH查询语义上可以作为子查询内联执行，则可以进行上述优化。  
  如果用户没有显示声明物化属性则遵守以下规则：如果CTE只在所属主干语句中被引用一次，且语义上支持内联执行，则会被改写为子查询内联执行，否则以CTE Scan的方式物化执行。
* **plan\_hint子句**
* 以/\*+\* /的形式在DELETE关键字后，用于对DELETE对应的语句块生成的计划进行hint调优，详细用法请参见章节：《[VASTDATA]Vastbase G100 V2.2管理员指南->性能调优->SQL调优指南->使用Plan-Hint进行调优》章节。每条语句中只有第一个/\*+ plan\_hint \*/注释块会作为hint生效，里面可以写多条hint。
* **ONLY**
* 如果指定ONLY则只有该表被删除；如果没有声明，则该表和它的所有子表将都被删除。
* **table\_name**
* 目标表的名称（可以有模式修饰）。
* 取值范围：已存在的表名。
* **partition\_clause**
* 指定分区删除操作
* PARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) } |
* SUBPARTITION { ( subpartition\_name ) | FOR ( subpartition\_value [, ...] ) }
* 关键字详见：SQL语法参考->SQL语法->SELECT章节。
* 示例详见：SQL语法参考->SQL语法->CREATE TABLE SUBPARTITION章节。
* **alias**
* 目标表的别名。
* 取值范围：字符串，符合标识符命名规范。
* **using\_list**
* using子句。
* **condition**
* 一个返回Boolean值的表达式，用于判断哪些行需要被删除。不建议使用int等数值类型作为condition，因为int等数值类型可以隐式转换为bool值（非0值隐式转换为true，0转换为false），可能导致非预期的结果。
* **WHERE CURRENT OF cursor\_name**
* 当前不支持，仅保留语法接口。
* **output\_expr**
* DELETE命令删除行之后计算输出结果的表达式。该表达式可以使用表的任意字段。可以使用\*返回被删除行的所有字段。
* **output\_name**
* 一个字段的输出名称。
* 取值范围：字符串，符合标识符命名规范。

**示例**

--创建表customer\_address\_bak。  
vastbase=# CREATE TABLE customer\_address\_bak AS TABLE customer\_address;  
  
--删除customer\_address\_bak中ca\_address\_sk小于14888的职员。  
vastbase=# DELETE FROM customer\_address\_bak WHERE ca\_address\_sk < 14888;  
  
--删除customer\_address\_bak中所有数据。  
vastbase=# DELETE FROM customer\_address\_bak;  
  
--删除customer\_address\_bak表。  
vastbase=# DROP TABLE customer\_address\_bak;

**优化建议**

* delete
* 如果要删除表中的所有记录，建议使用truncate语法。

### DECLARE

**功能描述**

DECLARE命令既可以定义一个游标，用于在一个大的查询里面检索少数几行数据，也可以作为一个匿名块的开始。

本节主要描述定义为游标的用法，开启匿名块的用法参考：SQL语法参考->SQL语法->BEGIN章节。

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

通常游标和SELECT一样返回文本格式。因为数据在系统内部是用二进制格式存储的，系统必须对数据做一定转换以生成文本格式。一旦数据是以文本形式返回，客户端应用需要把它们转换成二进制进行操作。使用FETCH语句，游标可以返回文本或二进制格式。

**注意事项**

* 游标命令只能在事务块里使用。
* 应该小心使用二进制游标。文本格式一般都比对应的二进制格式占用的存储空间大。二进制游标返回内部二进制形态的数据，可能更易于操作。如果想以文本方式显示数据，则以文本方式检索会为用户节约很多客户端的工作。比如，如果查询从某个整数列返回1，在缺省的游标里将获得一个字符串1，但在二进制游标里将得到一个4字节的包含该数值内部形式的数值（大端顺序）。

**语法格式**

* 定义游标

DECLARE cursor\_name [ BINARY ] [ NO SCROLL ]  
 CURSOR [ { WITH | WITHOUT } HOLD ] FOR query ;

* 开启匿名块

[DECLARE [declare\_statements]]   
 BEGIN  
 execution\_statements  
 END;  
 /

**参数说明**

* **cursor\_name**
* 将要创建的游标名。
* 取值范围：遵循数据库对象命名规范。
* **BINARY**
* 指明游标以二进制而不是文本格式返回数据。
* **NO SCROLL**
* 声明游标检索数据行的方式。
* NO SCROLL：声明该游标不能用于以倒序的方式检索数据行。
* 未声明：根据执行计划的不同，自动判断该游标是否可以用于以倒序的方式检索数据行。
* **WITH HOLD**
* **WITHOUT HOLD**
* 声明当创建游标的事务结束后，游标是否能继续使用。
* WITH HOLD：声明该游标在创建它的事务结束后仍可继续使用。
* WITHOUT HOLD：声明该游标在创建它的事务之外不能再继续使用，此游标将在事务结束时被自动关闭。
* 如果不指定WITH HOLD或WITHOUT HOLD，默认行为是WITHOUT HOLD。
* **query**
* 使用SELECT或VALUES子句指定游标返回的行。
* 取值范围：SELECT或VALUES子句。
* **declare\_statements**
* 声明变量，包括变量名和变量类型，如“sales\_cnt int”。
* **execution\_statements**
* 匿名块中要执行的语句。
* 取值范围：已存在的函数名称。

**示例**

定义游标示例请参考：SQL语法参考->SQL语法->FETCH章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->BEGIN、FETCH章节。

### DO

**功能描述**

执行匿名代码块。

代码块被看做是没有参数的一段函数体，返回值类型是void。它的解析和执行是同一时刻发生的。

**注意事项**

* 程序语言在使用之前，必须通过命令CREATE LANGUAGE安装到当前的数据库中。 plpgsql是默认的安装语言，其它语言安装时必须指定。
* 如果语言是不受信任的，用户必须有使用程序语言的USAGE权限，或者是系统管理员。

**语法格式**

DO [ LANGUAGE lang\_name ] code;

**参数说明**

* **lang\_name**
* 用来解析代码的程序语言的名称，如果缺省，默认的语言是plpgsql。
* **code**
* 程序语言代码可以被执行的。程序语言必须指定为字符串才行。

**示例**

--创建用户webuser。  
vastbase=# CREATE USER webuser PASSWORD 'xxxxxxxxx';  
  
--授予用户webuser对模式tpcds下视图的所有操作权限。  
vastbase=# DO $$DECLARE r record;  
BEGIN  
 FOR r IN SELECT c.relname table\_name,n.nspname table\_schema FROM pg\_class c,pg\_namespace n   
 WHERE c.relnamespace = n.oid AND n.nspname = 'tpcds' AND relkind IN ('r','v')  
 LOOP  
 EXECUTE 'GRANT ALL ON ' || quote\_ident(r.table\_schema) || '.' || quote\_ident(r.table\_name) || ' TO webuser';  
 END LOOP;  
END$$;  
  
  
--删除用户webuser。  
vastbase=# DROP USER webuser CASCADE;

### DROP AGGREGATE

**功能描述**

删除一个聚合函数。

**注意事项**

DROP AGGREGATE删除一个现存的聚合函数，执行这条命令的用户必须是该聚合函数的所有者。

**语法格式**

DROP AGGREGATE [ IF EXISTS ] name ( argtype [ , ... ] ) [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果指定的聚合不存在，那么发出一个 notice 而不是抛出一个错误。
* **name**
* 现存的聚合函数名（可以有模式修饰）。
* **argtype**
* 聚合函数操作的输入数据类型，要引用一个零参数聚合函数，请用\*代替输入数据类型列表。
* **CASCADE**
* 级联删除依赖于这个聚合函数的对象。
* **RESTRICT**
* 如果有任何依赖对象，则拒绝删除这个聚合函数。这是缺省处理。

**示例**

将integer类型的聚合函数myavg删除：

DROP AGGREGATE myavg(integer);

**兼容性**

SQL 标准里没有DROP AGGREGATE语句。

### DROP AUDIT POLICY

**功能描述**

删除一个审计策略。

**注意事项**

只有poladmin、sysadmin或初始用户才能进行此操作。

**语法格式**

DROP AUDIT POLICY [IF EXISTS] policy\_name;

**参数说明**

policy\_name

审计策略名称，需要唯一，不可重复。

取值范围：字符串，要符合标识符的命名规范。

**示例**

参考：SQL语法参考->SQL语法->CREATE AUDIT POLICY章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER AUDIT POLICY、CREATE AUDIT POLICY章节。

### DROP CAST

**功能描述**

删除一个类型转换。

**注意事项**

DROP CAST删除一个先前定义过的类型转换。

要能删除一个类型转换，你必须拥有源或者目的数据类型。这是和创建一个类型转换相同的权限。

**语法格式**

DROP CAST [ IF EXISTS ] (source\_type AS target\_type) [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果指定的转换不存在，那么发出一个 notice 而不是抛出一个错误。
* **source\_type**
* 类型转换里的源数据类型。
* **target\_type**
* 类型转换里的目标数据类型。
* **CASCADE**
* **RESTRICT**
* 这些键字没有任何效果，因为在类型转换上没有依赖关系。

**示例**

删除从text到int的转换：

DROP CAST (text AS int);

**兼容性**

DROP CAST遵循 SQL 标准。

### DROP CLIENT MASTER KEY

**功能描述**

删除一个客户端加密主密钥（CMK）。

**注意事项**

* 只有客户端加密主密钥所有者或者被授予了DROP权限的用户有权限执行命令，系统管理员默认拥有此权限。
* 该命令不仅删除数据库中的密钥对象，还会同时删除客户端指定路径下该密钥对象对应的密钥文件。

**语法格式**

DROP CLIENT MASTER KEY [ IF EXISTS ] client\_master\_key\_name [CASCADE];

**参数说明**

* **IF EXISTS**
* 如果指定的客户端加密主密钥不存在，则发出一个notice而不是抛出一个错误。
* **client\_master\_key\_name**
* 要删除的客户端加密主密钥名称。
* 取值范围：字符串，已存在的客户端加密主密钥对象的名称。
* **CASCADE**
* **CASCADE**：表示允许级联删除依赖于客户端加密主密钥的对象。
  + fig: **须知：**   
    在执行本语法的生命周期中，同时需要客户端和服务端更改状态，发生异常时可能存在服务端已删除密钥信息，但客户端未删除密钥文件的情况。  
    此时，客户端并不会在执行下一条语法的生命周期中，检查是否有期望被删除但却因发生异常而未被删除的密钥文件，而是需要用户定期检查密钥文件夹，对未被使用的密钥文件进行确认并处理。

**示例**

--删除客户端加密主密钥对象。  
vastbase=> DROP CLIENT MASTER KEY ImgCMK CASCADE;  
NOTICE: drop cascades to column setting: imgcek  
DROP CLIENT MASTER KEY

### DROP COLUMN ENCRYPTION KEY

**功能描述**

删除一个列加密密钥（CEK）。

**注意事项**

只有列加密密钥所有者或者被授予了DROP权限的用户有权限执行命令，系统管理员默认拥有此权限。

**语法格式**

DROP COLUMN ENCRYPTION KEY [ IF EXISTS ] column\_encryption\_key\_name [CASCADE];

**参数说明**

* **IF EXISTS**
* 如果指定的列加密密钥不存在，则发出一个notice而不是抛出一个错误。
* **column\_encryption\_key\_name**
* 要删除的列加密密钥名称。
* 取值范围：字符串，已存在的列加密密钥名称。

**示例**

--删除列加密密钥。  
vastbase=# DROP COLUMN ENCRYPTION KEY ImgCEK CASCADE;  
ERROR: cannot drop column setting: imgcek cascadely because encrypted column depend on it.  
HINT: we have to drop encrypted column: name, ... before drop column setting: imgcek cascadely.

### DROP DATA SOURCE

**功能描述**

删除一个Data Source对象。

**注意事项**

只有属主/系统管理员/初始用户才可以删除一个Data Source对象。

**语法格式**

DROP DATA SOURCE [IF EXISTS] src\_name [CASCADE | RESTRICT];

**参数说明**

* **src\_name**
* 待删除的Data Source对象名称。
* 取值范围：字符串，符合标识符命名规范。
* **IF EXISTS**
* 如果指定的Data Source不存在，则发出一个notice而不是报错。
* **CASCADE | RESTRICT**
* **CASCADE**：表示允许级联删除依赖于Data Source的对象。
* **RESTRICT**（缺省值）：表示有依赖于该Data Source的对象存在，则该Data Source无法删除。
  + 目前Data Source对象没有被依赖的对象，CASCADE和RESTRICT效果一样，保留此选项是为了向后兼容性。

**示例**

--创建Data Source对象。  
vastbase=# CREATE DATA SOURCE ds\_tst1;  
  
--删除Data Source对象。  
vastbase=# DROP DATA SOURCE ds\_tst1 CASCADE;  
vastbase=# DROP DATA SOURCE IF EXISTS ds\_tst1 RESTRICT;

**相关链接**

参考：SQL语法参考->SQL语法->CREATE DATA SOURCE、ALTER DATA SOURCE章节。

### DROP DATABASE

**功能描述**

删除一个数据库。

**注意事项**

* 只有数据库所有者或者被授予了数据库DROP权限的用户有权限执行DROP DATABASE命令，系统管理员默认拥有此权限。
* 不能对系统默认安装的三个数据库（POSTGRES、TEMPLATE0和TEMPLATE1）执行删除操作，系统做了保护。如果想查看当前服务中有哪几个数据库，可以用vsql的\l命令查看。
* 如果有用户正在与要删除的数据库连接，则删除操作失败。
* 不能在事务块中执行DROP DATABASE命令。
* 如果执行DROP DATABASE失败，事务回滚，需要再次执行一次DROP DATABASE IF EXISTS。

fig: **须知：**   
DROP DATABASE一旦执行将无法撤销，请谨慎使用。

**语法格式**

DROP DATABASE [ IF EXISTS ] database\_name ;

**参数说明**

* **IF EXISTS**
* 如果指定的数据库不存在，则发出一个notice而不是抛出一个错误。
* **database\_name**
* 要删除的数据库名称。
* 取值范围：字符串，已存在的数据库名称。

**示例**

参考：SQL语法参考->SQL语法->CREATE DATABASE章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE DATABASE章节。

### DROP DATABASE LINK

**功能描述**

删除一个数据库链接。

**注意事项**

无

**语法格式**

DROP DATABASE LINK dblink\_name;

**参数说明**

**dblink\_name**

连接名称，可自定义。

**示例**

参考：SQL语法参考->SQL语法->CREATE DATABASE LINK章节的示例。

**相关链接**

CREATE DATABASE LINK（参考：SQL语法参考->SQL语法->CREATE DATABASE LINK章节)。

### DROP DIRECTORY

**功能描述**

删除指定的directory对象。

**注意事项**

* 当enable\_access\_server\_directory=off时，只允许初始用户删除directory对象。
* 当enable\_access\_server\_directory=on时，具有SYSADMIN权限的用户、directory对象的属主、被授予了该directory的DROP权限的用户或者继承了内置角色gs\_rloe\_directory\_drop权限的用户可以删除directory。

**语法格式**

DROP DIRECTORY [ IF EXISTS ] directory\_name;

**参数说明**

* **directory\_name**
* 目录名称。
* 取值范围：已经存在的目录名。

**示例**

--创建目录。  
vastbase=# CREATE OR REPLACE DIRECTORY dir as '/tmp/';  
  
--删除目录。  
vastbase=# DROP DIRECTORY dir;

**相关链接**

参考：SQL语法参考->SQL语法->CREATE DIRECTORY、ALTER DIRECTORY章节。

### DROP EXTENSION

**功能描述**

删除一个扩展。

**注意事项**

* DROP EXTENSION命令从数据库中删除一个扩展。在删除扩展的过程中，构成扩展的组件也会一起删除。
* 必须是扩展的拥有者才能够使用DROP EXTENSION命令。

**语法格式**

DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 当使用IF EXISTS参数，如果扩展不存在时，不会抛出错误，而是产生一个通知。
* **name**
* 已经安装的扩展模块的名称。
* **CASCADE**
* 自动删除依赖于该扩展的对象。
* **RESTRICT**
* 如果有依赖于扩展的对象，则不允许删除次扩展（除非它所有的成员对象和其它扩展对象在一条DROP命令一起删除）。这是缺省行为。

**示例**

从当前数据库中删除扩展hstore

DROP EXTENSION hstore;

**说明：**

在当前数据库中，如果有使用hstore的对象的，这条命令就会失败，比如任一表中的字段使用hstore类型。这时增加CASCADE选项会强制删除扩展和依赖于扩展的对象。

### DROP FOREIGN TABLE

**功能描述**

删除指定的外表。

**注意事项**

DROP FOREIGN TABLE会强制删除指定的表，删除表后，依赖该表的索引会被删除，因此引用该表的函数和存储过程将无法执行。

**语法格式**

DROP FOREIGN TABLE [ IF EXISTS ]   
 table\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的表不存在，则发出一个notice而不是抛出一个错误。
* **table\_name**
* 表名称。
* 取值范围：已存在的表名。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖于表的对象（比如视图）。
* RESTRICT：如果存在依赖对象，则拒绝删除该表。这个是缺省。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER FOREIGN TABLE、CREATE FOREIGN TABLE章节。

### DROP FUNCTION

**功能描述**

删除一个已存在的函数。

**注意事项**

* 如果函数中涉及对临时表相关操作，则无法使用DROP FUNCTION删除函数。
* 只有函数的所有者或者被授予了函数DROP权限的用户才能执行DROP FUNCTION命令，系统管理员默认拥有该权限。
* 通过DROP PROCEDURE语法也能删除函数。

**语法格式**

DROP FUNCTION [ IF EXISTS ] function\_name   
[ ( [ {[ argname ] [ argmode ] argtype} [, ...] ] ) [ CASCADE | RESTRICT ] ];

**参数说明**

* **IF EXISTS**
* IF EXISTS表示，如果函数存在则执行删除操作，函数不存在也不会报错，只是发出一个notice。
* **function\_name**
* 要删除的函数名称。
* 取值范围：已存在的函数名。
* **argmode**
* 函数参数的模式。
* **argname**
* 函数参数的名称。
* **argtype**
* 函数参数的类型。

**示例**

参考：SQL语法参考->SQL语法->CREATE-FUNCTION章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER FUNCTION、CREATE FUNCTION章节。

### DROP GLOBAL CONFIGURATION

**功能描述**

删除系统表gs\_global\_config中的参数值。

**注意事项**

* 仅支持数据库初始用户运行此命令。
* 不支持删除关键字为weak\_password。

**语法格式**

DROP GLOBAL CONFIGURATION 参数名称，参数名称...;

**参数说明**

参数名称是gs\_global\_config中已经存在的参数，删除不存在的参数将报错。

### DROP GROUP

**功能描述**

删除用户组。

DROP GROUP是DROP ROLE的别名。

**注意事项**

DROP GROUP是Vastbase管理工具封装的接口，用来实现Vastbase管理。该接口不建议用户直接使用，以免对Vastbase状态造成影响。

**语法格式**

DROP GROUP [ IF EXISTS ] group\_name [, ...];

**参数说明**

参考：SQL语法参考->SQL语法->DROP ROLE章节的参数说明。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE GROUP、ALTER GROUP、DROP ROLE章节。

### DROP INDEX

**功能描述**

删除索引。

**注意事项**

只有索引的所有者或者拥有索引所在表的INDEX权限的用户有权限执行DROP INDEX命令，系统管理员默认拥有此权限。

**语法格式**

DROP INDEX [ CONCURRENTLY ] [ IF EXISTS ]   
 index\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **CONCURRENTLY**
* 以不加锁的方式删除索引。删除索引时，一般会阻塞其他语句对该索引所依赖表的访问。加此关键字，可实现删除过程中不做阻塞。
* 此选项只能指定一个索引的名称，并且CASCADE选项不支持。
* 普通DROP INDEX命令可以在事务内执行，但是DROP INDEX CONCURRENTLY不可以在事务内执行。
* **IF EXISTS**
* 如果指定的索引不存在，则发出一个notice而不是抛出一个错误。
* **index\_name**
* 要删除的索引名。
* 取值范围：已存在的索引。
* **CASCADE | RESTRICT**
* CASCADE：表示允许级联删除依赖于该索引的对象。
* RESTRICT（缺省值）：表示有依赖与此索引的对象存在，则该索引无法被删除。

**示例**

参考：SQL语法参考->SQL语法->CREATE INDEX章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER INDEX、CREATE INDEX章节。

### DROP MASKING POLICY

**功能描述**

删除脱敏策略。

**注意事项**

只有poladmin、sysadmin或初始用户才能执行此操作。

**语法格式**

DROP MASKING POLICY [IF EXISTS] policy\_name;

**参数说明**

**policy\_name**

审计策略名称，需要唯一，不可重复。

取值范围：字符串，要符合标识符的命名规范。

**示例**

--删除一个脱敏策略。  
vastbase=# DROP MASKING POLICY IF EXISTS maskpol1;  
  
--删除一组脱敏策略。  
vastbase=# DROP MASKING POLICY IF EXISTS maskpol1, maskpol2, maskpol3;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER MASKING POLICY、CREATE MASKING POLICY章节。

### DROP MATERIALIZED VIEW

**功能描述**

强制删除数据库中已有的物化视图。

**注意事项**

物化视图的所有者、物化视图所在模式的所有者、被授予了物化视图DROP权限的用户或拥有DROP ANY TABLE权限的用户才有权限执行DROP MATERIALIZED VIEW命令，系统管理员默认拥有此权限。

**语法格式**

DROP MATERIALIZED VIEW [ IF EXISTS ] mv\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的物化视图不存在，则发出一个notice而不是抛出一个错误。
* **mv\_name**
* 要删除的物化视图名称。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖此物化视图的对象。
* RESTRICT：如果有依赖对象存在，则拒绝删除此物化视图。此选项为缺省值。

**示例**

--删除名为my\_mv的物化视图。  
vastbase=# DROP MATERIALIZED VIEW my\_mv;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER MATERIALIZED VIEW、CREATE INCREMENTAL MATERIALIZED VIEW、CREATE MATERIALIZED VIEW、REFRESH INCREMENTAL MATERIALIZED VIEW、REFRESH MATERIALIZED VIEW章节。

### DROP MODEL

**功能描述**

删除一个已训练完成保存的模型对象。

**注意事项**

所删除模型可在系统表gs\_model\_warehouse中查看到。

**语法格式**

DROP MODEL model\_name;

**参数说明**

model\_name：模型名称。

取值范围：字符串，需要符合标识符的命名规范。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE MODEL、PREDICT BY章节。

### DROP OPERATOR

Vastbase不支持drop operator功能。

### DROP OWNED

**功能描述**

删除一个数据库角色所拥有的数据库对象。

**注意事项**

* 所有该角色在当前数据库里和共享对象（数据库、表空间）上的所有对象上的权限都将被撤销。
* DROP OWNED常常被用来为移除一个或者多个角色做准备。因为DROP OWNED只影响当前数据库中的对象，通常需要在包含将被移除角色所拥有的对象的每一个数据库中都执行这个命令。
* 使用CASCADE选项可能导致这个命令递归去删除由其他用户所拥有的对象。
* 角色所拥有的数据库、表空间将不会被移除。

**语法格式**

DROP OWNED BY name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **name**
* 角色名。
* **CASCADE | RESTRICT**
* CASCADE：级联删除所有依赖于被删除对象的对象。
* RESTRICT（缺省值）：拒绝删除那些有任何依赖对象存在的对象。

**相关链接**

参考：SQL语法参考->SQL语法->REASSIGN OWNED、DROP ROLE章节。

### DROP PACKAGE

**功能描述**

删除已存在的PACKAGE或者PACKAGE BODY。

**注意事项**

删除PACKAGE BODY后，PACKAGE内的存储过程及函数会同时失效。

**语法格式**

DROP PACKAGE [ IF EXISTS ] package\_name;  
DROP PACKAGE BODY [ IF EXISTS ] package\_name;

### DROP PACKAGE BODY

**功能描述**

删除已存在的PACKAGE BODY。

**注意事项**

删除PACKAGE BODY后，PACKAGE内的存储过程及函数会同时失效。

**语法格式**

DROP PACKAGE BODY [ IF EXISTS ] package\_name;

**参数说明**

**package\_name**

已存在的包名。

**示例**

1、创建一个CREATE PACKAGE SPECIFICATION。

CREATE OR REPLACE PACKAGE emp\_bonus IS

var1 int:=1; --公有变量

var2 int:=2;

PROCEDURE testpro1(var3 int);--公有存储过程，可以被外部调用

END ;

/

2、创建一个package body。

create or replace package body emp\_bonus is

var3 int:=3;

var4 int:=4;

procedure testpro1(var3 int)

is

begin

create table if not exists test1(col1 int);

insert into test1 values(var1);

insert into test1 values(var4);

end;

begin #实例化开始

var4:=9;

testpro1(var4);

end;

end;

/

3、删除package body。

drop package body emp\_bonus;

**相关链接**

DROP PACKAGE（参考：开发者指南->SQL语法参考->SQL语法->DROP PACKAGE章节)

### DROP PROCEDURE

**功能描述**

删除已存在的存储过程。

**注意事项**

通过DROP FUNCTION也能删除存储过程。

**语法格式**

DROP PROCEDURE [ IF EXISTS ] procedure\_name ;

**参数说明**

* **IF EXISTS**
* 如果指定的存储过程不存在，发出一个notice而不是抛出一个错误。
* **procedure\_name**
* 要删除的存储过程名称。
* 取值范围：已存在的存储过程名。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE PROCEDURE章节。

### DROP PUBLICATION

**功能描述**

从数据库中删除一个现有的发布。

**注意事项**

发布只能被其属主或系统管理员删除。

**语法格式**

DROP PUBLICATION [ IF EXISTS ] name [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果发布不存在，不要抛出一个错误，而是发出一个提示。
* **name**
* 现有发布的名称。
* **CASCADE|RESTRICT**
* 当前这些关键词没有任何作用，因为发布没有依赖关系。

**示例**

参考：SQL语法参考->SQL语法->CREATE-PUBLICATION章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER PUBLICATION、CREATE PUBLICATION章节。

### DROP RESOURCE LABEL

**功能描述**

删除资源标签。

**注意事项**

只有poladmin、sysadmin或初始用户才能执行此操作。

**语法格式**

DROP RESOURCE LABEL [IF EXISTS] policy\_name[, ...]\*;

**参数说明**

**label\_name**

资源标签名称；

取值范围：字符串，要符合标识符的命名规范。

**示例**

--删除一个资源标签。  
vastbase=# DROP RESOURCE LABEL IF EXISTS res\_label1;  
  
--删除一组资源标签。  
vastbase=# DROP RESOURCE LABEL IF EXISTS res\_label1, res\_label2, res\_label3;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER RESOURCE LABEL、CREATE RESOURCE LABEL章节。

### DROP ROLE

**功能描述**

删除指定的角色。

**语法格式**

DROP ROLE [ IF EXISTS ] role\_name [, ...];

**参数说明**

* **IF EXISTS**

如果指定的角色不存在，则发出一个notice而不是抛出一个错误。

* **role\_name**

要删除的角色名称。

* 取值范围：已存在的角色。

**注意事项**

只允许删除同类型管理员属性的用户。

**示例**

1、创建测试用户utest。

CREATE ROLE utest IDENTIFIED BY 'Bigdata@123';

2、删除测试用户。

DROP ROLE utest;

### DROP ROW LEVEL SECURITY POLICY

**功能描述**

删除表上某个行访问控制策略。

**注意事项**

仅表的所有者或者管理员用户才能删除表的行访问控制策略。

**语法格式**

DROP [ ROW LEVEL SECURITY ] POLICY [ IF EXISTS ] policy\_name ON table\_name [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果指定的行访问控制策略不存在，发出一个notice而不是抛出一个错误。
* **policy\_name**
* 要删除的行访问控制策略的名称。
* table\_name
  + 行访问控制策略所在的数据表名。
* CASCADE/RESTRICT
  + 仅适配此语法，无对象依赖于该行访问控制策略，CASCADE和RESTRICT效果相同。

**示例**

--创建数据表all\_data  
vastbase=# CREATE TABLE all\_data(id int, role varchar(100), data varchar(100));  
  
--创建行访问控制策略  
vastbase=# CREATE ROW LEVEL SECURITY POLICY all\_data\_rls ON all\_data USING(role = CURRENT\_USER);  
  
--删除行访问控制策略  
vastbase=# DROP ROW LEVEL SECURITY POLICY all\_data\_rls ON all\_data;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER ROW LEVEL SECURITY POLICY、CREATE ROW LEVEL SECURITY POLICY章节。

### DROP RULE

**功能描述**

删除一个重写规则。

**语法格式**

DROP RULE [ IF EXISTS ] name ON table\_name [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果该规则不存在，会抛出一个NOTICE。
* **name**
* 要删除的现存规则名称。
* **table\_name**
* 该规则应用的表名。
* **CASCADE**
* 自动级联删除依赖于此规则的对象。
* **RESTRICT**
* 缺省情况下，如果有任何依赖对象，则拒绝删除此规则。

**示例**

--删除重写规则newrule  
DROP RULE newrule ON mytable;

### DROP SCHEMA

**功能描述**

从数据库中删除模式。

**注意事项**

只有模式的所有者或者被授予了模式DROP权限的用户有权限执行DROP SCHEMA命令，系统管理员默认拥有此权限。

**语法格式**

DROP SCHEMA [ IF EXISTS ] schema\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的模式不存在，发出一个notice而不是抛出一个错误。
* **schema\_name**
* 模式的名称。
* 取值范围：已存在模式名。
* **CASCADE | RESTRICT**
* CASCADE：自动删除包含在模式中的对象。
* RESTRICT：如果模式包含任何对象，则删除失败（缺省行为）。

fig: **须知：**   
不要随意删除pg\_temp或pg\_toast\_temp开头的模式，这些模式是系统内部使用的，如果删除，可能导致无法预知的结果。

fig: **说明：**   
无法删除当前模式。如果要删除当前模式，须切换到其他模式下。

**示例**

参考：SQL语法参考->SQL语法->CREATE SCHEMA章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SCHEMA、CREATE SCHEMA章节。

### DROP SECURITY LABEL

**功能描述**

删除一个安全标签。

**注意事项**

无

**语法格式**

DROP SECURITY LABEL label\_name;

**参数说明**

**label\_name**

安全标签名称，删除时要求标签存在。

**示例**

参考：SQL语法参考->SQL语法->CREATE SECURITYLABEL章节的示例。

**相关链接**

CREATE SECURITYLABEL（参考：SQL语法参考->SQL语法->CREATE SECURITYLABEL章节）

### DROP SEQUENCE

**功能描述**

从当前数据库里删除序列。

**注意事项**

* 只有序列的所有者或者被授予了序列DROP权限的用户才能删除，系统管理员默认拥有该权限。
* 如果SEQUENCE被创建时使用了LARGE标识，DROP时也需要使用LARGE标识。

**语法格式**

DROP [ LARGE ] SEQUENCE [ IF EXISTS ] {[schema.]sequence\_name} [ , ... ] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的序列不存在，则发出一个notice而不是抛出一个错误。
* **name**
* 序列名称。
* **CASCADE**
* 级联删除依赖序列的对象。
* **RESTRICT**
* 如果存在任何依赖的对象，则拒绝删除序列。此项是缺省值。

**示例**

--创建一个名为serial的递增序列，从101开始。  
vastbase=# CREATE SEQUENCE serial START 101;  
  
--删除序列。  
vastbase=# DROP SEQUENCE serial;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SEQUENCE、CREATE SEQUENCE章节。

### DROP SERVER

**功能描述**

删除现有的一个数据服务器。

**注意事项**

只有server的所有者或者被授予了server的DROP权限的用户才可以删除，系统管理员默认拥有该权限。

**语法格式**

DROP SERVER [ IF EXISTS ] server\_name [ {CASCADE | RESTRICT} ] ;

**参数描述**

* **IF EXISTS**
* 如果指定的数据服务器不存在，则发出一个notice而不是抛出一个错误。
* **server\_name**
* 服务器名称。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖于server的对象 。
* RESTRICT（缺省值）：如果存在依赖对象，则拒绝删除该server。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SERVER、CREATE SERVER章节。

### DROP SUBSCRIPTION

**功能描述**

删除数据库实例中的一个订阅。

**注意事项**

* 只有系统管理员才可以删除订阅。
* 如果该待删除订阅与复制槽相关联，就不能在事务块内部执行DROP SUBSCRIPTION。

**语法格式**

DROP SUBSCRIPTION [ IF EXISTS ] name [ CASCADE | RESTRICT ]

**参数说明**

* **name**
* 要删除的订阅的名称。
* **CASCADE|RESTRICT**
* 当前这些关键词没有任何作用，因为订阅没有依赖关系。

**示例**

参考：SQL语法参考->SQL语法->CREATE SUBSCRIPTION章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SUBSCRIPTION、CREATE SUBSCRIPTION章节。

### DROP SYNONYM

**功能描述**

删除指定的SYNONYM对象。

**注意事项**

只有SYNONYM的所有者有权限执行DROP SYNONYM命令，系统管理员默认拥有此权限。

**语法格式**

DROP SYNONYM [ IF EXISTS ] synonym\_name [ CASCADE | RESTRICT ];

**参数描述**

* **IF EXISTS**
* 如果指定的同义词不存在，则发出一个notice而不是抛出一个错误。
* **synonym\_name**
* 同义词名字，可以带模式名。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖同义词的对象（比如视图）。
* RESTRICT：如果有依赖对象存在，则拒绝删除同义词。此选项为缺省值。

**示例**

参考：SQL语法参考->SQL语法->CREATE SYNONYM章节的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER SYNONYM、CREATE SYNONYM章节。

### DROP TABLE

**功能描述**

删除指定的表。

**注意事项**

* DROP TABLE会强制删除指定的表，删除表后，依赖该表的索引会被删除，而使用到该表的函数和存储过程将无法执行。删除分区表，会同时删除分区表中的所有分区。
* 表的所有者、被授予了表的DROP权限的用户或被授予DROP ANY TABLE权限的用户，有权删除指定表 ，系统管理员默认拥有该权限。

**语法格式**

DROP TABLE [ IF EXISTS ]   
 { [schema.]table\_name } [, ...] [ CASCADE | RESTRICT ] [ PURGE ];

**参数说明**

* **IF EXISTS**
* 如果指定的表不存在，则发出一个notice而不是抛出一个错误。
* **schema**
* 模式名称。
* **table\_name**
* 表名称。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖于表的对象（比如视图）。
* RESTRICT（缺省项）：如果存在依赖对象，则拒绝删除该表。这个是缺省。
* **PURGE**
* 该参数表示即使开启回收站功能，DROP表时，也会直接物理删除表，而不是将其放入回收站中。

**示例**

参考：SQL语法参考->SQL语法->CREATE TABLE章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TABLE、CREATE TABLE章节。

### DROP TABLESPACE

**功能描述**

删除一个表空间。

**注意事项**

* 只有表空间所有者或者被授予了表空间DROP权限的用户有权限执行DROP TABLESPACE命令，系统管理员默认拥有此权限。
* 在删除一个表空间之前，表空间里面不能有任何数据库对象，否则会报错。
* DROP TABLESPACE不支持回滚，因此，不能出现在事务块内部。
* 执行DROP TABLESPACE操作时，如果有另外的会话执行db查询操作，可能会由于tablespace事务的原因导致查询失败，请重新执行db查询操作。
* 如果执行DROP TABLESPACE失败，需要再次执行一次DROP TABLESPACE IF EXISTS。

**语法格式**

DROP TABLESPACE [ IF EXISTS ] tablespace\_name;

**参数说明**

* **IF EXISTS**
* 如果指定的表空间不存在，则发出一个notice而不是抛出一个错误。
* **tablespace\_name**
* 表空间的名称。
* 取值范围：已存在的表空间的名称。

**示例**

参考：SQL语法参考->SQL语法->CREATE TABLESPACE章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TABLESPACE、CREATE TABLESPACE章节。

### DROP TEXT SEARCH CONFIGURATION

**功能描述**

删除已有文本搜索配置。

**注意事项**

要执行这个命令，用户必须是该配置的所有者。

**语法格式**

DROP TEXT SEARCH CONFIGURATION [ IF EXISTS ] name [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的文本搜索配置不存在，那么发出一个notice而不是抛出一个错误。
* **name**
* 要删除的文本搜索配置名称（可有模式修饰）。
* **CASCADE**
* 级联删除依赖文本搜索配置的对象。
* **RESTRICT**
* 若有任何对象依赖文本搜索配置则拒绝删除它。这是默认情况。

**示例**

参考：SQL语法参考->SQL语法->CREATE TEXT SEARCH CONFIGURATION章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TEXT SEARCH CONFIGURATION、CREATE TEXT SEARCH CONFIGURATION章节。

### DROP TEXT SEARCH DICTIONARY

**功能描述**

删除全文检索词典。

**注意事项**

* 预定义词典不支持DROP操作。
* 只有词典的所有者可以执行DROP操作，系统管理员默认拥有此权限。
* 谨慎执行DROP...CASCADE操作，该操作将级联删除使用该词典的文本搜索配置（TEXT SEARCH CONFIGURATION）。

**语法格式**

DROP TEXT SEARCH DICTIONARY [ IF EXISTS ] name [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果指定的全文检索词典不存在，那么发出一个Notice而不是报错。
* **name**
* 要删除的词典名称（可指定模式名，否则默认在当前模式下）。
* 取值范围：已存在的词典名。
* **CASCADE**
* 自动删除依赖于该词典的对象，并依次删除依赖于这些对象的所有对象。
* 如果存在任何一个使用该词典的文本搜索配置，此DROP命令将不会成功。可添加CASCADE以删除引用该词典的所有文本搜索配置以及词典。
* **RESTRICT**
* 如果任何对象依赖词典，则拒绝删除该词典。这是缺省值。

**示例**

--删除词典english  
vastbase=# DROP TEXT SEARCH DICTIONARY english;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER TEXT SEARCH DICTIONARY、CREATE TEXT SEARCH DICTIONARY章节。

### DROP TRIGGER

**功能描述**

删除触发器。

**注意事项**

只有触发器的所有者可以执行DROP TRIGGER操作，系统管理员默认拥有此权限。

**语法格式**

DROP TRIGGER [ IF EXISTS ] trigger\_name ON table\_name [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的触发器不存在，则发出一个notice而不是抛出一个错误。
* **trigger\_name**
* 要删除的触发器名称。
* 取值范围：已存在的触发器。
* **table\_name**
* 要删除的触发器所在的表名称。
* 取值范围：已存在的含触发器的表。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖此触发器的对象。
* RESTRICT：如果有依赖对象存在，则拒绝删除此触发器。此选项为缺省值。

**示例**

参考：SQL语法参考->SQL语法->CREATE TRIGGER章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE TRIGGER、ALTER TRIGGER、ALTER TABLE章节。

### DROP TYPE

**功能描述**

删除一个用户定义的数据类型。

**注意事项**

只有类型的所有者或者被授予了类型DROP权限的用户有权限执行DROP TYPE命令，系统管理员默认拥有此权限。

**语法格式**

DROP TYPE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]

**参数说明**

* **IF EXISTS**
* 如果指定的类型不存在，那么发出一个notice而不是抛出一个错误。
* **name**
* 要删除的类型名(可以有模式修饰)。
* **CASCADE**
* 级联删除依赖该类型的对象(比如字段、函数、操作符等)。
* **RESTRICT**
* 如果有依赖对象，则拒绝删除该类型（缺省行为）。

**示例**

参考：SQL语法参考->SQL语法->CREATE TYPE章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE TYPE、ALTER TYPE章节。

### DROP USER

**功能描述**

删除用户，同时会删除同名的schema。

**注意事项**

* 须使用CASCADE级联删除依赖用户的对象（除数据库外）。当删除用户的级联对象时，如果级联对象处于锁定状态，则此级联对象无法被删除，直到对象被解锁或锁定级联对象的进程被杀死。
* 在Vastbase中，存在一个配置参数enable\_kill\_query，此参数在配置文件postgresql.conf中。此参数影响级联删除用户对象的行为：
* 当参数enable\_kill\_query为on ，且使用CASCADE模式删除用户时，会自动kill锁定用户级联对象的进程，并删除用户。
* 当参数enable\_kill\_query为off，且使用CASCADE模式删除用户时，会等待锁定级联对象的进程结束之后再删除用户。
* 在数据库中删除用户时，如果依赖用户的对象在其他数据库中或者依赖用户的对象是其他数据库，请用户先手动删除其他数据库中的依赖对象或直接删除依赖数据库，再删除用户。即drop user不支持跨数据库进行级联删除。
* 如果该用户被DATA SOURCE对象依赖时，无法直接级联删除该用户，需要手动删除对应的DATA SOURCE对象之后再删除该用户。

**语法格式**

DROP USER [ IF EXISTS ] user\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的用户不存在，发出一个notice而不是抛出一个错误。
* **user\_name**
* 待删除的用户名。
* 取值范围：已存在的用户名。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖用户的对象。
* RESTRICT：如果用户还有任何依赖的对象，则拒绝删除该用户（缺省行为）。

fig: **说明：**   
在Vastbase中，存在一个配置参数enable\_kill\_query，此参数在配置文件postgresql.conf中。此参数影响级联删除用户对象的行为：

* 当参数enable\_kill\_query为on ，且使用CASCADE模式删除用户时，会自动kill锁定用户级联对象的进程，并删除用户。
* 当参数enable\_kill\_query为off，且使用CASCADE模式删除用户时，会等待锁定级联对象的进程结束之后再删除用户。

**示例**

参考：SQL语法参考->SQL语法->CREATE USER章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER USER、CREATE USER章节。

### DROP USER MAPPING

**功能描述**

移除一个用于外部服务器的用户映射。

**语法格式**

DROP USER MAPPING [ IF EXISTS ] FOR { user\_name | USER | CURRENT\_USER | PUBLIC } SERVER server\_name;

**参数描述**

* **IF EXISTS**
* 如果该用户映射不存在则不要抛出一个错误，而是发出一个提示。
* **user\_name**
* 该映射的用户名。
* CURRENT\_USER和USER匹配当前用户的名称。PUBLIC被用来匹配系统中所有现存和未来的用户名。
* **server\_name**
* 用户映射的服务器名。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER USER MAPPING、CREATE USER MAPPING章节。

### DROP VIEW

**功能描述**

数据库中强制删除已有的视图。

**注意事项**

视图的所有者或者被授予了视图DROP权限的用户或拥有DROP ANY TABLE权限的用户，有权限执行DROP VIEW的命令，系统管理员默认拥有此权限。

**语法格式**

DROP VIEW [ IF EXISTS ] view\_name [, ...] [ CASCADE | RESTRICT ];

**参数说明**

* **IF EXISTS**
* 如果指定的视图不存在，则发出一个notice而不是抛出一个错误。
* **view\_name**
* 要删除的视图名称。
* 取值范围：已存在的视图。
* **CASCADE | RESTRICT**
* CASCADE：级联删除依赖此视图的对象（比如其他视图）。
* RESTRICT：如果有依赖对象存在，则拒绝删除此视图。此选项为缺省值。

**示例**

参考：SQL语法参考->SQL语法->CREATE VIEW章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->ALTER VIEW、CREATE VIEW章节。

### DROP WEAK PASSWORD DICTIONARY

**功能描述**

清空gs\_global\_config中的所有弱口令。

**注意事项**

只有初始用户、系统管理员和安全管理员拥有权限执行本语法。

**语法格式**

DROP WEAK PASSWORD DICTIONARY;

**参数说明**

无。

**示例**

参见CREATE WEAK PASSWORD DICTIONARY的示例。

**相关链接**

参考：SQL语法参考->SQL语法->CREATE WEAK PASSWORD DICTIONARY章节。

### EXECUTE

**功能描述**

执行一个前面准备好的预备语句。因为一个预备语句只在会话的生命期里存在，那么预备语句必须是在当前会话的前些时候用PREPARE语句创建的。

**注意事项**

如果创建预备语句的PREPARE语句声明了一些参数，那么传递给EXECUTE语句的必须是一个兼容的参数集，否则就会生成一个错误。

**语法格式**

EXECUTE name [ ( parameter [, ...] ) ];

**参数说明**

* **name**
* 要执行的预备语句的名称。
* **parameter**
* 给预备语句的一个参数的具体数值。它必须是一个和生成与创建这个预备语句时指定参数的数据类型相兼容的值的表达式。

**示例**

--创建表reason。  
vastbase=# CREATE TABLE tpcds.reason (   
 CD\_DEMO\_SK INTEGER NOT NULL,  
 CD\_GENDER character(16) ,  
 CD\_MARITAL\_STATUS character(100)  
)  
;  
  
--插入数据。  
vastbase=# INSERT INTO tpcds.reason VALUES(51, 'AAAAAAAADDAAAAAA', 'reason 51');  
  
--创建表reason\_t1。  
vastbase=# CREATE TABLE tpcds.reason\_t1 AS TABLE tpcds.reason;  
  
--为一个INSERT语句创建一个预备语句然后执行它。  
vastbase=# PREPARE insert\_reason(integer,character(16),character(100)) AS INSERT INTO tpcds.reason\_t1 VALUES($1,$2,$3);  
  
vastbase=# EXECUTE insert\_reason(52, 'AAAAAAAADDAAAAAA', 'reason 52');   
  
--删除表reason和reason\_t1。  
vastbase=# DROP TABLE tpcds.reason;  
vastbase=# DROP TABLE tpcds.reason\_t1;

### EXPLAIN

**功能描述**

显示SQL语句的执行计划。

执行计划将显示SQL语句所引用的表会采用什么样的扫描方式，如：简单的顺序扫描、索引扫描等。如果引用了多个表，执行计划还会显示用到的JOIN算法。

执行计划的最关键的部分是语句的预计执行开销，这是计划生成器估算执行该语句将花费多长的时间。

若指定了ANALYZE选项，则该语句会被执行，然后根据实际的运行结果显示统计数据，包括每个计划节点内时间总开销（毫秒为单位）和实际返回的总行数。这对于判断计划生成器的估计是否接近现实非常有用。

**注意事项**

* 在指定ANALYZE选项时，语句会被执行。如果用户想使用EXPLAIN分析INSERT、UPDATE、DELETE、CREATE TABLE AS或EXECUTE语句，而不想改动数据（执行这些语句会影响数据），请使用如下方法。

START TRANSACTION;  
EXPLAIN ANALYZE ...;  
ROLLBACK;

* 由于参数DETAIL、NODES、NUM\_NODES是分布式模式下的功能，在单机模式中是被禁止使用的。假如使用，会产生如下错误。

vastbase=# create table student(id int, name char(20));  
CREATE TABLE  
vastbase=# explain (nodes true) insert into student values(5,'a'),(6,'b');  
ERROR: unrecognized EXPLAIN option "nodes"  
vastbase=# explain (num\_nodes true) insert into student values(5,'a'),(6,'b');  
ERROR: unrecognized EXPLAIN option "num\_nodes"

**语法格式**

* 显示SQL语句的执行计划，支持多种选项，对选项顺序无要求。

EXPLAIN [ ( option [, ...] ) ] statement;

其中选项option子句的语法为。

ANALYZE [ boolean ] |  
 ANALYSE [ boolean ] |  
 VERBOSE [ boolean ] |  
 COSTS [ boolean ] |  
 CPU [ boolean ] |  
 DETAIL [ boolean ] |(不可用)  
 NODES [ boolean ] |(不可用)  
 NUM\_NODES [ boolean ] |(不可用)  
 BUFFERS [ boolean ] |  
 TIMING [ boolean ] |  
 PLAN [ boolean ] |  
 FORMAT { TEXT | XML | JSON | YAML }

* 显示SQL语句的执行计划，且要按顺序给出选项。

EXPLAIN { [ { ANALYZE | ANALYSE } ] [ VERBOSE ] | PERFORMANCE } statement;

**参数说明**

* **statement**
* 指定要分析的SQL语句。
* **ANALYZE boolean | ANALYSE boolean**
* 显示实际运行时间和其他统计数据。
* 取值范围：
* TRUE（缺省值）：显示实际运行时间和其他统计数据。
* FALSE：不显示。
* **VERBOSE boolean**
* 显示有关计划的额外信息。
* 取值范围：
* TRUE（缺省值）：显示额外信息。
* FALSE：不显示。
* **COSTS boolean**
* 包括每个规划节点的估计总成本，以及估计的行数和每行的宽度。
* 取值范围：
* TRUE（缺省值）：显示估计总成本和宽度。
* FALSE：不显示。
* **CPU boolean**
* 打印CPU的使用情况的信息。
* 取值范围：
* TRUE（缺省值）：显示CPU的使用情况。
* FALSE：不显示。
* **DETAIL boolean**（不可用）
* 打印数据库节点上的信息。
* 取值范围：
* TRUE（缺省值）：打印数据库节点的信息。
* FALSE：不打印。
* **NODES boolean**（不可用）
* 打印query执行的节点信息。
* 取值范围：
* TRUE（缺省值）：打印执行的节点的信息。
* FALSE：不打印。
* **NUM\_NODES boolean**（不可用）
* 打印执行中的节点的个数信息。
* 取值范围：
* TRUE（缺省值）：打印数据库节点个数的信息。
* FALSE：不打印。
* **BUFFERS boolean**
* 包括缓冲区的使用情况的信息。
* 取值范围：
* TRUE：显示缓冲区的使用情况。
* FALSE（缺省值）：不显示。
* **TIMING boolean**
* 包括实际的启动时间和花费在输出节点上的时间信息。
* 取值范围：
* TRUE（缺省值）：显示启动时间和花费在输出节点上的时间信息。
* FALSE：不显示。
* **PLAN**
* 是否将执行计划存储在plan\_table中。当该选项开启时，会将执行计划存储在PLAN\_TABLE中，不打印到当前屏幕，因此该选项为on时，不能与其他选项同时使用。
* 取值范围：
* ON（缺省值）：将执行计划存储在plan\_table中，不打印到当前屏幕。执行成功返回EXPLAIN SUCCESS。
* OFF：不存储执行计划，将执行计划打印到当前屏幕。
* **FORMAT**
* 指定输出格式。
* 取值范围：TEXT、XML、JSON和YAML。
* 默认值：TEXT。
* **PERFORMANCE**
* 使用此选项时，即打印执行中的所有相关信息。

**示例**

1、创建一个表tpcds.customer\_address\_p1。

CREATE TABLE tpcds.customer\_address\_p1 AS TABLE tpcds.customer\_address;

2、修改explain\_perf\_mode为normal。

SET explain\_perf\_mode=normal;

3、显示表简单查询的执行计划。

EXPLAIN SELECT \* FROM tpcds.customer\_address\_p1;

回显如下：

QUERY PLAN  
------------------------------------------------------------------------  
 Seq Scan on customer\_address\_p1 (cost=0.00..11.00 rows=100 width=788)  
  
(1 row)

4、以JSON格式输出的执行计划（explain\_perf\_mode为normal时）。

EXPLAIN(FORMAT JSON) SELECT \* FROM tpcds.customer\_address\_p1;

回显如下：

QUERY PLAN  
-----------------------------------------------  
 [ +  
 { +  
 "Plan": { +  
 "Node Type": "Seq Scan", +  
 "Relation Name": "customer\_address\_p1",+  
 "Alias": "customer\_address\_p1", +  
 "Startup Cost": 0.00, +  
 "Total Cost": 11.00, +  
 "Plan Rows": 100, +  
 "Plan Width": 788 +  
 } +  
 } +  
 ]  
(1 row)

5、如果有一个索引，当使用一个带索引WHERE条件的查询，可能会显示一个不同的计划。

EXPLAIN SELECT \* FROM tpcds.customer\_address\_p1 WHERE ca\_address\_sk=10000;

回显如下：

QUERY PLAN  
----------------------------------------------------------------------  
 Seq Scan on customer\_address\_p1 (cost=0.00..11.25 rows=1 width=788)  
 Filter: (ca\_address\_sk = 10000)  
(2 rows)

6、以YAML格式输出的执行计划（explain\_perf\_mode为normal时）。

EXPLAIN(FORMAT YAML) SELECT \* FROM tpcds.customer\_address\_p1 WHERE ca\_address\_sk=10000;

回显如下：

QUERY PLAN  
------------------------------------------  
 - Plan: +  
 Node Type: "Seq Scan" +  
 Relation Name: "customer\_address\_p1"+  
 Alias: "customer\_address\_p1" +  
 Startup Cost: 0.00 +  
 Total Cost: 11.25 +  
 Plan Rows: 1 +  
 Plan Width: 788 +  
 Filter: "(ca\_address\_sk = 10000)"  
(1 row)

7、禁止开销估计的执行计划。

EXPLAIN(COSTS FALSE)SELECT \* FROM tpcds.customer\_address\_p1 WHERE ca\_address\_sk=10000;

回显如下：

QUERY PLAN  
-----------------------------------  
 Seq Scan on customer\_address\_p1  
 Filter: (ca\_address\_sk = 10000)  
(2 rows)

8、带有聚集函数查询的执行计划。

EXPLAIN SELECT SUM(ca\_address\_sk) FROM tpcds.customer\_address\_p1 WHERE ca\_address\_sk<10000;

回显如下：

QUERY PLAN  
---------------------------------------------------------------------------  
 Aggregate (cost=11.33..11.34 rows=1 width=12)  
 -> Seq Scan on customer\_address\_p1 (cost=0.00..11.25 rows=33 width=4)  
 Filter: (ca\_address\_sk < 10000)  
(3 rows)

9、创建一个二级分区表。

CREATE TABLE range\_list  
(  
month\_code VARCHAR2 ( 30 ) NOT NULL ,  
dept\_code VARCHAR2 ( 30 ) NOT NULL ,  
user\_no VARCHAR2 ( 30 ) NOT NULL ,  
sales\_amt int  
)  
PARTITION BY RANGE (month\_code) SUBPARTITION BY LIST (dept\_code)  
(  
PARTITION p\_201901 VALUES LESS THAN( '201903' )  
(  
SUBPARTITION p\_201901\_a values ('1'),  
SUBPARTITION p\_201901\_b values ('2')  
),  
PARTITION p\_201902 VALUES LESS THAN( '201910' )  
(  
SUBPARTITION p\_201902\_a values ('1'),  
SUBPARTITION p\_201902\_b values ('2')  
)  
);

10、执行带有二级分区表的查询语句。

* Iterations 和 Sub Iterations分别标识遍历了几个一级分区和二级分区。
* Selected Partitions标识哪些一级分区被实际扫描，Selected Subpartitions: (p:s)标识第p个一级分区下s个二级分区被实际扫描，如果一级分区下所有二级分区都被扫描则s显示为ALL。

EXPLAIN SELECT \* FROM range\_list WHERE dept\_code = '1';

回显如下：

QUERY PLAN  
------------------------------------------------------------------------------  
 Partition Iterator (cost=0.00..8.81 rows=2 width=238)  
 Iterations: 2  
 -> Partitioned Seq Scan on range\_list (cost=0.00..8.81 rows=2 width=238)  
 Filter: ((dept\_code)::text = '1'::text)  
 Selected Subpartitions: 1,3  
(5 rows)

11、删除表

tpcds.customer\_address\_p1。  
  
DROP TABLE tpcds.customer\_address\_p1;

**相关链接**

参考：SQL语法参考->SQL语法->ANALYZE和ANALYSE章节。

### EXPLAIN PLAN

**功能描述**

通过EXPLAIN PLAN命令可以将查询执行的计划信息存储于PLAN\_TABLE表中。与EXPLAIN命令不同的是，EXPLAIN PLAN仅将计划信息进行存储，而不会打印到屏幕。

**语法格式**

EXPLAIN PLAN  
[ SET STATEMENT\_ID = string ]  
FOR statement ;

**参数说明**

* EXPLAIN中的PLAN选项表示需要将计划信息存储于PLAN\_TABLE中，存储成功将返回“EXPLAIN SUCCESS”。
* STATEMENT\_ID用户可以对查询设置标签，输入的标签信息也将存储于PLAN\_TABLE中。

fig: **说明：**   
用户在执行EXPLAIN PLAN时，如果没有进行SET STATEMENT\_ID，则默认为空值。同时，用户可输入的STATEMENT\_ID最大长度为30个字节，超过长度将会产生报错。

**注意事项**

* EXPLAIN PLAN不支持在数据库节点上执行。
* 对于执行错误的SQL无法进行计划信息的收集。
* PLAN\_TABLE中的数据是session级生命周期并且session隔离和用户隔离，用户只能看到当前session、当前用户的数据。

**示例**

使用EXPLAIN PLAN收集SQL语句的执行计划，通常包括以下步骤：

1、创建模式vb。

fig: **说明：**   
执行EXPLAIN PLAN 后会将计划信息自动存储于PLAN\_TABLE中，不支持对PLAN\_TABLE进行INSERT、UPDATE、ANALYZE等操作。  
PLAN\_TABLE详细介绍参考：系统表和系统视图->系统视图->PLAN\_TABLE章节。

DROP SCHEMA IF EXISTS vb CASCADE;  
CREATE SCHEMA vb;

2、创建测试表。

CREATE TABLE vb.t (  
 id integer primary key NOT NULL,  
 a integer DEFAULT NULL,  
 b integer DEFAULT NULL  
) ;

3、创建函数。

create or replace function vb.creatData() returns   
boolean AS  
$BODY$  
declare ii integer;  
 begin  
 II:=1;  
 FOR ii IN 1..500000 LOOP  
 INSERT INTO vb.t (id,a,b) VALUES (ii,ii,ii);  
 end loop;  
 return true;  
 end;  
$BODY$  
LANGUAGE plpgsql;

4、像表中插入数据。

select \* from vb.creatData() as testexplain;  
create index ind\_a on vb.T(a);

5、执行EXPLAIN PLAN 后会将计划信息自动存储于PLAN\_TABLE中。

explain plan SET STATEMENT\_ID = 'test\_ex\_plan01' FOR SELECT MAX(id),AVG(b) from vb.T where a between 3764 and 47491;

6、查询PLAN\_TABLE。

SELECT \* FROM PLAN\_TABLE;

返回结果为：

statement\_id | plan\_id | id | operation | options | object\_name | object\_type | object\_owner | projection |  
cost | cardinality  
  
----------------+------------------+----+--------------+-------------------+-------------+-------------+--------------+-----------------+-------  
-----------+-------------  
 test\_ex\_plan01 | 6473924464813365 | 1 | AGGREGATE | PLAIN | | | | max(id), avg(b) | 3380.2  
2577317446 | 1  
 test\_ex\_plan01 | 6473924464813365 | 2 | TABLE ACCESS | BITMAP HEAP SCAN | t | TABLE | vb | id, a, b | 3367.7  
1327317446 | 2500  
 test\_ex\_plan01 | 6473924464813365 | 3 | INDEX | BITMAP INDEX SCAN | ind\_a | INDEX | vb | |  
 53.25 | 2500  
(3 rows)

7、清理PLAN\_TABLE表中的数据。

DELETE FROM PLAN\_TABLE WHERE STATEMENT\_ID = 'test\_ex\_plan01';

### FETCH

**功能描述**

FETCH通过已创建的游标来检索数据。

每个游标都有一个供FETCH使用的关联位置。游标的关联位置可以在查询结果的第一行之前，或者在结果中的任意行，或者在结果的最后一行之后：

* 游标刚创建完之后，关联位置在第一行之前的。
* 在抓取了一些移动行之后，关联位置在检索到的最后一行上。
* 如果FETCH抓取完了所有可用行，它就停在最后一行后面，或者在反向抓取的情况下是停在第一行前面。
* FETCH ALL或FETCH BACKWARD ALL将总是把游标的关联位置放在最后一行或者在第一行前面。

**注意事项**

* 如果游标定义了NO SCROLL，则不允许使用例如FETCH BACKWARD之类的反向抓取。
* NEXT、PRIOR、FIRST、LAST、ABSOLUTE、RELATIVE形式在恰当地移动游标之后抓取一条记录。如果后面没有数据行，就返回一个空的结果，此时游标就会停在查询结果的最后一行之后（向后查询时）或者第一行之前（向前查询时）。
* FORWARD和BACKWARD形式在向前或者向后移动的过程中抓取指定的行数，然后把游标定位在最后返回的行上；或者是，如果count大于可用的行数，则在所有行之后（向后查询时）或者之前（向前查询时）。
* RELATIVE 0、FORWARD 0、BACKWARD 0都要求在不移动游标的前提下抓取当前行，也就是重新抓取最近刚抓取过的行。除非游标定位在第一行之前或者最后一行之后，这个动作都应该成功，而在那两种情况下，不返回任何行。
* 当FETCH的游标上涉及列存表时，不支持BACKWARD、PRIOR等涉及反向获取操作。

**语法格式**

FETCH [ direction { FROM | IN } ] cursor\_name;

其中direction子句为可选参数。

NEXT  
 | PRIOR  
 | FIRST  
 | LAST  
 | ABSOLUTE count  
 | RELATIVE count  
 | count  
 | ALL  
 | FORWARD  
 | FORWARD count  
 | FORWARD ALL  
 | BACKWARD  
 | BACKWARD count  
 | BACKWARD ALL

**参数说明**

* **direction\_clause**
* 定义抓取数据的方向。
* 取值范围：
* NEXT（缺省值）
  + 从当前关联位置开始，抓取下一行。
* PRIOR
  + 从当前关联位置开始，抓取上一行。
* FIRST
  + 抓取查询的第一行（和ABSOLUTE 1相同）。
* LAST
  + 抓取查询的最后一行（和ABSOLUTE -1相同）。
* ABSOLUTE count
  + 抓取查询中第count行。
  + ABSOLUTE抓取不会比用相对位移移动到需要的数据行更快，因为下层的实现必须遍历所有中间的行。
  + count取值范围：有符号的整数
* count为正数，就从查询结果的第一行开始，抓取第count行。
* count为负数，就从查询结果末尾抓取第abs(*count*)行。
* count为0时，定位在第一行之前。
* RELATIVE count
  + 从当前关联位置开始，抓取随后或前面的第count行。
  + 取值范围：有符号的整数
* count为正数就抓取当前关联位置之后的第count行。
* count为负数就抓取当前关联位置之前的第abs(count)行。
* 如果当前行没有数据的话，RELATIVE 0返回空。
* count
  + 抓取随后的count行（和FORWARD count一样）。
* ALL
  + 从当前关联位置开始，抓取所有剩余的行（和FORWARD ALL一样）。
* FORWARD
  + 抓取下一行（和NEXT一样）。
* FORWARD count
  + 从当前关联位置开始，抓取随后或前面的count行。
* FORWARD ALL
  + 从当前关联位置开始，抓取所有剩余行。
* BACKWARD
  + 从当前关联位置开始，抓取前面一行(和PRIOR一样) 。
* BACKWARD count
  + 从当前关联位置开始，抓取前面的count行（向后扫描）。
  + 取值范围：有符号的整数
* count为正数就抓取当前关联位置之前的count行。
* count为负数就抓取当前关联位置之后的abs（count）行。
* 如果有数据的话，BACKWARD 0重新抓取当前行。
* BACKWARD ALL
  + 从当前关联位置开始，抓取所有前面的行（向后扫描） 。
* **{ FROM | IN } cursor\_name**
* 使用关键字FROM或IN指定游标名称。
* 取值范围：已创建的游标的名称。

**示例**

--SELECT语句，用一个游标读取一个表。开始一个事务。  
vastbase=# START TRANSACTION;  
  
--建立一个名为cursor1的游标。  
vastbase=# CURSOR cursor1 FOR SELECT \* FROM tpcds.customer\_address ORDER BY 1;  
  
--抓取头3行到游标cursor1里。  
vastbase=# FETCH FORWARD 3 FROM cursor1;  
 ca\_address\_sk | ca\_address\_id | ca\_street\_number | ca\_street\_name | ca\_street\_type | ca\_suite\_number | ca\_city | ca\_county | ca\_state | ca\_zip | ca\_country | ca\_gmt\_offset | ca\_location\_type   
---------------+------------------+------------------+--------------------+-----------------+-----------------+-----------------+-----------------+----------+------------+---------------+---------------+----------------------  
 1 | AAAAAAAABAAAAAAA | 18 | Jackson | Parkway | Suite 280 | Fairfield | Maricopa County | AZ | 86192 | United States | -7.00 | condo   
 2 | AAAAAAAACAAAAAAA | 362 | Washington 6th | RD | Suite 80 | Fairview | Taos County | NM | 85709 | United States | -7.00 | condo   
 3 | AAAAAAAADAAAAAAA | 585 | Dogwood Washington | Circle | Suite Q | Pleasant Valley | York County | PA | 12477 | United States | -5.00 | single family   
(3 rows)  
  
--关闭游标并提交事务。  
vastbase=# CLOSE cursor1;  
  
--结束一个事务。  
vastbase=# END;  
  
--VALUES子句，用一个游标读取VALUES子句中的内容。开始一个事务。  
vastbase=# START TRANSACTION;  
  
--建立一个名为cursor2的游标。  
vastbase=# CURSOR cursor2 FOR VALUES(1,2),(0,3) ORDER BY 1;  
  
--抓取头2行到游标cursor2里。  
vastbase=# FETCH FORWARD 2 FROM cursor2;  
column1 | column2  
---------+---------  
0 | 3  
1 | 2  
(2 rows)  
  
--关闭游标并提交事务。  
vastbase=# CLOSE cursor2;  
  
--结束一个事务。  
vastbase=# END;  
  
--WITH HOLD游标的使用，开启事务。  
vastbase=# START TRANSACTION;  
  
--创建一个with hold游标。  
vastbase=# DECLARE cursor1 CURSOR WITH HOLD FOR SELECT \* FROM tpcds.customer\_address ORDER BY 1;  
  
--抓取头2行到游标cursor1里。  
vastbase=# FETCH FORWARD 2 FROM cursor1;  
 ca\_address\_sk | ca\_address\_id | ca\_street\_number | ca\_street\_name | ca\_street\_type | ca\_suite\_number | ca\_city | ca\_county | ca\_state | ca\_zip | ca\_country | ca\_gmt\_offset | ca\_location\_type   
---------------+------------------+------------------+--------------------+-----------------+-----------------+-----------------+-----------------+----------+------------+---------------+---------------+----------------------  
 1 | AAAAAAAABAAAAAAA | 18 | Jackson | Parkway | Suite 280 | Fairfield | Maricopa County | AZ | 86192 | United States | -7.00 | condo   
 2 | AAAAAAAACAAAAAAA | 362 | Washington 6th | RD | Suite 80 | Fairview | Taos County | NM | 85709 | United States | -7.00 | condo   
(2 rows)  
  
--结束事务。  
vastbase=# END;  
  
--抓取下一行到游标cursor1里。  
vastbase=# FETCH FORWARD 1 FROM cursor1;  
 ca\_address\_sk | ca\_address\_id | ca\_street\_number | ca\_street\_name | ca\_street\_type | ca\_suite\_number | ca\_city | ca\_county | ca\_state | ca\_zip | ca\_country | ca\_gmt\_offset | ca\_location\_type   
---------------+------------------+------------------+--------------------+-----------------+-----------------+-----------------+-----------------+----------+------------+---------------+---------------+----------------------  
 3 | AAAAAAAADAAAAAAA | 585 | Dogwood Washington | Circle | Suite Q | Pleasant Valley | York County | PA | 12477 | United States | -5.00 | single family   
(1 row)  
  
--关闭游标。  
vastbase=# CLOSE cursor1;

**相关链接**

参考：SQL语法参考->SQL语法->CLOSE、MOVE章节。

### GRANT

**功能描述**

对角色和用户进行授权操作。

使用GRANT命令进行用户授权包括以下三种场景：

* **将系统权限授权给角色或用户**
* 系统权限又称为用户属性，包括SYSADMIN、CREATEDB、CREATEROLE、AUDITADMIN、MONADMIN、OPRADMIN、POLADMIN和LOGIN。
* 系统权限一般通过CREATE/ALTER ROLE语法来指定。其中，SYSADMIN权限可以通过GRANT/REVOKE ALL PRIVILEGE授予或撤销。但系统权限无法通过ROLE和USER的权限被继承，也无法授予PUBLIC。
* **将数据库对象授权给角色或用户**
* 将数据库对象（表和视图、指定字段、数据库、函数、模式、表空间等）的相关权限授予特定角色或用户；

GRANT命令将数据库对象的特定权限授予一个或多个角色。这些权限会追加到已有的权限上。

关键字PUBLIC表示该权限要赋予所有角色，包括以后创建的用户。PUBLIC可以看做是一个隐含定义好的组，它总是包括所有角色。任何角色或用户都将拥有通过GRANT直接赋予的权限和所属的权限，再加上PUBLIC的权限。  
  
如果声明了WITH GRANT OPTION，则被授权的用户也可以将此权限赋予他人，否则就不能授权给他人。这个选项不能赋予PUBLIC，这是Vastbase特有的属性。  
  
Vastbase会将某些类型的对象上的权限授予PUBLIC。默认情况下，对表、表字段、序列、外部数据源、外部服务器、模式或表空间对象的权限不会授予PUBLIC，而以下这些对象的权限会授予PUBLIC：数据库的CONNECT权限和CREATE TEMP TABLE权限、函数的EXECUTE特权、语言和数据类型（包括域）的USAGE特权。当然，对象拥有者可以撤销默认授予PUBLIC的权限并专门授予权限给其他用户。为了更安全，建议在同一个事务中创建对象并设置权限，这样其他用户就没有时间窗口使用该对象。另外可参考安全加固指南的权限控制章节，对PUBLIC用户组的权限进行限制。这些初始的默认权限可以使用ALTER DEFAULT PRIVILEGES命令修改。  
  
对象的所有者缺省具有该对象上的所有权限，出于安全考虑所有者可以舍弃部分权限，但ALTER、DROP、COMMENT、INDEX、VACUUM以及对象的可再授予权限属于所有者固有的权限，隐式拥有。

* **将角色或用户的权限授权给其他角色或用户**
* 将一个角色或用户的权限授予一个或多个其他角色或用户。在这种情况下，每个角色或用户都可视为拥有一个或多个数据库权限的集合。
* 当声明了WITH ADMIN OPTION，被授权的用户可以将该权限再次授予其他角色或用户，以及撤销所有由该角色或用户继承到的权限。当授权的角色或用户发生变更或被撤销时，所有继承该角色或用户权限的用户拥有的权限都会随之发生变更。
* 数据库系统管理员可以给任何角色或用户授予/撤销任何权限。拥有CREATE ROLE权限的角色可以赋予或者撤销任何非系统管理员角色的权限。
* **将ANY权限授予给角色或用户**
* 将ANY权限授予特定的角色和用户，ANY权限的取值范围参见语法格式。当声明了WITH ADMIN OPTION，被授权的用户可以将该ANY权限再次授予其他角色/用户，或从其他角色/用户处回收该ANY权限。ANY权限可以通过角色被继承，但不能赋予PUBLIC。初始用户和三权分立关闭时的系统管理员用户可以给任何角色/用户授予或撤销ANY权限。
* 目前支持以下ANY权限：CREATE ANY TABLE、ALTER ANY TABLE、DROP ANY TABLE、SELECT ANY TABLE、INSERT ANY TABLE、UPDATE ANY TABLE、DELETE ANY TABLE、CREATE ANY SEQUENCE、CREATE ANY INDEX、CREATE ANY FUNCTION、EXECUTE ANY FUNCTION、 CREATE ANY PACKAGE、EXECUTE ANY PACKAGE、CREATE ANY TYPE。详细的ANY权限范围描述参考表1：ANY权限列表。

**注意事项**

* 不允许将ANY权限授予PUBLIC，也不允许从PUBLIC回收ANY权限。
* ANY权限属于数据库内的权限，只对授予该权限的数据库内的对象有效，例如SELECT ANY TABLE只允许用户查看当前数据库内的所有用户表数据，对其他数据库内的用户表无查看权限。
* 即使用户被授予ANY权限，也不能对私有用户下的对象进行访问操作（INSERT、DELETE、UPDATE、SELECT）。
* ANY权限与原有的权限相互无影响。
* 如果用户被授予CREATE ANY TABLE权限，在同名schema下创建表的属主是该schema的创建者，用户对表进行其他操作时，需要授予相应的操作权限。

**语法格式**

* 将表或视图的访问权限赋予指定的用户或角色。

GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM } [, ...]   
 | ALL [ PRIVILEGES ] }  
 ON { [ TABLE ] table\_name [, ...]  
 | ALL TABLES IN SCHEMA schema\_name [, ...] }  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]   
 [ WITH GRANT OPTION ];

* 将表中字段的访问权限赋予指定的用户或角色。

GRANT { {{ SELECT | INSERT | UPDATE | REFERENCES | COMMENT } ( column\_name [, ...] )} [, ...]   
 | ALL [ PRIVILEGES ] ( column\_name [, ...] ) }  
 ON [ TABLE ] table\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将序列的访问权限赋予指定的用户或角色，LARGE字段属性可选，赋权语句不区分序列是否为LARGE。

GRANT { { SELECT | UPDATE | USAGE | ALTER | DROP | COMMENT } [, ...]   
 | ALL [ PRIVILEGES ] }  
 ON { [ [ LARGE ] SEQUENCE ] sequence\_name [, ...]  
 | ALL SEQUENCES IN SCHEMA schema\_name [, ...] }  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]   
 [ WITH GRANT OPTION ];

* 将数据库的访问权限赋予指定的用户或角色。

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP | ALTER | DROP | COMMENT } [, ...]  
 | ALL [ PRIVILEGES ] }  
 ON DATABASE database\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将域的访问权限赋予指定的用户或角色。

GRANT { USAGE | ALL [ PRIVILEGES ] }  
 ON DOMAIN domain\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

fig: **说明：**   
本版本暂时不支持赋予域的访问权限。

* 将客户端加密主密钥CMK的访问权限赋予指定的用户或角色。

GRANT { { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }  
 ON CLIENT\_MASTER\_KEY client\_master\_key [, ...]   
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]   
 [ WITH GRANT OPTION ];

* 将列加密密钥CEK的访问权限赋予指定的用户或角色。

GRANT { { USAGE | DROP } [, ...] | ALL [ PRIVILEGES ] }  
 ON COLUMN\_ENCRYPTION\_KEY column\_encryption\_key [, ...]   
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]   
 [ WITH GRANT OPTION ];

* 将外部数据源的访问权限赋予给指定的用户或角色。

GRANT { USAGE | ALL [ PRIVILEGES ] }  
 ON FOREIGN DATA WRAPPER fdw\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将外部服务器的访问权限赋予给指定的用户或角色。

GRANT { { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON FOREIGN SERVER server\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将函数的访问权限赋予给指定的用户或角色。

GRANT { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON { FUNCTION {function\_name ( [ {[ argmode ] [ arg\_name ] arg\_type} [, ...] ] )} [, ...]  
 | ALL FUNCTIONS IN SCHEMA schema\_name [, ...] }  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将存储过程的访问权限赋予给指定的用户或角色。

GRANT { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON { PROCEDURE {proc\_name ( [ {[ argmode ] [ arg\_name ] arg\_type} [, ...] ] )} [, ...]  
 | ALL PROCEDURE IN SCHEMA schema\_name [, ...] }  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将过程语言的访问权限赋予给指定的用户或角色。

GRANT { USAGE | ALL [ PRIVILEGES ] }  
 ON LANGUAGE lang\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将大对象的访问权限赋予指定的用户或角色。

GRANT { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }  
 ON LARGE OBJECT loid [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

fig: **说明：**   
本版本暂时不支持大对象。

* 将模式的访问权限赋予指定的用户或角色。

GRANT { { CREATE | USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON SCHEMA schema\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

fig: **说明：**   
将模式中的表或者视图对象授权给其他用户时，需要将表或视图所属的模式的USAGE权限同时授予该用户，若没有该权限，则只能看到这些对象的名称，并不能实际进行对象访问。 同名模式下创建表的权限无法通过此语法赋予，可以通过将角色的权限赋予其他用户或角色的语法，赋予同名模式下创建表的权限。

* 将表空间的访问权限赋予指定的用户或角色。

GRANT { { CREATE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON TABLESPACE tablespace\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

* 将类型的访问权限赋予指定的用户或角色。

GRANT { { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
 ON TYPE type\_name [, ...]  
 TO { [ GROUP ] role\_name | PUBLIC } [, ...]  
 [ WITH GRANT OPTION ];

fig: **说明：**   
本版本暂时不支持赋予类型的访问权限。

* 将Data Source对象的权限赋予指定的角色。

GRANT { USAGE | ALL [PRIVILEGES]}  
 ON DATA SOURCE src\_name [, ...]  
 TO { [GROUP] role\_name | PUBLIC } [, ...]  
 [WITH GRANT OPTION];

* 将directory对象的权限赋予指定的角色。

GRANT { { READ | WRITE | ALTER | DROP } [, ...] | ALL [PRIVILEGES] }  
 ON DIRECTORY directory\_name [, ...]  
 TO { [GROUP] role\_name | PUBLIC } [, ...]  
 [WITH GRANT OPTION];

* 将package对象的权限赋予指定的角色。

GRANT { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [PRIVILEGES] }  
 ON PACKAGE package\_name [, ...]  
 TO { [GROUP] role\_name | PUBLIC } [, ...]  
 [WITH GRANT OPTION];

* 将角色的权限赋予其他用户或角色的语法。

GRANT role\_name [, ...]  
 TO role\_name [, ...]  
 [ WITH ADMIN OPTION ];

* 将sysadmin权限赋予指定的角色。

GRANT ALL { PRIVILEGES | PRIVILEGE }  
 TO role\_name;

* 将ANY权限赋予其他用户或角色的语法。

GRANT { CREATE ANY TABLE | ALTER ANY TABLE | DROP ANY TABLE | SELECT ANY TABLE | INSERT ANY TABLE | UPDATE ANY TABLE |  
 DELETE ANY TABLE | CREATE ANY SEQUENCE | CREATE ANY INDEX | CREATE ANY FUNCTION | EXECUTE ANY FUNCTION |  
 CREATE ANY PACKAGE | EXECUTE ANY PACKAGE | CREATE ANY TYPE } [, ...]  
 TO [ GROUP ] role\_name [, ...]  
 [ WITH ADMIN OPTION ];

**参数说明**

GRANT的权限分类如下所示。

* **SELECT**
* 允许对指定的表、视图、序列执行SELECT命令，update或delete时也需要对应字段上的select权限。
* **INSERT**
* 允许对指定的表执行INSERT命令。
* **UPDATE**
* 允许对声明的表中任意字段执行UPDATE命令。通常，update命令也需要select权限来查询出哪些行需要更新。SELECT… FOR UPDATE和SELECT… FOR SHARE除了需要SELECT权限外，还需要UPDATE权限。
* **DELETE**
* 允许执行DELETE命令删除指定表中的数据。通常，delete命令也需要select权限来查询出哪些行需要删除。
* **TRUNCATE**
* 允许执行TRUNCATE语句删除指定表中的所有记录。
* **REFERENCES**
* 创建一个外键约束，必须拥有参考表和被参考表的REFERENCES权限。
* **CREATE**
* 对于数据库，允许在数据库里创建新的模式。
* 对于模式，允许在模式中创建新的对象。如果要重命名一个对象，用户除了必须是该对象的所有者外，还必须拥有该对象所在模式的CREATE权限。
* 对于表空间，允许在表空间中创建表，允许在创建数据库和模式的时候把该表空间指定为缺省表空间。
* **CONNECT**
* 允许用户连接到指定的数据库。
* **EXECUTE**
* 允许使用指定的函数，以及利用这些函数实现的操作符。
* **USAGE**
* 对于过程语言，允许用户在创建函数的时候指定过程语言。
* 对于模式，USAGE允许访问包含在指定模式中的对象，若没有该权限，则只能看到这些对象的名称。
* 对于序列，USAGE允许使用nextval函数。
* 对于Data Source对象，USAGE是指访问权限，也是可赋予的所有权限，即USAGE与ALL PRIVILEGES等价。
* **ALTER**
* 允许用户修改指定对象的属性，但不包括修改对象的所有者和修改对象所在的模式。
* **DROP**
* 允许用户删除指定的对象。
* **COMMENT**
* 允许用户定义或修改指定对象的注释。
* **INDEX**
* 允许用户在指定表上创建索引，并管理指定表上的索引，还允许用户对指定表执行REINDEX和CLUSTER操作。
* **VACUUM**
* 允许用户对指定的表执行ANALYZE和VACUUM操作。
* **ALL PRIVILEGES**
* 一次性给指定用户/角色赋予所有可赋予的权限。只有系统管理员有权执行GRANT ALL PRIVILEGES。

GRANT的参数说明如下所示。

* **role\_name**
* 已存在用户名称。
* **table\_name**
* 已存在表名称。
* **column\_name**
* 已存在字段名称。
* **schema\_name**
* 已存在模式名称。
* **database\_name**
* 已存在数据库名称。
* **function\_name**
* 已存在函数名称。
* **procedure\_name**
* 已存在存储过程名称。
* **sequence\_name**
* 已存在序列名称。
* **domain\_name**
* 已存在域类型名称。
* **fdw\_name**
* 已存在外部数据包名称。
* **lang\_name**
* 已存在语言名称。
* **type\_name**
* 已存在类型名称。
* **src\_name**
* 已存在的Data Source对象名称。
* **argmode**
* 参数模式。
* 取值范围：字符串，要符合标识符命名规范。
* **arg\_name**
* 参数名称。
* 取值范围：字符串，要符合标识符命名规范。
* **arg\_type**
* 参数类型。
* 取值范围：字符串，要符合标识符命名规范。
* **loid**
* 包含本页的大对象的标识符。
* 取值范围：字符串，要符合标识符命名规范。
* **tablespace\_name**
* 表空间名称。
* **client\_master\_key**
* 客户端加密主密钥的名称。
* 取值范围：字符串，要符合标识符命名规范。
* **column\_encryption\_key**
* 列加密密钥的名称。
* 取值范围：字符串，要符合标识符命名规范。
* **directory\_name**
* 目录名称。
* 取值范围：字符串，要符合标识符命名规范。
* **WITH GRANT OPTION**
* 如果声明了WITH GRANT OPTION，则被授权的用户也可以将此权限赋予他人，否则就不能授权给他人。这个选项不能赋予PUBLIC。

非对象所有者给其他用户授予对象权限时，命令按照以下规则执行：

* 如果用户没有该对象上指定的权限，命令立即失败。
* 如果用户有该对象上的部分权限，则GRANT命令只授予他有授权选项的权限。
* 如果用户没有可用的授权选项，GRANT ALL PRIVILEGES形式将发出一个警告信息，其他命令形式将发出在命令中提到的且没有授权选项的相关警告信息。

fig: **说明：**   
数据库系统管理员可以访问所有对象，而不会受对象的权限设置影响。这个特点类似Unix系统的root的权限。和root一样，除了必要的情况外，建议不要总是以系统管理员身份进行操作。  
不允许对表分区进行GRANT操作，对分区表进行GRANT操作会引起告警。

* WITH ADMIN OPTION
* 对于角色，当声明了WITH ADMIN OPTION，被授权的用户可以将该角色再授予其他角色/用户，或从其他角色/用户回收该角色。
* 对于ANY权限，当声明了WITH ADMIN OPTION，被授权的用户可以将该ANY权限再授予其他角色/用户，或从其他角色/用户回收该ANY权限。

**表 1** ANY权限列表

| **系统权限名称** | **描述** |
| --- | --- |
| CREATE ANY TABLE | 用户能够在public模式和用户模式下创建表或视图。如果想要创建serial类型列的表，还需要授予创建序列的权限。 |
| ALTER ANY TABLE | 用户拥有对public模式和用户模式下表或视图的ALTER权限。如果想要修改表的唯一索引为表增加主键约束或唯一约束，还需要授予该表的索引权限。 |
| DROP ANY TABLE | 用户拥有对public模式和用户模式下表或视图的DROP权限。 |
| SELECT ANY TABLE | 用户拥有对public模式和用户模式下表或视图的SELETCT权限，仍然受行级访问控制限制。 |
| UPDATE ANY TABLE | 用户拥有对public模式和用户模式下表或视图的UPDATE权限，仍然受行级访问控制限制。 |
| INSERT ANY TABLE | 用户拥有对public模式和用户模式下表或视图的INSERT权限。 |
| DELETE ANY TABLE | 用户拥有对public模式和用户模式下表或视图的DELETE权限，仍然受行级访问控制限制。 |
| CREATE ANY FUNCTION | 用户能够在用户模式下创建函数或存储过程。 |
| EXECUTE ANY FUNCTION | 用户拥有在public模式和用户模式下函数或存储过程的EXECUTE权限。 |
| CREATE ANY PACKAGE | 用户能够在public模式和用户模式下创建PACKAGE。 |
| EXECUTE ANY PACKAGE | 用户拥有在public模式和用户模式下PACKAGE的EXECUTE权限。 |
| CREATE ANY TYPE | 用户能够在public模式和用户模式下创建类型。 |
| CREATE ANY SEQUENCE | 用户能够在public模式和用户模式下创建序列。 |
| CREATE ANY INDEX | 用户能够在public模式和用户模式下创建索引。如果在某表空间创建分区表索引，需要授予用户该表空间的创建权限。 |

**说明：**

用户被授予任何一种ANY权限后，用户对public模式和用户模式具有USAGE权限，对除public之外的系统模式没有USAGE权限。

**示例**

**示例1：将系统权限授权给用户或者角色。**

创建名为joe的用户，并将sysadmin权限授权给他。

vastbase=# CREATE USER joe PASSWORD 'xxxxxxxxx';  
vastbase=# GRANT ALL PRIVILEGES TO joe;

授权成功后，用户joe会拥有sysadmin的所有权限。

**示例2：将对象权限授权给用户或者角色**。

1、销joe用户的sysadmin权限，然后将模式tpcds的使用权限和表tpcds.reason的所有权限授权给用户joe。

vastbase=# REVOKE ALL PRIVILEGES FROM joe;  
vastbase=# GRANT USAGE ON SCHEMA tpcds TO joe;  
vastbase=# GRANT ALL PRIVILEGES ON tpcds.reason TO joe;

授权成功后，joe用户就拥有了tpcds.reason表的所有权限，包括增删改查等权限。

2、将tpcds.reason表中r\_reason\_sk、r\_reason\_id、r\_reason\_desc列的查询权限，r\_reason\_desc的更新权限授权给joe。

vastbase=# GRANT select (r\_reason\_sk,r\_reason\_id,r\_reason\_desc),update (r\_reason\_desc) ON tpcds.reason TO joe;

授权成功后，用户joe对tpcds.reason表中r\_reason\_sk，r\_reason\_id的查询权限会立即生效。如果joe用户需要拥有将这些权限授权给其他用户的权限，可以通过以下语法对joe用户进行授权。

vastbase=# GRANT select (r\_reason\_sk, r\_reason\_id) ON tpcds.reason TO joe WITH GRANT OPTION;

将数据库Vastbase的连接权限授权给用户joe，并给予其在Vastbase中创建schema的权限，而且允许joe将此权限授权给其他用户。

vastbase=# GRANT create,connect on database Vastbase TO joe WITH GRANT OPTION;

创建角色tpcds\_manager，将模式tpcds的访问权限授权给角色tpcds\_manager，并授予该角色在tpcds下创建对象的权限，不允许该角色中的用户将权限授权给其他人。

vastbase=# CREATE ROLE tpcds\_manager PASSWORD 'xxxxxxxxx';  
vastbase=# GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds\_manager;

将表空间tpcds\_tbspc的所有权限授权给用户joe，但用户joe无法将权限继续授予其他用户。

vastbase=# CREATE TABLESPACE tpcds\_tbspc RELATIVE LOCATION 'tablespace/tablespace\_1';  
vastbase=# GRANT ALL ON TABLESPACE tpcds\_tbspc TO joe;

**示例3：将用户或者角色的权限授权给其他用户或角色。**

1、创建角色manager，将joe的权限授权给manager，并允许该角色将权限授权给其他人。

vastbase=# CREATE ROLE manager PASSWORD 'xxxxxxxxx';  
vastbase=# GRANT joe TO manager WITH ADMIN OPTION;

2、创建用户senior\_manager，将用户manager的权限授权给该用户。

vastbase=# CREATE ROLE senior\_manager PASSWORD 'xxxxxxxxx';  
vastbase=# GRANT manager TO senior\_manager;

3、撤销权限，并清理用户。

vastbase=# REVOKE manager FROM joe;  
vastbase=# REVOKE senior\_manager FROM manager;  
vastbase=# DROP USER manager;

**示例4：将CMK或者CEK的权限授权给其他用户或角色。**

1、连接密态数据库。

vsql -r  
vastbase=# CREATE CLIENT MASTER KEY MyCMK1 WITH ( KEY\_STORE = localkms , KEY\_PATH = "key\_path\_value" , ALGORITHM = RSA\_2048);  
CREATE CLIENT MASTER KEY  
vastbase=# CREATE COLUMN ENCRYPTION KEY MyCEK1 WITH VALUES (CLIENT\_MASTER\_KEY = MyCMK1, ALGORITHM = AEAD\_AES\_256\_CBC\_HMAC\_SHA256);  
CREATE COLUMN ENCRYPTION KEY

2、创建角色newuser，将密钥的权限授权给newuser。

vastbase=# CREATE USER newuser PASSWORD 'xxxxxxxxx';  
CREATE ROLE  
vastbase=# GRANT ALL ON SCHEMA public TO newuser;  
GRANT  
vastbase=# GRANT USAGE ON COLUMN\_ENCRYPTION\_KEY MyCEK1 to newuser;  
GRANT  
vastbase=# GRANT USAGE ON CLIENT\_MASTER\_KEY MyCMK1 to newuser;  
GRANT

3、设置该用户连接数据库，使用该CEK创建加密表。

vastbase=# SET SESSION AUTHORIZATION newuser PASSWORD 'xxxxxxxxx';  
vastbase=> CREATE TABLE acltest1 (x int, x2 varchar(50) ENCRYPTED WITH (COLUMN\_ENCRYPTION\_KEY = MyCEK1, ENCRYPTION\_TYPE = DETERMINISTIC));  
CREATE TABLE  
vastbase=> SELECT has\_cek\_privilege('newuser', 'MyCEK1', 'USAGE');  
 has\_cek\_privilege  
-------------------  
 t  
(1 row)

4、撤销权限，并清理用户。

vastbase=# REVOKE USAGE ON COLUMN\_ENCRYPTION\_KEY MyCEK1 FROM newuser;  
vastbase=# REVOKE USAGE ON CLIENT\_MASTER\_KEY MyCMK1 FROM newuser;  
vastbase=# DROP TABLE newuser.acltest1;  
vastbase=# DROP COLUMN ENCRYPTION KEY MyCEK1;  
vastbase=# DROP CLIENT MASTER KEY MyCMK1;  
vastbase=# DROP SCHEMA IF EXISTS newuser CASCADE;  
vastbase=# REVOKE ALL ON SCHEMA public FROM newuser;  
vastbase=# DROP ROLE IF EXISTS newuser;

**示例：撤销上述授予的权限，并清理角色和用户。**

vastbase=# REVOKE ALL PRIVILEGES ON tpcds.reason FROM joe;  
vastbase=# REVOKE ALL PRIVILEGES ON SCHEMA tpcds FROM joe;  
vastbase=# REVOKE ALL ON TABLESPACE tpcds\_tbspc FROM joe;  
vastbase=# DROP TABLESPACE tpcds\_tbspc;  
vastbase=# REVOKE USAGE,CREATE ON SCHEMA tpcds FROM tpcds\_manager;  
vastbase=# DROP ROLE tpcds\_manager;  
vastbase=# DROP ROLE senior\_manager;  
vastbase=# DROP USER joe CASCADE;

**相关链接**

参考：SQL语法参考->SQL语法->REVOKE、ALTER DEFAULT PRIVILEGES章节。

### MERGE

**功能描述**

通过MERGE INTO语句，将目标表和源表中数据针对关联条件进行匹配，若关联条件匹配时对目标表进行UPDATE，无法匹配时对目标表执行INSERT。此语法可以很方便地用来合并执行UPDATE和INSERT，避免多次执行。

**注意事项**

进行MERGE INTO操作的用户需要同时拥有目标表的UPDATE和INSERT权限，以及源表的SELECT权限。

**语法格式**

MERGE [/\*+ plan\_hint \*/] INTO table\_name [ partition\_clause ] [ [ AS ] alias ]

USING { { table\_name | view\_name } | subquery } [ [ AS ] alias ]

ON ( condition )

[

WHEN MATCHED THEN

UPDATE SET { column\_name = { expression | subquery | DEFAULT } |

( column\_name [, ...] ) = ( { expression | subquery | DEFAULT } [, ...] ) } [, ...]

[ WHERE condition ]

]

[

WHEN NOT MATCHED THEN

INSERT { DEFAULT VALUES |

[ ( column\_name [, ...] ) ] VALUES ( { expression | subquery | DEFAULT } [, ...] ) [, ...] [ WHERE condition ] }

];

其中partition\_clause可以是：

PARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) } |

SUBPARTITION { ( subpartition\_name ) | FOR ( subpartition\_value [, ...] ) }

**参数说明**

* **plan\_hint子句**
* 以/\*+\*/的形式在MERGE关键字后，用于对MERGE对应的语句块生成的计划进行hint调优，详细用法请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->性能调优->SQL调优指南->使用Plan Hint进行调优》章节。
* 每条语句中只有第一个/+ plan\_hint \*/注释块会作为hint生效，里面可以写多条hint。
* **INTO子句**
* 指定正在更新或插入的目标表。
* **talbe\_name**

目标表的表名。

* **Alias**

目标表的别名。

取值范围：字符串，符合标识符命名规范。

* **partition\_clause**
* 指定分区MERGE操作：

PARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) } |  
SUBPARTITION { ( subpartition\_name ) | FOR ( subpartition\_value [, ...] ) }

关键字详见：SQL语法参考->SQL语法->SELECT章节。

如果value子句的值和指定分区不一致，会抛出异常。

示例详见：SQL语法参考->SQL语法->CREATE TABLE SUBPARTITION章节。

* **alias**
* 目标表的别名。
* 取值范围：字符串，符合标识符命名规范。
* **USING子句**
* 指定源表，源表可以为表、视图或子查询。
* **ON子句**
* 关联条件，用于指定目标表和源表的关联条件。不支持更新关联条件中的字段。
* **WHEN MATCHED子句**
* 当源表和目标表中数据针对关联条件可以匹配上时，选择WHEN MATCHED子句进行UPDATE操作。
* 不支持更新系统表、系统列。
* **WHEN NOT MATCHED子句**
* 当源表和目标表中数据针对关联条件无法匹配时，选择WHEN NOT MATCHED子句进行INSERT操作。
* 不支持INSERT子句中包含多个VALUES。
* WHEN MATCHED和WHEN NOT MATCHED子句顺序可以交换，可以缺省其中一个，但不能同时缺省，不支持同时指定两个WHEN MATCHED或WHEN NOT MATCHED子句。
* **DEFAULT**
* 用对应字段的缺省值填充该字段。
* 如果没有缺省值，则为NULL。
* **WHERE condition**
* UPDATE子句和INSERT子句的条件，只有在条件满足时才进行更新操作，可缺省。不支持WHERE条件中引用系统列。不建议使用int等数值类型作为condition，因为int等数值类型可以隐式转换为bool值（非0值隐式转换为true，0转换为false），可能导致非预期的结果。

**示例**

1、创建目标表products和源表newproducts，并插入数据。

CREATE TABLE products

(

product\_id INTEGER,

product\_name VARCHAR2(60),

category VARCHAR2(60)

);

INSERT INTO products VALUES (1501, 'vivitar 35mm', 'electrncs');

INSERT INTO products VALUES (1502, 'olympus is50', 'electrncs');

INSERT INTO products VALUES (1600, 'play gym', 'toys');

INSERT INTO products VALUES (1601, 'lamaze', 'toys');

INSERT INTO products VALUES (1666, 'harry potter', 'dvd');

CREATE TABLE newproducts

(

product\_id INTEGER,

product\_name VARCHAR2(60),

category VARCHAR2(60)

);

INSERT INTO newproducts VALUES (1502, 'olympus camera', 'electrncs');

INSERT INTO newproducts VALUES (1601, 'lamaze', 'toys');

INSERT INTO newproducts VALUES (1666, 'harry potter', 'toys');

INSERT INTO newproducts VALUES (1700, 'wait interface', 'books');

2、进行MERGE INTO操作。

MERGE INTO products p

USING newproducts np

ON (p.product\_id = np.product\_id)

WHEN MATCHED THEN

UPDATE SET p.product\_name = np.product\_name, p.category = np.category WHERE p.product\_name != 'play gym'

WHEN NOT MATCHED THEN

INSERT VALUES (np.product\_id, np.product\_name, np.category) WHERE np.category = 'books';

提交回显结果为：

MERGE 4

3、查询更新后的结果。

SELECT \* FROM products ORDER BY product\_id;

显示结果为：

product\_id | product\_name | category

------------+----------------+-----------

1501 | vivitar 35mm | electrncs

1502 | olympus camera | electrncs

1600 | play gym | toys

1601 | lamaze | toys

1666 | harry potter | toys

1700 | wait interface | books

(6 rows)

### INSERT

**功能描述**

向表中添加一行或多行数据。

**注意事项**

* 只有拥有表INSERT权限的用户，才可以向表中插入数据。用户被授予insert any table权限，相当于用户对除系统模式之外的任何模式具有USAGE权限，并且拥有这些模式下表的INSERT权限
* 如果使用RETURNING子句，用户必须要有该表的SELECT权限。
* 如果使用ON DUPLICATE KEY UPDATE，用户必须要有该表的SELECT、UPDATE权限，唯一约束（主键或唯一索引）的SELECT权限。
* 如果使用query子句插入来自查询里的数据行，用户还需要拥有在查询里使用的表的SELECT权限。
* 当连接到TD兼容的数据库时，td\_compatible\_truncation参数设置为on时，将启用超长字符串自动截断功能，在后续的insert语句中（不包含外表的场景下），对目标表中char和varchar类型的列上插入超长字符串时，系统会自动按照目标表中相应列定义的最大长度对超长字符串进行截断。
* fig: **说明：**   
  如果向字符集为字节类型编码（SQL\_ASCII，LATIN1等）的数据库中插入多字节字符数据（如汉字等），且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

**语法格式**

[ WITH [ RECURSIVE ] with\_query [, ...] ]  
INSERT [/\*+ plan\_hint \*/] INTO table\_name [partition\_clause] [ AS alias ] [ ( column\_name [, ...] ) ]  
 { DEFAULT VALUES  
 | VALUES {( { expression | DEFAULT } [, ...] ) }[, ...]   
 | query }  
 [ ON DUPLICATE KEY UPDATE { NOTHING | { column\_name = { expression | DEFAULT } } [, ...] [ WHERE condition ] }]  
 [ RETURNING {\* | {output\_expression [ [ AS ] output\_name ] }[, ...]} ];

**参数说明**

* **WITH [ RECURSIVE ] with\_query [, ...]**
* 用于声明一个或多个可以在主查询中通过名称引用的子查询，相当于临时表。
* 如果声明了RECURSIVE，那么允许SELECT子查询通过名称引用它自己。

1、其中with\_query的详细格式为：

with\_query\_name [ ( column\_name [, ...] ) ] AS [ [ NOT ] MATERIALIZED ]  
 ( {select | values | insert | update | delete} )

2、with\_query\_name指定子查询生成的结果集名称，在查询中可使用该名称访问子查询的结果集。

3、column\_name指定子查询结果集中显示的列名。

4、每个子查询可以是SELECT，VALUES，INSERT，UPDATE或DELETE语句。

5、用户可以使用MATERIALIZED / NOT MATERIALIZED对CTE进行修饰。

* 如果声明为MATERIALIZED，WITH查询将被物化，生成一个子查询结果集的拷贝，在引用处直接查询该拷贝，因此WITH子查询无法和主干SELECT语句进行联合优化（如谓词下推、等价类传递等），对于此类场景可以使用NOT MATERIALIZED进行修饰，如果WITH查询语义上可以作为子查询内联执行，则可以进行上述优化。
* 如果用户没有显示声明物化属性则遵守以下规则：如果CTE只在所属主干语句中被引用一次，且语义上支持内联执行，则会被改写为子查询内联执行，否则以CTE Scan的方式物化执行。

fig: **说明：**   
INSERT ON DUPLICATE KEY UPDATE不支持WITH及WITH RECURSIVE子句。

* **plan\_hint子句**
* 以/\*+ \*/的形式在INSERT关键字后，用于对INSERT对应的语句块生成的计划进行hint调优，详细用法请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->性能调优->SQL调优指南->使用Plan Hint进行调优》章节。每条语句中只有第一个/\*+ plan\_hint \*/注释块会作为hint生效，里面可以写多条hint。
* **table\_name**
* 要插入数据的目标表名。
* 取值范围：已存在的表名。
* **partition\_clause**
* 指定分区插入操作

PARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) } |  
 SUBPARTITION { ( subpartition\_name ) | FOR ( subpartition\_value [, ...] ) }

关键字详见SQL语法参考->SQL语法->SELECT章节。

如果value子句的值和指定分区不一致，会抛出异常。

示例详见：SQL语法参考->SQL语法->CREATE TABLE SUBPARTITION章节。

* **column\_name**
* 目标表中的字段名：
* 字段名可以有子字段名或者数组下标修饰。
* 没有在字段列表中出现的每个字段，将由系统默认值，或者声明时的默认值填充，若都没有则用NULL填充。例如，向一个复合类型中的某些字段插入数据的话，其他字段将是NULL。
* 目标字段（column\_name）可以按顺序排列。如果没有列出任何字段，则默认全部字段，且顺序为表声明时的顺序。
* 如果value子句和query中只提供了N个字段，则目标字段为前N个字段。
* value子句和query提供的值在表中从左到右关联到对应列。
* 取值范围：已存在的字段名。
* **expression**
* 赋予对应column的一个有效表达式或值：
* 如果是INSERT ON DUPLICATE KEY UPDATE语句下，expression可以为VALUES(column\_name)或EXCLUDED.column\_name用来表示引用冲突行对应的column\_name字段的值。需注意，其中VALUES(column\_name)不支持嵌套在表达式中（例如VALUES(column\_name)+1），但EXCLUDED不受此限制。
* 向表中字段插入单引号 “ ' ”时需要使用单引号自身进行转义。
* 如果插入行的表达式不是正确的数据类型，系统试图进行类型转换，若转换不成功，则插入数据失败，系统返回错误信息。
* **DEFAULT**
* 对应字段名的缺省值。如果没有缺省值，则为NULL。
* **query**
* 一个查询语句（SELECT语句），将查询结果作为插入的数据。
* **RETURNING**
* 返回实际插入的行，RETURNING列表的语法与SELECT的输出列表一致。
* **注意：**INSERT ON DUPLICATE KEY UPDATE不支持RETURNING子句。
* **output\_expression**
* INSERT命令在每一行都被插入之后用于计算输出结果的表达式。
* 取值范围：该表达式可以使用table的任意字段。可以使用\*返回被插入行的所有字段。
* **output\_name**
* 字段的输出名称。
* 取值范围：字符串，符合标识符命名规范。
* **ON DUPLICATE KEY UPDATE**
* 对于带有唯一约束（UNIQUE INDEX或PRIMARY KEY）的表，如果插入数据违反唯一约束，则对冲突行执行UPDATE子句完成更新，对于不带唯一约束的表，则仅执行插入。UPDATE时，若指定NOTHING则忽略此条插入，可通过“EXCLUDE.” 或者 "VALUES()" 来选择源数据相应的列。
* 支持触发器，触发器执行顺序由实际执行流程决定：
* 执行insert：触发 before insert、after insert触发器。
* 执行update：触发before insert、before update、after update触发器。
* 执行update nothing：触发before insert触发器。
* 不支持延迟生效（DEFERRABLE）的唯一约束或主键。
* 如果表中存在多个唯一约束，如果所插入数据违反多个唯一约束，对于检测到冲突的第一行进行更新，其他冲突行不更新（检查顺序与索引维护具有强相关性，一般先创建的索引先进行冲突检查）。
* 如果插入多行，这些行均与表中同一行数据存在唯一约束冲突，则按照顺序，第一条执行插入或更新，之后依次执行更新。
* 主键、唯一索引列不允许UPDATE。
* 不支持列存，不支持外表、内存表。
* expression支持使用子查询表达式，其语法与功能同UPDATE。子查询表达式中支持使用“EXCLUDED.”来选择源数据相应的列。

**示例**

--创建表reason\_t2。  
vastbase=# CREATE TABLE reason\_t2  
(  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
);  
  
--向表中插入一条记录。  
vastbase=# INSERT INTO reason\_t2(r\_reason\_sk, r\_reason\_id, r\_reason\_desc) VALUES (1, 'AAAAAAAABAAAAAAA', 'reason1');  
  
--向表中插入一条记录，和上一条语法等效。  
vastbase=# INSERT INTO reason\_t2 VALUES (2, 'AAAAAAAABAAAAAAA', 'reason2');  
  
--向表中插入多条记录。  
vastbase=# INSERT INTO reason\_t2 VALUES (3, 'AAAAAAAACAAAAAAA','reason3'),(4, 'AAAAAAAADAAAAAAA', 'reason4'),(5, 'AAAAAAAAEAAAAAAA','reason5');  
  
--向表中插入reason中r\_reason\_sk小于5的记录。  
vastbase=# INSERT INTO reason\_t2 SELECT \* FROM reason WHERE r\_reason\_sk <5;  
  
--对表创建唯一索引  
vastbase=# CREATE UNIQUE INDEX reason\_t2\_u\_index ON reason\_t2(r\_reason\_sk);  
  
--向表中插入多条记录，如果冲突则更新冲突数据行中r\_reason\_id字段为'BBBBBBBBCAAAAAAA'。  
vastbase=# INSERT INTO reason\_t2 VALUES (5, 'BBBBBBBBCAAAAAAA','reason5'),(6, 'AAAAAAAADAAAAAAA', 'reason6') ON DUPLICATE KEY UPDATE r\_reason\_id = 'BBBBBBBBCAAAAAAA';  
  
--删除表reason\_t2。  
vastbase=# DROP TABLE reason\_t2;

**优化建议**

* VALUES
* 通过insert语句批量插入数据时，建议将多条记录合并入一条语句中执行插入，以提高数据加载性能。例如，INSERT INTO sections VALUES (30, 'Administration', 31, 1900)、(40, 'Development', 35, 2000)、 (50, 'Development' , 60 , 2001)。

### LOCK

**功能描述**

LOCK TABLE获取表级锁。

Vastbase在为一个引用了表的命令自动请求锁时，尽可能选择最小限制的锁模式。如果用户需要一种更为严格的锁模式，可以使用LOCK命令。例如，一个应用是在Read Committed隔离级别上运行事务，并且它需要保证表中的数据在事务的运行过程中不被修改。为实现这个目的，则可以在查询之前对表使用SHARE锁模式进行锁定。这样将防止数据不被并发修改，从而保证后续的查询可以读到已提交的持久化的数据。因为SHARE锁模式与任何写操作需要的ROW EXCLUSIVE模式冲突，并且LOCK TABLE name IN SHARE MODE语句将等到所有当前持有ROW EXCLUSIVE模式锁的事务提交或回滚后才能执行。因此，一旦获得该锁，就不会存在未提交的写操作，并且其他操作也只能等到该锁释放之后才能开始。

**注意事项**

* LOCK TABLE只能在一个事务块的内部有用，因为锁在事务结束时就会被释放。出现在任意事务块外面的LOCK TABLE都会报错。
* 如果没有声明锁模式，缺省为最严格的模式ACCESS EXCLUSIVE。
* LOCK TABLE ... IN ACCESS SHARE MODE需要在目标表上有SELECT权限。所有其他形式的LOCK需要UPDATE和/或DELETE权限。
* 没有UNLOCK TABLE命令，锁总是在事务结束时释放。
* LOCK TABLE只处理表级的锁，因此那些带“ROW”字样的锁模式都是有歧义的。这些模式名称通常可理解为用户试图在一个被锁定的表中获取行级的锁。同样，ROW EXCLUSIVE模式也是一个可共享的表级锁。注意，只要是涉及到LOCK TABLE ，所有锁模式都有相同的语意，区别仅在于规则中锁与锁之间是否冲突，规则请参见表1：冲突的锁模式。
* 如果没有打开xc\_maintenance\_mode参数，那么对系统表申请ACCESS EXCLUSIVE级别锁将报错。

**语法格式**

LOCK [ TABLE ] {[ ONLY ] name [, ...]| {name [ \* ]} [, ...]}  
 [ IN {ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE} MODE ]  
 [ NOWAIT ];

**参数说明**

**表 1** 冲突的锁模式

| **请求的锁模式/当前锁模式** | **ACCESS SHARE** | **ROW SHARE** | **ROW EXCLUSIVE** | **SHARE UPDATE EXCLUSIVE** | **SHARE** | **SHARE ROW EXCLUSIVE** | **EXCLUSIVE** | **ACCESS EXCLUSIVE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ACCESS SHARE | - | - | - | - | - | - | - | X |
| ROW SHARE | - | - | - | - | - | - | X | X |
| ROW EXCLUSIVE | - | - | - | - | X | X | X | X |
| SHARE UPDATE EXCLUSIVE | - | - | - | X | X | X | X | X |
| SHARE | - | - | X | X | - | X | X | X |
| SHARE ROW EXCLUSIVE | - | - | X | X | X | X | X | X |
| EXCLUSIVE | - | X | X | X | X | X | X | X |
| ACCESS EXCLUSIVE | X | X | X | X | X | X | X | X |

LOCK的参数说明如下所示：

* **name**
* 要锁定的表的名称，可以有模式修饰。
* LOCK TABLE命令中声明的表的顺序就是上锁的顺序。
* 取值范围：已存在的表名。
* **ONLY**
* 如果指定ONLY，只有该表被锁定。如果没有声明，该表和他的所有子表将都被锁定。
* **ACCESS SHARE**
* ACCESS锁只允许对表进行读取，而禁止对表进行修改。所有对表进行读取而不修改的SQL语句都会自动请求这种锁。例如，SELECT命令会自动在被引用的表上请求一个这种锁。
* **ROW SHARE**
* 与EXCLUSIVE和ACCESS EXCLUSIVE锁模式冲突。
* SELECT FOR UPDATE和SELECT FOR SHARE命令会自动在目标表上请求ROW SHARE锁（且所有被引用但不是FOR SHARE/FOR UPDATE的其他表上，还会自动加上ACCESS SHARE锁）。
* **ROW EXCLUSIVE**
* 与ROW SHARE锁相同，ROW EXCLUSIVE允许并发读取表，但是禁止修改表中数据。UPDATE，DELETE，INSERT命令会自动在目标表上请求这个锁（且所有被引用的其他表上还会自动加上的ACCESS SHARE锁）。通常情况下，所有会修改表数据的命令都会请求表的ROW EXCLUSIVE锁。
* **SHARE UPDATE EXCLUSIVE**
* 这个模式保护一个表的模式不被并发修改，以及禁止在目标表上执行垃圾回收命令（VACUUM ）。
* VACUUM（不带FULL选项），ANALYZE，CREATE INDEX CONCURRENTLY命令会自动请求这样的锁。
* **SHARE**
* SHARE锁允许并发的查询，但是禁止对表进行修改。
* CREATE INDEX（不带CONCURRENTLY选项）语句会自动请求这种锁。
* **SHARE ROW EXCLUSIVE**
* SHARE ROW EXCLUSIVE锁禁止对表进行任何的并发修改，而且是独占锁，因此一个会话中只能获取一次。
* 任何SQL语句都不会自动请求这个锁模式。
* **EXCLUSIVE**
* EXCLUSIVE锁允许对目标表进行并发查询，但是禁止任何其他操作。
* 这个模式只允许并发加ACCESS SHARE锁，也就是说，只有对表的读动作可以和持有这个锁模式的事务并发执行。
* 任何SQL语句都不会在用户表上自动请求这个锁模式。然而在某些操作的时候，会在某些系统表上请求它。
* **ACCESS EXCLUSIVE**
* 这个模式保证其所有者（事务）是可以访问该表的唯一事务。
* ALTER TABLE，DROP TABLE，TRUNCATE，REINDEX命令会自动请求这种锁。
* 在LOCK TABLE命令没有明确声明需要的锁模式时，它是缺省锁模式。
* **NOWAIT**
* 声明LOCK TABLE不去等待任何冲突的锁释放，如果无法立即获取该锁，该命令退出并且发出一个错误信息。
* 在不指定NOWAIT的情况下获取表级锁时，如果有其他互斥锁存在的话，则等待其他锁的释放。

**示例**

--在执行删除操作时对一个有主键的表进行 SHARE ROW EXCLUSIVE 锁。  
vastbase=# CREATE TABLE reason\_t1 AS TABLE reason;  
  
vastbase=# START TRANSACTION;  
  
vastbase=# LOCK TABLE reason\_t1 IN SHARE ROW EXCLUSIVE MODE;  
  
vastbase=# DELETE FROM reason\_t1 WHERE r\_reason\_desc IN(SELECT r\_reason\_desc FROM reason\_t1 WHERE r\_reason\_sk < 6 );  
  
vastbase=# DELETE FROM reason\_t1 WHERE r\_reason\_sk = 7;  
  
vastbase=# COMMIT;  
  
--删除表reason\_t1。  
vastbase=# DROP TABLE reason\_t1;

### MOVE

**功能描述**

MOVE在不检索数据的情况下重新定位一个游标。MOVE的作用类似于FETCH命令，但只是重定位游标而不返回行。

**注意事项**

无。

**语法格式**

MOVE [ direction [ FROM | IN ] ] cursor\_name;

其中direction子句为可选参数。

NEXT  
 | PRIOR  
 | FIRST  
 | LAST  
 | ABSOLUTE count  
 | RELATIVE count  
 | count  
 | ALL  
 | FORWARD  
 | FORWARD count  
 | FORWARD ALL  
 | BACKWARD  
 | BACKWARD count  
 | BACKWARD ALL

**参数说明**

MOVE命令的参数与FETCH的相同，详细请参见：SQL语法参考->SQL语法->FETCH章节中的参数说明。

fig: **说明：**   
成功完成时，MOVE命令将返回一个“MOVE count”的标签，count是一个使用相同参数的FETCH命令会返回的行数（可能为零）。

**示例**

--开始一个事务。  
vastbase=# START TRANSACTION;  
  
--定义一个名为cursor1的游标。  
vastbase=# CURSOR cursor1 FOR SELECT \* FROM tpcds.reason;  
  
--忽略游标cursor1的前3行。  
vastbase=# MOVE FORWARD 3 FROM cursor1;  
  
--抓取游标cursor1的前4行。  
vastbase=# FETCH 4 FROM cursor1;  
 r\_reason\_sk | r\_reason\_id | r\_reason\_desc   
-------------+------------------+------------------------------------------------------------------------------------------------------  
 4 | AAAAAAAAEAAAAAAA | Not the product that was ordred   
 5 | AAAAAAAAFAAAAAAA | Parts missing   
 6 | AAAAAAAAGAAAAAAA | Does not work with a product that I have   
 7 | AAAAAAAAHAAAAAAA | Gift exchange   
(4 rows)  
  
--关闭游标。  
vastbase=# CLOSE cursor1;  
  
--结束一个事务。  
vastbase=# END;

**相关链接**

参考：SQL语法参考->SQL语法->CLOSE、FETCH章节。

### PREDICT BY

**功能描述**

利用完成训练的模型进行推测任务。

**注意事项**

调用的模型名称在系统表gs\_model\_warehouse中可查看到。

**语法格式**

PREDICT BY model\_name [ (FEATURES attribute [, attribute] +]) ]

**参数说明**

* model\_name
* 用于推测任务的模型名称。
* 取值范围：字符串，需要符合标识符的命名规则。
* attribute
* 推测任务的输入特征列名。
* 取值范围：字符串，需要符合标识符的命名规则。

**示例**

SELECT id, PREDICT BY price\_model (FEATURES size,lot), price  
FROM houses;

**相关链接**

参考：SQL语法参考->SQL语法->CREATE MODEL、DROP MODEL章节。

### PREPARE

**功能描述**

创建一个预备语句。

预备语句是服务端的对象，可以用于优化性能。在执行PREPARE语句的时候，指定的查询被解析、分析、重写。当随后发出EXECUTE语句的时候，预备语句被规划和执行。这种设计避免了重复解析、分析工作。PREPARE语句创建后在整个数据库会话期间一直存在，一旦创建成功，即便是在事务块中创建，事务回滚，PREPARE也不会删除。只能通过显式调用DEALLOCATE进行删除，会话结束时，PREPARE也会自动删除。

**注意事项**

无。

**语法格式**

PREPARE name [ ( data\_type [, ...] ) ] AS statement;

**参数说明**

* **name**
* 指定预备语句的名称。它必须在该会话中是唯一的。
* **data\_type**
* 参数的数据类型。
* **statement**
* 是SELECT INSERT、UPDATE、DELETE、MERGE INTO或VALUES语句之一。

**示例**

请参见：SQL语法参考->SQL语法->EXECUTE章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->DEALLOCATE章节。

### PREPARE TRANSACTION

**功能描述**

为当前事务做两阶段提交的准备。

在命令之后，事务就不再和当前会话关联了；它的状态完全保存在磁盘上，它被提交成功的可能性非常高，即使是在请求提交之前数据库发生了崩溃也如此。

一旦准备好了，一个事务就可以在稍后用COMMIT PREPARED或 ROLLBACK PREPARED命令分别进行提交或者回滚。这些命令可以从任何会话中发出，而不光是最初执行事务的那个会话。

从发出命令的会话的角度来看，PREPARE TRANSACTION不同于ROLLBACK：在执行它之后，就不再有活跃的当前事务了，并且预备事务的效果无法见到（在事务提交的时候其效果会再次可见）。

如果PREPARE TRANSACTION因为某些原因失败，那么它就会变成一个ROLLBACK，当前事务被取消。

**注意事项**

* 事务功能由数据库自动维护，不应显式使用事务功能。
* 在运行PREPARE TRANSACTION命令时，必须在postgresql.conf配置文件中增大max\_prepared\_transactions的数值。建议至少将其设置为等于max\_connections，这样每个会话都可以有一个等待中的预备事务。

**语法格式**

PREPARE TRANSACTION transaction\_id;

**参数说明**

**transaction\_id**

待提交事务的标识符，用于后面在COMMIT PREPARED或ROLLBACK PREPARED的时候标识这个事务。它不能和任何当前预备事务已经使用了的标识符同名。

取值范围：标识符必须以字符串文本的方式书写，并且必须小于200字节长。

**相关链接**

参考：SQL语法参考->SQL语法->COMMIT PREPARED、ROLLBACK PREPARED章节。

### PURGE

**功能描述**

使用PURGE语句可以实现如下功能：

* 从回收站中清理表或索引，并释放对象相关的全部空间。
* 清理回收站。
* 清理回收站中指定表空间的对象。

**注意事项**

* 清除（PURGE）操作支持：表（PURGE TABLE）、索引（PURGE INDEX）、回收站（PURGE RECYCLEBIN）。
* 执行PURGE操作的权限要求如下：
* PURGE TABLE：用户必须是表的所有者，且用户必须拥有表所在模式的USAGE权限，系统管理员默认拥有此权限。
* PURGE INDEX：用户必须是索引的所有者，用户必须拥有索引所在模式的USAGE权限，系统管理员默认拥有此权限。
* PURGE RECYCLEBIN：普通用户只能清理回收站中当前用户拥有的对象，且用户必须拥有对象所在模式的USAGE权限，系统管理员默认可以清理回收站所有对象。

**语法格式**

PURGE { TABLE [schema\_name.]table\_name   
 | INDEX index\_name   
 | RECYCLEBIN   
 }

**参数说明**

* [ schema\_name. ]
* 模式名。
* TABLE [ schema\_name. ] table\_name
* 清空回收站中指定的表。
* INDEX index\_name
* 清空回收站中指定的索引。
* RECYCLEBIN
* 清空回收站中的对象。

**示例**

-- 创建表空间reason\_table\_space  
vastbase=# CREATE TABLESPACE REASON\_TABLE\_SPACE1 owner tpcds RELATIVE location 'tablespace/tsp\_reason1';  
-- 在表空间创建表tpcds.reason\_t1  
vastbase=# CREATE TABLE tpcds.reason\_t1  
 (  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
 ) tablespace reason\_table\_space1;  
-- 在表空间创建表tpcds.reason\_t2  
vastbase=# CREATE TABLE tpcds.reason\_t2  
 (  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
 ) tablespace reason\_table\_space1;  
-- 在表空间创建表tpcds.reason\_t3  
vastbase=# CREATE TABLE tpcds.reason\_t3  
 (  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
 ) tablespace reason\_table\_space1;  
-- 对表tpcds.reason\_t1创建索引  
vastbase=# CREATE INDEX index\_t1 on tpcds.reason\_t1(r\_reason\_id);   
vastbase=# DROP TABLE tpcds.reason\_t1;  
vastbase=# DROP TABLE tpcds.reason\_t2;  
vastbase=# DROP TABLE tpcds.reason\_t3;  
--查看回收站  
vastbase=# SELECT rcyname,rcyoriginname,rcytablespace FROM GS\_RECYCLEBIN;  
 rcyname | rcyoriginname | rcytablespace   
-----------------------+---------------+---------------  
 BIN$16409$2CEE988==$0 | reason\_t1 | 16408  
 BIN$16412$2CF2188==$0 | reason\_t2 | 16408  
 BIN$16415$2CF2EC8==$0 | reason\_t3 | 16408  
 BIN$16418$2CF3EC8==$0 | index\_t1 | 0  
(4 rows)  
--PURGE清除表  
vastbase=# PURGE TABLE tpcds.reason\_t3;  
vastbase=# SELECT rcyname,rcyoriginname,rcytablespace FROM GS\_RECYCLEBIN;  
 rcyname | rcyoriginname | rcytablespace   
-----------------------+---------------+---------------  
 BIN$16409$2CEE988==$0 | reason\_t1 | 16408  
 BIN$16412$2CF2188==$0 | reason\_t2 | 16408  
 BIN$16418$2CF3EC8==$0 | index\_t1 | 0  
(3 rows)  
--PURGE清除索引  
vastbase=# PURGE INDEX tcpds.index\_t1;  
vastbase=# SELECT rcyname,rcyoriginname,rcytablespace FROM GS\_RECYCLEBIN;  
 rcyname | rcyoriginname | rcytablespace   
-----------------------+---------------+---------------  
 BIN$16409$2CEE988==$0 | reason\_t1 | 16408  
 BIN$16412$2CF2188==$0 | reason\_t2 | 16408  
(2 rows)  
--PURGE清除回收站所有对象  
vastbase=# PURGE recyclebin;  
vastbase=# SELECT rcyname,rcyoriginname,rcytablespace FROM GS\_RECYCLEBIN;  
 rcyname | rcyoriginname | rcytablespace   
-----------------------+---------------+---------------  
(0 rows)

### REASSIGN OWNED

**功能描述**

修改数据库对象的属主。

REASSIGN OWNED要求系统将所有old\_roles拥有的数据库对象的属主更改为new\_role。

**注意事项**

* REASSIGN OWNED常用于在删除角色之前的准备工作。
* 执行REASSIGN OWNED需要有原角色和目标角色上的权限。

**语法格式**

REASSIGN OWNED BY old\_role [, ...] TO new\_role;

**参数说明**

* **old\_role**
* 旧属主的角色名。
* **new\_role**
* 将要成为这些对象属主的新角色的名称。

**示例**

无。

### REFRESH INCREMENTAL MATERIALIZED VIEW

**功能描述**

REFRESH INCREMENTAL MATERIALIZED VIEW会以增量刷新的方式对物化视图进行刷新。

**注意事项**

* 增量刷新仅支持增量物化视图。
* 刷新物化视图需要当前用户拥有基表的SELECT权限。

**语法格式**

REFRESH INCREMENTAL MATERIALIZED VIEW mv\_name;

**参数说明**

* **mv\_name**
* 要刷新的物化视图的名称。

**示例**

--创建一个普通表  
vastbase=# CREATE TABLE my\_table (c1 int, c2 int);  
--创建增量物化视图  
vastbase=# CREATE INCREMENTAL MATERIALIZED VIEW my\_imv AS SELECT \* FROM my\_table;  
--基表写入数据  
vastbase=# INSERT INTO my\_table VALUES(1,1),(2,2);  
--对增量物化视图my\_imv进行增量刷新  
vastbase=# REFRESH INCREMENTAL MATERIALIZED VIEW my\_imv;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER MATERIALIZED VIEW、CREATE INCREMENTAL MATERIALIZED VIEW、CREATE MATERIALIZED VIEW、CREATE TABLE、DROP MATERIALIZED VIEW、REFRESH MATERIALIZED VIEW章节。

### MATERIALIZED VIEW

**功能描述**

REFRESH MATERIALIZED VIEW会以全量刷新的方式对物化视图进行刷新。

**注意事项**

* 全量刷新既可以对全量物化视图执行，也可以对增量物化视图执行。
* 刷新物化视图需要当前用户拥有基表的SELECT权限。

**语法格式**

REFRESH MATERIALIZED VIEW mv\_name;

**参数说明**

* **mv\_name**
* 要刷新的物化视图的名称。

**示例**

--创建一个普通表  
vastbase=# CREATE TABLE my\_table (c1 int, c2 int);  
--创建全量物化视图  
vastbase=# CREATE MATERIALIZED VIEW my\_mv AS SELECT \* FROM my\_table;  
--创建增量物化视图  
vastbase=# CREATE INCREMENTAL MATERIALIZED VIEW my\_imv AS SELECT \* FROM my\_table;  
--基表写入数据  
vastbase=# INSERT INTO my\_table VALUES(1,1),(2,2);  
--对全量物化视图my\_mv进行全量刷新  
vastbase=# REFRESH MATERIALIZED VIEW my\_mv;  
--对增量物化视图my\_imv进行全量刷新  
vastbase=# REFRESH MATERIALIZED VIEW my\_imv;

**相关链接**

参考：SQL语法参考->SQL语法->ALTER MATERIALIZED VIEW、CREATE INCREMENTAL MATERIALIZED VIEW、REATE MATERIALIZED VIEW、CREATE TABLE、DROP MATERIALIZED VIEW、REFRESH INCREMENTAL MATERIALIZED VIEW章节。

### REINDEX

**功能描述**

为表中的数据重建索引。

在以下几种情况下需要使用REINDEX重建索引：

* 索引崩溃，并且不再包含有效的数据。
* 索引变得“臃肿”，包含大量的空页或接近空页。
* 为索引更改了存储参数（例如填充因子），并且希望这个更改完全生效。
* 使用CONCURRENTLY选项创建索引失败，留下了一个“非法”索引。

**注意事项**

* REINDEX DATABASE和SYSTEM这种形式的重建索引不能在事务块中执行。
* REINDEX CONCURRENTLY这种形式的重建索引不能在事务块中执行。

**语法格式**

* 重建普通索引。

REINDEX { INDEX | [INTERNAL] TABLE | DATABASE | SYSTEM } [CONCURRENTLY] name [ FORCE ];

* 重建索引分区。

REINDEX { INDEX| [INTERNAL] TABLE} [CONCURRENTLY] name  
 PARTITION partition\_name [ FORCE ];

**参数说明**

* **INDEX**
* 重新建立指定的索引。
* **INTERNAL TABLE**
* 重建列存表或Hadoop内表的Desc表的索引，如果表有从属的“TOAST”表，则这个表也会重建索引。
* **TABLE**
* 重新建立指定表的所有索引，如果表有从属的“TOAST”表，则这个表也会重建索引。如果表上有索引已经被alter unusable失效，则这个索引无法被重新创建。
* **DATABASE**
* 重建当前数据库里的所有索引。
* **SYSTEM**
* 在系统库上重建所有系统表上的索引。不会处理在用户表上的索引。
* **CONCURRENTLY**
* 以不阻塞DML的方式重建索引（加ShareUpdateExclusiveLock锁）。重建索引时，一般会阻塞其他语句对该索引所依赖表的访问。指定此关键字，可以实现重建过程中不阻塞DML。
* 此选项只能指定一个索引的名称。
* 普通REINDEX命令可以在事务内执行，但是REINDEX CONCURRENTLY不可以在事务内执行。
* 列存表、全局分区表和临时表不支持CONCURRENTLY方式重建索引。
* REINDEX SYSTEM CONCURRENTLY不会执行任何操作，因为系统表不支持在线重建索引。
* 重建索引时指定此关键字，需要执行先后两次对该表的全表扫描来完成build，第一次扫描的时候创建新索引，不阻塞读写操作；第二次扫描的时候合并更新第一次扫描到目前为止发生的变更。
* 因为需要执行两次对表的扫描和build，且必须等待现有的所有可能对该表执行修改的事务结束，所以该索引的重建比正常耗时更长，同时带来的CPU和I/O消耗对其他业务也会造成影响。
* 如果在索引构建时发生失败，那会留下一个“不可用”的索引。这个索引会被查询忽略，但它仍消耗更新开销。这种情况推荐的恢复方法是删除该索引并尝试再次CONCURRENTLY重建索引。
* 由于在第二次扫描之后，索引构建必须等待任何持有早于第二次扫描拿的快照的事务终止，而且建索引时加的ShareUpdateExclusiveLock锁（4级）会和大于等于4级的锁冲突，因此在创建这类索引时，容易引发卡住（hang）或者死锁问题。例如：
* 两个会话对同一个表重建CONCURRENTLY索引，会引起死锁问题；
* 两个会话，一个对表重建CONCURRENTLY索引，一个drop table，会引起死锁问题；
* 三个会话，会话1先对表a加锁，不提交，会话2接着对表b重建CONCURRENTLY索引，会话3接着对表a执行写入操作，在会话1事务未提交之前，会话2会一直被阻塞；
* 将事务隔离级别设置成可重复读（默认为读已提交），起两个会话，会话1起事务对表a执行写入操作，不提交，会话2对表b重建CONCURRENTLY索引，在会话1事务未提交之前，会话2会一直被阻塞。
* **name**
* 需要重建索引的索引、表、数据库的名称。表和索引可以有模式修饰。
* REINDEX DATABASE和SYSTEM只能重建当前数据库的索引，所以name必须和当前数据库名称相同。
* **FORCE**
* 无效选项，会被忽略。
* **partition\_name**
* 需要重建索引的分区的名称或者索引分区的名称。
* 取值范围：
* 如果前面是REINDEX INDEX，则这里应该指定索引分区的名称；
* 如果前面是REINDEX TABLE，则这里应该指定分区的名称；
* 如果前面是REINDEX INTERNAL TABLE，则这里应该指定列存分区表的分区的名称。

REINDEX DATABASE和SYSTEM这种形式的重建索引不能在事务块中执行。

**示例**

--创建一个行存表customer\_t1，并在customer\_t1表上的c\_customer\_sk字段创建索引。  
vastbase=# CREATE TABLE customer\_t1  
(  
 c\_customer\_sk integer not null,  
 c\_customer\_id char(16) not null,  
 c\_current\_cdemo\_sk integer ,  
 c\_current\_hdemo\_sk integer ,  
 c\_current\_addr\_sk integer ,  
 c\_first\_shipto\_date\_sk integer ,  
 c\_first\_sales\_date\_sk integer ,  
 c\_salutation char(10) ,  
 c\_first\_name char(20) ,  
 c\_last\_name char(30) ,  
 c\_preferred\_cust\_flag char(1) ,  
 c\_birth\_day integer ,  
 c\_birth\_month integer ,  
 c\_birth\_year integer ,  
 c\_birth\_country varchar(20) ,  
 c\_login char(13) ,  
 c\_email\_address char(50) ,  
 c\_last\_review\_date char(10)  
)  
WITH (orientation = row)  
  
vastbase=# CREATE INDEX customer\_index1 ON customer\_t1 (c\_customer\_sk);  
  
vastbase=# INSERT INTO customer\_t1 SELECT \* FROM customer WHERE c\_customer\_sk < 10;  
  
--重建一个单独索引。  
vastbase=# REINDEX INDEX customer\_index1;  
  
--实时重建一个单独索引。  
vastbase=# REINDEX INDEX CONCURRENTLY customer\_index1;  
  
--重建表customer\_t1上的所有索引。  
vastbase=# REINDEX TABLE customer\_t1;  
  
--实时重建表customer\_t1上的所有索引。  
vastbase=# REINDEX TABLE CONCURRENTLY customer\_t1;  
  
--删除tpcds.customer\_t1表。  
vastbase=# DROP TABLE customer\_t1;

**优化建议**

* INTERNAL TABLE
* 此种情况大多用于故障恢复，不建议进行并发操作。
* DATABASE
* 不能在事务中reindex database。
* SYSTEM
* 不能在事务中reindex系统表。

### RELEASE SAVEPOINT

**功能描述**

RELEASE SAVEPOINT删除一个当前事务先前定义的保存点。

把一个保存点删除就令其无法作为回滚点使用，除此之外它没有其它用户可见的行为。它并不能撤销在保存点建立起来之后执行的命令的影响。要撤销那些命令可以使用ROLLBACK TO SAVEPOINT 。在不再需要的时候删除一个保存点可以令系统在事务结束之前提前回收一些资源。

RELEASE SAVEPOINT也删除所有在指定的保存点建立之后的所有保存点。

**注意事项**

* 不能RELEASE一个没有定义的保存点，语法上会报错。
* 如果事务在回滚状态，则不能释放保存点。
* 如果多个保存点拥有同样的名称，只有最近定义的那个才被释放。

**语法格式**

RELEASE [ SAVEPOINT ] savepoint\_name;

**参数说明**

**savepoint\_name**

要删除的保存点的名称。

**示例**

--创建一个新表。  
vastbase=# CREATE TABLE tpcds.table1(a int);  
  
--开启事务。  
vastbase=# START TRANSACTION;  
  
--插入数据。  
vastbase=# INSERT INTO tpcds.table1 VALUES (3);  
  
--建立保存点。  
vastbase=# SAVEPOINT my\_savepoint;  
  
--插入数据。  
vastbase=# INSERT INTO tpcds.table1 VALUES (4);  
  
--删除保存点。  
vastbase=# RELEASE SAVEPOINT my\_savepoint;  
  
--提交事务。  
vastbase=# COMMIT;  
  
--查询表的内容，会同时看到3和4。  
vastbase=# SELECT \* FROM tpcds.table1;  
  
--删除表。  
vastbase=# DROP TABLE tpcds.table1;

**相关链接**

参考：SQL语法参考->SQL语法->SAVEPOINT、ROLLBACK TO SAVEPOINT章节。

### RESET

**功能描述**

RESET将指定的运行时参数恢复为缺省值。这些参数的缺省值是指postgresql.conf配置文件中所描述的参数缺省值。

RESET命令与如下命令的作用相同：

SET configuration\_parameter TO DEFAULT

**注意事项**

RESET的事务性行为和SET相同：它的影响将会被事务回滚撤销。

**语法格式**

RESET {configuration\_parameter | CURRENT\_SCHEMA | TIME ZONE | TRANSACTION ISOLATION LEVEL | SESSION AUTHORIZATION | ALL };

**参数说明**

* **configuration\_parameter**
* 运行时参数的名称。
* 取值范围：可以使用SHOW ALL命令查看运行时参数。
* **CURRENT\_SCHEMA**
* 当前模式。
* **TIME ZONE**
* 时区。
* **TRANSACTION ISOLATION LEVEL**
* 事务的隔离级别。
* **SESSION AUTHORIZATION**
* 当前会话的用户标识符。
* **ALL**
* 所有运行时参数。

**示例**

--把timezone设为缺省值。  
vastbase=# RESET timezone;  
  
--把所有参数设置为缺省值。  
vastbase=# RESET ALL;

**相关链接**

参考：SQL语法参考->SQL语法->SET、SHOW章节。

### REVOKE

**功能描述**

REVOKE用于撤销一个或多个角色的权限。

**注意事项**

非对象所有者试图在对象上REVOKE权限，命令按照以下规则执行：

* 如果授权用户没有该对象上的权限，则命令立即失败。
* 如果授权用户有部分权限，则只撤销那些有授权选项的权限。
* 如果授权用户没有授权选项，REVOKE ALL PRIVILEGES形式将发出一个错误信息，而对于其他形式的命令而言，如果是命令中指定名称的权限没有相应的授权选项，该命令将发出一个警告。
* 不允许对表分区进行REVOKE操作，对分区表进行REVOKE操作会引起告警。

**语法格式**

* 回收指定表或视图上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | ALTER | DROP | COMMENT | INDEX | VACUUM }[, ...]   
   | ALL [ PRIVILEGES ] }  
   ON { [ TABLE ] table\_name [, ...]  
   | ALL TABLES IN SCHEMA schema\_name [, ...] }  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收表上指定字段权限。
* REVOKE [ GRANT OPTION FOR ]  
   { {{ SELECT | INSERT | UPDATE | REFERENCES | COMMENT } ( column\_name [, ...] )}[, ...]   
   | ALL [ PRIVILEGES ] ( column\_name [, ...] ) }  
   ON [ TABLE ] table\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定序列上权限，LARGE字段属性可选，回收语句不区分序列是否为LARGE。
* REVOKE [ GRANT OPTION FOR ]  
   { { SELECT | UPDATE | ALTER | DROP | COMMENT }[, ...]   
   | ALL [ PRIVILEGES ] }  
   ON { [ [ LARGE ] SEQUENCE ] sequence\_name [, ...]  
   | ALL SEQUENCES IN SCHEMA schema\_name [, ...] }  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定数据库上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { CREATE | CONNECT | TEMPORARY | TEMP | ALTER | DROP | COMMENT } [, ...]   
   | ALL [ PRIVILEGES ] }  
   ON DATABASE database\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定域上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { USAGE | ALL [ PRIVILEGES ] }  
   ON DOMAIN domain\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定客户端加密主密钥上的权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { USAGE | DROP } [, ...] | ALL [PRIVILEGES] }  
   ON CLIENT\_MASTER\_KEYS client\_master\_keys\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定列加密密钥上的权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { USAGE | DROP } [, ...] | ALL [PRIVILEGES]}  
   ON COLUMN\_ENCRYPTION\_KEYS column\_encryption\_keys\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定目录上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { READ | WRITE | ALTER |DROP } [, ...] | ALL [ PRIVILEGES ] }  
   ON DIRECTORY directory\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定外部数据源上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { USAGE | ALL [ PRIVILEGES ] }  
   ON FOREIGN DATA WRAPPER fdw\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定外部服务器上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON FOREIGN SERVER server\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定函数上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON { FUNCTION {function\_name ( [ {[ argmode ] [ arg\_name ] arg\_type} [, ...] ] )} [, ...]  
   | ALL FUNCTIONS IN SCHEMA schema\_name [, ...] }  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定存储过程上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON { PROCEDURE {proc\_name ( [ {[ argmode ] [ arg\_name ] arg\_type} [, ...] ] )} [, ...]  
   | ALL PROCEDURE IN SCHEMA schema\_name [, ...] }  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定过程语言上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { USAGE | ALL [ PRIVILEGES ] }  
   ON LANGUAGE lang\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定大对象上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }  
   ON LARGE OBJECT loid [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定模式上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { CREATE | USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON SCHEMA schema\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定表空间上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { CREATE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON TABLESPACE tablespace\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收指定类型上权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { USAGE | ALTER | DROP | COMMENT } [, ...] | ALL [ PRIVILEGES ] }  
   ON TYPE type\_name [, ...]  
   FROM { [ GROUP ] role\_name | PUBLIC } [, ...]  
   [ CASCADE | RESTRICT ];
* 回收Data Source对象上的权限。
* REVOKE [ GRANT OPTION FOR ]  
   { USAGE | ALL [PRIVILEGES] }  
   ON DATA SOURCE src\_name [, ...]  
   FROM {[GROUP] role\_name | PUBLIC} [, ...]  
   [ CASCADE | RESTRICT ];
* 回收package对象的权限。
* REVOKE [ GRANT OPTION FOR ]  
   { { EXECUTE | ALTER | DROP | COMMENT } [, ...] | ALL [PRIVILEGES] }  
   ON PACKAGE package\_name [, ...]  
   FROM {[GROUP] role\_name | PUBLIC} [, ...]  
   [ CASCADE | RESTRICT ];
* 按角色回收角色上的权限。
* REVOKE [ ADMIN OPTION FOR ]  
   role\_name [, ...] FROM role\_name [, ...]  
   [ CASCADE | RESTRICT ];
* 回收角色上的sysadmin权限。
* REVOKE ALL { PRIVILEGES | PRIVILEGE } FROM role\_name;
* 回收ANY权限。
* REVOKE [ ADMIN OPTION FOR ]  
   { CREATE ANY TABLE | ALTER ANY TABLE | DROP ANY TABLE | SELECT ANY TABLE | INSERT ANY TABLE | UPDATE ANY TABLE |  
   DELETE ANY TABLE | CREATE ANY SEQUENCE | CREATE ANY INDEX | CREATE ANY FUNCTION | EXECUTE ANY FUNCTION |  
   CREATE ANY PACKAGE | EXECUTE ANY PACKAGE | CREATE ANY TYPE } [, ...]  
   FROM [ GROUP ] role\_name [, ...];

**参数说明**

* 关键字PUBLIC表示一个隐式定义的拥有所有角色的组。
* 权限类别和参数说明，请参见：SQL语法参考->SQL语法->GRANT章节中的参数说明。
* 任何特定角色拥有的特权包括直接授予该角色的特权、从该角色作为其成员的角色中得到的权限以及授予给PUBLIC的权限。因此，从PUBLIC收回SELECT特权并不一定会意味着所有角色都会失去在该对象上的SELECT特权，那些直接被授予的或者通过另一个角色被授予的角色仍然会拥有它。类似地，从一个用户收回SELECT后，如果PUBLIC仍有SELECT权限，该用户还是可以使用SELECT。
* 指定GRANT OPTION FOR时，只撤销对该权限授权的权力，而不撤销该权限本身。
* 如用户A拥有某个表的UPDATE权限，及WITH GRANT OPTION选项，同时A把这个权限赋予了用户B，则用户B持有的权限称为依赖性权限。当用户A持有的权限或者授权选项被撤销时，必须声明CASCADE，将所有依赖性权限都撤销。
* 一个用户只能撤销由它自己直接赋予的权限。例如，如果用户A被指定授权（WITH ADMIN OPTION）选项，且把一个权限赋予了用户B，然后用户B又赋予了用户C，则用户A不能直接将C的权限撤销。但是，用户A可以撤销用户B的授权选项，并且使用CASCADE。这样，用户C的权限就会自动被撤销。另外一个例子：如果A和B都赋予了C同样的权限，则A可以撤销他自己的授权选项，但是不能撤销B的，因此C仍然拥有该权限。
* 如果执行REVOKE的角色持有的权限是通过多层成员关系获得的，则具体是哪个包含的角色执行的该命令是不确定的。在这种场合下，最好的方法是使用SET ROLE成为特定角色，然后执行REVOKE，否则可能导致删除了不想删除的权限，或者是任何权限都没有删除。

**示例**

参考：SQL语法参考->SQL语法->GRANT章节中的示例。

**相关链接**

参考：SQL语法参考->SQL语法->GRANT章节。

### ROLLBACK

**功能描述**

回滚当前事务并取消当前事务中的所有更新。

在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销，数据库状态回到事务开始前。

**语法格式**

ROLLBACK [ WORK | TRANSACTION ];

**参数说明**

WORK | TRANSACTION：可选关键字。除了增加可读性，没有任何其他作用。

**注意事项**

无。

**示例**

1、 创建测试表test。

CREATE TABLE test(id int);

2、 开启一个事务 。

START TRANSACTION;

3、 插入数据。

INSERT INTO test(id) VALUES(1);

4、 查询当前数据。

select \* from test;

当结果显示如下信息，则表示插入成功。

id   
----  
 1  
(1 row)

5、 取消所有更改 。

ROLLBACK;

6、 查询数据进行验证。

select \* from test;

当结果显示如下信息，则表示回滚成功。

id   
----  
(0 rows)

### ROLLBACK PREPARED

**功能描述**

取消一个先前为两阶段提交准备好的事务。

**注意事项**

* 该功能仅在维护模式(GUC参数xc\_maintenance\_mode为on时)下可用。该模式谨慎打开，一般供维护人员排查问题使用，一般用户不应使用该模式。
* 要想回滚一个预备事务，必须是最初发起事务的用户，或者是系统管理员。
* 事务功能由数据库自动维护，不应显式使用事务功能。

**语法格式**

ROLLBACK PREPARED transaction\_id ;

**参数说明**

**transaction\_id**

待提交事务的标识符。它不能和任何当前预备事务已经使用了的标识符同名。

**相关链接**

参考：SQL语法参考->SQL语法->COMMIT PREPARED、PREPARE TRANSACTION章节。

### ROLLBACK TO SAVEPOINT

**功能描述**

ROLLBACK TO SAVEPOINT用于回滚到一个保存点，隐含地删除所有在该保存点之后建立的保存点。

回滚所有指定保存点建立之后执行的命令。保存点仍然有效，并且需要时可以再次回滚到该点。

**注意事项**

* 不能回滚到一个未定义的保存点，语法上会报错。
* 在保存点方面，游标有一些非事务性的行为。任何在保存点里打开的游标都会在回滚掉这个保存点之后关闭。如果一个前面打开了的游标在保存点里面，并且游标被一个FETCH命令影响，而这个保存点稍后回滚了，那么这个游标的位置仍然在FETCH让它指向的位置（也就是FETCH不会被回滚）。关闭一个游标的行为也不会被回滚给撤消掉。如果一个游标的操作导致事务回滚，那么这个游标就会置于不可执行状态，所以，尽管一个事务可以用ROLLBACK TO SAVEPOINT重新恢复，但是游标不能再使用了。
* 使用ROLLBACK TO SAVEPOINT回滚到一个保存点。使用RELEASE SAVEPOINT删除一个保存点，但是保留该保存点建立后执行的命令的效果。

**语法格式**

ROLLBACK [ WORK | TRANSACTION ] TO [ SAVEPOINT ] savepoint\_name;

**参数说明**

savepoint\_name：回滚截至的保存点。

**示例**

--撤销 my\_savepoint 建立之后执行的命令的影响。  
vastbase=# START TRANSACTION;  
vastbase=# SAVEPOINT my\_savepoint;  
vastbase=# ROLLBACK TO SAVEPOINT my\_savepoint;  
--游标位置不受保存点回滚的影响。  
vastbase=# DECLARE foo CURSOR FOR SELECT 1 UNION SELECT 2;  
vastbase=# SAVEPOINT foo;  
vastbase=# FETCH 1 FROM foo;  
 ?column?   
----------  
 1  
vastbase=# ROLLBACK TO SAVEPOINT foo;  
vastbase=# FETCH 1 FROM foo;  
 ?column?   
----------  
 2  
vastbase=# RELEASE SAVEPOINT my\_savepoint;  
vastbase=# COMMIT;

**相关链接**

参考：SQL语法参考->SQL语法->SAVEPOINT、RELEASE SAVEPOINT章节。

### SAVEPOINT

**功能描述**

SAVEPOINT用于在当前事务里建立一个新的保存点。

保存点是事务中的一个特殊记号，它允许将那些在它建立后执行的命令全部回滚，把事务的状态恢复到保存点所在的时刻。

**注意事项**

* 使用ROLLBACK TO SAVEPOINT回滚到一个保存点。使用RELEASE SAVEPOINT删除一个保存点，但是保留该保存点建立后执行的命令的效果。
* 保存点只能在一个事务块里面建立。在一个事务里面可以定义多个保存点。
* 由于节点故障或者通信故障引起的节点线程或进程退出导致的报错，以及由于COPY FROM操作中源数据与目标表的表结构不一致导致的报错，均不能正常回滚到保存点之前，而是整个事务回滚。
* SQL标准要求，使用savepoint建立一个同名保存点时，需要自动删除前面那个同名保存点。在Vastbase数据库里，我们将保留旧的保存点，但是在回滚或者释放的时候，只使用最近的那个。释放了新的保存点将导致旧的再次成为ROLLBACK TO SAVEPOINT和RELEASE SAVEPOINT可以访问的保存点。除此之外，SAVEPOINT是完全符合SQL标准的。

**语法格式**

SAVEPOINT savepoint\_name;

**参数说明**

savepoint\_name：新建保存点的名称。

**示例**

--创建一个新表。  
vastbase=# CREATE TABLE table1(a int);  
  
--开启事务。  
vastbase=# START TRANSACTION;  
  
--插入数据。  
vastbase=# INSERT INTO table1 VALUES (1);  
  
--建立保存点。  
vastbase=# SAVEPOINT my\_savepoint;  
  
--插入数据。  
vastbase=# INSERT INTO table1 VALUES (2);  
  
--回滚保存点。  
vastbase=# ROLLBACK TO SAVEPOINT my\_savepoint;  
  
--插入数据。  
vastbase=# INSERT INTO table1 VALUES (3);  
  
--提交事务。  
vastbase=# COMMIT;  
  
--查询表的内容，会同时看到1和3,不能看到2，因为2被回滚。  
vastbase=# SELECT \* FROM table1;  
  
--删除表。  
vastbase=# DROP TABLE table1;  
  
--创建一个新表。  
vastbase=# CREATE TABLE table2(a int);  
  
--开启事务。  
vastbase=# START TRANSACTION;  
  
--插入数据。  
vastbase=# INSERT INTO table2 VALUES (3);  
  
--建立保存点。  
vastbase=# SAVEPOINT my\_savepoint;  
  
--插入数据。  
vastbase=# INSERT INTO table2 VALUES (4);  
  
--回滚保存点。  
vastbase=# RELEASE SAVEPOINT my\_savepoint;  
  
--提交事务。  
vastbase=# COMMIT;  
  
--查询表的内容，会同时看到3和4。  
vastbase=# SELECT \* FROM table2;  
  
--删除表。  
vastbase=# DROP TABLE table2;

**相关链接**

参考：SQL语法参考->SQL语法->RELEASE SAVEPOINT、ROLLBACK TO SAVEPOINT章节。

### SECURITY LABEL

**功能描述**

定义或更改应用于对象的安全标签。

**注意事项**

无

**语法格式**

{

TABLE object\_name |

FUNCTION function\_name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ] |

MATERIALIZED VIEW object\_name |

PROCEDURE procedure\_name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ] |

ROLE object\_name |

SEQUENCE object\_name |

TRIGGER object\_name OF table\_name |

VIEW object\_name |

COLUMN table\_name.column.name

} IS 'label' | NULL;

**参数说明**

* object\_name

对象名称。

* function\_name

函数名称。

* argmode

函数参数的模式。

取值范围：IN，OUT，INOUT或VARIADIC（用于声明数组类型的参 数），缺省值是IN。只有OUT模式的参数后面能跟VARIADIC。并且 OUT和INOUT模式的参数不能用在RETURNS TABLE的函数定义 中。

* argname

函数参数的名称。

取值范围：字符串，要符合标识符命令规范。

* argtype

函数参数的类型。

* procedure\_name

存储过程名称。

**示例**

更改一个表的安全标签。

SECURITY LABEL FOR vastbase ON TABLE mytable IS 'system\_u:object\_r:sepgsql\_table\_t:s0';

**相关链接**

无

### SELECT

**功能描述**

SELECT用于从表或视图中查询数据。

SELECT语句就像叠加在数据库表上的过滤器，利用SQL关键字从数据表中过滤出用户需要的数据。

**注意事项**

* 需要对每个在SELECT命令中使用的字段具有SELECT权限。
* 使用FOR UPDATE，FOR NO KEY UPDATE，FOR SHARE或FOR KEY SHARE还要求UPDATE权限。

**语法格式**

* 查询数据

[ WITH [ RECURSIVE ] with\_query [, ...] ]  
SELECT [/\*+ plan\_hint \*/] [ ALL | DISTINCT [ ON ( expression[, ...] ) ] ]  
{ \* | {expression [ [ AS ] output\_name ]} [, ...] }  
[ FROM from\_item [, ...] ]  
[ WHERE condition ]  
[ GROUP BY grouping\_element [, ...] ]  
[ HAVING condition [, ...] ]  
[ WINDOW {window\_name AS ( window\_definition )} [, ...] ]  
[ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]  
[ ORDER BY {expression [ [ ASC | DESC | USING operator ] |nlssort\_expression\_clause ] [ NULLS { FIRST | LAST } ]} [, ...]]  
[ LIMIT { [offset,] count | ALL } ]  
[ OFFSET start [ ROW | ROWS] ]  
[ FETCH { FIRST| NEXT } [ count ] { ROW | ROWS } ONLY ]  
[ {FOR { UPDATE | SHARE } [ OF table\_name [, ...] ] [ NOWAIT]} [...] ];

condition和expression中可以使用targetlist中表达式的别名。

* 只能同一层引用。
* 只能引用targetlist中的别名。
* 只能是后面的表达式引用前面的表达式。
* 不能包含volatile函数。
* 不能包含Window function函数。
* 不支持在join on条件中引用别名。
* targetlist中有多个要应用的别名则报错。
* 其中子查询with\_query为：

with\_query\_name [ ( column\_name [, ...] ) ]  
AS ( {select | values | insert | update | delete} )

* 其中指定查询源from\_item为：

{[ ONLY ] table\_name [ \* ] [ partition\_clause ] [ [ AS ]alias [ ( column\_alias [, ...] ) ] ]  
[ TABLESAMPLE sampling\_method ( argument [, ...] ) [ REPEATABLE (seed ) ] ]|( select ) [ AS ] alias [ ( column\_alias [, ...] ) ]  
|with\_query\_name [ [ AS ] alias [ ( column\_alias [, ...] ) ]]  
|function\_name ( [ argument [, ...] ] ) [ AS ] alias [ (column\_alias [, ...]   
| column\_definition [, ...] ) ]  
|function\_name ( [ argument [, ...] ] ) AS ( column\_definition[, ...] )  
|from\_item [ NATURAL ] join\_type from\_item [ ON join\_condition |USING ( join\_column [, ...] ) ]}

* 其中group子句为：

expression  
| ( expression [, ...] )  
| ROLLUP ( { expression | ( expression [, ...] ) } [, ...] )  
| CUBE ( { expression | ( expression [, ...] ) } [, ...] )  
| GROUPING SETS ( grouping\_element [, ...] )

* 其中指定分区partition\_clause为：

PARTITION { ( partition\_name ) |  
FOR ( partition\_value [, ...] ) }

* 其中设置排序方式nlssort\_expression\_clause为：

NLSSORT ( column\_name, ' NLS\_SORT = { SCHINESE\_PINYIN\_M |generic\_m\_ci } ' )

* 简化版查询语法，功能相当于select \* from table\_name。

TABLE { ONLY {(table\_name)| table\_name} | table\_name [ \* ]};

**参数说明**

* WITH [ RECURSIVE ] with\_query [, ...]
* 用于声明一个或多个可以在主查询中通过名称引用的子查询，相当于临时表。如果声明了RECURSIVE，那么允许SELECT子查询通过名称引用它自己。
* with\_query的详细格式为：

with\_query\_name [ ( column\_name [, ...]) ] AS ( {select | values | insert | update | delete} )

* with\_query\_name指定子查询生成的结果集名称，在查询中可使用该名称访问子查询的结果集。
* column\_name指定子查询结果集中显示的列名。
* 每个子查询可以是SELECT，VALUES，INSERT，UPDATE或DELETE语句。
* plan\_hint子句
* 以/\**+\**/的形式在SELECT关键字后，用于对SELECT对应的语句块生成的计划进行hint调优。
* ALL
* 声明返回所有符合条件的行，是默认行为，可以省略该关键字。
* DISTINCT [ ON ( expression [, …] ) ]
* 从SELECT的结果集中删除所有重复的行，使结果集中的每行都是唯一的。
* ON ( expression [, …] )只保留那些在给出的表达式上运算出相同结果的行集合中的第一行。

DISTINCT ON表达式是使用与ORDER BY相同的规则进行解释的。除非使用了ORDER BY来保证需要的行首先出现，否则，"第一行" 是不可预测的。

* SELECT列表
* 指定查询表中列名，可以是部分列或者是全部（使用通配符\*表示）。 通过使用子句AS output\_name可以为输出字段取个别名，这个别名通常用于输出字段的显示。列名可以用下面几种形式表达：
* 手动输入列名，多个列之间用英文逗号（,）分隔。
* 可以是FROM子句里面计算出来的字段。
* FROM子句
* 为SELECT声明一个或者多个源表。 FROM子句涉及的元素如下所示。
* table\_name
  + 表名或视图名，名称前可加上模式名，如：schema\_name.table\_name。
* alias
  + 给表或复杂的表引用起一个临时的表别名，以便被其余的查询引用。
  + 别名用于缩写或者在自连接中消除歧义。如果提供了别名，它就会完全隐藏表的实际名。
* TABLESAMPLE sampling\_method ( argument [, …] ) [ REPEATABLE (seed ) ]
  + table\_name之后的TABLESAMPLE子句表示应该用指定的sampling\_method来检索表中行的子集。可选的REPEATABLE子句指定一个用于产生采样方法中随机数的种子数。种子值可以是任何非空常量值。如果查询时表没有被更改，指定相同种子和argument值的两个查询将会选择该表相同的采样。但是不同的种子值通常将会产生不同的采样。如果没有给出REPEATABLE，则会基于一个系统产生的种子为每一个查询选择一个新的随机采样。
* column\_alias
  + 列别名。
* PARTITION
  + 查询分区表的某个分区的数据。
* partition\_name
  + 分区名。
* partition\_value
  + 指定的分区键值。在创建分区表时，如果指定了多个分区键，可以通过PARTITION FOR子句指定的这一组分区键的值，唯一确定一个分区。
* subquery
  + FROM子句中可以出现子查询，创建一个临时表保存子查询的输出。
* with\_query\_name
  + WITH子句同样可以作为FROM子句的源，可以通过WITH查询的名称对其进行引用。
* function\_name
  + 函数名称。函数调用也可以出现在FROM子句中。
* ON join\_condition
  + 连接条件，用于限定连接中的哪些行是匹配的。如：

ON left\_table.a = right\_table.a

* USING(join\_column[，…])
* ON left\_table.a = right\_table.a AND left\_table.b = right\_table.b …的简写。要求对应的列必须同名。
* NATURAL
* NATURAL是具有相同名称的两个表的所有列的USING列表的简写。
* from\_item
* 用于连接的查询源对象的名称。
* join\_type
* 有5种类型，如下所示。
* [ INNER ] JOIN：一个JOIN子句组合两个FROM项。可使用圆括弧以决定嵌套的顺序。如果没有圆括弧，JOIN从左向右嵌套。在任何情况下，JOIN都比逗号分隔的FROM项绑定得更紧。
* LEFT [ OUTER ] JOIN：返回笛卡尔积中所有符合连接条件的行，再加上左表中通过连接条件没有匹配到右表行的那些行。这样，左边的行将扩展为生成表的全长，方法是在那些右表对应的字段位置填上NULL。请注意，只在计算匹配的时候，才使用JOIN子句的条件，外层的条件是在计算完毕之后 施加的。
* RIGHT [ OUTER ] JOIN：返回所有内连接的结果行，加上每个不匹配的右边行（左边用NULL 扩展）。这只是一个符号上的方便，因为总是可以把它转换成一个LEFT OUTER JOIN，只要把左边和右边的输入互换位置即可。
* FULL [ OUTER ] JOIN：返回所有内连接的结果行，加上每个不匹配的左边行（右边用NULL扩展），再加上每个不匹配的右边行（左边用NULL扩展）。
* CROSS JOIN：CROSS JOIN等效于INNER JOIN ON（TRUE） ，即没有被条件删除的行。这种连接类型只是符号上的方便，因为它们与简单的FROM和WHERE的效果相同。

必须为INNER和OUTER连接类型声明一个连接条件，即NATURAL ON，join\_condition，USING (join\_column [， …]) 之一。但是它们不能出现在CROSS JOIN中。其中CROSS JOIN和INNER JOIN生成一个简单的笛卡尔积，和在FROM的顶层列出两个项的结果相同。

* WHERE子句
* WHERE子句构成一个行选择表达式，用来缩小SELECT查询的范围。condition是返回值为布尔型的任意表达式，任何不满足该条件的行都不会被检索。
* WHERE子句中可以通过指定“（+）”操作符的方法将表的连接关系转换为外连接。但是不建议用户使用这种用法，因为这并不是SQL的标准语法，在做平台迁移的时候可能面临语法兼容性的问题。同时，使用“（+）”有很多限制：
* “（+）”只能出现在where子句中。
* 如果from子句中已经有指定表连接关系，那么不能再在where子句中使用“（+）”。
* “（+）”只能作用在表或者视图的列上，不能作用在表达式上。
* 如果表A和表B有多个连接条件，那么必须在所有的连接条件中指定“（+）”，否则“（+）”将不会生效，表连接会转化成内连接，并且不给出任何提示信息。
* 在任何外连接的 WHERE 子句中，无论指定哪种形式，都不能将带有"（+）"的列与子查询进行比较。
* 如果“（+）”作用的表，不在当前查询或者子查询的from子句中，则会报错。如果“（+）”作用的对端的表不存在，则不报错，同时连接关系会转化为内连接。
* ''(+)"作用的表达式不能直接通过"OR"连接。
* 如果“（+）”作用的列是和一个常量的比较关系， 那么这个表达式会成为join条件的一部分。
* 同一个表不能对应多个外表。
* “（+）”只能出现"比较表达式"，"NOT表达式"，"ANY表达式"，"ALL表达式"，"IN表达式"，"NULLIF表达式"，"IS DISTINCT FROM表达式"，"IS OF"表达式。“（+）”不能出现在其他类型表达式中，并且这些表达式中不允许出现通过"AND"和"OR"连接的表达式。
* “（+）”只能转化为左外连接或者右外连接，不能转化为全连接，即不能在一个表达式的两个表上同时指定“（+）”。

对于WHERE子句的LIKE操作符，当LIKE中要查询特殊字符“%”、“\_”、“\”的时候需要使用反斜杠“\”来进行转义。

* GROUP BY子句
* 将查询结果按某一列或多列的值分组，值相等的为一组。
* CUBE ( { expression | ( expression [, …] ) } [, …] )：CUBE是自动对groupby子句中列出的字段进行分组汇总，结果集将包含维度列中各值的所有可能组合，以及与这些维度值组合相匹配的基础行中的聚合值。它会为每个分组返回一行汇总信息，用户可以使用CUBE来产生交叉表值。比如，在CUBE子句中给出三个表达式（n =3），运算结果为2^n= 2^3 = 8组。以n个表达式的值分组的行称为常规行，其余的行称为超级聚集行。
* GROUPING SETS ( grouping\_element [, …] )：GROUPING SETS子句是GROUP BY子句的进一步扩展，它可以使用户指定多个GROUP BY选项。这样做可以通过裁剪用户不需要的数据组来提高效率。当用户指定了所需的数据组时，数据库不需要执行完整CUBE或ROLLUP生成的聚合集合。

SELECT子句中指定字段必须要包含在group by语句后，作为分组的依据，如果没有包含则会报错。除非SELECT子句中的字段包含在聚集函数中，因为对于未分组的字段，可能会返回多个数值。

* HAVING子句
* 与GROUP BY子句配合用来选择特殊的组。HAVING子句将组的一些属性与一个常数值比较，只有满足HAVING子句中的逻辑表达式的组才会被提取出来。
* WINDOW子句
* 一般形式为：

WINDOW window\_name AS ( window\_definition ) [，...]

window\_name是可以被随后的窗口定义所引用的名称，window\_definition可以是以下的形式：

[ existing\_window\_name ]  
[ PARTITION BY expression [, ...] ]  
[ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS {FIRST | LAST } ] [, ...] ]   
[ frame\_clause]

frame\_clause为窗函数定义一个窗口框架window frame，窗函数（并非所有）依赖于框架，window frame是当前查询行的一组相关行。frame\_clause可以是以下的形式：

[ RANGE | ROWS ] frame\_start  
[ RANGE | ROWS ] BETWEEN frame\_start AND frame\_end

frame\_start和frame\_end可以是：

UNBOUNDED PRECEDING  
value PRECEDING  
CURRENT ROW   
value FOLLOWING  
UNBOUNDED FOLLOWING

对列存表的查询目前只支持row\_number窗口函数，不支持frame\_clause。

* UNION子句
* UNION计算多个SELECT语句返回行集合的并集。
* UNION子句有如下约束条件：
* 除非声明了ALL子句，否则缺省的UNION结果不包含重复的行。
* 同一个SELECT语句中的多个UNION操作符是从左向右计算的，除非用圆括弧进行了标识
* FOR UPDATE不能在UNION的结果或输入中声明。
* 一般表达式：

select\_statement UNION [ALL] select\_statement

* select\_statement可以是任何没有ORDER BY、LIMIT、FOR UPDATE子句的SELECT语句。
* 如果用圆括弧包围，ORDER BY和LIMIT可以附着在子表达式里。
* INTERSECT子句
* INTERSECT计算多个SELECT语句返回行集合的交集，不含重复的记录。
* INTERSECT子句有如下约束条件：
* 同一个SELECT语句中的多个INTERSECT操作符是从左向右计算的，除非用圆括弧进行了标识。
* 当对多个SELECT语句的执行结果进行UNION和INTERSECT操作的时候，会优先处理INTERSECT。
* 一般形式：

select\_statement INTERSECT select\_statement

select\_statement可以是任何没有FOR UPDATE子句的SELECT语句。

* EXCEPT子句
* EXCEPT子句有如下的通用形式：

select\_statement EXCEPT [ ALL ] select\_statement

select\_statement是任何没有FOR UPDATE子句的SELECT表达式。

* EXCEPT操作符计算存在于左边SELECT语句的输出而不存在于右边SELECT语句输出的行。
* EXCEPT的结果不包含任何重复的行，除非声明了ALL选项。使用ALL时，一个在左边表中有m个重复而在右边表中有n个重复的行将在结果中出现max(m-n,0)次。
* 除非用圆括弧指明顺序，否则同一个SELECT语句中的多个EXCEPT操作符是从左向右计算的。EXCEPT和UNION的绑定级别相同。
* 不能给EXCEPT的结果或者任何EXCEPT的输入声明FOR UPDATE子句。
* MINUS子句
* 与EXCEPT子句具有相同的功能和用法。
* ORDER BY子句
* 对SELECT语句检索得到的数据进行升序或降序排序。对于ORDER BY表达式中包含多列的情况：
* 首先根据最左边的列进行排序，如果这一列的值相同，则根据下一个表达式进行比较，依此类推。
* 如果对于所有声明的表达式都相同，则按随机顺序返回。
* ORDER BY中排序的列必须包括在SELECT语句所检索的结果集的列中。

**须知：**如果要支持中文拼音排序和不区分大小写排序，需要在初始化数据库时指定编码格式为UTF-8或GBK。 命令如下:

initdb --E UTF8 --D ../data --locale=zh\_CN.UTF-8或initdb --E GBK --D ../data -- locale=zh\_CN.GBK

注意：若要使用pg\_zhtrgm插件，initdb时指定的 -E和--locale参数对应的编码规则必须一致。

* LIMIT子句

LIMIT { count | ALL }

当limit后面跟一个参数的时候，该参数表示要取的行数。 当limit后面跟两个参数的时候，第一个数表示要跳过的行数，后一位表示要取的行数。

* OFFSET子句

OFFSET start { ROW | ROWS }

start声明开始返回行之前忽略的行数。当limit和offset组合使用的时候，limit后面只能有一个参数，表示要取的行数,offset表示要跳过的行数。

* FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY
* 如果不指定count，默认值为1，FETCH子句限定返回查询结果从第一行开始的总行数。
* FOR UPDATE子句
* FOR UPDATE子句将对SELECT检索出来的行进行加锁。这样避免它们在当前事务结束前被其他事务修改或者删除，即其他企图UPDATE、DELETE、 SELECT FOR UPDATE这些行的事务将被阻塞，直到当前事务结束。
* 为了避免操作等待其他事务提交，可使用NOWAIT选项，如果被选择的行不能立即被锁住，执行SELECT FOR UPDATE NOWAIT将会立即汇报一个错误，而不是等待。
* FOR NO KEY UPDATE行为与FOR UPDATE类似，不过获得的锁较弱：这种锁将不会阻塞尝试在相同行上获得锁的SELECT FOR KEY SHARE命令。任何不获取FOR UPDATE锁的UPDATE也会获得这种锁模式。FOR SHARE的行为类似，只是它在每个检索出来的行上要求一个共享锁，而不是一个排他锁。一个共享锁阻塞其它事务执行UPDATE、DELETE、SELECT，不阻塞SELECT FOR SHARE。
* 如果在FOR UPDATE或FOR SHARE中明确指定了表名称，则只有这些指定的表被锁定，其他在SELECT中使用的表将不会被锁定。否则，将锁定该命令中所有使用的表。
* 如果FOR UPDATE或FOR SHARE应用于一个视图或者子查询，它同样将锁定所有该视图或子查询中使用到的表。
* 多个FOR UPDATE和FOR SHARE子句可以用于为不同的表指定不同的锁定模式。
* 如果一个表中同时出现（或隐含同时出现）在FOR UPDATE和FOR SHARE子句中，则按照FOR UPDATE处理。类似的，如果影响一个表的任意子句中出现了NOWAIT，该表将按照NOWAIT处理。

对列存表的查询不支持for update/share。

* NLS\_SORT
* 指定某字段按照特殊方式排序。目前仅支持中文拼音格式排序和不区分大小写排序。
* 取值范围：
* SCHINESE\_PINYIN\_M，按照中文拼音排序。如果要支持此排序方式，在创建数据库时需要指定编码格式为"GBK"，否则排序无效。
* generic\_m\_ci，不区分大小写排序。
* PARTITION子句
* 查询某个分区表中相应分区的数据。

**示例**

通过子查询得到一张临时表temp\_t，然后查询表temp\_t中的所有数据。

WITH temp\_t(name,isdba) AS (SELECT usename,usesuper FROM pg\_user) SELECT \* FROM temp\_t;

创建表test并插入测试数据。

CREATE TABLE test(id int,name varchar);  
INSERT INTO test(id,name) VALUES('1','zhangsan');  
INSERT INTO test(id,name) VALUES('2','lisi');  
INSERT INTO test(id,name) VALUES('3','zhangsan');  
INSERT INTO test(id,name) VALUES('4','wangwu');  
INSERT INTO test(id,name) VALUES('5','xiaoming');  
INSERT INTO test(id,name) VALUES('6','xiaohong');  
INSERT INTO test(id,name) VALUES('7','LISI');

查询test表的所有name记录，且去除重复。

SELECT DISTINCT(name) FROM test;

LIMIT子句示例：获取表中一条记录。

SELECT \* FROM test LIMIT 1;

查询所有记录，且按字母升序排列。

SELECT name FROM test ORDER BY name;

GROUP BY子句示例：根据查询条件过滤，并对结果进行分组。

SELECT name, AVG(id) FROM test GROUP BY name HAVING AVG(id) > 2;

GROUP BY CUBE子句示例：根据查询条件过滤，并对结果进行分组汇总。

SELECT name,AVG(id) FROM test GROUP BY CUBE(id,name);

GROUP BY GROUPING SETS子句示例:根据查询条件过滤，并对结果进行分组汇总。

SELECT name,AVG(id) FROM test GROUP BY GROUPING SETS((id,name),id);

UNION子句示例：将表test里name字段中的内容以'l'开头和以'z'开头的进行合并。

SELECT id, name FROM test WHERE test.name LIKE 'l%' UNION SELECT id, name FROM test WHERE test.name LIKE 'z%';

NLS\_SORT子句示例：中文拼音排序。

SELECT \* FROM test ORDER BY NLSSORT(name, 'NLS\_SORT = SCHINESE\_PINYIN\_M');

不区分大小写排序:

SELECT \* FROM test ORDER BY NLSSORT(name, 'NLS\_SORT = generic\_m\_ci');

通过表别名，从pg\_user和pg\_user\_status这两张表中获取数据。

SELECT a.usename,b.locktime FROM pg\_user a,pg\_user\_status b WHERE a.usesysid=b.roloid;

FULL JOIN子句示例：将pg\_user和pg\_user\_status这两张表的数据进行全连接显示，即数据的合集。

SELECT a.usename,b.locktime,a.usesuper FROM pg\_user a FULL JOIN pg\_user\_status b on a.usesysid=b.roloid;

创建分区表test\_p

CREATE TABLE test\_p  
(  
r\_reason\_sk integer,  
r\_reason\_id character(16),  
r\_reason\_desc character(100)  
)  
PARTITION BY RANGE (r\_reason\_sk)  
(  
partition P\_05\_BEFORE values less than (05),  
partition P\_15 values less than (15),  
partition P\_25 values less than (25),  
partition P\_35 values less than (35),  
partition P\_45\_AFTER values less than (MAXVALUE)  
)  
;

插入数据。

INSERT INTO test\_p  
values(3,'AAAAAAAABAAAAAAA','reason  
1'),(10,'AAAAAAAABAAAAAAA','reason  
2'),(4,'AAAAAAAABAAAAAAA','reason  
3'),(10,'AAAAAAAABAAAAAAA','reason  
4'),(10,'AAAAAAAABAAAAAAA','reason  
5'),(20,'AAAAAAAACAAAAAAA','reason  
6'),(30,'AAAAAAAACAAAAAAA','reason 7');

PARTITION子句示例：从test\_p的表分区P\_05\_BEFORE中获取数据。

SELECT \* FROM test\_p PARTITION (P\_05\_BEFORE);

GROUP BY子句示例：按r\_reason\_id分组统计test\_p表中的记录数。

SELECT COUNT(\*),r\_reason\_id FROM test\_p GROUP BY r\_reason\_id;

HAVING子句示例：按r\_reason\_id分组统计tpcds.reason\_p表中的记录，并只显示r\_reason\_id个数大于2的信息。

SELECT COUNT(\*) c,r\_reason\_id FROM test\_p GROUP BY r\_reason\_id HAVING c>2;

IN子句示例：按r\_reason\_id分组统计tpcds.reason\_p表中的r\_reason\_id个数，并只显示r\_reason\_id值为AAAAAAAABAAAAAAA或AAAAAAAADAAAAAAA的个数。

SELECT COUNT(\*),r\_reason\_id FROM test\_p GROUP BY  
r\_reason\_id HAVING r\_reason\_id IN('AAAAAAAABAAAAAAA','AAAAAAAADAAAAAAA');

INTERSECT子句示例：查询r\_reason\_id等于AAAAAAAABAAAAAAA，并且r\_reason\_sk小于5的信息。

SELECT \* FROM test\_p WHERE r\_reason\_id='AAAAAAAABAAAAAAA' INTERSECT SELECT \* FROM test\_p WHERE r\_reason\_sk<5;

EXCEPT子句示例：查询r\_reason\_id等于AAAAAAAABAAAAAAA，并且去除r\_reason\_sk小于4的信息。

SELECT \* FROM test\_p WHERE r\_reason\_id='AAAAAAAABAAAAAAA' EXCEPT SELECT \* FROM test\_p WHERE r\_reason\_sk<4;

通过在where子句中指定“（+）”来实现左连接。

SELECT test\_p.r\_reason\_sk ,test.id FROM test\_p,test WHERE test\_p.r\_reason\_sk=test.id(+);

通过在where子句中指定“（+）”来实现右连接。

SELECT test\_p.r\_reason\_sk ,test.id FROM test\_p,test WHERE test\_p.r\_reason\_sk(+)=test.id;

通过在where子句中指定“（+）”来实现左连接，并且增加连接条件。

SELECT test\_p.r\_reason\_sk ,test.id FROM test\_p,test WHERE test\_p.r\_reason\_sk=test.id(+) AND test.id(+) < 1 ORDER BY 1 LIMIT 1 ;

* 不支持在where子句中指定“（+）”的同时使用内层嵌套AND/OR的表达式。
* where子句在不支持表达式宏指定“（+）”会报错。
* where子句在表达式的两边都指定“（+）”会报错。

删除表。

DROP TABLE test\_p,test;

### SELECT INTO

**功能描述**

SELECT INTO用于根据查询结果创建一个新表，并且将查询到的数据插入到新表中。

数据并不返回给客户端，这一点和普通的SELECT不同。新表的字段具有和SELECT的输出字段相同的名称和数据类型。

**注意事项**

CREATE TABLE AS的作用和SELECT INTO类似，且提供了SELECT INTO所提供功能的超集。建议使用CREATE TABLE AS语法替代SELECT INTO，因为SELECT INTO不能在存储过程中使用。

**语法格式**

[ WITH [ RECURSIVE ] with\_query [, ...] ]  
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]  
 { \* | {expression [ [ AS ] output\_name ]} [, ...] }  
 INTO [ [ GLOBAL | LOCAL ] [ TEMPORARY | TEMP ] | UNLOGGED ] [ TABLE ] new\_table  
 [ FROM from\_item [, ...] ]  
 [ WHERE condition ]  
 [ GROUP BY expression [, ...] ]  
 [ HAVING condition [, ...] ]  
 [ WINDOW {window\_name AS ( window\_definition )} [, ...] ]  
 [ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]  
 [ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort\_expression\_clause ] [ NULLS { FIRST | LAST } ]} [, ...] ]  
 [ LIMIT { count | ALL } ]  
 [ OFFSET start [ ROW | ROWS ] ]  
 [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]  
 [ {FOR { UPDATE | SHARE } [ OF table\_name [, ...] ] [ NOWAIT | WAIT N]} [...] ];

**参数说明**

* **new\_table**
* new\_table指定新建表的名称。
* **UNLOGGED**
* 指定表为非日志表。在非日志表中写入的数据不会被写入到预写日志中，这样就会比普通表快很多。但是，它也是不安全的，非日志表在冲突或异常关机后会被自动删截。非日志表中的内容也不会被复制到备用服务器中。在该类表中创建的索引也不会被自动记录。
* 使用场景：非日志表不能保证数据的安全性，用户应该在确保数据已经做好备份的前提下使用，例如系统升级时进行数据的备份。
* 故障处理：当异常关机等操作导致非日志表上的索引发生数据丢失时，用户应该对发生错误的索引进行重建。
* **GLOBAL | LOCAL**
* 创建临时表时可以在TEMP或TEMPORARY前指定GLOBAL或LOCAL关键字。如果指定GLOBAL关键字，Vastbase会创建全局临时表，否则Vastbase会创建本地临时表。
* **TEMPORARY | TEMP**
* 如果指定TEMP或TEMPORARY关键字，则创建的表为临时表。临时表分为全局临时表和本地临时表两种类型。创建临时表时如果指定GLOBAL关键字则为全局临时表，否则为本地临时表。
* 全局临时表的元数据对所有会话可见，会话结束后元数据继续存在。会话与会话之间的用户数据、索引和统计信息相互隔离，每个会话只能看到和更改自己提交的数据。全局临时表有两种模式：一种是基于会话级别的（ON COMMIT PRESERVE ROWS），当会话结束时自动清空用户数据；一种是基于事务级别的（ON COMMIT PRESERVE ROWS），当执行commit或rollback时自动清空用户数据。建表时如果没有指定ON COMMIT选项，则缺省为会话级别。与本地临时表不同，全局临时表建表时可以指定非pg\_temp\_开头的schema。
* 由于临时表只在当前会话创建，对于涉及对临时表操作的DDL语句，会产生DDL失败的报错。因此，建议DDL语句中不要对临时表进行操作。TEMP和TEMPORARY等价。
* fig: **须知：**
* 本地临时表通过每个会话独立的以pg\_temp开头的schema来保证只对当前会话可见，因此，不建议用户在日常操作中手动删除以pg\_tem、pg\_toast\_temp开头的schema。
* 如果建表时不指定TEMPORARY/TEMP关键字，而指定表的schema为当前会话的pg\_temp\_开头的schema，则此表会被创建为临时表。
* ALTER/DROP全局临时表和索引，如果其它会话正在使用它，禁止操作。
* 全局临时表的DDL只会影响当前会话的用户数据和索引。例如truncate、reindex、analyze只对当前会话有效。

fig: **说明：**   
SELECT INTO的其它参数可以参考：SQL语法参考->SQL语法->SELECT章节中的参数说明。

**示例**

--将reason表中r\_reason\_sk小于5的值加入到新建表中。  
vastbase=# SELECT \* INTO reason\_t1 FROM reason WHERE r\_reason\_sk < 5;  
INSERT 0 6  
  
--删除reason\_t1表。  
vastbase=# DROP TABLE reason\_t1;

**相关链接**

参考：SQL语法参考->SQL语法->SELECT章节。

### SET

**功能描述**

用于修改运行时配置参数。

**注意事项**

大多数运行时参数都可以用SET在运行时设置，但有些则在服务运行过程中或会话开始之后不能修改。

**语法格式**

* 设置所处的时区。

SET [ SESSION | LOCAL ] TIME ZONE { timezone | LOCAL | DEFAULT };

* 设置所属的模式。

SET [ SESSION | LOCAL ]   
 {CURRENT\_SCHEMA { TO | = } { schema | DEFAULT }  
 | SCHEMA 'schema'};

* 设置客户端编码集。

SET [ SESSION | LOCAL ] NAMES encoding\_name;

* 设置XML的解析方式。

SET [ SESSION | LOCAL ] XML OPTION { DOCUMENT | CONTENT };

* 设置其他运行时参数。

SET [ LOCAL | SESSION ]  
 { {config\_parameter { { TO | = } { value | DEFAULT }   
 | FROM CURRENT }}};

**参数说明**

* **SESSION**
* 声明的参数只对当前会话起作用。如果SESSION和LOCAL都没出现，则SESSION为缺省值。
* 如果在事务中执行了此命令，命令的产生影响将在事务回滚之后消失。如果该事务已提交，影响将持续到会话的结束，除非被另外一个SET命令重置参数。
* **LOCAL**
* 声明的参数只在当前事务中有效。在COMMIT或ROLLBACK之后，会话级别的设置将再次生效。
* 不论事务是否提交，此命令的影响只持续到当前事务结束。一个特例是：在一个事务里面，即有SET命令，又有SET LOCAL命令，且SET LOCAL在SET后面，则在事务结束之前，SET LOCAL命令会起作用，但事务提交之后，则是SET命令会生效。
* **TIME ZONE timezone**
* 用于指定当前会话的本地时区。
* 取值范围：有效的本地时区。该选项对应的运行时参数名称为TimeZone，DEFAULT缺省值为PRC。
* **CURRENT\_SCHEMA**
* **schema**
* CURRENT\_SCHEMA用于指定当前的模式。
* 取值范围：已存在模式名称。如果模式名不存在，会导致CURRENT\_SCHEMA值为空。
* **SCHEMA schema**
* 同CURRENT\_SCHEMA。此处的schema是个字符串。
* 例如：set schema 'public'。
* **NAMES encoding\_name**
* 用于设置客户端的字符编码。等价于set client\_encoding to encoding\_name。
* 取值范围：有效的字符编码。该选项对应的运行时参数名称为client\_encoding，默认编码为UTF8。
* **XML OPTION option**
* 用于设置XML的解析方式。
* 取值范围：CONTENT（缺省）、DOCUMENT。
* **config\_parameter**
* 可设置的运行时参数的名称。可用的运行时参数可以使用SHOW ALL命令查看。
* fig: **说明：**   
  部分通过SHOW ALL查看的参数不能通过SET设置。如max\_datanodes。
* **value**
* config\_parameter的新值。可以声明为字符串常量、标识符、数字，或者逗号分隔的列表。DEFAULT用于把这些参数设置为它们的缺省值。

**示例**

--设置模式搜索路径。   
vastbase=# SET search\_path TO tpcds, public;  
  
--把日期时间风格设置为传统的 POSTGRES 风格(日在月前)。  
vastbase=# SET datestyle TO postgres,dmy;

**相关链接**

参考：SQL语法参考->SQL语法->RESET、SHOW章节。

### SET CONSTRAINTS

**功能描述**

SET CONSTRAINTS设置当前事务检查行为的约束条件。

IMMEDIATE约束是在每条语句后面进行检查。DEFERRED约束一直到事务提交时才检查。每个约束都有自己的模式。

从创建约束条件开始，一个约束总是设定为DEFERRABLE INITIALLY DEFERRED、DEFERRABLE INITIALLY IMMEDIATE、NOT DEFERRABLE三个特性之一。第三种总是IMMEDIATE，并且不会受SET CONSTRAINTS影响。前两种以指定的方式启动每个事务，但是其行为可以在事务里用SET CONSTRAINTS改变。

带着一个约束名列表的SET CONSTRAINTS改变这些约束的模式（都必须是可推迟的）。如果有多个约束匹配某个名称，则所有都会被影响。SET CONSTRAINTS ALL改变所有可推迟约束的模式。

当SET CONSTRAINTS把一个约束从DEFERRED改成IMMEDIATE的时候，新模式反作用式地起作用：任何将在事务结束准备进行的数据修改都将在SET CONSTRAINTS的时候执行检查。如果违反了任何约束，SET CONSTRAINTS都会失败（并且不会修改约束模式）。因此，SET CONSTRAINTS可以用于强制在事务中某一点进行约束检查。

检查约束总是不可推迟的。

**注意事项**

SET CONSTRAINTS只在当前事务里设置约束的行为。因此，如果用户在事务块（START TRANSACTION/COMMIT对）之外执行这个命令，它将没有任何作用。

**语法格式**

SET CONSTRAINTS { ALL | { name } [, ...] } { DEFERRED | IMMEDIATE } ;

**参数说明**

* **name**
* 约束名。
* 取值范围：已存在的约束名。可以在系统表pg\_constraint中查到。
* **ALL**
* 所有约束。
* **DEFERRED**
* 约束一直到事务提交时才检查。
* **IMMEDIATE**
* 约束在每条语句后进行检查。

**示例**

--设置所有约束在事务提交时检查。  
vastbase=# SET CONSTRAINTS ALL DEFERRED;

### SET ROLE

**功能描述**

设置当前会话的当前用户标识符。

**注意事项**

* 当前会话的用户必须是指定的rolename角色的成员，但系统管理员可以选择任何角色。
* 使用这条命令，它可能会增加一个用户的权限，也可能会限制一个用户的权限。如果会话用户的角色有INHERITS属性，则它自动拥有它能SET ROLE变成的角色的所有权限；在这种情况下，SET ROLE实际上是删除了所有直接赋予会话用户的权限，以及它的所属角色的权限，只剩下指定角色的权限。另一方面，如果会话用户的角色有NOINHERITS属性，SET ROLE删除直接赋予会话用户的权限，而获取指定角色的权限。

**语法格式**

* 设置当前会话的当前用户标识符。

SET [ SESSION | LOCAL ] ROLE role\_name PASSWORD 'password';

* 重置当前用户标识为当前会话用户标识符。

RESET ROLE;

**参数说明**

* **SESSION**
* 声明这个命令只对当前会话起作用，此参数为缺省值。
* **LOCAL**
* 声明该命令只在当前事务中有效。
* **role\_name**
* 角色名。
* 取值范围：字符串，要符合标识符的命名规范。
* **password**
* 角色的密码。要求符合密码的命名规则。
* **RESET ROLE**
* 用于重置当前用户标识。

**示例**

--创建角色paul。  
vastbase=# CREATE ROLE paul IDENTIFIED BY 'xxxxxxxxx';  
  
--设置当前用户为paul。  
vastbase=# SET ROLE paul PASSWORD 'xxxxxxxxx';  
  
--查看当前会话用户，当前用户。  
vastbase=# SELECT SESSION\_USER, CURRENT\_USER;  
  
--重置当前用户。  
vastbase=# RESET role;  
  
--删除用户。  
vastbase=# DROP USER paul;

### SET SESSION AUTHORIZATION

**功能描述**

把当前会话里的会话用户标识和当前用户标识都设置为指定的用户。

**注意事项**

只有在初始会话用户有系统管理员权限的时候，会话用户标识符才能改变。否则，只有在指定了被认证的用户名的情况下，系统才接受该命令。

**语法格式**

* 为当前会话设置会话用户标识符和当前用户标识符。

SET [ SESSION | LOCAL ] SESSION AUTHORIZATION role\_name PASSWORD 'password';

* 重置会话和当前用户标识符为初始认证的用户名。

{SET [ SESSION | LOCAL ] SESSION AUTHORIZATION DEFAULT  
| RESET SESSION AUTHORIZATION};

**参数说明**

* **SESSION**
* 声明这个命令只对当前会话起作用。
* **LOCAL**
* 声明该命令只在当前事务中有效。
* **role\_name**
* 用户名。
* 取值范围：字符串，要符合标识符的命名规范。
* **password**
* 角色的密码。要求符合密码的命名规则。
* **DEFAULT**
* 重置会话和当前用户标识符为初始认证的用户名。

**示例**

--创建角色paul。  
vastbase=# CREATE ROLE paul IDENTIFIED BY 'xxxxxxxxx';  
  
--设置当前用户为paul。  
vastbase=# SET SESSION AUTHORIZATION paul password 'xxxxxxxxx';  
  
--查看当前会话用户，当前用户。  
vastbase=# SELECT SESSION\_USER, CURRENT\_USER;  
  
--重置当前用户。  
vastbase=# RESET SESSION AUTHORIZATION;  
  
--删除用户。  
vastbase=# DROP USER paul;

**相关参考**

参考：SQL语法参考->SQL语法->SET ROLE章节。

### SET TRANSACTION

**功能描述**

为事务设置特性。事务特性包括事务隔离级别、事务访问模式（读/写或者只读）。可以设置当前事务的特性（LOCAL)，也可以设置会话的默认事务特性（SESSION）。

**语法格式**

设置事务的隔离级别、读写模式。

{ SET [ LOCAL ] TRANSACTION|SET SESSION CHARACTERISTICS AS TRANSACTION }  
 { ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }  
 { READ WRITE | READ ONLY } } [, ...]

**参数说明**

* LOCAL：声明该命令只在当前事务中有效。
* SESSION：声明这个命令只对当前会话起作用。
* 取值范围：字符串，要符合标识符的命名规范。
* ISOLATION\_LEVEL：指定事务隔离级别，该参数决定当一个事务中存在其他并发运行事务时能够看到什么数据。
* 在事务中第一个数据修改语句（SELECT、INSERT、DELETE、UPDATE、FETCH、COPY)执行之后，当前事务的隔离级别就不能再次设置。
* 取值范围：
* READ COMMITTED：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
* READ UNCOMMITTED：读未提交隔离级别，也就是说事务所作的修改在未提交前，其他并发事务是可以读到的。
* REPEATABLE READ：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
* SERIALIZABLE：Vastbase目前功能上不支持此隔离级别，等价于REPEATABLE READ。
* READ WRITE | READ ONLY：指定事务访问模式（读/写或者只读）。

**注意事项**

设置当前事务特性需要在事务中执行（即执行SET TRANSACTION之前需要执行START TRANSACTION或者BEGIN），否则设置不生效。

**示例**

开启一个事务，设置事务的隔离级别为READ COMMITTED，访问模式为READ ONLY。

START TRANSACTION;   
SET LOCAL TRANSACTION ISOLATION LEVEL READ COMMITTED READ ONLY;   
COMMIT;

### SHOW

**功能描述**

SHOW将显示当前运行时参数的数值。

**注意事项**

无。

**语法格式**

SHOW   
{   
 configuration\_parameter |  
 VARIABLES LIKE var\_name|   
 CURRENT\_SCHEMA |   
 TIME ZONE |   
 TRANSACTION ISOLATION LEVEL |   
 SESSION AUTHORIZATION |   
 ALL   
};

**参数说明**

显示变量的参数请参见：SQL语法参考->SQL语法->RESET章节中的参数说明。

**示例**

--显示 timezone 参数值。  
SHOW timezone;  
  
--显示所有参数。  
SHOW ALL;  
  
--显示参数名中包含”var”的所有参数  
SHOW VARIABLES LIKE var;

**相关链接**

参考：SQL语法参考->SQL语法->SET、RESET章节。

### SHUTDOWN

**功能描述**

SHUTDOWN将关闭当前连接的数据库节点。

**注意事项**

仅拥有管理员权限的用户可以运行此命令。

**语法格式**

SHUTDOWN [ SMART | FAST | IMMEDIATE ]

**参数说明**

* **smart**
* 不接受新的连接，且等待已有连接全部断开后，关闭数据库节点。
* **fast**
* 不等待客户端中断连接，将所有活跃事务回滚并且强制断开客户端，然后关闭数据库节点。
* **immediate**
* 强行关闭，在下次重新启动的时候将导致故障恢复。

**示例**

--关闭当前数据库节点。  
vastbase=# SHUTDOWN;  
  
--使用fast模式关闭当前数据库节点。  
vastbase=# SHUTDOWN FAST;

### SNAPSHOT

**功能描述**

针对多用户情况下，对数据进行统一的版本控制。

**注意事项**

* 当快照选用增量存储方式时，各个快照中具有依赖关系。删除快照需要按照依赖顺序进行删除。
* snapshot特性用于团队不同成员间维护数据，涉及管理员和普通用户之间的数据转写。所以在私有用户、三权分立(enableSeparationOfDuty=ON)等状态下，数据库不支持snapshot功能特性。
* 当需要稳定可用的快照用于AI训练等任务时，用户需要将快照发布。
* 本特性相关GUC 参数有（以下参数详见：开发者指南->GUC参数说明->AI特性章节）：
  + db4ai\_snapshot\_mode：快照存储模型分为MSS和CSS两种
  + db4ai\_snapshot\_version\_delimiter：用于设定版本分隔符，默认为“@”
  + db4ai\_snapshot\_version\_separator：用于设定子版本分隔符，默认为“.”。

**语法格式**

1、创建快照。

可以采用“CREATE SNAPSHOT … AS”以及“CREATE SNAPSHOT … FROM”语句创建数据表快照。

* CREATE SNAPSHOT AS

CREATE SNAPSHOT <qualified\_name> [@ <version | ident | sconst>]  
 [COMMENT IS <sconst>}  
 AS query;

* CREATE SNAPSHOT FROM

CREATE SNAPSHOT <qualified\_name> [@ <version | ident | sconst>]  
 FROM @ <version | ident | sconst>  
 [COMMENT IS <sconst>}  
 USING (  
 { INSERT [INTO SNAPSHOT] …  
 | UPDATE [SNAPSHOT] [AS <alias>] SET … [FROM …] [WHERE …]  
 | DELETE [FROM SNAPSHOT] [AS <alias>] [USING …] [WHERE …]  
 | ALTER [SNAPSHOT] { ADD … | DROP … } [, …]  
 } [; …]  
 );

2、删除快照。

PURGE SNAPSHOT

PURGE SNAPSHOT <qualified\_name> @ <version | ident | sconst>;

3、快照采样。

SAMPLE SNAPSHOT

SAMPLE SNAPSHOT <qualified\_name> @ <version | ident | sconst>  
 [STRATIFY BY attr\_list]  
 { AS <label> AT RATIO <num> [COMMENT IS <comment>] } [, …]

4、快照发布。

PUBLISH SNAPSHOT

PUBLISH SNAPSHOT <qualified\_name> @ <version | ident | sconst>;

5、快照存档。

ARCHIVE SNAPSHOT

ARCHIVE SNAPSHOT <qualified\_name> @ <version | ident | sconst>;

**参数说明**

* qualified\_name
* 创建snapshot的名称。
* 取值范围：字符串，需要符合标识符命名规则。
* version
* (可省略)snapshot的版本号，当省略设置。系统会自动顺延编号。
* 取值范围：字符串，数字编号配合分隔符。

**示例**

1、创建测试表。

create table t1(id int,name text);

2、创建新快照s1。

create snapshot s1@1.0 comment is 'first version' as select \* from t1;

3、从现有快照创建快照修订。

create snapshot s1@3.0 from @1.0 comment is 'inherits from @1.0' using (INSERT VALUES(6,6), (7,7); DELETE WHERE id = 1);

返回结果为：

schema | name

--------+--------

public | s1@3.0

(1 row)

4、删除一个快照。

purge snapshot s1@1.0;

5、快照采样。

sample snapshot s1@3.0 stratify by id as nick at ratio .5;

返回结果为：

schema | name

--------+------------

public | s1nick@3.0

(1 row)

6、发布一个快照。

publish snapshot s1@3.0;

返回结果为：

schema | name

--------+--------

public | s1@3.0

(1 row)

7、存档快照。

archive snapshot s1@3.0;

**相关链接**

无。

### START TRANSACTION

**功能描述**

通过START TRANSACTION启动事务。如果声明了隔离级别、读写模式，那么新事务就使用这些特性，类似执行了SET TRANSACTION。

**语法格式**

格式一：START TRANSACTION格式

START TRANSACTION  
 [ { ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }  
 | { READ WRITE | READ ONLY }  
 } [, ...] ];

格式二：BEGIN格式

BEGIN [ WORK | TRANSACTION ]  
 [ { ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | SERIALIZABLE | REPEATABLE READ }  
 | { READ WRITE | READ ONLY }  
 } [, ...] ];

**参数说明**

* WORK | TRANSACTIONBEGIN：格式中的可选关键字，没有实际作用。
* ISOLATION LEVEL：指定事务隔离级别，它决定当一个事务中存在其他并发运行事务时它能够看到什么数据。
* 在事务中第一个数据修改语句（SELECT、 INSERT、DELETE、UPDATE、FETCH、COPY）执行之后，事务隔离级别就不能再次设置。
* 取值范围：
* READ COMMITTED：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
* READ UNCOMMITTED：读未提交隔离级别，也就是说事务所作的修改在未提交前，其他并发事务是可以读到的。
* REPEATABLE READ：可重复读隔离级别，仅仅看到事务开始之前提交的数据，它不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
* SERIALIZABLE：Vastbase目前功能上不支持此隔离级别，等价于REPEATABLE READ。
* READ WRITE | READ ONLY：指定事务访问模式（读/写或者只读）。

**注意事项**

无。

**示例**

1、创建测试表test。

CREATE TABLE test(id int);

2、以不同方式启动事务

* 以默认方式（START TRANSACTION）启动事务，并在操作后提交。

START TRANSACTION;   
SELECT \* FROM test;  
END;

* 以默认方式（BEGIN）启动事务，并在操作后提交。

BEGIN;   
SELECT \* FROM test;   
END;

* 以隔离级别为READ COMMITTED，读/写方式启动事务，并在操作后提交。

START TRANSACTION ISOLATION LEVEL READ COMMITTED READ WRITE;   
SELECT \* FROM test;   
COMMIT;

3、删除测试表

DROP TABLE test;

### TIMECAPSULE TABLE

**功能描述**

在人为操作或应用程序错误时，使用TIMECAPSULE TABLE语句恢复可将表恢复到一个早期状态。

表可以闪回到过去的时间点，这依赖于系统中保存的旧版本数据。此外Vastbase数据库不能恢复到通过DDL操作改变了表结构的早期状态。

**注意事项**

* TIMECAPSULE TABLE语句的用法：从回收站中闪回。
* 回收站记录了DROP和TRUNCATE的对象数据。TO BEFORE DROP和TO BEFORE TRUNCATE就是从回收站中闪回。
* 不支持闪回表的对象类型：系统表、列存表、内存表、DFS表、全局临时表、本地临时表、UNLOGGED表、序列表、hashbucket表。
* 闪回点和当前点之间，执行过修改表结构或影响物理存储的语句（DDL、DCL、VACUUM FULL），闪回失败。
* 执行闪回删除需要用户具有如下权限：用户必须具有垃圾对象所在schema的create和usage权限，并且用户必须是schema的所有者或者是垃圾对象的所有者。
* 执行闪回TRUNCATE需要用户具有如下权限：用户必须具有垃圾对象所在schema的create和usage权限，并且用户必须是schema的所有者或者是垃圾对象的所有者，另外用户必须具有垃圾对象的TRUNCATE权限。
* 不适用闪回drop/truncate功能的场景或表：
* 回收站关闭场景：enable\_recyclebin = off；
* 系统处于维护态（xc\_maintenance\_mode = on）或升级场景；
* 多对象删除场景：DROP/TRUNCATE TABLE命令同时指定多个对象；
* 系统表、列存表、内存表、DFS表、全局临时表、本地临时表、UNLOGGED表、序列表、hashbucket表。

**语法格式**

TIMECAPSULE TABLE [ schema.]table\_name TO {BEFORE { DROP [RENAME TO table\_name] | TRUNCATE } }

**参数说明**

* **schema\_name**
* 指定模式包含的表。如果缺省，则为当前模式。
* **table\_name**
* 指定表名。
* **TO BEFORE DROP**
* 使用这个子句检索回收站中已删除的表及其子对象。
* 你可以指定原始用户指定的表的名称，或对象删除时数据库分配的系统生成名称。
* 回收站中系统生成的对象名称是唯一的。因此，如果指定系统生成名称，那么数据库检索指定的对象。使用“select \* from gs\_recyclebin;”语句查看回收站中的内容。
* 如果指定了用户指定的名称，且如果回收站中包含多个该名称的对象，然后数据库检索回收站中最近移动的对象。如果想要检索更早版本的表，你可以这样做：
* 指定你想要检索的表的系统生成名称。
* 执行TIMECAPSULE TABLE ... TO BEFORE DROP语句，直到你要检索的表。
* 恢复DROP表时，只恢复基表名，其他子对象名均保持回收站对象名。用户可根据需要，执行DDL命令手工调整子对象名。
* 回收站对象不支持DML、DCL、DDL等写操作，不支持DQL查询操作（后续支持）。
* recyclebin\_retention\_time配置参数用于设置回收站对象保留时间，超过该时间的回收站对象将被自动清理。
* **RENAME TO**
* 为从回收站中检索的表指定一个新名称。
* **TRUNCATE**
* 闪回到TRUNCATE之前。

**示例**

--当前数据库已存在tpcds模式  
-- 删除表tpcds.reason\_t2  
DROP TABLE IF EXISTS tpcds.reason\_t2;  
-- 创建表tpcds.reason\_t2  
vastbase=# CREATE TABLE tpcds.reason\_t2  
 (  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
 );  
   
--向表tpcds.reason\_t2中插入记录  
vastbase=# INSERT INTO tpcds.reason\_t2 VALUES (1, 'AA', 'reason1'),(2, 'AB', 'reason2'),(3, 'AC', 'reason3');  
INSERT 0 3  
  
--清空tpcds.reason\_t2表中的数据  
vastbase=# TRUNCATE TABLE tpcds.reason\_t2;  
  
--查询tpcds.reason\_t2表中的数据  
vastbase=# select \* from tpcds.reason\_t2;  
 r\_reason\_sk | r\_reason\_id | r\_reason\_desc   
-------------+-------------+---------------  
(0 rows)  
  
--执行闪回TRUNCATE  
vastbase=# TIMECAPSULE TABLE tpcds.reason\_t2 to BEFORE TRUNCATE;  
  
vastbase=# select \* from tpcds.reason\_t2;  
 r\_reason\_sk | r\_reason\_id | r\_reason\_desc   
-------------+------------------+------------------------------------------------------------------------------------------------------  
 1 | AA | reason1   
 2 | AB | reason2   
 3 | AC | reason3   
(3 rows)  
--删除表tpcds.reason\_t2  
vastbase=# DROP TABLE tpcds.reason\_t2;  
  
--执行闪回DROP  
vastbase=# TIMECAPSULE TABLE tpcds.reason\_t2 to BEFORE DROP;  
TimeCapsule Table

### TRUNCATE

**功能描述**

清理表数据，TRUNCATE快速地从表中删除所有行。

它和在目标表上进行无条件的DELETE有同样的效果，但由于TRUNCATE不做表扫描，因而快得多。在大表上操作效果更明显。

**注意事项**

* TRUNCATE TABLE在功能上与不带WHERE子句DELETE语句相同：二者均删除表中的全部行。
* TRUNCATE TABLE比DELETE速度快且使用系统和事务日志资源少：
* DELETE语句每次删除一行，并在事务日志中为所删除每行记录一项。
* TRUNCATE TABLE通过释放存储表数据所用数据页来删除数据，并且只在事务日志中记录页的释放。
* TRUNCATE、DELETE、DROP三者的差异如下：
* TRUNCATE TABLE，删除内容，释放空间，但不删除定义。
* DELETE TABLE，删除内容，不删除定义，不释放空间。
* DROP TABLE，删除内容和定义，释放空间。

**语法格式**

* 清理表数据。

TRUNCATE [ TABLE ] [ ONLY ] {table\_name [ \* ]} [, ... ]  
 [ CONTINUE IDENTITY ] [ CASCADE | RESTRICT][PURGE]};

* 清理表分区的数据。

ALTER TABLE [ IF EXISTS ] { [ ONLY ] table\_name   
 | table\_name \*   
 | ONLY ( table\_name ) }   
 TRUNCATE PARTITION { partition\_name   
 | FOR ( partition\_value [, ...] ) } [ UPDATE GLOBAL INDEX ];

**参数说明**

* **ONLY**
* 如果声明ONLY，只有指定的表会被清空。如果没有声明ONLY，这个表以及其所有子表（若有）会被清空。
* **table\_name**
* 目标表的名称（可以有模式修饰）。
* 取值范围：已存在的表名。
* **CONTINUE IDENTITY**
* 不改变序列的值。这是缺省值。
* **CASCADE | RESTRICT**
* CASCADE：级联清空所有由于CASCADE而被添加到组中的表。
* RESTRICT（缺省值）：完全清空。
* **PURGE**：默认将表数据放入回收站中，PURGE直接清理。
* **partition\_name**
* 目标分区表的分区名。
* 取值范围：已存在的分区名。
* **partition\_value**
* 指定的分区键值。
* 通过PARTITION FOR子句指定的这一组值，可以唯一确定一个分区。
* 取值范围：需要进行删除数据分区的分区键的取值范围。
* fig: **须知：**   
  使用PARTITION FOR子句时，partition\_value所在的整个分区会被清空。
* **UPDATE GLOBAL INDEX**
* 如果使用该参数，则会更新分区表上的所有全局索引，以确保使用全局索引可以查询出正确的数据；如果不使用该参数，则分区表上的所有全局索引将会失效。

**示例**

--创建表。  
vastbase=# CREATE TABLE tpcds.reason\_t1 AS TABLE tpcds.reason;  
  
--清空表tpcds.reason\_t1。  
vastbase=# TRUNCATE TABLE tpcds.reason\_t1;  
  
--删除表。  
vastbase=# DROP TABLE tpcds.reason\_t1;

--创建分区表。  
vastbase=# CREATE TABLE tpcds.reason\_p  
(  
 r\_reason\_sk integer,  
 r\_reason\_id character(16),  
 r\_reason\_desc character(100)  
)PARTITION BY RANGE (r\_reason\_sk)  
(  
 partition p\_05\_before values less than (05),  
 partition p\_15 values less than (15),  
 partition p\_25 values less than (25),  
 partition p\_35 values less than (35),  
 partition p\_45\_after values less than (MAXVALUE)  
);  
  
--插入数据。  
vastbase=# INSERT INTO tpcds.reason\_p SELECT \* FROM tpcds.reason;  
  
--清空分区p\_05\_before。  
vastbase=# ALTER TABLE tpcds.reason\_p TRUNCATE PARTITION p\_05\_before;  
  
--清空分区p\_15。  
vastbase=# ALTER TABLE tpcds.reason\_p TRUNCATE PARTITION for (13);  
  
--清空分区表。  
vastbase=# TRUNCATE TABLE tpcds.reason\_p;  
  
--删除表。  
vastbase=# DROP TABLE tpcds.reason\_p;

### UPDATE

**功能描述**

更新表中的数据。UPDATE修改满足条件的所有行中指定的字段值，WHERE子句声明条件，SET子句指定的字段会被修改，没有出现的字段则保持它们的原值。

**注意事项**

* 表的所有者、拥有表UPDATE权限的用户或拥有UPDATE ANY TABLE权限的用户，有权更新表中的数据，系统管理员默认拥有此权限。
* 对expression或condition条件里涉及到的任何表要有SELECT权限。
* 对于列存表，暂时不支持RETURNING子句。
* 列存表不支持结果不确定的更新(non-deterministic update)。试图对列存表用多行数据更新一行时会报错。
* 列存表的更新操作，旧记录空间不会回收，需要执行VACUUM FULL table\_name进行清理。
* 对于列存复制表，暂不支持UPDATE操作。

**语法格式**

UPDATE [/\*+ plan\_hint \*/] [ ONLY ] table\_name [ \* ] [ [ AS ] alias ]  
SET {column\_name = { expression | DEFAULT }   
 |( column\_name [, ...] ) = {( { expression | DEFAULT } [, ...] ) |sub\_query }}[, ...]  
 [ FROM from\_list] [ WHERE condition ]  
 [ RETURNING {\*   
 | {output\_expression [ [ AS ] output\_name ]} [, ...] }];  
  
where sub\_query can be:  
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]  
{ \* | {expression [ [ AS ] output\_name ]} [, ...] }  
[ FROM from\_item [, ...] ]  
[ WHERE condition ]  
[ GROUP BY grouping\_element [, ...] ]  
[ HAVING condition [, ...] ]

**参数说明**

* **plan\_hint子句**
* 以/\*+ /的形式在UPDATE关键字后，用于对UPDATE对应的语句块生成的计划进行hint调优，详细用法请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->性能调优->SQL调优指南->使用Plan Hint进行调优》章节。每条语句中只有第一个/+ plan\_hint \*/注释块会作为hint生效，里面可以写多条hint。
* **table\_name**
* 要更新的表名，可以使用模式修饰。
* 取值范围：已存在的表名称。
* **partition\_clause**
* 指定分区更新操作
* PARTITION { ( partition\_name ) | FOR ( partition\_value [, ...] ) } |
* SUBPARTITION { ( subpartition\_name ) | FOR ( subpartition\_value [, ...] ) }
* 关键字详见：SQL语法参考->SQL语法->SELECT章节。
* 示例详见：SQL语法参考->SQL语法->CREATE TABLE SUBPARTITION章节。
* **alias**
* 目标表的别名。
* 取值范围：字符串，符合标识符命名规范。
* **column\_name**
* 要修改的字段名。
* 支持使用目标表的别名加字段名来引用这个字段。例如：
* UPDATE foo AS f SET f.col\_name = 'namecol';
* 取值范围：已存在的字段名。
* **expression**
* 赋给字段的值或表达式。
* **DEFAULT**
* 用对应字段的缺省值填充该字段。
* 如果没有缺省值，则为NULL。
* **sub\_query**
* 子查询。
* 使用同一数据库里其他表的信息来更新一个表可以使用子查询的方法。其中SELECT子句具体介绍请参考：SQL语法参考->SQL语法->SELECT章节。
* **from\_list**
* 一个表的表达式列表，允许在WHERE条件里使用其他表的字段。与在一个SELECT语句的FROM子句里声明表列表类似。
* fig: **须知：**   
  目标表绝对不能出现在from\_list里，除非在使用一个自连接（此时它必须以from\_list的别名出现）。
* **condition**
* 一个返回Boolean类型结果的表达式。只有这个表达式返回true的行才会被更新。不建议使用int等数值类型作为condition，因为int等数值类型可以隐式转换为bool值（非0值隐式转换为true，0转换为false），可能导致非预期的结果。
* **output\_expression**
* 在所有需要更新的行都被更新之后，UPDATE命令用于计算返回值的表达式。
* 取值范围：使用任何table以及FROM中列出的表的字段。\*表示返回所有字段。
* **output\_name**
* 字段的返回名称。

**示例**

--创建表student1。  
vastbase=# CREATE TABLE student1  
(  
 stuno int,  
 classno int   
);  
  
--插入数据。  
vastbase=# INSERT INTO student1 VALUES(1,1);  
vastbase=# INSERT INTO student1 VALUES(2,2);  
vastbase=# INSERT INTO student1 VALUES(3,3);  
  
--查看数据。  
vastbase=# SELECT \* FROM student1;  
  
--直接更新所有记录的值。  
vastbase=# UPDATE student1 SET classno = classno\*2;  
  
--查看数据。  
vastbase=# SELECT \* FROM student1;  
  
--删除表。  
vastbase=# DROP TABLE student1;

### VACUUM

**功能描述**

VACUUM回收表或B-Tree索引中已经删除的行所占据的存储空间。在一般的数据库操作里，那些已经DELETE的行并没有从它们所属的表中物理删除；在完成VACUUM之前它们仍然存在。因此有必要周期地运行VACUUM，特别是在经常更新的表上。

**注意事项**

* 如果没有参数，VACUUM处理当前数据库里用户拥有相应权限的每个表。如果参数指定了一个表，VACUUM只处理指定的那个表。
* 要对一个表进行VACUUM操作，通常用户必须是表的所有者或者被授予了指定表VACUUM权限的用户，默认系统管理员有该权限。数据库的所有者允许对数据库中除了共享目录以外的所有表进行VACUUM操作（该限制意味着只有系统管理员才能真正对一个数据库进行VACUUM操作）。VACUUM命令会跳过那些用户没有权限的表进行垃圾回收操作。
* VACUUM不能在事务块内执行。
* 建议生产数据库经常清理（至少每晚一次），以保证不断地删除失效的行。尤其是在增删了大量记录之后，对受影响的表执行VACUUM ANALYZE命令是一个很好的习惯。这样将更新系统目录为最近的更改，并且允许查询优化器在规划用户查询时有更好地选择。
* 不建议日常使用FULL选项，但是可以在特殊情况下使用。例如在用户删除了一个表的大部分行之后，希望从物理上缩小该表以减少磁盘空间占用。VACUUM FULL通常要比单纯的VACUUM收缩更多的表尺寸。FULL选项并不清理索引，所以推荐周期性的运行REINDEX命令。实际上，首先删除所有索引，再运行VACUUM FULL命令，最后重建索引通常是更快的选择。如果执行此命令后所占用物理空间无变化（未减少），请确认是否有其他活跃事务（删除数据事务开始之前开始的事务，并在VACUUM FULL执行前未结束）存在，如果有等其他活跃事务退出进行重试。
* VACUUM会导致I/O流量的大幅增加，这可能会影响其他活动会话的性能。因此，有时候会建议使用基于开销的VACUUM延迟特性。
* 如果指定了VERBOSE选项，VACUUM将打印处理过程中的信息，以表明当前正在处理的表。各种有关当前表的统计信息也会打印出来。但是对于列存表执行VACUUM操作，指定了VERBOSE选项，无信息输出。
* 当含有带括号的选项列表时，选项可以以任何顺序写入。如果没有括号，则选项必须按语法显示的顺序给出。
* VACUUM和VACUUM FULL时，会根据参数vacuum\_defer\_cleanup\_age延迟清理行存表记录，即不会立即清理刚刚删除的元组。
* VACUUM ANALYZE先执行一个VACUUM操作，然后给每个选定的表执行一个ANALYZE。对于日常维护脚本而言，这是一个很方便的组合。
* 简单的VACUUM（不带FULL选项）只是简单地回收空间并且令其可以再次使用。这种形式的命令可以和对表的普通读写并发操作，因为没有请求排他锁。VACUUM FULL执行更广泛的处理，包括跨块移动行，以便把表压缩到最少的磁盘块数目里。这种形式要慢许多并且在处理的时候需要在表上施加一个排他锁。
* VACUUM列存表内部执行的操作包括三个：迁移delta表中的数据到主表、VACUUM主表的delta表、VACUUM主表的desc表。该操作不会回收delta表的存储空间，如果要回收delta表的冗余存储空间，需要对该列存表执行VACUUM DELTAMERGE。
* 同时执行多个VACUUM FULL可能出现死锁。
* 如果没有打开xc\_maintenance\_mode参数，那么VACUUM FULL操作将跳过所有系统表。
* 执行DELETE后立即执行VACUUM FULL命令不会回收空间。执行DELETE后再执行1000个非SELECT事务，或者等待1s后再执行1个事务，之后再执行VACUUM FULL命令空间才会回收。

**语法格式**

* 回收空间并更新统计信息，对关键字顺序无要求。

VACUUM [ ( { FULL | FREEZE | VERBOSE | {ANALYZE | ANALYSE }} [,...] ) ]  
 [ table\_name [ (column\_name [, ...] ) ] ] [ PARTITION ( partition\_name ) | SUBPARTITION ( subpartition\_name ) ];

* 仅回收空间，不更新统计信息。

VACUUM [ FULL [COMPACT] ] [ FREEZE ] [ VERBOSE ] [ table\_name ]   
 [ PARTITION ( partition\_name ) | SUBPARTITION ( subpartition\_name ) ];

* 回收空间并更新统计信息，且对关键字顺序有要求。

VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] { ANALYZE | ANALYSE } [ VERBOSE ]   
 [ table\_name [ (column\_name [, ...] ) ] ] [ PARTITION ( partition\_name )| SUBPARTITION ( subpartition\_name ) ];

**参数说明**

* **FULL**
* 选择“FULL”清理，这样可以恢复更多的空间，但是需要耗时更多，并且在表上施加了排他锁。
* fig: **说明：**   
  使用FULL参数会导致统计信息丢失，如果需要收集统计信息，请在VACUUM FULL语句中加上analyze关键字。
* **FREEZE**
* 指定FREEZE相当于执行VACUUM时将vacuum\_freeze\_min\_age参数设为0。
* **VERBOSE**
* 为每个表打印一份详细的清理工作报告。
* **ANALYZE | ANALYSE**
* 更新用于优化器的统计信息，以决定执行查询的最有效方法。
* **table\_name**
* 要清理的表的名称（可以有模式修饰）。
* 取值范围：要清理的表的名称。缺省时为当前数据库中的所有表。
* **column\_name**
* 要分析的具体的字段名称，需要配合analyze选项使用。
* 取值范围：要分析的具体的字段名称。缺省时为所有字段。
* **PARTITION**
* COMPACT和PARTITION参数不能同时使用。
* **partition\_name**
* 要清理的表的一级分区名称。缺省时为所有一级分区。
* **subpartition\_name**
* 要清理的表的二级分区名称。缺省时为所有二级分区
* **DELTAMERGE**
* 只针对列存表，将列存表的delta table中的数据转移到主表存储上。对列存表而言，此操作受enable\_delta\_store和“SQL语法参考->SQL语法->CREATE-TABLE”章节中的参数说明中的deltarow\_threshold控制。

**示例**

--在表reason上创建索引。  
vastbase=# CREATE UNIQUE INDEX ds\_reason\_index1 ON reason(r\_reason\_sk);  
  
--对带索引的表reason执行VACUUM操作。  
vastbase=# VACUUM (VERBOSE, ANALYZE) reason;  
  
--删除索引。  
vastbase=# DROP INDEX ds\_reason\_index1 CASCADE;  
vastbase=# DROP TABLE reason;

**优化建议**

vacuum

* VACUUM不能在事务块内执行。
* 建议生产数据库经常清理（至少每晚一次），以保证不断地删除失效的行。尤其是在增删了大量记录后，对相关表执行VACUUM ANALYZE命令。
* 不建议日常使用FULL选项，但是可以在特殊情况下使用。例如，一个例子就是在用户删除了一个表的大部分行之后，希望从物理上缩小该表以减少磁盘空间占用。
* 执行VACUUM FULL操作时，建议首先删除相关表上的所有索引，再运行VACUUM FULL命令，最后重建索引。

### VALUES

**功能描述**

根据给定的值表达式计算一个或一组行的值。它通常用于在一个较大的命令内生成一个“常数表”。

**注意事项**

* 应当避免使用VALUES返回数量非常大的结果行，否则可能会遭遇内存耗尽或者性能低下。出现在INSERT中的VALUES是一个特殊情况，因为目标字段类型可以从INSERT的目标表获知，并不需要通过扫描VALUES列表来推测，所以在此情况下可以处理非常大的结果行。
* 如果指定了多行，那么每一行都必须拥有相同的元素个数。

**语法格式**

VALUES {( expression [, ...] )} [, ...]  
 [ ORDER BY { sort\_expression [ ASC | DESC | USING operator ] } [, ...] ]  
 [ LIMIT { count | ALL } ]  
 [ OFFSET start [ ROW | ROWS ] ]  
 [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ];

**参数说明**

* **expression**
* 用于计算或插入结果表指定地点的常量或者表达式。
* 在一个出现在INSERT顶层的VALUES列表中，expression可以被DEFAULT替换以表示插入目的字段的缺省值。除此以外，当VALUES出现在其他场合的时候是不能使用DEFAULT的。
* **sort\_expression**
* 一个表示如何排序结果行的表达式或者整数常量。
* **ASC**
* 指定按照升序排列。
* **DESC**
* 指定按照降序排列。
* **operator**
* 一个排序操作符。
* **count**
* 返回的最大行数。
* **OFFSET start { ROW | ROWS }**
* 声明返回的最大行数，而start声明开始返回行之前忽略的行数。
* **FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY**
* FETCH子句限定返回查询结果从第一行开始的总行数，count的缺省值为1。

**示例**

参考：SQL语法参考->SQL语法->INSERT章节。

### WITH FUNCTION

**功能描述**

with function功能实现了在with语句中定义临时函数，在后续的子语句中可以重复使用该函数。

**注意事项**

无

**语法格式**

WITH { FUNCTION func\_name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default\_expr ] [, ...] ] )

[ RETURNS rettype

| RETURNS TABLE ( column\_name column\_type [, ...] ) ] AS 'definition' } [, ...]

{ SelectStmt }

**参数说明**

* **func\_name**

自定义的函数名称。

取值范围：字符串，要符合标识符命令规范。

* **argmode**

函数参数的模式。

取值范围：IN，OUT，INOUT或VARIADIC（用于声明数组类型的参 数），缺省值是IN。只有OUT模式的参数后面能跟VARIADIC。并且 OUT和INOUT模式的参数不能用在RETURNS TABLE的函数定义中。

* **argname**

函数参数的名称。

取值范围：字符串，要符合标识符命令规范。

* **argtype**

函数参数的类型。

* **rettype**

函数返回值的数据类型。如果存在OUT或IN OUT参数，可以省略 RETURNS子句。如果出现了，那么它必须和输出参数隐含的结果类型 兼容：如果有多个输出参数，必须是 RECORD， 如果只有一个输出参数， 则与其相同。

* **column\_name**

字段名称。

* **column\_type**

字段类型。

* **definition**

一个定义函数的字符串长，含义取决于语言。它可以是一个内部函数名 称、一个指向某个目标文件的路径、一个SQL查询、一个过程语言文 本。

**示例**

使用WITH FUNCTION字句调用函数。

WITH FUNCTION withfunc (x integer) RETURNS INTEGER AS $$

BEGIN

RETURN x+1;

END

$$,

FUNCTION withfunc2(x INTEGER, y INTEGER) RETURNS INTEGER AS $$

BEGIN

RETURN x+y;

END;

$$,

FUNCTION withfunc3(x TEXT) RETURNS TEXT AS $$

BEGIN

RETURN x || '-test';

END;

$$

SELECT withfunc(1),withfunc2(2,3),withfunc3('4');

当结果显示如下，则表示函数调用成功：

withfunc | withfunc2 | withfunc3

----------+-----------+-----------

2 | 5 | 4-test

(1 row)

**相关链接**

无

## 类型转换概述

在SQL语言中，每个数据都与一个决定其行为和用法的数据类型相关。Vastbase提供一个可扩展的数据类型系统，该系统比其它SQL实现更具通用性和灵活性。因而，Vastbase中大多数类型转换是由通用规则来管理的，这种做法允许使用混合类型的表达式。

Vastbase扫描/分析器只将词法元素分解成五个基本种类：整数、浮点数、字符串、标识符和关键字。大多数非数字类型首先表现为字符串。SQL语言的定义允许将常量字符串声明为具体的类型。例如下面查询：

SELECT text 'Origin' AS "label", point '(0,0)' AS "value";   
 label | value   
--------+-------   
 Origin | (0,0)   
(1 row)

示例中有两个文本常量，类型分别为text和point。如果没有为字符串文本声明类型，则该文本首先被定义成一个unknown类型。

在Vastbase分析器里，有四种基本的SQL结构需要独立的类型转换规则：

* 函数调用

多数SQL类型系统是建筑在一套丰富的函数上的。函数调用可以有一个或多个参数。因为SQL允许函数重载，所以不能通过函数名直接找到要调用的函数，分析器必须根据函数提供的参数类型选择正确的函数。

* 操作符

SQL允许在表达式上使用前缀或后缀（单目）操作符，也允许表达式内部使用双目操作符（两个参数）。像函数一样，操作符也可以被重载，因此操作符的选择也和函数一样取决于参数类型。

* 值存储

INSERT和UPDATE语句将表达式结果存入表中。语句中的表达式类型必须和目标字段的类型一致或者可以转换为一致。

* UNION，CASE和相关构造

因为联合SELECT语句中的所有查询结果必须在一列里显示出来，所以每个SELECT子句中的元素类型必须相互匹配并转换成一个统一类型。类似地，一个CASE构造的结果表达式必须转换成统一的类型，这样整个CASE表达式会有一个统一的输出类型。同样的要求也存在于ARRAY构造以及GREATEST和LEAST函数中。

系统表pg\_cast存储了有关数据类型之间的转换关系以及如何执行这些转换的信息。详细信息请参见“PG\_CAST”章节。

语义分析阶段会决定表达式的返回值类型并选择适当的转换行为。数据类型的基本类型分类，包括：Boolean，numeric，string，bitstring，datetime，timespan，geometric和network。每种类型都有一种或多种首选类型用于解决类型选择的问题。根据首选类型和可用的隐含转换，就可能保证有歧义的表达式（那些有多个候选解析方案的）得到有效的方式解决。

所有类型转换规则都是建立在下面几个基本原则上的：

* 隐含转换决不能有奇怪的或不可预见的输出。
* 如果一个查询不需要隐含的类型转换，分析器和执行器不应该进行更多的额外操作。这就是说，任何一个类型匹配、格式清晰的查询不应该在分析器里耗费更多的时间，也不应该向查询中引入任何不必要的隐含类型转换调用。
* 另外，如果一个查询在调用某个函数时需要进行隐式转换，当用户定义了一个有正确参数的函数后，解释器应该选择使用新函数。

### 操作符

**操作符类型解析**

1、从系统表pg\_operator中选出要考虑的操作符。如果可以找到一个参数类型以及参数个数都一致的操作符，那么这个操作符就是最终使用的操作符。如果找到了多个备选的操作符，我们将从中选择一个最合适的。

2、寻找最优匹配。

* 抛弃那些输入类型不匹配并且也不能隐式转换成匹配的候选操作符。unknown文本在这种情况下可以转换成任何东西。如果只剩下一个候选项，则用之，否则继续下一步。
* 遍历所有候选操作符，保留那些输入类型匹配最准确的。此时，域被看作和他们的基本类型相同。如果没有一个操作符能被保留，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。
* 遍历所有候选操作符，保留那些需要类型转换时接受（属于输入数据类型的类型范畴的）首选类型位置最多的操作符。如果没有接受首选类型的操作符，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。
* 如果有任何输入参数是unknown类型，检查剩余的候选操作符对应参数位置的类型范畴。在每一个能够接受字符串类型范畴的位置使用string类型（这种对字符串的偏爱是合适的，因为unknown文本确实像字符串）。另外，如果所有剩下的候选操作符都接受相同的类型范畴，则选择该类型范畴，否则抛出一个错误（因为在没有更多线索的条件下无法作出正确的选择）。现在抛弃不接受选定的类型范畴的候选操作符，然后，如果任意候选操作符在某个给定的参数位置接受一个首选类型，则抛弃那些在该参数位置接受非首选类型的候选操作符。如果没有一个操作符能被保留，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。
* 如果同时有unknown和已知类型的参数，并且所有已知类型的参数都是相同的类型，那么假设unknown参数也是那种类型，并检查哪个候选操作符在unknown参数位置接受那个类型。如果只有一个操作符符合，那么使用它。否则，产生一个错误。

**示例**

* 阶乘操作符类型解析。在系统表中里只有一个阶乘操作符（后缀!），它以bigint作为参数。扫描器给下面查询表达式的参数赋予bigint的初始类型：

SELECT 40 ! AS "40 factorial";

结果显示如下：

40 factorial   
--------------------------------------------------   
 815915283247897734345611269596115894272000000000   
(1 row)

分析器对参数做类型转换，查询等效于：

SELECT CAST(40 AS bigint) ! AS "40 factorial";

* 字符串连接操作符类型分析。一种字符串风格的语法既可以用于字符串也可以用于复杂的扩展类型。未声明类型的字符串将被所有可能的候选操作符匹配。有一个未声明的参数的例子：

SELECT text 'abc' || 'def' AS "text and unknown";

结果显示如下：

text and unknown   
------------------   
 abcdef   
(1 row)

本例中分析器寻找两个参数都是text的操作符。确实有这样的操作符，两个参数都是text类型。

下面是连接两个未声明类型的值：

SELECT 'abc' || 'def' AS "unspecified";

结果显示如下：

unspecified   
-------------   
 abcdef   
(1 row)

因为查询中没有声明任何类型，所以本例中对类型没有任何初始提示。因此，分析器查找所有候选操作符，发现既存在接受字符串类型范畴的操作符也存在接受位串类型范畴的操作符。因为字符串类型范畴是首选，所以选择字符串类型范畴的首选类型text作为解析未知类型文本的声明类型。

* 绝对值和取反操作符类型分析。Vastbase操作符表里面有几条记录对应于前缀操作符@，它们都用于为各种数值类型实现绝对值操作。其中之一用于float8类型，它是数值类型范畴中的首选类型。因此，在面对unknown输入的时候， Vastbase会使用该类型：

SELECT @ '-4.5' AS "abs";

结果显示如下：

abs   
-----   
 4.5   
(1 row)

此处，系统在应用选定的操作符之前隐式的转换unknown类型的文字为float8类型。

* 数组包含操作符类型分析。这里是解决一个操作符带有一个已知和一个未知类型输入的例子：

SELECT array[1,2] <@ '{1,2,3}' as "is subset";

结果显示如下：

is subset   
-----------   
 t   
(1 row)

Vastbase操作符表有几条记录对应于中缀操作符<@，但是只有两个可以在左侧接受一个整数数组的操作符是数组包含(anyarray <@ anyarray) 和范围包含(anyelement <@ anyrange)的。因为没有多态的伪类型(参阅12.3.15伪类型)是首选的，所以解析器不能解决这个基础上的歧义。然而，最后一个解析规则告诉用户，假设未知类型的文字是和另外一个输入相同的类型，也就是，整数数组。现在只有两个操作符中的一个可以匹配，所以选择数组包含。（如果用户选择了范围包含，用户将得到一个错误，因为字符串没有正确的格式成为范围的文字。）

### UNION,CASE和相关构造

SQL UNION构造必须把那些可能不太相似的类型匹配起来成为一个结果集。解析算法分别应用于联合查询的每个输出字段。INTERSECT和EXCEPT构造对不相同的类型使用和UNION相同的算法进行解析。CASE、ARRAY、VALUES、GREATEST和LEAST构造也使用同样的算法匹配它的部件表达式并且选择一个结果数据类型。

**UNION，CASE和相关构造解析**

* 如果所有输入都是相同的类型，并且不是unknown类型，那么解析成这种类型。
* 如果所有输入都是unknown类型则解析成text类型（字符串类型范畴的首选类型）。否则，忽略unknown输入。
* 如果输入不属于同一个类型范畴（unknown类型除外），失败。
* 如果输入类型是同一个类型范畴，则选择该类型范畴的首选类型（例外：union操作会选择第一个分支的类型作为所选类型）。

fig: **说明：**

系统表pg\_type中typcategory表示数据类型范畴typispreferred表示是否是typcategory分类中的首选类型。

* 把所有输入转换为所选的类型（对于字符串保持原有长度）。如果从给定的输入到所选的类型没有隐式转换则失败。
* 若输入中含json、txid\_snapshot、sys\_refcursor或几何类型，则不能进行union。

**对于case和coalesce，在TD兼容模式下的处理**

* 如果所有输入都是相同的类型，并且不是unknown类型，那么解析成这种类型。
* 如果所有输入都是unknown类型则解析成text类型。
* 如果输入字符串（包括unknown，unknown当text来处理）和数字类型，那么解析成字符串类型，如果是其他不同的类型范畴，则报错。
* 如果输入类型是同一个类型范畴，则选择该类型的优先级较高的类型。
* 把所有输入转换为所选的类型。如果从给定的输入到所选的类型没有隐式转换则失败。

**对于case，在ORA兼容模式下的处理**

decode(expr, search1, result1, search2, result2, …, defresult)，也即 case expr when search1 then result1 when search2 then result2 else defresult end; 在ORA兼容模式下的处理，将整个表达式最终的返回值类型定为result1的数据类型，或者与result1同类型范畴的更高精度的数据类型。（例如，numeric与int同属数值类型范畴，但numeric比int精度要高，具有更高优先级）

* 将result1的数据类型置为最终的返回值类型preferType，其所属类型范畴为preferCategory。
* 依次考虑result2、result3直至defresult的数据类型。如果其类型范畴也是preferCategory，即与result1具有相同的类型范畴，则判断其精度（优先级）是否高于preferType，如果高于，则将preferType更新为更高精度的数据类型；如果其类型范畴不是preferCategory，则判断其数据类型是否可以隐式转换为preferType，不可以则报错。
* 将最终preferType记录的数据类型作为整个表达式最终的返回值类型；表达式的结果向此类型进行隐式转换。
* 为了兼容一种特殊情况，即表示了超大数字的字符类型向数值类型转换的情况，例如select decode(1, 2, 2, “53465465676465454657567678676”)，大数超过了bigint、double等的表示范围。所以，当result1的类型范畴为数值类型时，将返回值的类型直接置为numeric，以兼容此种特殊情况。
* 数值类型的优先级排序：numeric>float8>float4>int8>int4>int2>int1
* 字符类型的优先级排序：text>varchar=nvarchar2>bpchar>char
* 日期类型的优先级排序：timestamptz>timestamp>smalldatetime>date>abstime>timetz>time
* 日期跨度类型的优先级排序：interval>tinterval>reltime

**示例**

* Union中的待定类型解析。这里，unknown类型文本'b'将被解析成text类型。

SELECT text 'a' AS "text" UNION SELECT 'b';

结果显示如下：

text   
------   
 a   
 b   
(2 rows)

* 简单Union中的类型解析。文本1.2的类型为numeric，而且integer类型的1可以隐含地转换为numeric，因此使用这个类型。

SELECT 1.2 AS "numeric" UNION SELECT 1;

结果显示如下：

numeric   
---------   
 1   
 1.2   
(2 rows)

* 转置Union中的类型解析。这里，因为类型real不能被隐含转换成integer，但是integer可以隐含转换成real，那么联合的结果类型将是real。

SELECT 1 AS "real" UNION SELECT CAST('2.2' AS REAL);

结果显示如下：

real   
------   
 1   
 2.2   
(2 rows)

* TD模式下，coalesce参数输入int和varchar类型，那么解析成varchar类型。

1、在TD模式下，创建TD兼容模式的数据库td\_1并切换数据库为td\_1。

CREATE DATABASE td\_1 dbcompatibility = 'C';   
\c td\_1

2、创建表t1。

CREATE TABLE t1(a int, b varchar(10));

3、查看coalesce参数输入int和varchar类型的查询语句的执行计划。

EXPLAIN VERBOSE select coalesce(a, b) from t1;

当结果显示如下信息，则表示验证完成。

QUERY PLAN   
---------------------------------------------------------------------------------------   
 Seq Scan on public.t1 (cost=0.00..24.18 rows=1134 width=42)   
 Output: (COALESCE((a)::varchar, b)  
(2 rows)

4、删除测试库

\c vastbase  
DROP DATABASE td\_1;

### 函数

**函数类型解析**

1、从系统表pg\_proc中选择所有可能被选到的函数。如果使用了一个不带模式修饰的函数名称，那么认为该函数是那些在当前搜索路径中的函数。如果给出一个带修饰的函数名，那么只考虑指定模式中的函数。如果搜索路径中找到了多个不同参数类型的函数。将从中选择一个合适的函数。

2、查找和输入参数类型完全匹配的函数。如果找到一个，则用之。如果输入的实参类型都是unknown类型，则不会找到匹配的函数。

3、如果未找到完全匹配，请查看该函数是否为一个特殊的类型转换函数。

4、寻找最优匹配。

* 抛弃那些输入类型不匹配并且也不能隐式转换成匹配的候选函数。unknown文本在这种情况下可以转换成任何东西。如果只剩下一个候选项，则用之，否则继续下一步。
* 遍历所有候选函数，保留那些输入类型匹配最准确的。此时，域被看作和它们的基本类型相同。如果没有一个函数能准确匹配，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。
* 遍历所有候选函数，保留那些需要类型转换时接受首选类型位置最多的函数。如果没有接受首选类型的函数，则保留所有候选。如果只剩下一个候选项，则用之，否则继续下一步。
* 如果有任何输入参数是unknown类型，检查剩余的候选函数对应参数位置的类型范畴。在每一个能够接受字符串类型范畴的位置使用string类型（这种对字符串的偏爱是合适的，因为unknown文本确实像字符串）。另外，如果所有剩下的候选函数都接受相同的类型范畴，则选择该类型范畴，否则抛出一个错误（因为在没有更多线索的条件下无法作出正确的选择）。现在抛弃不接受选定的类型范畴的候选函数，然后，如果任意候选函数在那个范畴接受一个首选类型，则抛弃那些在该参数位置接受非首选类型的候选函数。如果没有一个候选符合这些测试则保留所有候选。如果只有一个候选函数符合，则使用它；否则，继续下一步。
* 如果同时有unknown和已知类型的参数，并且所有已知类型的参数有相同的类型，假设unknown参数也是这种类型，检查哪个候选函数可以在unknown参数位置接受这种类型。如果正好一个候选符合，那么使用它。否则，产生一个错误。

**示例**

* 圆整函数参数类型解析。只有一个round函数有两个参数（第一个是numeric，第二个是integer）。所以下面的查询自动把第一个类型为integer的参数转换成numeric类型。

SELECT round(4, 4);

结果显示如下：

round   
--------   
 4.0000   
(1 row)

实际上它被分析器转换成：

SELECT round(CAST (4 AS numeric), 4);

因为带小数点的数值常量初始时被赋予numeric类型，因此下面的查询将不需要类型转换，并且可能会略微高效一些：

SELECT round(4.0, 4);

* 子字符串函数类型解析。有好几个substr函数，其中一个接受text和integer类型。如果用一个未声明类型的字符串常量调用它，系统将选择接受string类型范畴的首选类型（也就是text类型）的候选函数。

SELECT substr('1234', 3);

结果显示如下：

substr   
--------   
 34   
(1 row)

如果该字符串声明为varchar类型，就像从表中取出来的数据一样，分析器将试着将其转换成text类型：

SELECT substr(varchar '1234', 3);

结果显示如下：

substr   
--------   
 34   
(1 row)

被分析器转换后实际上变成：

SELECT substr(CAST (varchar '1234' AS text), 3);

分析器从pg\_cast表中了解到text和varchar是二进制兼容的，意思是说一个可以传递给接受另一个的函数而不需要做任何物理转换。因此，在这种情况下，实际上没有做任何类型转换。

如果以integer为参数调用函数，分析器将试图将其转换成text类型：

SELECT substr(1234, 3);

结果显示如下：

substr   
--------   
 34   
(1 row)

被分析器转换后实际上变成：

SELECT substr(CAST (1234 AS text), 3);

结果显示如下：

substr   
--------   
 34   
(1 row)

### 值存储

**值存储数据类型解析**

1、查找与目标字段准确的匹配。

2、试着将表达式直接转换成目标类型。如果已知这两种类型之间存在一个已注册的转换函数，那么直接调用该转换函数即可。如果表达式是一个未知类型文本，该文本字符串的内容将交给目标类型的输入转换过程。

3、检查一下看目标类型是否有长度转换。长度转换是一个从某类型到自身的转换。如果在pg\_cast表里面找到一个，那么在存储到目标字段之前先在表达式上应用。这样的转换函数总是接受一个额外的类型为integer的参数，它接收目标字段的atttypmod值（实际上是其声明长度，atttypmod的解释随不同的数据类型而不同），并且它可能接受一个Boolean类型的第三个参数，表示转换是显式的还是隐式的。转换函数负责施加那些长度相关的语义，比如长度检查或者截断。

**示例**

character存储类型转换。对一个目标列定义为character(20)的语句，下面的示例显示存储值的长度正确：

1、创建测试表。

CREATE TABLE value\_storage\_t1 (   
 VS\_COL1 CHARACTER(20)   
);

2、插入测试数据。

INSERT INTO value\_storage\_t1 VALUES('abcdef');

3、查询结果。

SELECT VS\_COL1, octet\_length(VS\_COL1) FROM value\_storage\_t1;

当结果显示如下信息，则表示存储值长度正确。

vs\_col1 | octet\_length   
----------------------+--------------   
 abcdef | 20   
(1 row)   
)

4、删除测试表。

DROP TABLE value\_storage\_t1;

这里真正发生的事情是两个unknown文本缺省解析成text，这样就允许||操作符解析成text连接。然后操作符的text结果转换成bpchar("空白填充的字符型"， character类型内部名称)以匹配目标字段类型。不过，从text到bpchar的转换是二进制兼容的，这样的转换是隐含的并且实际上不做任何函数调用。最后，在系统表里找到长度转换函数bpchar(bpchar, integer, Boolean) 并且应用于该操作符的结果和存储的字段长。这个类型相关的函数执行所需的长度检查和额外的空白填充。

## 全文检索

### 介绍

### 全文检索概述

**功能描述**

文本搜索操作符在数据库中已存在多年。Vastbase为文本数据类型提供~、~\*、LIKE和ILIKE操作符用于匹配字符进行检索，同时可以通过使用词典和索引满足现代信息系统所要求的许多必要属性。

**注意事项**

* 每个分词长度必须小于2K字节。
* tsvector结构（分词+位置）的长度必须小于1M字节。
* tsvector的位置值必须大于0，且小于等于16,383字节。
* 每个分词在文档中位置数必须小于256，若超过将舍弃后面的位置信息。
* tsquery中的关键字及对应运算符最大支持到32768字节。
* 没有语义支持，即使是英语。由于要识别派生词并不是那么容易，因此正则表达式也不能满足要求。如，satisfies和satisfy，当使用正则表达式寻找satisfy时，并不会查询到包含satisfies的文档。用户可以使用OR搜索多种派生形式，但过程非常繁琐。并且有些词会有上千的派生词，因此容易出错。
* 没有对搜索结果的分类（排序）。当搜索出成千的文档时，查找效率很低。
* 由于没有索引的支持，每一次的搜索需要遍历所有的文档，整体搜索比较缓慢。使用全文索引可以对文档进行预处理，并且可以使后续的搜索更快速。预处理过程包括：
* 将文档解析成token。为每个文档标记不同类别的token是非常有必要的，例如：数字、文字、复合词、电子邮件地址，这样就可以做不同的处理。原则上token的类别依赖于具体的应用，但对于大多数的应用来说，可以使用一组预定义的token类。
* 将token转换为词素。词素像token一样是一个字符串，但它已经标准化处理，这样同一个词的不同形式是一样的。例如，标准化通常包括：将大写字母转换成小写字母、删除后缀（如英语中的s或者es）。这将允许通过搜索找到同一个词的不同形式，不需要繁琐地输入所有可能的变形样式。同时，这一步通常会删除停用词。这些停用词通常因为太常见而对搜索无用。（总之，token是文档文本的原片段，而词素被认为是有用的索引和搜索词。）Vastbase使用词典执行这一步，且提供了各种标准的词典。
* 保存搜索优化后的预处理文档。比如，每个文档可以呈现为标准化词素的有序组合。伴随词素，通常还需要存储词素位置信息以用于邻近排序。因此文档包含的查询词越密集其排序越高。
* 词典能够对token如何标准化做到细粒度控制。使用合适的词典，可以定义不被索引的停用词。
* 数据类型tsvector用于存储预处理文档，tsquery用于存储查询条件，详细请参见“SQL语法参考->数据类型->文本搜索类型”章节。为这些数据类型提供的函数和操作符请参见“SQL语法参考->数据类型->文本检索函数和操作符”章节。其中最重要的是匹配运算符@@，将在“基本文本匹配”章节中介绍。

#### 分词器

**功能描述**

全文检索功能还可以做更多事情：忽略索引某个词（停用词），处理同义词和使用复杂解析，例如：不仅基于空格的解析。这些功能通过文本搜索分词器控制。Vastbase支持多语言的预定义的分词器，并且可以创建分词器（\dF命令显示了所有可用分词器）。

为了更方便的建立自定义文本搜索分词器，可以通过简单的数据库对象建立分词器。 Vastbase文本搜索功能提供了四种类型与分词器相关的数据库对象：

* 文本搜索解析器将文档分解为token，并且分类每个token（例如：词和数字）。
* 文本搜索词典将token转换成规范格式并且丢弃停用词。
* 文本搜索模板提供潜在的词典功能：一个词典指定一个模板，并且为模板设置参数。
* 文本搜索分词器选择一个解析器，并且使用一系列词典规范化语法分析器产生的token。

**使用方法**

* 在安装期间选择一个合适的分词器，并且在postgresql.conf中相应的设置default\_text\_search\_config。
* 如果为了Vastbase使用同一个文本搜索分词器可以使用postgresql.conf中的值。
* 如果需要在Vastbase中使用不同分词器，可以使用ALTER DATABASE … SET在任一数据库进行配置。
* 用户也可以在每个会话中设置default\_text\_search\_config。

**注意事项**

每个依赖于分词器的文本搜索函数有一个可选的配置参数，用以明确声明所使用的分词器。仅当忽略这个参数的时候，才使用default\_text\_search\_config。

#### 基本文本匹配

**功能描述**

Vastbase的全文检索基于匹配算子@@，当一个tsvector(document)匹配到一个tsquery(query)时，则返回true。其中，tsvector(document)和tsquery(query)两种数据类型可以任意排序。

**示例**

* 查询验证结果。

SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector @@ 'cat & rat'::tsquery AS RESULT;

结果返回如下：

result   
----------   
 t   
(1 row)

SELECT 'fat & cow'::tsquery @@ 'a fat cat sat on a mat and ate a fat rat'::tsvector AS RESULT;

结果返回如下：

result   
----------   
 f   
(1 row)

* tsquery不仅是文本，且比tsvector包含的要多。tsquery包含已经标注化为词条的搜索词，同时可能是使用AND、OR、或NOT操作符连接的多个术语。详细请参见“SQL语法参考->数据类型->文本搜索类型”章节。函数to\_tsquery和plainto\_tsquery对于将用户书写文本转换成适合的tsquery是非常有用的，比如将文本中的词标准化。类似地，to\_tsvector用于解析和标准化文档字符串。因此，实际中文本搜索匹配看起来更像这样：

SELECT to\_tsvector('fat cats ate fat rats') @@ to\_tsquery('fat & rat') AS RESULT;

结果返回如下：

result   
----------   
 t   
(1 row)

需要注意的是，下面这种方式是不可行的：

SELECT 'fat cats ate fat rats'::tsvector @@ to\_tsquery('fat & rat')AS RESULT;

结果返回如下：

result   
----------   
 f   
(1 row)

* 由于tsvector没有对rats进行标准化，所以rats不匹配rat。

@@操作符也支持text输入，允许一个文本字符串的显示转换为tsvector或者在简单情况下忽略tsquery。可用形式是：

tsvector @@ tsquery   
tsquery @@ tsvector   
text @@ tsquery   
text @@ text

我们已经看到了前面两种，形式text @@ tsquery等价于to\_tsvector(text) @@ tsquery，而text @@ text等价于to\_tsvector(text) @@ plainto\_tsquery(text)。

#### 文档

**功能描述**

文档是全文搜索系统的搜索单元，例如：杂志上的一篇文章或电子邮件消息。文本搜索引擎必须能够解析文档，而且可以存储父文档的关联词素（关键词）。后续，这些关联词素用来搜索包含查询词的文档。

在Vastbase中，文档通常是一个数据库表中一行的文本字段，或者这些字段的可能组合（级联）。文档可能存储在多个表中或者需动态获取。换句话说，一个文档由被索引化的不同部分构成，因此无法存储为一个整体，如下面示例所示。

另外一种可能是：文档在文件系统中作为简单的文本文件存储。在这种情况下，数据库可以用于存储全文索引并且执行搜索，同时可以使用一些唯一标识从文件系统中检索文档。然而，从数据库外部检索文件需要拥有系统管理员权限或者特殊函数支持。因此，还是将所有数据保存在数据库中比较方便。同时，将所有数据保存在数据库中可以方便地访问文档元数据以便于索引和显示。

**注意事项**

为了实现文本搜索目的，必须将每个文档减少至预处理后的tsvector格式。搜索和相关性排序都是在tsvector形式的文档上执行的。原始文档只有在被选中要呈现给用户时才会被检索。因此，我们常将tsvector说成文档，但是很显然其实它只是完整文档的一种紧凑表示。

**示例**

1、创建测试表并插入数据。

create table date\_dim(  
d\_dow int,  
d\_dom int,  
d\_fy\_week\_seq int  
);  
  
insert into date\_dim values(5,6,1);  
insert into date\_dim values(0,8,1);  
insert into date\_dim values(2,3,1);  
insert into date\_dim values(3,4,1);  
insert into date\_dim values(4,5,1);  
insert into date\_dim values(1,2,1);  
insert into date\_dim values(6,7,1);  
insert into date\_dim values(4,5,2);  
insert into date\_dim values(1,2,2);  
insert into date\_dim values(6,7,2);

2、查询数据。

SELECT d\_dow || '-' || d\_dom || '-' || d\_fy\_week\_seq AS identify\_serials FROM date\_dim WHERE d\_fy\_week\_seq = 1;

结果显示如下

identify\_serials   
--------------------  
 5-6-1   
 0-8-1   
 2-3-1   
 3-4-1   
 4-5-1   
 1-2-1   
 6-7-1   
(7 rows)

实际上，在这些示例查询中，应该使用coalesce防止一个独立的NULL属性导致整个文档的NULL结果。

### 表和索引

#### 创建GIN索引

**功能描述**

为了加速文本搜索，可以创建GIN索引。

**注意事项**

* to\_tsvector()函数有两个版本。只输一个参数的版本和输两个参数的版本。只输一个参数时，系统默认采用default\_text\_search\_config所指定的分词器。
* 创建索引时必须使用to\_tsvector的两参数版本。只有指定了分词器名称的全文检索函数才可以在索引表达式中使用。这是因为索引的内容必须不受default\_text\_search\_config的影响，否则索引内容可能不一致。由于default\_text\_search\_config的值可以随时调整，从而导致不同条目生成的tsvector采用了不同的分词器，并且没有办法区分究竟使用了哪个分词器。正确地转储和恢复这样的索引也是不可能的。
* 因为在上述创建索引中to\_tsvector使用了两个参数，只有当查询时也使用了两个参数，且参数值与索引中相同时，才会使用该索引。也就是说，WHERE to\_tsvector('english', body) @@ 'a & b' 可以使用索引，但WHERE to\_tsvector(body) @@ 'a & b'不能使用索引。这确保只使用这样的索引——索引各条目是使用相同的分词器创建的。

**示例**

* 方法一：

1、创建GIN索引，索引中的分词器名称由另一列指定时可以建立更复杂的表达式索引。

CREATE INDEX pgweb\_idx\_1 ON tsearch.pgweb USING gin(to\_tsvector('ngram', body));

2、索引可以连接列。

CREATE INDEX pgweb\_idx\_2 ON tsearch.pgweb USING gin(to\_tsvector('english', title || ' ' || body));

* 方法二：

1、创建一个单独的tsvector列控制to\_tsvector的输出。下面的例子是title和body的连接，当其它是NULL的时候，使用coalesce确保一个字段仍然会被索引：

ALTER TABLE tsearch.pgweb ADD COLUMN textsearchable\_index\_col tsvector;   
UPDATE tsearch.pgweb SET textsearchable\_index\_col = to\_tsvector('english', coalesce(title,'') || ' ' || coalesce(body,''));

2、为加速搜索创建一个GIN索引。

CREATE INDEX textsearch\_idx\_3 ON tsearch.pgweb USING gin(textsearchable\_index\_col);

3、执行一个快速全文搜索。

SELECT title   
FROM tsearch.pgweb   
WHERE textsearchable\_index\_col @@ to\_tsquery('north & america')   
ORDER BY last\_mod\_date DESC   
LIMIT 10;

结果显示如下：

title   
--------  
 Canada  
 Mexico  
(2 rows)

相比于一个表达式索引，单独列方法的一个优势是：它没有必要在查询时明确指定分词器以便能使用索引。正如上面例子所示，查询可以依赖于default\_text\_search\_config。另一个优势是搜索比较快速，因为它没有必要重新利用to\_tsvector调用来验证索引匹配。表达式索引方法更容易建立，且它需要较少的磁盘空间，因为tsvector形式没有明确存储。

#### 搜索表

**功能描述**

在不使用索引的情况下也可以进行全文检索。

**示例**

创建测试表并插入数据。

DROP SCHEMA IF EXISTS tsearch CASCADE;  
CREATE SCHEMA tsearch;  
CREATE TABLE tsearch.pgweb(id int, body text, title text, last\_mod\_date date);  
   
INSERT INTO tsearch.pgweb VALUES(1, 'China, officially the People''s Republic of China (PRC), located in Asia, is the world''s most populous state.', 'China', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(2, 'America is a rock band, formed in England in 1970 by multi-instrumentalists Dewey Bunnell, Dan Peek, and Gerry Beckley.', 'America', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(3, 'England is a country that is part of the United Kingdom. It shares land borders with Scotland to the north and Wales to the west.', 'England', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(4, 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania, and numerous smaller islands.', 'Australia', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(6, 'Japan is an island country in East Asia.', 'Japan', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(7, 'Germany, officially the Federal Republic of Germany, is a sovereign state and federal parliamentary republic in central-western Europe.', 'Germany', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(8, 'France, is a sovereign state comprising territory in western Europe and several overseas regions and territories.', 'France', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(9, 'Italy officially the Italian Republic, is a unitary parliamentary republic in Europe.', 'Italy', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(10, 'India, officially the Republic of India, is a country in South Asia.', 'India', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(11, 'Brazil, officially the Federative Republic of Brazil, is the largest country in both South America and Latin America.', 'Brazil', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(12, 'Canada is a country in the northern half of North America.', 'Canada', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(13, 'Mexico, officially the United Mexican States, is a federal republic in the southern part of North America.', 'Mexico', '2010-1-1');

* 示例1：一个简单查询，将body字段中包含america的每一行打印出来。像America这样的相关词也会被找到，因为这些词都被处理成了相同标准的词条。

SELECT id, body, title FROM tsearch.pgweb WHERE to\_tsvector('english', body) @@ to\_tsquery('english', 'america');

结果显示如下：

id | body | title   
----+-------------------------------------------------------------------------------------------------------------------------+---------  
 2 | America is a rock band, formed in England in 1970 by multi-instrumentalists Dewey Bunnell, Dan Peek, and Gerry Beckley. | America  
 11 | Brazil, officially the Federative Republic of Brazil, is the largest country in both South America and Latin America. | Brazil  
 12 | Canada is a country in the northern half of North America. | Canada  
 13 | Mexico, officially the United Mexican States, is a federal republic in the southern part of North America. | Mexico  
(4 rows)

上面的查询指定english配置来解析和规范化字符串。当然也可以省略此配置，通过default\_text\_search\_config进行配置设置。

查询default\_text\_search\_config参数。

SHOW default\_text\_search\_config;   
   
 default\_text\_search\_config   
----------------------------   
 pg\_catalog.english   
(1 row)

执行查询

SELECT id, body, title FROM tsearch.pgweb WHERE to\_tsvector(body) @@ to\_tsquery('america');

结果显示如下：

id | body | title   
----+-------------------------------------------------------------------------------------------------------------------------+---------  
 2 | America is a rock band, formed in England in 1970 by multi-instrumentalists Dewey Bunnell, Dan Peek, and Gerry Beckley. | America  
 11 | Brazil, officially the Federative Republic of Brazil, is the largest country in both South America and Latin America. | Brazil  
 12 | Canada is a country in the northern half of North America. | Canada  
 13 | Mexico, officially the United Mexican States, is a federal republic in the southern part of North America. | Mexico  
(4 rows)

* 示例2：一个复杂查询：检索出在title或者body字段中包含north和america的最近10篇文档：

SELECT title FROM tsearch.pgweb WHERE to\_tsvector(title || ' ' || body) @@ to\_tsquery('north & america') ORDER BY last\_mod\_date DESC LIMIT 10;

结果显示如下：

title   
--------   
 Mexico   
 Canada   
(2 rows)

* 为了清晰，举例中没有调用coalesce函数在两个字段中查找包含NULL的行。
* 以上例子均在没有索引的情况下进行查询。对于大多数应用程序来说，这个方法很慢。因此除了偶尔的特定搜索，文本搜索在实际使用中通常需要创建索引。

#### 索引使用约束

下面是一个使用索引的例子：

create table table1 (c\_int int,c\_bigint bigint,c\_varchar varchar,c\_text text) with(orientation=row);   
create text search configuration ts\_conf\_1(parser=POUND);   
create text search configuration ts\_conf\_2(parser=POUND) with(split\_flag='%');   
set default\_text\_search\_config='ts\_conf\_1';   
create index idx1 on table1 using gin(to\_tsvector(c\_text));   
set default\_text\_search\_config='ts\_conf\_2';   
create index idx2 on table1 using gin(to\_tsvector(c\_text));

查询测试

select c\_varchar,to\_tsvector(c\_varchar) from table1 where to\_tsvector(c\_text) @@ plainto\_tsquery('￥#@……&\*\*') and to\_tsvector(c\_text) @@ plainto\_tsquery('某公司 ') and c\_varchar is not null order by 1 desc limit 3;

该例子的关键点是表table1的同一个列c\_text上建立了两个gin索引：idx1和idx2，但这两个索引是在不同default\_text\_search\_config的设置下建立的。该例子和同一张表的同一个列上建立普通索引的不同之处在于：

* gin索引使用了不同的parser（即分隔符不同），那么idx1和idx2的索引数据是不同的；
* 在同一张表的同一个列上建立的多个普通索引的索引数据是相同的。

因此当执行同一个查询时，使用idx1和idx2查询出的结果是不同的。

**使用约束**

通过上面的例子，索引使用满足如下条件时：

* 在同一个表的同一个列上建立了多个gin索引；
* 这些gin索引使用了不同的parser（即分隔符不同）；
* 在查询中使用了该列，且执行计划中使用索引进行扫描；

为了避免使用不同gin索引导致查询结果不同的问题，需要保证在物理表的一列上只有一个gin索引可用。

### 测试和调试文本搜索

#### 分词器测试

**功能描述**

函数ts\_debug允许简单测试文本搜索分词器。

**语法格式**

ts\_debug([ config regconfig, ] document text,  
 OUT alias text,  
 OUT description text,  
 OUT token text,  
 OUT dictionaries regdictionary[],  
 OUT dictionary regdictionary,  
 OUT lexemes text[])  
 returns setof record

**参数说明**

ts\_debug为文本解析器标识的每个token返回一行记录。记录中的列分别是：

* alias：text类型，token的别名。
* description：text类型，token的描述。
* token：text类型，token的文本内容。
* dictionaries：regdictionary数组类型，是分词器为token选定的词典。
* dictionary：regdictionary类型，用来识别token的词典。如果为空，则不做识别。
* lexemes：text数组类型，词典识别token时生成的词素。如果为空，则不生成词素。空数组（{}）意味着token将被识别成停用词。

**注意事项**

ts\_debug显示document的每个token信息，token是由解析器生成，由指定的词典进行处理。如果忽略对应参数，则使用config指定的分词器或者default\_text\_search\_config指定的分词器。

**示例**

SELECT \* FROM ts\_debug('english','a fat cat sat on a mat - it ate a fat rats');

结果返回如下：

alias | description | token | dictionaries | dictionary | lexemes   
-----------+-----------------+-------+----------------+--------------+---------   
 asciiword | Word, all ASCII | a | {english\_stem} | english\_stem | {}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | fat | {english\_stem} | english\_stem | {fat}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | cat | {english\_stem} | english\_stem | {cat}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | sat | {english\_stem} | english\_stem | {sat}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | on | {english\_stem} | english\_stem | {}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | a | {english\_stem} | english\_stem | {}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | mat | {english\_stem} | english\_stem | {mat}   
 blank | Space symbols | | {} | |   
 blank | Space symbols | - | {} | |   
 asciiword | Word, all ASCII | it | {english\_stem} | english\_stem | {}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | ate | {english\_stem} | english\_stem | {ate}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | a | {english\_stem} | english\_stem | {}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | fat | {english\_stem} | english\_stem | {fat}   
 blank | Space symbols | | {} | |   
 asciiword | Word, all ASCII | rats | {english\_stem} | english\_stem | {rat}   
(24 rows)

#### 解析器测试

* 函数ts\_parse可以直接测试文本搜索解析器。

ts\_parse(parser\_name text, document text,  
 OUT tokid integer, OUT token text) returns setof record

ts\_parse解析指定的document并返回一系列的记录，一条记录代表一个解析生成的token。每条记录包括标识token类型的tokid及token文本。例如：

SELECT \* FROM ts\_parse('default', '123 - a number');

结果返回如下：

tokid | token  
-------+--------  
 22 | 123  
 12 |  
 12 | -  
 1 | a  
 12 |  
 1 | number  
(6 rows)

* 函数ts\_token\_type返回指定解析器的token类型及其描述信息。

ts\_token\_type(parser\_name text, OUT tokid integer,  
 OUT alias text, OUT description text) returns setof record

ts\_token\_type返回一个表，这个表描述了指定解析器可以识别的每种token类型。

对于每个token类型，表中给出了三个字段：

* tokid：用于解析器标记对应的token类型。
* alias：命名分词器命令中的token类型。
* description：简单描述。

SELECT \* FROM ts\_token\_type('default');

结果返回如下：

tokid | alias | description   
-------+-----------------+------------------------------------------   
 1 | asciiword | Word, all ASCII   
 2 | word | Word, all letters   
 3 | numword | Word, letters and digits   
 4 | email | Email address   
 5 | url | URL   
 6 | host | Host   
 7 | sfloat | Scientific notation   
 8 | version | Version number   
 9 | hword\_numpart | Hyphenated word part, letters and digits   
 10 | hword\_part | Hyphenated word part, all letters   
 11 | hword\_asciipart | Hyphenated word part, all ASCII   
 12 | blank | Space symbols   
 13 | tag | XML tag   
 14 | protocol | Protocol head   
 15 | numhword | Hyphenated word, letters and digits   
 16 | asciihword | Hyphenated word, all ASCII   
 17 | hword | Hyphenated word, all letters   
 18 | url\_path | URL path   
 19 | file | File or path name   
 20 | float | Decimal notation   
 21 | int | Signed integer   
 22 | uint | Unsigned integer   
 23 | entity | XML entity   
(23 rows)

### 词典

#### 词典概述

词典用于定义停用词（stop words），即全文检索时不搜索哪些词。

词典还可以用于对同一个词的不同形式进行规范化，这样同一个词的不同派生形式都可以进行匹配。规范化后的词称为词位（lexeme）。

除了提高检索质量外，词的规范化和删除停用词可以减少文档tsvector格式的大小， 从而提高性能。词的规范化和删除停用词并不总是具有语言学意义，用户可以根据应用环境在词典定义文件中自定义规范化和删除规则。

一个词典是一个程序，接收标记（token）作为输入，并返回：

* 如果token在词典中已知，返回对应lexeme数组（注意，一个标记可能对应多个lexeme）。
* 一个lexeme。一个新token会代替输入token被传递给后继词典（当前词典可被称为过滤词典）。
* 如果token在词典中已知，但它是一个停用词，返回空数组。
* 如果词典不能识别输入的token，返回NULL。

Vastbase提供了多种语言的预定义字典，同时提供了五种预定义的词典模板，分别是Simple，Synonym，Thesaurus，Ispell和Snowball，可用于创建自定义参数的新词典。

在使用全文检索时，建议用户：

* 可以在文本搜索配置中定义一个解析器，以及一组用于处理该解析器的输出标记词典。对于解析器返回的每个标记类型，可以在配置中指定不同的词典列表进行处理。当解析器输出一种类型的标记后，在对应列表的每个字典中会查阅该标记，直到某个词典识别它。如果它被识别为一个停用词， 或者没有任何词典识别，该token将被丢弃，即不被索引或检索到。通常情况下，第一个返回非空结果的词典决定了最终结果，后继词典将不会继续处理。但是一个过滤类型的词典可以依据规则替换输入token，然后将替换后的token传递给后继词典进行处理。
* 配置字典列表的一般规则是，第一个位置放置一个应用范围最小的、最具体化定义的词典，其次是更一般化定义的词典， 最后是一个普适定义的词典，比如Snowball词干词典或Simple词典。在下面例子中，对于一个针对天文学的文本搜索配置astro\_en，可以定义标记类型asciiword（ASCII词）对应的词典列表为：天文术语的Synonym同义词词典， Ispell英语词典和Snowball 英语词干词典。

ALTER TEXT SEARCH CONFIGURATION astro\_en   
 ADD MAPPING FOR asciiword WITH astro\_syn, english\_ispell, english\_stem;

过滤类型的词典可以放置在词典列表中除去末尾的任何地方，放置在末尾时是无效的。使用这些词典对标记进行部分规范化，可以有效简化后继词典的处理。

#### Simple词典

**功能描述**

Simple词典首先将输入标记转换为小写字母，然后检查停用词表。如果识别为停用词则返回空数组，即表示该标记会被丢弃。否则，输入标记的小写形式作为规范化后的lexeme返回。此外，Simple词典可通过设置参数Accept为false（默认值true），将非停用词报告为未识别，传递给后继词典继续处理。

**注意事项**

* 大多数词典的功能依赖于词典定义文件，词典定义文件名仅支持小写字母、数字、下划线组合。
* 临时模式pg\_temp下不允许创建词典。
* 词典定义文件的字符集编码必须为UTF-8格式。实际应用时，如果与数据库的字符编码格式不一致，在读入词典定义文件时会进行编码转换。
* 通常情况下，每个session仅读取词典定义文件一次，当且仅当在第一次使用该词典时。需要修改词典文件时，可通过ALTER TEXT SEARCH DICTIONARY命令进行词典定义文件的更新和重新加载。

**示例**

1、创建Simple词典。

CREATE TEXT SEARCH DICTIONARY public.simple\_dict (   
 TEMPLATE = pg\_catalog.simple,   
 STOPWORDS = english   
);

其中，停用词表文件全名为english.stop。关于创建simple词典的语法和更多参数，请参见章节“CREATE TEXT SEARCH DICTIONARY”。

2、使用Simple词典。

SELECT ts\_lexize('public.simple\_dict','YeS');

结果显示如下

ts\_lexize   
-----------   
 {yes}   
(1 row)

SELECT ts\_lexize('public.simple\_dict','The');

结果显示如下：

ts\_lexize   
-----------  
 {}  
(1 row)

3、设置参数ACCEPT=false，使Simple词典返回NULL，而不是返回非停用词的小写形式。

ALTER TEXT SEARCH DICTIONARY public.simple\_dict ( Accept = false );

查询验证。

SELECT ts\_lexize('public.simple\_dict','YeS');

结果显示如下：

ts\_lexize   
-----------   
(1 row)

查询验证。

SELECT ts\_lexize('public.simple\_dict','The');

结果显示如下：

ts\_lexize   
-----------   
 {}   
(1 row)

#### Snowball词典

Snowball词典模板支持词干分析词典，基于Martin Porter的Snowball项目，内置有许多语言的词干分析算法。Vastbase中预定义有多种语言的Snowball词典，可通过系统表章节“PG\_TS\_DICT”查看预定义的词干分析词典以及支持的语言词干分析算法。

无论是否可以简化，Snowball词典将标示所有输入为已识别，因此它应当被放置在词典列表的最后。把Snowball词典放在任何其他词典前面会导致后继词典失效，因为输入token不会通过Snowball词典进入到下一个词典。

#### Synonym词典

**功能描述**

Synonym词典用于定义、识别token的同义词并转化，不支持词组（词组形式的同义词可用Thesaurus词典定义，详细请参见章节“Thesaurus词典”）。

**示例**

* 1、Synonym词典可用于解决语言学相关问题，例如，为避免使单词"Paris"变成"pari"，可在Synonym词典文件中定义一行"Paris paris"，并将该词典放置在预定义的english\_stem词典之前。

SELECT \* FROM ts\_debug('english', 'Paris');

结果返回如下：

alias | description | token | dictionaries | dictionary | lexemes   
-----------+-----------------+-------+----------------+--------------+---------   
 asciiword | Word, all ASCII | Paris | {english\_stem} | english\_stem | {pari}   
(1 row)

2、创建词典，FILEPATH以dicts文件实际目录为准。

CREATE TEXT SEARCH DICTIONARY my\_synonym (   
 TEMPLATE = synonym,   
 SYNONYMS = my\_synonyms,   
 FILEPATH = 'file:///home/dicts/'   
);

3、修改词典。

ALTER TEXT SEARCH CONFIGURATION english   
 ALTER MAPPING FOR asciiword   
 WITH my\_synonym, english\_stem;

4、 查询验证。

SELECT \* FROM ts\_debug('english', 'Paris');

结果显示如下：

alias | description | token | dictionaries | dictionary | lexemes   
-----------+-----------------+-------+---------------------------+------------+---------   
 asciiword | Word, all ASCII | Paris | {my\_synonym,english\_stem} | my\_synonym | {paris}   
(1 row)

5、 查询验证。

SELECT \* FROM ts\_debug('english', 'paris');

结果显示如下：

alias | description | token | dictionaries | dictionary | lexemes   
-----------+-----------------+-------+---------------------------+------------+---------   
 asciiword | Word, all ASCII | Paris | {my\_synonym,english\_stem} | my\_synonym | {paris}   
(1 row)

6、修改词典。

ALTER TEXT SEARCH DICTIONARY my\_synonym ( CASESENSITIVE=true);

7、查询验证。

SELECT \* FROM ts\_debug('english', 'Paris');

结果显示如下：

alias | description | token | dictionaries | dictionary | lexemes   
-----------+-----------------+-------+---------------------------+------------+---------   
 asciiword | Word, all ASCII | Paris | {my\_synonym,english\_stem} | my\_synonym | {paris}   
(1 row)

8、查询验证 。

SELECT \* FROM ts\_debug('english', 'paris');

结果显示如下：

alias | description | token | dictionaries | dictionary | lexemes   
----------+-----------------+-------+---------------------------+------------+---------   
 asciiword | Word, all ASCII | Paris | {my\_synonym,english\_stem} | my\_synonym | {pari}   
(1 row)

其中，同义词词典文件全名为my\_synonyms.syn，所在目录为当前连接数据库主节点的/home/dicts/（以实际路径为准）下。关于创建词典的语法和更多参数，请参见章节“ALTER TEXT SEARCH DICTIONARY”。

* 星号（\*）可用于词典文件中的同义词结尾，表示该同义词是一个前缀。在to\_tsvector()中该星号将被忽略，但在to\_tsquery()中会匹配该前缀并对应输出结果（参照章节“附加功能->处理查询”）。

假设词典文件synonym\_sample.syn内容如下：

postgres pgsql   
postgresql pgsql   
postgre pgsql   
gogle googl   
indices index\*

1、创建并使用词典。

CREATE TEXT SEARCH DICTIONARY syn (   
 TEMPLATE = synonym,   
 SYNONYMS = synonym\_sample   
);

2、验证结果。

SELECT ts\_lexize('syn','indices');

结果返回如下：

ts\_lexize   
-----------   
 {index}   
(1 row)

3、创建词典并修改。

CREATE TEXT SEARCH CONFIGURATION tst (copy=simple);   
ALTER TEXT SEARCH CONFIGURATION tst ALTER MAPPING FOR asciiword WITH syn;

4、验证结果。

SELECT to\_tsvector('tst','indices');

结果返回如下：

to\_tsvector   
-------------   
 'index':1   
(1 row)

SELECT to\_tsquery('tst','indices');

结果返回如下：

to\_tsquery   
------------   
 'index':\*   
(1 row)

SELECT 'indexes are very useful'::tsvector;

结果返回如下：

tsvector   
---------------------------------   
 'are' 'indexes' 'useful' 'very'   
(1 row)

SELECT 'indexes are very useful'::tsvector @@ to\_tsquery('tst','indices');

结果返回如下：

?column?   
----------   
 t   
(1 row)

#### Thesaurus词典

**功能描述**

Thesaurus词典，也叫做分类词典（缩写为TZ），是一组定义了词以及词组间关系的集合，包括广义词（BT）、狭义词（NT）、首选词、非首选词、相关词等。根据词典文件中的定义，TZ词典用一个指定的短语替换对应匹配的所有短语，并且可选择保留原始短语进行索引。TZ词典实际上是Synonym词典的一个扩展，增加了短语支持。

**注意事项**

* 由于TZ词典需要识别短语，所以在处理过程中必须保存当前状态并与解析器进行交互，以决定是否处理下一个token或是结束当前识别。此外，TZ词典配置时需谨慎，如果设置TZ词典仅处理asciiword类型的token，则类似one 7的分类词典定义将不会生效，因为uint类型的token不会传给TZ词典处理。
* 在索引期间要用到分类词典，因此分类词典参数中的任何变化都要求重新索引。对于其他大多数类型的词典来说，类似添加或删除停用词这种修改并不需要强制重新索引。

**示例**

1、创建一个名为thesaurus\_astro的TZ词典。

以一个简单的天文学词典thesaurus\_astro为例，其中定义了两组天文短语及其同义词如下：

supernovae stars : sn   
crab nebulae : crab

执行如下语句创建TZ词典：

CREATE TEXT SEARCH DICTIONARY thesaurus\_astro (   
 TEMPLATE = thesaurus,   
 DictFile = thesaurus\_astro,   
 Dictionary = pg\_catalog.english\_stem,   
 FILEPATH = 'file:///home/dicts/'   
);

其中，词典定义文件全名为thesaurus\_astro.ths，所在目录为当前连接数据库主节点的/home/dicts/下 。子词典pg\_catalog.english\_stem是预定义的Snowball类型的英语词干词典，用于规范化输入词，子词典自身相关配置（例如停用词等）不在此处显示。关于创建词典的语法和更多参数，请参考章节“CREATE TEXT SEARCH DICTIONARY”。

2、创建词典后，将其绑定到对应文本搜索配置中需要处理的token类型上：

ALTER TEXT SEARCH CONFIGURATION russian   
 ALTER MAPPING FOR asciiword, asciihword, hword\_asciipart   
 WITH thesaurus\_astro, english\_stem;

3、使用TZ词典。

* 测试TZ词典。

ts\_lexize函数对于测试TZ词典作用不大，因为该函数是按照单个token处理输入。可以使用plainto\_tsquery、to\_tsvector、to\_tsquery函数测试TZ词典，这些函数能够将输入分解成多个token（to\_tsquery函数需要将输入加上引号）。

SELECT plainto\_tsquery('russian','supernova star');

结果显示如下：

plainto\_tsquery   
-----------------   
 'sn'   
(1 row)

SELECT to\_tsvector('russian','supernova star');

结果显示如下：

to\_tsvector   
-------------   
 'sn':1   
(1 row)

SELECT to\_tsquery('russian','''supernova star''');

结果显示如下：

to\_tsquery   
------------   
 'sn'   
(1 row)

其中，supernova star匹配了词典thesaurus\_astro定义中的supernovae stars，这是因为在thesaurus\_astro词典定义中指定了Snowball类型的子词典english\_stem，该词典移除了e和s。

* 如果同时需要索引原始短语，只要将其同时放置在词典定义文件中对应定义的右侧即可（在thesaurus\_astro文件中修改），如下：

supernovae stars : sn supernovae stars

数据库中执行以下语句修改词典并查询验证。

ALTER TEXT SEARCH DICTIONARY thesaurus\_astro (   
 DictFile = thesaurus\_astro,   
 FILEPATH = 'file:///home/dicts/');   
 SELECT plainto\_tsquery('russian','supernova star');

结果返回如下：

plainto\_tsquery   
-----------------------------   
 'sn' & 'supernova' & 'star'   
(1 row)

#### 词典测试

**功能描述**

函数ts\_lexize用于进行词典测试。

**注意事项**

ts\_lexize(dict regdictionary, token text) returns text[]如果输入的token可以被词典识别，那么ts\_lexize返回词素的数组；如果token可以被词典识别到它是一个停用词，则返回空数组；如果是一个不可识别的词则返回NULL。

**示例**

SELECT ts\_lexize('english\_stem', 'stars');

结果返回如下：

ts\_lexize   
-----------   
 {star}

查询不可识别值。

SELECT ts\_lexize('english\_stem', 'a');

结果返回如下：

ts\_lexize   
-----------   
 {}

#### Ispell词典

**功能描述**

Ispell词典模板支持词法词典，它可以把一个词的各种语言学形式规范化成相同的词位。比如，一个Ispell英语词典可以匹配搜索词bank的词尾变化和词形变化，如banking、banked、banks、banks'和bank's等。

Vastbase不提供任何预定义的Ispell类型词典或词典文件。dict文件和affix文件支持多种开源词典格式，包括Ispell、MySpell和Hunspell等。

**示例**

1、获取词典定义文件和词缀文件。

用户可以使用开源词典，直接获取的开源词典后缀名可能为.aff和.dic，此时需要将扩展名改为.affix和.dict。此外，对于某些词典文件，还需要使用下面的命令把字符转换成 UTF-8 编码，比如挪威语词典：

iconv -f ISO\_8859-1 -t UTF-8 -o nn\_no.affix nn\_NO.aff   
iconv -f ISO\_8859-1 -t UTF-8 -o nn\_no.dict nn\_NO.dic

2、创建Ispell词典。

CREATE TEXT SEARCH DICTIONARY norwegian\_ispell (   
 TEMPLATE = ispell,   
 DictFile = nn\_no,   
 AffFile = nn\_no,   
 FilePath = 'file:///home/dicts'   
);

其中，词典文件全名为nn\_no.dict和nn\_no.affix，所在目录为当前连接数据库主节点的/home/dicts/下 。关于创建词典的语法和更多参数，请参见章节“CREATE TEXT SEARCH DICTIONARY”。

3、使用Ispell词典进行复合词拆分。

SELECT ts\_lexize('norwegian\_ispell', 'sjokoladefabrikk');

结果显示如下：

ts\_lexize   
---------------------   
 {sjokolade,fabrikk}   
(1 row)

MySpell不支持复合词，Hunspell对复合词有较好的支持。Vastbase仅持Hunspell中基本的复合词操作。通常情况下，Ispell词典能够识别的词是一个有限集合，其后应该配置一个更广义的词典，例如一个可以识别所有词的Snowball词典。

#### 配置示例

文本搜索配置（Text Search Configuration），指定了将文档转换成tsvector过程中所必需的组件：

* 解析器，用于把文本分解成标记token；
* 词典列表，用于将每个token转换成词位lexeme。

每次to\_tsvector或to\_tsquery函数调用时，都需要指定一个文本搜索配置来指定具体的处理过程。GUC参数章节“default\_text\_search\_config”指定了默认的文本搜索配置，当文本搜索函数中没有显式指定文本搜索配置参数时，将会使用该默认值进行处理。

Vastbase中预定义有一些可用的文本搜索配置，用户也可创建自定义的文本搜索配置。此外，为了便于管理文本搜索对象，还提供有多个vsql元命令，可以显示有关文本搜索对象的信息（详细请参见《工具参考》中“vsql”章节）。

**操作步骤**

1、创建一个文本搜索配置ts\_conf，复制预定义的文本搜索配置english。

CREATE TEXT SEARCH CONFIGURATION ts\_conf ( COPY = pg\_catalog.english );

2、创建Synonym词典。

假设同义词词典定义文件pg\_dict.syn内容如下：

postgres pg   
pgsql pg   
postgresql pg

执行如下语句创建Synonym词典：

CREATE TEXT SEARCH DICTIONARY pg\_dict (   
 TEMPLATE = synonym,   
 SYNONYMS = pg\_dict,   
 FILEPATH = 'file:///home/dicts'   
 );

3、创建一个Ispell词典english\_ispell（词典定义文件来自开源词典）。

CREATE TEXT SEARCH DICTIONARY english\_ispell (   
 TEMPLATE = ispell,   
 DictFile = english,   
 AffFile = english,   
 StopWords = english,   
 FILEPATH = 'file:///home/dicts'   
);

4、设置文本搜索配置ts\_conf，修改某些类型的token对应的词典列表。关于token类型的详细信息，请参见章节“解析器”。

ALTER TEXT SEARCH CONFIGURATION ts\_conf   
 ALTER MAPPING FOR asciiword, asciihword, hword\_asciipart,   
 word, hword, hword\_part   
 WITH pg\_dict, english\_ispell, english\_stem;

5、在文本搜索配置中，选择设置不索引或搜索某些token类型。

ALTER TEXT SEARCH CONFIGURATION ts\_conf   
 DROP MAPPING FOR email, url, url\_path, sfloat, float;

6、使用文本检索调测函数ts\_debug()对所创建的词典配置ts\_conf进行测试。

SELECT \* FROM ts\_debug('ts\_conf', '   
PostgreSQL, the highly scalable, SQL compliant, open source object-relational   
database management system, is now undergoing beta testing of the next   
version of our software.   
');

7、可以设置当前session使用ts\_conf作为默认的文本搜索配置。此设置仅在当前session有效。

\dF+ ts\_conf

结果返回如下：

Text search configuration "public.ts\_conf"   
Parser: "pg\_catalog.default"   
 Token | Dictionaries   
-----------------+-------------------------------------   
 asciihword | pg\_dict,english\_ispell,english\_stem   
 asciiword | pg\_dict,english\_ispell,english\_stem   
 file | simple   
 host | simple   
 hword | pg\_dict,english\_ispell,english\_stem   
 hword\_asciipart | pg\_dict,english\_ispell,english\_stem   
 hword\_numpart | simple   
 hword\_part | pg\_dict,english\_ispell,english\_stem   
 int | simple   
 numhword | simple   
 numword | simple   
 uint | simple   
 version | simple   
 word | pg\_dict,english\_ispell,english\_stem

SET default\_text\_search\_config = 'public.ts\_conf';   
SHOW default\_text\_search\_config;   
 default\_text\_search\_config   
----------------------------   
 public.ts\_conf   
(1 row)

#### 停用词

**功能描述**

停用词是很常见的词，几乎出现在每一个文档中，并且没有区分值。因此，在全文搜索的语境下可忽视它们。停用词处理逻辑和词典类型相关。例如，Ispell词典会先对标记进行规范化，然后再查看停用词表，而Snowball词典会最先检查输入标记是否为停用词。

**示例**

* 每个英文文本包含像a和the的单词，因此没必要将它们存储在索引中。然而，停用词影响tsvector中的位置，同时位置也会影响相关度。

SELECT to\_tsvector('english','in the list of stop words');

结果显示如下：

to\_tsvector   
----------------------------   
 'list':3 'stop':5 'word':6

* 位置1、2、4是停用词，所以不显示。为包含和不包含停用词的文档计算出的排序是完全不同的：

vastbase=# SELECT ts\_rank\_cd (to\_tsvector('english','in the list of stop words'), to\_tsquery('list & stop'));   
  
 ts\_rank\_cd   
------------   
 .05

vastbase=# SELECT ts\_rank\_cd (to\_tsvector('english','list stop words'), to\_tsquery('list & stop'));   
  
 ts\_rank\_cd   
------------   
 .1

### 附加功能

#### 查询重写

**功能描述**

ts\_rewrite函数族可以从tsquery中搜索一个特定的目标子查询，并在该子查询每次出现的地方都替换为另一个子查询。 实际上这只是通过字串替换而得到的一个特定tsquery版本。目标子查询和替换查询组合起来可以被认为是一个重写规则。一组类似的重写规则可以为搜索提供强大的帮助。例如，可以使用同义词扩大搜索范围（例如，new york, big apple, nyc, gotham）或限制搜索范围在用户直接感兴趣的热点话题上。

**语法格式**

* ts\_rewrite的这种形式只适用于一个单一的重写规则：任何出现目标子查询的地方都被无条件替换。

ts\_rewrite (query tsquery, target tsquery, substitute tsquery) returns tsquery

* ts\_rewrite的这种形式接受一个起始查询和SQL查询命令。 这里的查询命令是文本字串形式，必须产生两个tsquery列。查询结果的每一行，第一个字段的值（目标子查询） 都会被第二个字段（替代子查询）替换。

ts\_rewrite (query tsquery, select text) returns tsquery

当多个规则需要重写时，重写顺序非常重要； 因此在实践中需要使用ORDER BY将源查询按照某些字段进行排序。

**示例**

举一个现实生活中天文学上的例子。我们将使用表驱动的重写规则扩大supernovae的查询范围：

1、创建测试表并插入数据。

CREATE TABLE tsearch.aliases (id int, t tsquery, s tsquery);   
INSERT INTO tsearch.aliases VALUES(1, to\_tsquery('supernovae'), to\_tsquery('supernovae|sn'));

2、查询数据。

SELECT ts\_rewrite(to\_tsquery('supernovae & crab'), 'SELECT t, s FROM tsearch.aliases');

结果返回如下：

ts\_rewrite   
---------------------------------   
 'crab' & ( 'supernova' | 'sn' )

3、可以通过更新表修改重写规则。

UPDATE tsearch.aliases   
SET s = to\_tsquery('supernovae|sn & !nebulae')   
WHERE t = to\_tsquery('supernovae');

4、查询数据。

SELECT ts\_rewrite(to\_tsquery('supernovae & crab'), 'SELECT t, s FROM tsearch.aliases');

结果显示如下：

ts\_rewrite   
---------------------------------------------   
 'crab' & ( 'supernova' | 'sn' & !'nebula' )

5、需要重写的规则越多，重写操作就越慢。因为它要检查每一个可能匹配的规则。为了过滤明显的非候选规则，可以使用tsquery类型的操作符来实现。在下面的例子中， 我们只选择那些可能与原始查询匹配的规则。

SELECT ts\_rewrite('a & b'::tsquery, 'SELECT t,s FROM tsearch.aliases WHERE ''a & b''::tsquery @> t');

结果显示如下：

ts\_rewrite   
------------   
 'b' & 'a'   
(1 row)

#### 处理tsvector

Vastbase提供了用来操作tsvector类型的函数和操作符。

* tsvector || tsvector
* tsvector连接操作符返回一个新的tsvector类型，它综合了两个tsvector中词素和位置信息，并保留词素的位置信息和权重标签。右侧的tsvector的起始位置位于左侧tsvector的最后位置，因此，新生成的tsvector几乎等同于将两个原始文档字串连接后进行to\_tsvector操作。（这个等价是不准确的，因为任何从左边tsvector中删除的停用词都不会影响结果，但是，在使用文本连接时，则会影响词素在右侧tsvector中的位置。）  
  相较于对文本进行连接后再执行to\_tsvector操作，使用tsvector类型进行连接操作的优势在于，可以对文档的不同部分使用不同配置进行解析。因为setweight函数会对给定的tsvector中的语素进行统一设置，如果想要对文档的不同部分设置不同的权重，需要在连接之前对文本进行解析和权重设置。
* setweight(vector tsvector, weight "char") returns tsvector
* setweight返回一个输入tsvector的副本，其中每一个位置都使用给定的权重做了标记。权值可以为A、B、C或D（D是tsvector副本的默认权重，并且不在输出中呈现）。当对tsvector进行连接操作时，这些权重标签将会被保留，文档不同部分以不同的权重进行排序。
* length(vector tsvector) returns integer
* 返回vector中的词素的数量。
* strip(vector tsvector) returns tsvector
* 返回一个tsvector类型，其中包含输入的tsvector的同义词，但不包含任何位置和权重信息。虽然在相关性排序中，这里返回的tsvector要比未拆分的tsvector的作用小很多，但它通常都比未拆分的tsvector小的多。

**注意事项**

权重标签作用于位置，而不是词素。如果传入的tsvector已经被剥离了位置信息，那么setweight函数将什么都不做。

#### 处理查询

Vastbase提供了函数和操作符用来操作tsquery类型的查询。

* tsquery && tsquery

返回两个给定查询tsquery的与结果。

* tsquery || tsquery

返回两个给定查询tsquery的或结果。

* !! tsquery

返回给定查询tsquery的非结果。

* numnode(query tsquery) returns integer

返回tsquery中的节点数目（词素加操作符），这个函数在检查查询是否有效（返回值大于0），或者只包含停用词（返回值等于0）时，是有用的。例如：

SELECT numnode(plainto\_tsquery('the any'));

结果显示如下：

NOTICE: text-search query contains only stop words or doesn't contain lexemes, ignored   
CONTEXT: referenced column: numnode   
 numnode   
---------   
 0

SELECT numnode('foo & bar'::tsquery);

结果显示如下：

numnode   
---------   
 3

* querytree(query tsquery) returns text

返回可用于索引搜索的tsquery部分，该函数对于检测非索引查询是有用的（例如只包含停用词或否定项）。例如:

SELECT querytree(to\_tsquery('!defined'));

结果显示如下

querytree   
-----------   
 T   
(1 row)

#### 收集文献统计

**功能描述**

函数ts\_stat可用于检查配置和查找候选停用词。

**语法格式**

ts\_stat(sqlquery text, [ weights text, ]  
 OUT word text, OUT ndoc integer,  
 OUT nentry integer) returns setof record

**参数说明**

* sqlquery：是一个包含SQL查询语句的文本，该SQL查询将返回一个tsvector。
* ts\_stat执行SQL查询语句并返回一个包含tsvector中每一个不同的语素（词）的统计信息。返回信息包括：
* word text：词素。
* ndoc integer：词素在文档（tsvector）中的编号。
* nentry integer：词素出现的频率。

**示例**

1、创建测试表并插入数据。

DROP SCHEMA IF EXISTS tsearch CASCADE;  
CREATE SCHEMA tsearch;  
CREATE TABLE tsearch.pgweb(id int, body text, title text, last\_mod\_date date);  
  
INSERT INTO tsearch.pgweb VALUES(1, 'China, officially the People''s Republic of China (PRC), located in Asia, is the world''s most populous state.', 'China', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(2, 'America is a rock band, formed in England in 1970 by multi-instrumentalists Dewey Bunnell, Dan Peek, and Gerry Beckley.', 'America', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(3, 'England is a country that is part of the United Kingdom. It shares land borders with Scotland to the north and Wales to the west.', 'England', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(4, 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania, and numerous smaller islands.', 'Australia', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(6, 'Japan is an island country in East Asia.', 'Japan', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(7, 'Germany, officially the Federal Republic of Germany, is a sovereign state and federal parliamentary republic in central-western Europe.', 'Germany', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(8, 'France, is a sovereign state comprising territory in western Europe and several overseas regions and territories.', 'France', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(9, 'Italy officially the Italian Republic, is a unitary parliamentary republic in Europe.', 'Italy', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(10, 'India, officially the Republic of India, is a country in South Asia.', 'India', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(11, 'Brazil, officially the Federative Republic of Brazil, is the largest country in both South America and Latin America.', 'Brazil', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(12, 'Canada is a country in the northern half of North America.', 'Canada', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(13, 'Mexico, officially the United Mexican States, is a federal republic in the southern part of North America.', 'Mexico', '2010-1-1');

2、如果设置了权重条件，只有标记了对应权重的词素才会统计频率。例如，在一个文档集中检索使用频率最高的十个单词。

SELECT \* FROM ts\_stat('SELECT to\_tsvector(''english'',title) FROM tsearch.pgweb WHERE id < 10') ORDER BY nentry DESC, ndoc DESC, word LIMIT 10;

结果显示如下：

word | ndoc | nentry   
-----------+------+--------  
 america | 1 | 1  
 australia | 1 | 1  
 china | 1 | 1  
 england | 1 | 1  
 franc | 1 | 1  
 germani | 1 | 1  
 itali | 1 | 1  
 japan | 1 | 1  
(8 rows)

3、同样的情况，但是只计算权重为A或者B的单词使用频率。

SELECT \* FROM ts\_stat('SELECT to\_tsvector(''english'', title) FROM tsearch.pgweb WHERE id < 10', 'a') ORDER BY nentry DESC, ndoc DESC, word LIMIT 10;

结果显示如下：

word | ndoc | nentry   
------+------+--------  
(0 rows)

### 控制文本搜索

#### 高亮搜索结果

**功能描述**

搜索结果的理想显示是：列出每篇文档中与搜索相关的部分，并标识为什么与查询相关。搜索引擎能够显示标识了搜索词的文档片段。Vastbase提供了函数ts\_headline支持这部分功能。

**语法格式**

ts\_headline([ config regconfig, ] document text, query tsquery [, options text ]) returns text

**参数说明**

* config：ts\_headline的输入是带有查询条件的文档，其返回文档中的摘录，在摘录中查询词是高亮显示的。用来解析文档的分词器由config参数指定。如果省略config，则使用default\_text\_search\_config的值所指定的分词器。
* options：指定options字符串时，需由一个或多个option=value对组成，且必须用逗号分隔。options可以是下面的选项：
* StartSel，StopSel：分隔文档中出现的查询词，以区别于其他摘录词。当包含有空格或逗号时，必须用双引号将字符串引起来。
* MaxWords，MinWords：定义摘录的最长和最短值。
* ShortWord：在摘录的开始和结束会丢弃此长度或更短的词。默认值3会消除常见的英语冠词。
* HighlightAll：布尔标志。如果为真，整个文档将作为摘录。忽略前面三个参数的值。
* MaxFragments：要显示的文本摘录或片段的最大数量。默认值0表示选择非片段的摘录生成方法。 大于0的值表示选择基于片段的摘录生成。此方法查找带有尽可能多查询词的文本片段，并显示查询词周围的上下文片段。因此，查询词临近每个片段的中间，且查询词两边都有词。每个片段至多有MaxWords，并且长度为ShortWord或更短的词在每一个片段开始和结束被丢弃。如果在文档中没有找到所有的查询词，则文档中开头将显示MinWords单片段。
* 不声明选项时，采用下面的缺省值：

StartSel=<b>, StopSel=</b>,  
MaxWords=35, MinWords=15, ShortWord=3, HighlightAll=FALSE,  
MaxFragments=0, FragmentDelimiter=" ... "

**示例**

SELECT ts\_headline('english',   
'The most common type of search   
is to find all documents containing given query terms   
and return them in order of their similarity to the   
query.',   
to\_tsquery('english', 'query & similarity'));

结果返回如下

ts\_headline   
------------------------------------------------------------   
  
 containing given <b>query</b> terms   
 and return them in order of their <b>similarity</b> to the   
 <b>query</b>.   
(1 row)

ts\_headline使用原始文档，而不是tsvector摘录，因此使用起来会慢，应慎重使用。

#### 解析查询

**功能描述**

Vastbase提供了函数to\_tsquery和plainto\_tsquery将查询转换为tsquery数据类型，to\_tsquery提供比plainto\_tsquery更多的功能，但对其输入要求更严格。

**语法格式**

to\_tsquery([ config regconfig, ] querytext text) returns tsquery

**注意事项**

to\_tsquery从querytext中创建一个tsquery，querytext必须由布尔运算符& (AND)，| (OR)和! (NOT)分割的单个token组成。这些运算符可以用圆括弧分组。换句话说，to\_tsquery输入必须遵循tsquery输入的通用规则，具体请参见“SQL语法参考->数据类型->文本搜索类型”章节。不同的是基本tsquery以token表面值作为输入，而to\_tsquery使用指定或默认分词器将每个token标准化成词素，并依据分词器丢弃属于停用词的token。

**示例**

基本tsquery以token表面值作为输入，而to\_tsquery使用指定或默认分词器将每个token标准化成词素，并依据分词器丢弃属于停用词的token。比如：

SELECT to\_tsquery('english', 'The & Fat & Rats');   
  
to\_tsquery   
---------------   
 'fat' & 'rat'   
(1 row)

像在基本tsquery中的输入一样，weight(s)可以附加到每个词素来限制它只匹配那些有相同weight(s)的tsvector词素。比如：

SELECT to\_tsquery('english', 'Fat | Rats:AB');   
  
 to\_tsquery   
------------------   
 'fat' | 'rat':AB   
(1 row)

同时，也可以附加到词素来指定前缀匹配：

SELECT to\_tsquery('supern:\*A & star:A\*B');   
  
to\_tsquery   
--------------------------   
 'supern':\*A & 'star':\*AB   
(1 row)

这样的词素将匹配tsquery中指定字符串和权重的项。

plainto\_tsquery([ config regconfig, ] querytext text) returns tsquery

plainto\_tsquery将未格式化的文本querytext变换为tsquery。类似于to\_tsvector，文本被解析并且标准化，然后在存在的词之间插入&(AND)布尔算子。比如：

SELECT plainto\_tsquery('english', 'The Fat Rats');   
  
 plainto\_tsquery   
-----------------   
'fat' & 'rat'   
(1 row)

请注意，plainto\_tsquery无法识别布尔运算符、权重标签，或在其输入中的前缀匹配标签：

SELECT plainto\_tsquery('english', 'The Fat & Rats:C');   
  
 plainto\_tsquery   
---------------------   
 'fat' & 'rat' & 'c'   
(1 row)

在这里，所有输入的标点符号作为空格符号丢弃。

#### 解析文档

**功能描述**

Vastbase中提供了to\_tsvector函数把文档处理成tsvector数据类型。

**语法格式**

to\_tsvector([ config regconfig, ] document text) returns tsvector

**to\_tsvector函数**

to\_tsvector将文本文档解析为token，再将token简化到词素，并返回一个tsvector。其中tsvector中列出了词素及它们在文档中的位置。文档是根据指定的或默认的文本搜索分词器进行处理的。

to\_tsvector函数内部调用一个解析器，将文档的文本分解成token并给每个token指定一个类型。对于每个token，有一系列词典可供查询。词典系列因token类型的不同而不同。识别token的第一本词典将发出一个或多个标准词素来表示token。例如：

* rats变成rat因为词典认为词rats是rat的复数形式。
* 有些词被作为停用词（请参考“全文检索->词典->停用词”章节），这样它们就会被忽略，因为它们出现得太过频繁以致于搜索中没有用处。比如例子中的a、on和it。
* 如果没有词典识别token，那么它也被忽略。在这个例子中，符号“-”被忽略，因为词典没有给它分配token类型（空间符号），即空间记号永远不会被索引。

语法解析器、词典和要索引的token类型由选定的文本搜索分词器决定。可以在同一个数据库中有多种不同的分词器，以及提供各种语言的预定义分词器。在以上例子中，使用缺省分词器english。

函数setweight可以给tsvector的记录加权重，权重是字母A、B、C、D之一。这通常用于标记来自文档不同部分的记录，比如标题、正文。之后，这些信息可以用于排序搜索结果。

因为to\_tsvector(NULL)会返回空，当字段可能是空的时候，建议使用coalesce。

**示例**

* 示例1：结果tsvector不包含词a、on或者it，rats变成rat，并且忽略标点符号-。

SELECT to\_tsvector('english', 'a fat cat sat on a mat - it ate a fat rats');

结果显示如下：

to\_tsvector   
-----------------------------------------------------  
 'ate':9 'cat':3 'fat':2,11 'mat':7 'rat':12 'sat':4  
(1 row)

* 示例2：结构化文档创建tsvector的方法：

CREATE TABLE tsearch.tt (id int, title text, keyword text, abstract text, body text, ti tsvector);   
  
INSERT INTO tsearch.tt(id, title, keyword, abstract, body) VALUES (1, 'China', 'Beijing', 'China','China, officially the People''s Republic of China (PRC), located in Asia, is the world''s most populous state.');   
  
UPDATE tsearch.tt SET ti =   
 setweight(to\_tsvector(coalesce(title,'')), 'A') ||   
 setweight(to\_tsvector(coalesce(keyword,'')), 'B') ||   
 setweight(to\_tsvector(coalesce(abstract,'')), 'C') ||   
 setweight(to\_tsvector(coalesce(body,'')), 'D');

结果显示如下：

id | title | keyword | abstract | body |   
 ti   
----+-------+---------+----------+--------------------------------------------------------------------------------------------------------------+--------------------------------------  
---------------------------------------------------------------------------------------  
 1 | China | Beijing | China | China, officially the People's Republic of China (PRC), located in Asia, is the world's most populous state. | 'asia':15 'beij':2B 'china':1A,3C,4,1  
1 'locat':13 'offici':5 'peopl':7 'popul':21 'prc':12 'republ':9 'state':22 'world':18

#### 排序查询结果

**功能描述**

排序试图针对特定查询衡量文档的相关度，从而将众多的匹配文档中相关度最高的文档排在最前。Vastbase提供了两个预置的排序函数。函数考虑了词法、距离和结构信息；也就是，他们考虑查询词在文档中出现的频率、紧密程度以及他们出现的地方在文档中的重要性。然而，相关性的概念是模糊的，并且是跟应用强相关的。不同的应用程序可能需要额外的信息来排序，比如，文档的修改时间、内置的排序函数等。也可以开发自己的排序函数或者采用附加因素组合这些排序函数的结果来满足特定需求。

**预置排序函数**

ts\_rank([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ]) returns float4

基于词素匹配率对vector进行排序：

ts\_rank\_cd([ weights float4[], ] vector tsvector, query tsquery [, normalization integer ]) returns float4

该函数需要位置信息的输入。因此它不能在“剥离”tsvector值的情况下运行—它将总是返回零。

**权重**

* 对于这两个函数，可选的weights参数提供给词加权重的能力，词的权重大小取决于所加的权值。权重阵列指定在排序时为每类词汇加多大的权重。如果没有提供weights，则使用缺省值：{0.1, 0.2, 0.4, 1.0}。

{D-weight, C-weight, B-weight, A-weight}

* 通常的权重是用来标记文档特殊领域的词，如标题或最初的摘要，所以相对于文章主体中的词它们有着更高或更低的重要性。
* 由于较长的文档有更多的机会包含查询词，因此有必要考虑文档的大小。例如，包含有5个搜索词的一百字文档比包含有5个搜索词的一千字文档相关性更高。两个预置的排序函数都采用了一个整型的标准化选项来定义文档长度是否影响排序及如何影响。这个整型选项控制多个行为，所以它是一个屏蔽字：可以使用|指定一个或多个行为（例如，2|4）。
* 0（缺省）表示：跟长度大小没有关系
* 1 表示：排名（rank）除以（文档长度的对数+1）
* 2表示：排名除以文档的长度
* 4表示：排名除以两个扩展词间的调和平均距离。只能使用ts\_rank\_cd实现
* 8表示：排名除以文档中单独词的数量
* 16表示：排名除以单独词数量的对数+1
* 32表示：排名除以排名本身+1
* 当指定多个标志位时，会按照所列的顺序依次进行转换。

**注意事项**

需要特别注意的是，排序函数不使用任何全局信息，所以不可能产生一个某些情况下需要的1%或100%的理想标准值。标准化选项32 (rank/(rank+1))可用于所有规模的从零到一之间的排序，当然，这只是一个表面变化；它不会影响搜索结果的排序。

**示例**

创建测试表并插入数据。

DROP SCHEMA IF EXISTS tsearch CASCADE;  
CREATE SCHEMA tsearch;  
CREATE TABLE tsearch.pgweb(id int, body text, title text, last\_mod\_date date);  
  
INSERT INTO tsearch.pgweb VALUES(1, 'China, officially the People''s Republic of China (PRC), located in Asia, is the world''s most populous state.', 'China', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(2, 'America is a rock band, formed in England in 1970 by multi-instrumentalists Dewey Bunnell, Dan Peek, and Gerry Beckley.', 'America', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(3, 'England is a country that is part of the United Kingdom. It shares land borders with Scotland to the north and Wales to the west.', 'England', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(4, 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania, and numerous smaller islands.', 'Australia', '2010-1-1');   
INSERT INTO tsearch.pgweb VALUES(6, 'Japan is an island country in East Asia.', 'Japan', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(7, 'Germany, officially the Federal Republic of Germany, is a sovereign state and federal parliamentary republic in central-western Europe.', 'Germany', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(8, 'France, is a sovereign state comprising territory in western Europe and several overseas regions and territories.', 'France', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(9, 'Italy officially the Italian Republic, is a unitary parliamentary republic in Europe.', 'Italy', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(10, 'India, officially the Republic of India, is a country in South Asia.', 'India', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(11, 'Brazil, officially the Federative Republic of Brazil, is the largest country in both South America and Latin America.', 'Brazil', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(12, 'Canada is a country in the northern half of North America.', 'Canada', '2010-1-1');  
INSERT INTO tsearch.pgweb VALUES(13, 'Mexico, officially the United Mexican States, is a federal republic in the southern part of North America.', 'Mexico', '2010-1-1');

* 仅选择排名前十的匹配：

SELECT id, title, ts\_rank\_cd(to\_tsvector(body), query) AS rank   
FROM tsearch.pgweb, to\_tsquery('america') query   
WHERE query @@ to\_tsvector(body)   
ORDER BY rank DESC   
LIMIT 10;

结果显示如下：

id | title | rank   
----+---------+------   
 11 | Brazil | .2   
 2 | America | .1   
 12 | Canada | .1   
 13 | Mexico | .1   
(4 rows)

* 使用标准化排序的相同例子：

SELECT id, title, ts\_rank\_cd(to\_tsvector(body), query, 32 /\* rank/(rank+1) \*/ ) AS rank   
FROM tsearch.pgweb, to\_tsquery('america') query   
WHERE query @@ to\_tsvector(body)   
ORDER BY rank DESC   
LIMIT 10;

结果显示如下

id | title | rank   
----+---------+----------   
 11 | Brazil | .166667   
 2 | America | .0909091   
 12 | Canada | .0909091   
 13 | Mexico | .0909091   
(4 rows)

* 使用中文分词法排序查询的例子：

1、创建测试表并插入数据。

CREATE TABLE ts\_ngram(id int, body text);   
INSERT INTO ts\_ngram VALUES(1, '中文');   
INSERT INTO ts\_ngram VALUES(2, '中文检索');   
INSERT INTO ts\_ngram VALUES(3, '检索中文');

2、精确匹配 。

SELECT id, body, ts\_rank\_cd(to\_tsvector('ngram',body), query) AS rank FROM ts\_ngram, to\_tsquery('中文') query WHERE query @@ to\_tsvector(body);

结果显示如下：

id | body | rank   
----+------+------   
 1 | 中文 | .1   
(1 row)

3、模糊匹配

SELECT id, body, ts\_rank\_cd(to\_tsvector('ngram',body), query) AS rank FROM ts\_ngram, to\_tsquery('中文') query WHERE query @@ to\_tsvector('ngram',body);

结果显示如下：

id | body | rank   
----+----------+------   
 3 | 检索中文 | .1   
 1 | 中文 | .1   
 2 | 中文检索 | .1   
(3 rows)

排序要遍历每个匹配的tsvector，因此资源消耗多，可能会因为I/O限制导致排序慢。可是这是很难避免的，因为实际查询中通常会有大量的匹配。

### 解析器

**功能描述**

文本搜索解析器负责将原文档文本分解为多个token，并标识每个token的类型。这里的类型集由解析器本身定义。注意，解析器并不修改文本，它只是确定合理的单词边界。由于这一限制，人们更需要定制词典，而不是为每个应用程序定制解析器。

目前Vastbase提供了三个内置的解析器，分别为pg\_catalog.default、pg\_catalog.ngram、pg\_catalog.pound，其中pg\_catalog.default适用于英文分词场景，pg\_catalog.ngram、pg\_catalog.pound是为了支持中文全文检索功能新增的两种解析器，适用于中文及中英混合分词场景。

内置解析器pg\_catalog.default，它能识别23种token类型，如下表所示：

| **别名** | **描述** | **示例** |
| --- | --- | --- |
| asciiword | Word, all ASCII letters | elephant |
| word | Word, all letters | mañana |
| numword | Word, letters and digits | beta1 |
| asciihword | Hyphenated word, all ASCII | up-to-date |
| hword | Hyphenated word, all letters | lógico-matemática |
| numhword | Hyphenated word, letters and digits | openGauss-beta1 |
| hword\_asciipart | Hyphenated word part, all ASCII | openGauss in the context openGauss-beta1 |
| hword\_part | Hyphenated word part, all letters | lógico or matemática in the context lógico-matemática |
| hword\_numpart | Hyphenated word part, letters and digits | beta1 in the context openGauss-beta1 |
| email | Email address | foo@example.com |
| protocol | Protocol head | http:// |
| url | URL | example.com/stuff/index.html |
| host | Host | example.com |
| url\_path | URL path | /stuff/index.html, in the context of a URL |
| file | File or path name | /usr/local/foo.txt, if not within a URL |
| sfloat | Scientific notation | -1.23E+56 |
| float | Decimal notation | -1.234 |
| int | Signed integer | -1234 |
| uint | Unsigned integer | 1234 |
| version | Version number | 8.3.0 |
| tag | XML tag | <a href=“dictionaries.html”> |
| entity | XML entity | &amp; |
| blank | Space symbols | (any whitespace or punctuation not otherwise recognized) |

**注意事项**

* 对于解析器来说，一个“字母”的概念是由数据库的语言区域设置，即lc\_ctype设置决定的。只包含基本ASCII字母的词被报告为一个单独的token类型，因为这类词有时需要被区分出来。大多数欧洲语言中，对token类型word和asciiword的处理方法是类似的。
* email不支持某些由RFC 5322定义的有效电子邮件字符。具体来说，可用于email用户名的非字母数字字符仅包含句号、破折号和下划线。

解析器可能对同一内容进行重叠token。例如，包含连字符的单词将作为一个整体进行报告，其组件也会分别被报告。

SELECT alias, description, token FROM ts\_debug('english','foo-bar-beta1');

结果显示如下：

alias | description | token   
-----------------+------------------------------------------+---------------   
 numhword | Hyphenated word, letters and digits | foo-bar-beta1   
 hword\_asciipart | Hyphenated word part, all ASCII | foo   
 blank | Space symbols | -   
 hword\_asciipart | Hyphenated word part, all ASCII | bar   
 blank | Space symbols | -   
 hword\_numpart | Hyphenated word part, letters and digits | beta1

这种行为是有必要的，因为它支持搜索整个复合词和各组件。这里是另一个例子：

SELECT alias, description, token FROM ts\_debug('english','http://example.com/stuff/index.html');

结果显示如下：

alias | description | token   
----------+---------------+------------------------------   
 protocol | Protocol head | http://   
 url | URL | example.com/stuff/index.html   
 host | Host | example.com   
 url\_path | URL path | /stuff/index.html

N-gram是一种机械分词方法，适用于无语义中文分词场景。N-gram分词法可以保证分词的完备性，但是为了照顾所有可能，把很多不必要的词也加入到索引中，导致索引项增加。N-gram支持中文编码包括GBK、UTF-8。内置6种token类型，如表2 所示。

表2 token类型

| **别名** | **描述** |
| --- | --- |
| zh\_words | chinese words |
| en\_word | english word |
| numeric | numeric data |
| alnum | alnum string |
| grapsymbol | graphic symbol |
| multisymbol | multiple symbol |

Pound是一种固定格式分词方法，适用于无语意但待解析文本以固定分隔符分割开来的中英文分词场景。支持中文编码包括GBK、UTF8，支持英文编码包括ASCII。内置6种token类型，如表3所示；支持5种分隔符，如表4所示，在用户不进行自定义设置的情况下分隔符默认为“#”。Pound限制单个token长度不能超过256个字符。

表3 token类型

| **别名** | **描述** |
| --- | --- |
| zh\_words | chinese words |
| en\_word | english word |
| numeric | numeric data |
| alnum | alnum string |
| grapsymbol | graphic symbol |
| multisymbol | multiple symbol |

表4 分隔符类型

| **分隔符** | **描述** |
| --- | --- |
| @ | Special character |
| # | Special character |
| $ | Special character |
| % | Special character |
| / | Special character |

## 聚合函数嵌套

**功能描述**

Vastbase支持聚合函数嵌套使用。其中内层聚合函数用于计算通过group by子句定义的分组的聚合结果，再将内层聚合函数返回的结果集使用外层聚合函数再进行一次聚合计算，从而返回整体的结果。

**语法格式**

aggregate\_name1(aggregate\_name2(expression[,…][order\_by\_clause]))  
aggregate\_name1(aggregate\_name2(ALL expression [,…][order\_by\_clause]))  
aggregate\_name1(aggregate\_name2(DISTINCT expression[,…][order\_by\_clause]))  
aggregate\_name1(aggregate\_name2(\*))  
aggregate\_name1(aggregate\_name2([expression[,…]]))

**参数说明**

* aggregate\_name1：预定义的聚合函数名称1，内层函数。
* aggregate\_name2：预定义的聚合函数名称2，外层函数。
* expression：自身不包含聚合函数表达式的任意值表达式。

**注意事项**

聚合函数嵌套的用法约束有如下限制：

* 必须含有group by子句，否则报错。例如：

select avg(max(salary)) from empsalary;

* 查询列中同时包括嵌套聚合函数和非嵌套聚合函数时报错。例如：

select avg(max(salary)),max(salary) from empsalary group by department\_id;

* 查询列中同时包括嵌套聚合函数和一般列时报错。例如：

select avg(max(salary)),salary from empsalary group by department\_id;

**示例**

1、创建测试表empsalary。

create table empsalary (  
depname varchar,  
empno int,  
salary int  
);

2、插入数据。

insert into empsalary values('develop',10,5200);  
insert into empsalary values('develop',10,5200);  
insert into empsalary values('saleseeee',10,5000);  
insert into empsalary values('personnel11',5,500);  
insert into empsalary values('personnel11',5,500);  
insert into empsalary values('personnel11',5,500);  
insert into empsalary values('personnel1',5,500);  
insert into empsalary values('sales',1,5200);  
insert into empsalary values('sales',1,5200);  
insert into empsalary values('selaseee',1,5000);

3、调用聚合嵌套函数1。

select max(max(Empno)),max(sum(salary)) from empsalary group by depname;

返回结果如下，则表示调用成功：

max | max  
-----------  
10 | 10400  
(1 row)

4、调用聚合嵌套函数2。

select max(sum(Empno)),max(sum(salary)) from empsalary group by depname;

返回结果如下，则表示调用成功：

max | max   
-----+-------  
 20 | 10400  
(1 row)

## 普通表和分区表在线重建索引

**功能描述**

支持在线rebuild普通行存表索引和行存一级分区表索引，其中行存一级分区表索引包括本地分区索引和全局索引。

**语法格式**

ALTER INDEX index\_name  
REBUILD [ PARTITION partition\_name ] CONCURRENTLY;  
  
REINDEX INDEX CONCURRENTLY  
index\_name [ PARTITION partiotion\_name ];

**参数说明**

* index\_name：自定义的索引名称。
* CONCURRENTLY：并行的含义是不会阻塞DML和DQL语句。

**注意事项**

* 目前仅支持并行重建普通行存表和行存一级分区表。
* 支持行级一级分区表上local和global类型的主表索引重建。
* 支持行级一级分区表上local类型索引在某一特定分区上的分区索引的重建。
* 不支持行级一级分区表上global类型索引在某一特定分区上的重建。

**示例**

1、创建测试表和索引，并插入数据。

CREATE TABLE t\_rebuild\_row(id int,name varchar(20));  
INSERT INTO t\_rebuild\_row values(1,'liuyi');  
INSERT INTO t\_rebuild\_row values(2,'chener');  
INSERT INTO t\_rebuild\_row values(3,'zhangsan');  
INSERT INTO t\_rebuild\_row values(4,'lisi');  
INSERT INTO t\_rebuild\_row values(5,'wangwu');  
INSERT INTO t\_rebuild\_row values(6,'zhaoliu');  
INSERT INTO t\_rebuild\_row values(7,'sunqi');  
INSERT INTO t\_rebuild\_row values(8,'zhouba');  
INSERT INTO t\_rebuild\_row values(9,'wujiu');  
CREATE INDEX idx\_row ON t\_rebuild\_row (id);

2、查看索引是否创建成功。

\d t\_rebuild\_row

返回结果如下，则表示索引创建成功：

Table "public.t\_rebuild\_row"  
 Column | Type | Modifiers  
--------+-------------+-----------  
 id | integer |  
 name | varchar(20) |  
Indexes:  
 "idx\_row" btree (id) TABLESPACE pg\_default

3、设置扫描开关。

set enable\_seqscan =off;

4、检查执行计划是否为索引扫描。

explain select /\*+tablescan(t\_rebuild\_row) \*/ count(\*) from t\_rebuild\_row where id <6;  
explain select /\*+indexonlyscan(t\_rebuild\_row idx\_row )\*/ count(\*) from t\_rebuild\_row where id <6;

两个执行计划分别为顺序扫描及索引扫描,返回结果为：

QUERY PLAN  
-----------------------------------------------------------------------------------------  
 Aggregate (cost=1000000000111.26..1000000000111.27 rows=1 width=8)  
 -> Seq Scan on t\_rebuild\_row (cost=10000000000.00..1000000000111.25 rows=3 width=0)  
 Filter: (id < 6)  
(3 rows)  
  
 QUERY PLAN  
----------------------------------------------------------------------------------------  
 Aggregate (cost=8.31..8.32 rows=1 width=8)  
 -> Index Only Scan using idx\_row on t\_rebuild\_row (cost=0.00..8.30 rows=3 width=0)  
 Index Cond: (id < 6)  
(3 rows)

5、打开两个会话，会话1不断执行DML和DQL操作。

do language plpgsql $$   
declare   
begin   
 for i in 5..100000 loop   
 INSERT INTO t\_rebuild\_row values(10,'zhengshi');  
 DELETE FROM t\_rebuild\_row WHERE id=10;  
 perform name FROM t\_rebuild\_row WHERE id<6 order by name;  
 commit;   
 end loop;   
end;   
$$;

6、切换至会话2，重建索引。

ALTER INDEX idx\_row REBUILD concurrently;

或

REINDEX INDEX CONCURRENTLY idx\_row;

7、会话2中执行如下命令，检查索引是否创建成功(会话1执行过程中，会话2重建索引成功，且会话1不报错)。

explain select /\*+tablescan(t\_rebuild\_row) \*/ count(\*) from t\_rebuild\_row where id <6;  
explain select /\*+indexonlyscan(t\_rebuild\_row idx\_row )\*/ count(\*) from t\_rebuild\_row where id <6;

两个执行计划分别为顺序扫描及索引扫描,返回结果如下则表示索引创建成功：

QUERY PLAN  
--------------------------------------------------------------------------  
 Aggregate (cost=1390.03..1390.04 rows=1 width=8)  
 -> Seq Scan on t\_rebuild\_row (cost=0.00..1332.40 rows=23051 width=0)  
 Filter: (id < 6)  
(3 rows)  
  
  
 QUERY PLAN  
-----------------------------------------------------------------------------------------------  
 Aggregate (cost=2637.27..2637.28 rows=1 width=8)  
 -> Index Only Scan using idx\_row on t\_rebuild\_row (cost=0.00..2579.64 rows=23051 width=0)  
 Index Cond: (id < 6)  
(3 rows)

8、在会话2中执行如下命令，检查两者的结果是否相同。

select /\*+tablescan(t\_rebuild\_row) \*/ count(\*) from t\_rebuild\_row where id <6;  
select /\*+indexonlyscan(t\_rebuild\_row idx\_row )\*/ count(\*) from t\_rebuild\_row where id <6;

返回结果相同，均为：

count  
-------  
 5  
(1 row)

9、还原扫描开关。

reset enable\_seqscan ;

## SQL并行查询

一条查询语句在所涉及的表中的数据量达到阈值时，由规划器生成的计划去执行扫描时性能会比较差。因此需要使用多个worker线程并行各自去执行扫描表操作，在Vastbase G100 V2.2版本中支持并行顺序扫描、并行索引扫描、并行嵌套查询、并行哈希连接。

**并行顺序扫描**

**功能描述**

并行顺序扫描就是利用多核CPU的并行处理能力，将一个表的顺序查询语句由多个并行worker线程去执行查询操作。

**注意事项**

* 不支持临时表、外表、分区表、非行存表的并行查询。
* 不支持CET查询，并行join，并行子查询等。

**示例**

1、创建测试表test并插入数据。

create table test(id int,name text,age int);  
insert into test values(generate\_series(1,1000000),'vastbase',34);

2、指定分析表test。

analyze test;

3、设置并行度。

set query\_dop=5;

4、打印并行查询计划。

explain select count(\*) from test where id<100;

返回结果如下，则表示支持该功能：

QUERY PLAN   
   
---------------------------------------------------------------------------------  
-----  
 Aggregate (cost=8100.25..8100.26 rows=1 width=8)  
 -> Streaming(type: LOCAL GATHER dop: 1/5) (cost=8100.25..8100.26 rows=1 widt  
h=8)  
 -> Aggregate (cost=8100.25..8100.26 rows=1 width=8)  
 -> Seq Scan on test (cost=0.00..8100.00 rows=100 width=0)  
 Filter: (id < 100)  
(5 rows)

**并行索引扫描**

**功能描述**

并行索引扫描就是利用多核CPU的并行处理能力，索引查询可以由多个并行worker线程执行。

当一条sql语句的条件子句中涉及的属性是索引时，查询编译器就会生成并行索引扫描路径，根据代价估算得到并行索引扫描计划，执行器在选择扫描计划时，会将索引扫描计划分给多个worker来执行。

**注意事项**

* 不支持临时表、外表、分区表、非行存表的并行查询。
* 不支持CET查询。
* 只支持并行btree索引扫描，bitmap，gin，gist等索引查询不支持。

**示例**

1、创建测试表并插入数据。

create table parallel\_t2(a int,b int);  
insert into parallel\_t2 values(generate\_series(1,100000), generate\_series(1,100000));

2、创建btree索引。

create index t2\_idx on parallel\_t2 using btree(a);

3、设置扫描开关和并行代价。

set enable\_seqscan to off;  
set enable\_bitmapscan to off;

4、.打印并行执行计划。

explain (costs off) select count(b) from parallel\_t2 where a > 5000;

返回结果为：

QUERY PLAN  
----------------------------------------------  
 Aggregate  
 -> Index Scan using t2\_idx on parallel\_t2  
 Index Cond: (a > 5000)  
(3 rows)

explain (costs off) select count(b) from parallel\_t2 where a < 5000;

返回结果为：

QUERY PLAN  
----------------------------------------------  
 Aggregate  
 -> Index Scan using t2\_idx on parallel\_t2  
 Index Cond: (a < 5000)  
(3 rows)

5、执行查询。

select count(b) from parallel\_t2 where a > 5000;

返回结果为：

count  
-------  
 95000  
(1 row)

select count(b) from parallel\_t2 where a < 5000;

返回结果为：

count  
-------  
 4999  
(1 row)

**并行嵌套查询**

**功能描述**

并行嵌套查询就是利用多核CPU的并行处理能力，将一个多表连接查询语句由多个并行worker线程执行。

**注意事项**

* 不支持临时表、外表、分区表、非行存表的并行查询。
* 不支持CET查询，并行子查询等。

**示例**

1、创建测试表并插入数据。

create table test\_a (id int, col varchar(20));  
create table test\_b (id int, col int);  
create table t3(id int, id2 int, id3 int);  
insert into test\_a values(generate\_series(1,200000), 'xiaohong');  
insert into test\_b values(generate\_series(100,200000), 20);  
insert into t3 values(generate\_series(1, 200000), generate\_series(1, 100), generate\_series(1, 10));

2、设置并行开关。

set enable\_nestloop=on;  
set enable\_mergejoin=off;  
set enable\_hashjoin=off;  
set query\_dop=2;

3、打印嵌套循环并行查询计划。

explain (costs off)   
select col from test\_a where id in (select id from test\_b where id in (select id from t3));

返回结果为：

QUERY PLAN  
----------------------------------------------------------------------------------  
 Streaming(type: LOCAL GATHER dop: 1/2)  
 -> Nested Loop Semi Join  
 Join Filter: (test\_a.id = test\_b.id)  
 -> Streaming(type: LOCAL REDISTRIBUTE dop: 2/2)  
 -> Seq Scan on test\_a  
 -> Materialize  
 -> Streaming(type: LOCAL REDISTRIBUTE dop: 2/2)  
 -> Nested Loop Semi Join  
 Join Filter: (test\_b.id = t3.id)  
 -> Streaming(type: LOCAL REDISTRIBUTE dop: 2/2)  
 -> Seq Scan on test\_b  
 -> Materialize  
 -> Streaming(type: LOCAL REDISTRIBUTE dop: 2/2)  
 -> Seq Scan on t3  
(14 rows)

**并行哈希连接**

**功能描述**

并行哈希连接是利用多核CPU的并行处理能力，将一个生成哈希连接的执行计划由多个Worker执行。一个表连接语句规划器会生成哈希连接计划，并行哈希连接功能是执行器在执行时，将哈希连接的操作由多个Worker并行执行完成。

**注意事项**

* 不支持临时表、外表、分区表、非行存表的并行查询。
* 不支持CET查询，并行子查询等。

**示例**

1、创建两张表。

create table parallel\_hashjoin\_test\_a (id int);  
create table parallel\_hashjoin\_test\_b (id int);

2、插入数据。

insert into parallel\_hashjoin\_test\_a select n from generate\_series(1,1000) n;  
insert into parallel\_hashjoin\_test\_b select n from generate\_series(1,10) n;

3、收集两张表的统计信息。

analyse parallel\_hashjoin\_test\_a;  
analyse parallel\_hashjoin\_test\_b;

4、打印两张表连接查询的查询计划。

explain (costs off) select \* from parallel\_hashjoin\_test\_a left outer join parallel\_hashjoin\_test\_b on parallel\_hashjoin\_test\_a.id = parallel\_hashjoin\_test\_b.id where parallel\_hashjoin\_test\_a.id < 10 order by parallel\_hashjoin\_test\_a.id;

返回结果如下，则表示支持该功能：

QUERY PLAN  
--------------------------------------------------------------------------------  
 Sort  
 Sort Key: parallel\_hashjoin\_test\_a.id  
 -> Hash Left Join  
 Hash Cond: (parallel\_hashjoin\_test\_a.id = parallel\_hashjoin\_test\_b.id)  
 -> Seq Scan on parallel\_hashjoin\_test\_a  
 Filter: (id < 10)  
 -> Hash  
 -> Seq Scan on parallel\_hashjoin\_test\_b  
(8 rows)

5、执行两张表连接查询。

select \* from parallel\_hashjoin\_test\_a left outer join parallel\_hashjoin\_test\_b on parallel\_hashjoin\_test\_a.id = parallel\_hashjoin\_test\_b.id where parallel\_hashjoin\_test\_a.id < 10 order by parallel\_hashjoin\_test\_a.id;

返回结果为:

id | id  
----+----  
 1 | 1  
 2 | 2  
 3 | 3  
 4 | 4  
 5 | 5  
 6 | 6  
 7 | 7  
 8 | 8  
 9 | 9  
(9 rows)

## 附录

### GIN索引

#### GIN索引介绍

GIN（Generalized Inverted Index）通用倒排索引。设计为处理索引项为组合值的情况，查询时需要通过索引搜索出出现在组合值中的特定元素值。例如，文档是由多个单词组成，需要查询出文档中包含的特定单词。

使用item表示索引的组合值，key表示一个元素值。GIN用来存储和搜索key，而不是item。

GIN索引存储一系列（key，posting list）键值对，这里的posting list是一组出现key的行ID。由于每个item都可能包含多个key，同一个行ID可能会出现在多个posting list中，而每个key值只被存储一次，所以在相同的key在item中出现多次的情况下，GIN索引是非常简洁的。

因为GIN索引的访问方式不需要了解他的运行方式，所以GIN索引是通用的。GIN索引使用为特殊数据类型定义的策略。策略定义了如何从索引选项和查询条件中抽出key，以及如何确定在查询中包含某些key值的行是否实际满足查询条件。

#### GIN索引扩展性

GIN索引的接口实现了一个高层次的抽象，要求访问用户仅需要实现被访问数据类型的语义。GIN层自身可以处理并发操作、记录日志、搜索树结构的任务。

定义GIN索引的访问方式所要做的事情就是实现多个用户定义的方法，这些方法定义了键在树中的行为、键与键之间的关系、需要索引的item、能够使用索引的查询。简而言之，GIN索引将扩展性与普遍性、代码重用、清晰的接口结合在了一起。

实现GIN索引的操作符类有如下四个方法：

* int compare(Datum a, Datum b)
* 比较两个key（不是索引的item）然后返回一个小于零、零或大于零的值，分别表示第一个key小于、等于或大于第二个key。NULL不会被传入这个函数。
* Datum \*extractValue(Datum itemValue, int32 \*nkeys, bool \*\*nullFlags)
* 给定一个要被索引的item，返回一个对应key的数组。返回key的数目必须存储在*nkeys中。如果任何key都可能为NULL，还要分配包含*nkeys个布尔元素的数组，将地址存储到*nullFlags，并且根据需要设置NULL值。 如果所有key都是非NULL，可以让*nullFlags保持为NULL（他的初始值）。如果item不包含任何key，返回值可以为NULL。
* Datum \*extractQuery(Datum query, int32 \*nkeys, StrategyNumber n, bool **pmatch, Pointer** extra\_data, bool \*\*nullFlags, int32 \*searchMode)
* 给定一个被查询的值，返回一个对应的key的数组。也就是说，query是可索引操作符右侧的值，而该操作符左侧是被索引的字段。 n是操作符类中操作符的策略号。通常，extractQuery需要参考n来决定query的数据类型以及抽取键值的方法。返回key的个数必须存放在*nkeys中。如果任何key都可能为NULL，还要分配包含*nkeys个布尔元素的数组，将地址存储到*nullFlags，并且根据需要设置NULL值。 如果所有key都是非NULL的，可以让*nullFlags保持为NULL（他的初始值）。如果query不包含任何key，返回值可以为NULL。
* searchMode是一个输出参数，他允许extractQuery指定一些关于如何执行搜索的细节。如果设置*searchMode为GIN\_SEARCH\_MODE\_DEFAULT（这也是调用函数前此参数的初始化值），只有那些至少返回一个key的item才能被考虑作为候选匹配项。如果设置*searchMode为GIN\_SEARCH\_MODE\_INCLUDE\_EMPTY，除了包含至少一个匹配key的item之外，根本不包含任何key的item也被考虑作为候选匹配项。（这个模式对于实现像“是否是子集”这样的操作是有用的）如果设置\*searchMode为GIN\_SEARCH\_MODE\_ALL，索引中所有非NULL的item都被考虑作为候选匹配项，不管他们是否匹配返回key中的任何一个。
* pmatch是一个允许支持部分匹配的输出参数。如果使用此参数，extractQuery必须分配有*nkeys个布尔元素的数组，并把数组地址保存到*pmatch。如果需要部分匹配相应的key，则数组的每个元素应该设置为TRUE；如果不需要匹配，则设置为FALSE。如果设置\*pmatch为NULL，则假设GIN不需要部分匹配。在函数调用前这个值被初始化为NULL，因此，对于不支持部分匹配的操作符类，可以忽略这个参数。
* extra\_data是一个允许extractQuery以consistent和comparePartial的方式传递额外数据的输出参数。如果使用他，extractQuery必须分配一个包含*nkeys个Pointer元素的数组，并把数组地址保存到*extra\_data，然后把他想附加的东西存储到各个独立的指针中。在函数调用前这个值初始化为NULL，因此，对于不需要附加数据的操作符类，可以忽略这个参数。如果设置了\*extra\_data，那么以consistent方式传递整个数组，使用comparePartial方式传递适当的元素。
* bool consistent(bool check[], StrategyNumber n, Datum query, int32 nkeys, Pointer extra\_data[], bool \*recheck, Datum queryKeys[], bool nullFlags[])
* 如果被索引项满足StrategyNumber为n的查询操作符则返回TRUE。这个函数并不直接访问被索引项的值，因为GIN并没有精确的把项目保存下来，但是需要知道从查询中提取的哪些键值出现在给定的被索引项中。 check数组的长度是nkeys，这个与query调用extractQuery函数返回的键值的数目相同。如果索引项包含了相应的查询键，check数组中对应的元素值就是TRUE。比如，如果(check[i] == TRUE)，那么意味着extractQuery的结果数组的第i个键出现在索引项中。考虑可能会用到consistent方式，原始的query也被作为参数传入进来。与此相同的还有extractQuery函数返回的queryKeys[]和nullFlags[]。 extra\_data是extractQuery函数返回的额外数据数组，如果没有的话就是NULL。
* 当extractQuery在queryKeys[]中返回一个NULL的键值，如果被索引项包含NULL键值，相应的check[]中的元素是TRUE。也就是说，check[]的语义很像IS NOT DISTINCT FROM。如果需要知道是通常值匹配还是NULL匹配，consistent函数可以检查相应的nullFlags[]元素。
* 成功执行后，如果堆元组需要针对查询运算符进行重新检查，*recheck需要设置为TRUE，如果索引测试已经是精确的了，则设为FALSE。也就是说，FALSE的返回值确保堆元组不匹配这个查询；设置*recheck为FALSE的TRUE的返回值确保堆元组匹配这个查询；设置\*recheck为TRUE的TRUE的返回值意味着堆元组可能匹配这个查询，因此需要通过直接对照原始索引项对查询运算符进行获取和重新检查。

GIN操作符类可以可选地提供第五个函数。

* int comparePartial(Datum partial\_key, Datum key, StrategyNumber n, Pointer extra\_data)
* 比较一个部分匹配查询键和一个索引键。返回一个整型值，它的符号代表了不同的含义：小于0意味着索引键不匹配查询，但是索引扫描应该继续； 0意味着索引键匹配查询；大于0指示应该终止索引扫描，因为不可能再有更多的匹配。在需要确定何时结束扫描的语义的情况下，这里提供了生成部分一致查询的操作符的策略号n。同样的，extra\_data是extractQuery生成的额外数据数组中的相应元素，如果没有对应的元素，则为NULL。 NULL的键永远不会被传入这个函数。

为了支持"部分匹配"查询，一个操作符类必须提供comparePartial方法，并且当遇到部分匹配查询时，他的extractQuery方法必须设置pmatch参数。

上面的各种Datum值的实际数据类型根据操作符类的不同而不同。传入到extractValue中的项目值总是操作符类的输入类型，所有的键值类型必须是这个类的STORAGE类型。传入到extractQuery和consistent的query参数的类型是由策略号识别的类成员操作符的右操作数的输入类型。他不需要和项目类型相同，只要可以从中抽取出正确类型的键值。

#### GIN索引实现

在内部，GIN索引包含一个在键上构造的B-tree索引，每个键是一个或多个被索引项的一个元素（比如，一个数组的一个成员）。并且页面上每个元组包含了堆指针的B-tree的一个指针（一个posting tree），当列表小到足以和键值一起存储到一个索引元组中时，则是堆指针的一个简单列表（一个posting list）。

多列GIN索引通过在组合值（列号、键值）上建立一个单个的B-tree实现。不同列的键值可以有不同的类型。

**GIN快速更新技术**

由于倒排索引的本身特性影响，更新一个GIN索引可能会比较慢。插入或更新一个堆行可能导致许多往索引的插入。当对表执行VACUUM后，或者如果待处理实体的列表太大了（大于work\_mem），这些实体被使用和初始索引创建时用到的相同的bulk插入方法，移动到主要的GIN数据结构。即使把额外的VACUUM开销算进去，这也大大提升了GIN索引更新的速度。而且，这种额外开销的工作可以通过后台进程而不是前端查询来处理。

这种方法的主要缺点在于搜索时除了常规的索引还必须要扫描待处理实体的列表。因此，大的待处理实体的列表会显著的拖慢搜索。另一个缺点是，虽然大多数更新很快，但是一个导致待处理列表（pending list）变得“太大”的更新将引发一个立即清理，并因此比起其它更新会非常慢。恰当的使用autovacuum可以弱化这两个问题。

如果一致的响应时间（清理实体速度和更新速度的响应时间）比更新速度更重要，可以通过把GIN索引的存储参数FASTUPDATE设置为off而不使用待处理实体。详细请参考：SQL语法参考->SQL语法->CREATE INDEX章节。

**部分匹配算法**

GIN可以支持"部分匹配"查询。即：查询并不决定单个或多个键的一个精确的匹配，而是，可能的匹配落在一个合理的狭窄键值范围内（根据compare支持函数决定的键值排序顺序）。此时，extractQuery方法并不返回一个用于精确匹配的键值，取而代之的是，返回一个要被搜索的键值范围的下边界，并且设置pmatch为true。然后，使用comparePartial方式扫描这个键值范围。comparePartial必须为一个相匹配的索引键返回0，如果不匹配但依然在被搜索范围内时返回小于0的值，对超过可以匹配的范围的索引键则返回大于0的值。

#### GIN提示与技巧

创建vs插入

由于可能要为每个项目插入很多键，所以GIN索引的插入可能比较慢。对于向表中大量插入的操作，我们建议先删除GIN索引，在完成插入之后再重建索引。与GIN索引创建、查询性能相关的GUC参数如下：

* maintenance\_work\_mem
* GIN索引的构建时间对maintenance\_work\_mem的设置非常敏感。
* work\_mem
* 在向启用了FASTUPDATE的GIN索引执行插入操作的期间，只要待处理实体列表的大小超过了work\_mem，系统就会清理这个列表。为了避免可观察到的响应时间的大起大落，让待处理实体列表在后台被清理是比较合适的（比如通过autovacuum）。前端清理操作可以通过增加work\_mem或者执行autovacuum来避免。然而，扩大work\_mem意味着如果发生了前端清理，那么他的执行时间将更长。
* gin\_fuzzy\_search\_limit
* 开发GIN索引的主要目的是为了让Vastbase支持高度可伸缩的全文索引，并且常常会遇见全文索引返回海量结果的情形。而且，这经常发生在查询高频词的时候，因而这样的结果集没什么用处。因为从磁盘读取大量记录并对其进行排序会消耗大量资源，这在产品环境下是不能接受的。为了控制这种情况，GIN索引有一个可配置的返回结果行数的软上限的配置参数gin\_fuzzy\_search\_limit。缺省值0表示没有限制。如果设置了非零值，那么返回结果就是从完整结果集中随机选择的一部分。"软上限"的意思是返回结果的实际数量可能与指定的限制有偏差，这取决于查询和系统随机数生成器的质量。

### 扩展函数

下表列举了Vastbase中支持的扩展函数，不作为商用特性交付，仅供参考。

| **分类** | **函数名称** | **描述** |
| --- | --- | --- |
| 访问权限查询函数 | has\_sequence\_privilege(user, sequence, privilege) | 指定用户是否有访问序列的权限 |
| has\_sequence\_privilege(sequence, privilege) | 当前用户是否有访问序列的权限 |
| 触发器函数 | pg\_get\_triggerdef(oid) | 为触发器获取CREATE [ CONSTRAINT ] TRIGGER命令 |
| pg\_get\_triggerdef(oid, boolean) | 为触发器获取CREATE [ CONSTRAINT ] TRIGGER命令 |

### 扩展语法

Vastbase提供的扩展语法如下。

**表 1** 扩展SQL语法

| **类别** | **语法关键字** | **描述** |
| --- | --- | --- |
| 创建表CREATE TABLE | INHERITS ( parent\_table [, … ] ) | 支持继承表。 |
| column\_constraint：  REFERENCES reftable  [ ( refcolumn ) ] [ MATCH  FULL | MATCH PARTIAL |  MATCH SIMPLE ][ ON  DELETE action ] [ ON  UPDATE action ] | 支持用REFERENCES reftable[ ( refcolumn ) ] [ MATCH FULL |MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] 为表创建外键约束。 |
| 加载模块 | CREATE EXTENSION | 把一个新的模块（例如DBLINK）加载进当前数据库中。 |
| DROP EXTENSION | 删除已加载的模块。 |
| 聚集函数 | CREATE AGGREGATE | 定义一个新的聚集函数。 |
| ALTER AGGREGATE | 修改一个聚集函数的定义。 |
| DROP AGGREGATE | 删除一个现存的聚集函数。 |

# 系统表和系统视图

本章主要介绍Vastbase的系统表和系统视图。

## 系统表和系统视图概述

系统表是Vastbase存放结构元数据的地方，它是Vastbase数据库系统运行控制信息的来源，是数据库系统的核心组成部分。

系统视图提供了查询系统表和访问数据库内部状态的方法。

系统表和系统视图要么只对管理员可见，要么对所有用户可见。后续的系统表和系统视图章节有些标识了需要管理员权限，这些系统表和系统视图只有管理员可以查询。

用户可以删除后重新创建这些表、增加列、插入和更新数值，但是用户修改系统表会导致系统信息的不一致，从而导致系统控制紊乱。正常情况下不应该由用户手工修改系统表或系统视图，或者手工重命名系统表或系统视图所在的模式，而是由SQL语句关联的系统表操作自动维护系统表信息。

用户应该禁止对系统表进行增删改等操作，人为对系统表的修改或破坏可能会导致系统各种异常情况甚至Vastbase不可用。

## 系统表

### GS\_ASP

GS\_ASP显示被持久化的ACTIVE SESSION PROFILE样本，该表只在系统库下查询，在用户库下查询无数据。

**表 1** GS\_ASP字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sampleid | bigint | 采样ID。 |
| sample\_time | timestamp with time zone | 采样的时间。 |
| need\_flush\_sample | boolean | 该样本是否需要刷新到磁盘。-t（true）：表示需要。-f（false）：表示不需要。 |
| databaseid | oid | 数据库ID。 |
| thread\_id | bigint | 线程的ID。 |
| sessionid | bigint | 会话的ID。 |
| start\_time | timestamp with time zone | 会话的启动时间。 |
| event | text | 具体的事件名称。 |
| lwtid | integer | 当前线程的轻量级线程号。 |
| psessionid | bigint | streaming线程的父线程。 |
| tlevel | integer | streaming线程的层级。与执行计划的层级（id）相对应。 |
| smpid | integer | smp执行模式下并行线程的并行编号。 |
| userid | oid | session用户的id。 |
| application\_name | text | 应用的名字。 |
| client\_addr | inet | client端的地址。 |
| client\_hostname | text | client端的名字。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号。 |
| query\_id | bigint | debug query id。 |
| unique\_query\_id | bigint | unique query id。 |
| user\_id | oid | unique query的key中的user\_id。 |
| cn\_id | integer | 表示下发该unique sql的节点id。unique query的key中的cn\_id。 |
| unique\_query | text | -规范化后的Unique SQL文本串。 |
| locktag | text | 会话等待锁信息，可通过locktag\_decode解析。 |
| lockmode | text | 会话等待锁模式：LW\_EXCLUSIVE：排他锁LW\_SHARED：共享锁LW\_WAIT\_UNTIL\_FREE：等待LW\_EXCLUSIVE可用 |
| block\_sessionid | bigint | 如果会话正在等待锁，阻塞该会话获取锁的会话标识。 |
| wait\_status | text | 描述event列的更多详细信息。 |
| global\_sessionid | text | 全局会话ID。 |

### GS\_AUDITING\_POLICY

GS\_AUDITING\_POLICY系统表记录统一审计的主体信息，每条记录对应一个设计策略。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_AUDITING\_POLICY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| polname | name | 策略名称，需要唯一，不可重复。 |
| polcomments | name | 策略描述字段，记录策略相关的描述信息，通过COMMENTS关键字体现。 |
| modifydate | timestamp without time zone | 策略创建或修改的最新时间戳。 |
| polenabled | boolean | 用来表示策略启动开关。 |
| pollevel | name | 表示策略等级。 |

### GS\_AUDITING\_POLICY\_ACCESS

GS\_AUDITING\_POLICY\_ACCESS系统表记录与DML数据库相关操作的统一审计信息。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1**GS\_AUDITING\_POLICY\_ACCESS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| accesstype | name | DML数据库操作相关类型。例如SELECT、INSERT、DELETE等。 |
| labelname | name | 资源标签名称。对应系统表gs\_auditing\_policy中的polname字段。 |
| policyoid | oid | 所属审计策略的Oid，对应系统表GS\_AUDITING\_POLICY中的oid。 |
| modifydate | timestamp without time zone | 创建或修改的最新时间戳。 |

### GS\_AUDITING\_POLICY\_FILTERS

GS\_AUDITING\_POLICY\_FILTERS系统表记录统一审计相关的过滤策略相关信息，每条记录对应一个设计策略。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_AUDITING\_POLICY\_FILTERS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| filtertype | name | 过滤类型。目前值仅为logical\_expr。 |
| labelname | name | 名称。目前值仅为logical\_expr。 |
| policyoid | oid | 所属审计策略的Oid，对应系统表GS\_AUDITING\_POLICY中的oid。 |
| modifydate | timestamp | 创建或修改的最新时间戳。 |
| logicaloperator | text | 过滤条件的逻辑字符串。 |

### GS\_AUDITING\_POLICY\_PRIVILEGES

GS\_AUDITING\_POLICY\_PRIVILEGES系统表记录统一审计DDL数据库相关操作信息，每条记录对应一个设计策略。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_AUDITING\_POLICY\_PRIVI字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| privilegetype | name | DDL数据库操作相关类型。例如CREATE、ALTER、DROP等。 |
| labelname | name | 资源标签名称。对应系统表gs\_auditing\_policy中的polname字段。 |
| policyoid | oid | 对应审计策略系统表GS\_AUDITING\_POLICY中的oid。 |
| modifydate | timestamp | 创建或修改的最新时间戳。 |

### GS\_CLIENT\_GLOBAL\_KEYS

GS\_CLIENT\_GLOBAL\_KEYS系统表记录密态等值特性中客户端加密主密钥相关信息，每条记录对应一个客户端加密主密钥。

**表 1** GS\_CLIENT\_GLOBAL\_KEYS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| global\_key\_name | name | 客户端加密主密钥（cmk）名称。 |
| key\_namespace | oid | 包含此客户端加密主密钥（cmk）的命名空间OID。 |
| key\_owner | oid | 客户端加密主密钥（cmk）的所有者。 |
| key\_acl | aclitem[] | 创建该密钥时所拥有的访问权限。 |
| create\_date | timestamp | 创建密钥的时间。 |

### GS\_CLIENT\_GLOBAL\_KEYS\_ARGS

GS\_CLIENT\_GLOBAL\_KEYS\_ARGS系统表记录密态等值特性中客户端加密主密钥相关元数据信息，每条记录对应客户端加密主密钥的一个键值对信息。

**表 1** GS\_CLIENT\_GLOBAL\_KEYS\_ARGS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| global\_key\_id | oid | 客户端加密主密钥（cmk）oid。 |
| function\_name | name | 值为encryption。 |
| key | name | 客户端加密主密钥（cmk）的元数据信息对应的名称。 |
| value | bytea | 客户端加密主密钥（cmk）的元数据信息名称的值。 |

### GS\_COLUMN\_KEYS

GS\_COLUMN\_KEYS系统表记录密态等值特性中列加密密钥相关信息，每条记录对应一个列加密密钥。

**表 1** GS\_COLUMN\_KEYS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| column\_key\_name | name | 列加密密钥（cek）名称。 |
| column\_key\_distributed\_id | oid | 根据加密密钥（cek）全称域名hash值得到的id。 |
| global\_key\_id | oid | 外键。客户端加密主密钥（cmk）的oid。 |
| key\_namespace | oid | 包含此列加密密钥（cek）的命名空间oid。 |
| key\_owner | oid | 列加密密钥（cek）的所有者。 |
| create\_date | timestamp | 创建列加密密钥的时间。 |
| key\_acl | aclitem[] | 创建该列加密密钥时所拥有的访问权限。 |

### GS\_COLUMN\_KEYS\_ARGS

GS\_COLUMN\_KEYS\_ARGS系统表记录密态等值特性中客户端加密主密钥相关元数据信息，每条记录对应客户端加密主密钥的一个键值对信息。

**表 1** GS\_COLUMN\_KEYS\_ARGS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| column\_key\_id | oid | 列加密密钥（cek）oid。 |
| function\_name | name | 值为encryption。 |
| key | name | 列加密密钥（cek）的元数据信息对应的名称。 |
| value | bytea | 列加密密钥（cek）的元数据信息名称的值。 |

### GS\_DB\_PRIVILEGE

GS\_DB\_PRIVILEGE系统表记录ANY权限的授予情况，每条记录对应一条授权信息。

**表 1** GS\_DB\_PRIVILEGE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段，必须明确选择）。 |
| roleid | oid | 用户标识。 |
| privilege\_type | text | 用户拥有的ANY权限，取值参考ANY权限列表。 |
| admin\_option | boolean | 是否具有privilege\_type列记录的ANY权限的再授权权限。-t：表示具有。-f：表示不具有。 |

### GS\_ENCRYPTED\_COLUMNS

GS\_ENCRYPTED\_COLUMNS系统表记录密态等值特性中表的加密列相关信息，每条记录对应一条加密列信息。

**表 1** GS\_ENCRYPTED\_COLUMNS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| rel\_id | oid | 表的OID。 |
| column\_name | name | 加密列的名称。 |
| column\_key\_id | oid | 外键，列加密密钥的OID。 |
| encryption\_type | int1 | 加密类型，取值为2（DETERMINISTIC）或者1（RANDOMIZED）。 |
| data\_type\_original\_oid | oid | 加密列的原始数据类型id，参考系统表PG\_TYPE中的oid。 |
| data\_type\_original\_mod | int4 | 加密列的原始数据类型修饰符，参考系统表PG\_ATTRIBUTE中的atttypmod。其值对那些不需要的类型data\_type\_original\_mod通常为-1。 |
| create\_date | timestamp | 创建加密列的时间。 |

### GS\_ENCRYPTED\_PROC

GS\_ENCRYPTED\_PROC系统表提供了密态函数/存储过程函数参数、返回值的原始数据类型，加密列等信息。

**表 1** GS\_ENCRYPTED\_PROC字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段）。 |
| func\_id | oid | function的oid，对应pg\_proc系统表中的oid行标识符。 |
| prorettype\_orig | int4 | 返回值的原始数据类型。 |
| proargcachedcol | oidvector | 函数INPUT参数对应的加密列的oid，对应gs\_encrypted\_columns系统表中的oid行标识符。 |
| proallargtypes\_orig | oid[] | 所有函数参数的原始数据类型。 |

### GS\_GLOBAL\_CHAIN

GS\_GLOBAL\_CHAIN系统表记录用户对防篡改用户表的修改操作信息，每条记录对应一次表级修改操作。具有审计管理员权限的用户可以查询此系统表，所有用户均不允许修改此系统表。

**表 1** GS\_GLOBAL\_CHAIN字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| blocknum | bigint | 区块号，当前用户操作在账本中记录的序号。 |
| dbname | name | 数据库名称。被修改的防篡改用户表所属的database。 |
| username | name | 用户名，执行用户表修改操作的用户名。 |
| starttime | timestamp with time zone | 用户操作执行的最新时间戳。 |
| relid | oid | 用户表Oid，被修改的防篡改用户表Oid。 |
| relnsp | name | 模式Oid，被修改的防篡改用户表所属的namesapce oid。 |
| relname | name | 用户表名，被修改的防篡改用户表名。 |
| relhash | hash16 | 当前操作产生的表级别hash变化量。 |
| globalhash | hash32 | 全局摘要，由当前行信息与前一行globalhash计算而来，将整个表串联起来，用于验证GS\_GLOBAL\_CHAIN数据完整性。 |
| txcommand | text | 被记录操作的SQL语句。 |

### GS\_GLOBAL\_CONFIG

GS\_GLOBAL\_CONFIG记录了数据库实例初始化时，用户指定的参数值。除此之外，还存放了用户设置的弱口令，支持数据库初始用户通过ALTER和DROP语法对系统表中的参数进行写入、修改和删除。

**表 1** GS\_GLOBAL\_CONFIG字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | name | 数据库实例初始化时系统内置的指定参数名称。当前版本第一行默认为buckets\_len，第二行起存放弱口令名称。 |
| value | text | 数据库实例初始化时系统内置的指定参数值。当前版本第一行默认为bucketmap长度；第二行起存放弱口令。 |

### GS\_MASKING\_POLICY

GS\_MASKING\_POLICY系统表记录动态数据脱敏策略的主体信息，每条记录对应一个脱敏策略。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_MASKING\_POLICY表字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| polname | name | 策略名称，唯一不可重复。 |
| polcomments | name | 策略描述字段，记录策略相关的描述信息，通过COMMENTS关键字体现。 |
| modifydate | timestamp | 策略创建或修改的最新时间戳。 |
| polenabled | Boolean | 策略启动开关。 |

### GS\_MASKING\_POLICY\_ACTIONS

GS\_MASKING\_POLICY\_ACTIONS系统表记录动态数据脱敏策略中相应的脱敏策略包含的脱敏行为，一个脱敏策略对应着该表的一行或多行记录。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_MASKING\_POLICY\_ACTIONS表字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| actiontype | name | 脱敏函数，标识脱敏策略使用的脱敏函数。 |
| actparams | name | 向脱敏函数中传递的参数信息。 |
| actlabelname | name | 被脱敏的label名称。 |
| policyoid | oid | 该条记录所属的脱敏策略oid，对应GS\_MASKING\_POLICY中的oid。 |
| modifydate | timestamp | 该条记录创建或修改的最新时间戳。 |

### GS\_MASKING\_POLICY\_FILTERS

GS\_MASKING\_POLICY\_FILTERS系统表记录动态数据脱敏策略对应的用户过滤条件，当用户条件满足FILTER条件时，对应的脱敏策略才会生效。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

**表 1** GS\_MASKING\_POLICY\_FILTERS表字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| filtertype | name | 过滤类型。目前值仅为logical\_expr。 |
| filterlabelname | name | 过滤范围。目前值仅为logical\_expr。 |
| policyoid | oid | 该条用户过滤条件所属的脱敏策略oid，对应GS\_MASKING\_POLICY中的oid。 |
| modifydate | timestamp | 该条用户过滤条件创建或修改的最新时间戳。 |
| logicaloperator | text | 过滤条件的波兰表达式。 |

### GS\_MATVIEW

GS\_MATVIEW系统表提供了关于数据库中每一个物化视图的信息。

**表 1** GS\_MATVIEW字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| matviewid | oid | 物化视图的oid。 |
| mapid | oid | 物化视图map表的oid，map表为物化视图关联表，与物化视图一一对应。全量物化视图不存在对应的map表，该字段为0。 |
| ivm | boolean | 物化视图的类型，t为增量物化视图，f为全量物化视图。 |
| needrefresh | boolean | 保留字段。 |
| refreshtime | timestamp | 物化视图上一次刷新时间，若未刷新则为null。仅对增量物化视图维护该字段，全量物化视图为null。 |

### GS\_MATVIEW\_DEPENDENCY

GS\_MATVIEW\_DEPENDENCY系统表提供了关于数据库中每一个增量物化视图、基表和mlog表的关联信息。全量物化视图不存在与基表对应的mlog表，不会写入记录。

**表 1** GS\_MATVIEW\_DEPENDENCY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐藏属性，必须明确选择）。 |
| matviewid | oid | 物化视图的oid。 |
| relid | oid | 物化视图基表的oid。 |
| mlogid | oid | 物化视图mlog表的oid，mlog表为物化视图日志表，与基表一一对应。 |
| mxmin | int4 | 保留字段。 |

### GS\_MODEL\_WAREHOUSE

GS\_MODEL\_WAREHOUSE系统表用于存储AI引擎训练模型，其中包含模型，训练过程的详细描述。

**表 1** GS\_MODEL\_WAREHOUSE字段

| **名称** | **数据类型** | **描述** |
| --- | --- | --- |
| oid | oid | 隐含列。 |
| modelname | name | 唯一约束。 |
| modelowner | oid | 模型拥有者的OID。 |
| createtime | timestamp | 模型创建的时间。 |
| processedtuples | int | 训练涉及的元组数。 |
| discardedtuples | int | 未参加训练的不合格元组数。 |
| preprocesstime | real | 数据预处理时长。 |
| exectime | real | 训练时长。 |
| iterations | int | 迭代轮次。 |
| outputtype | oid | 模型输出的数据类型OID。 |
| modeltype | text | AI算子的类型名称。 |
| query | text | 创建模型所执行的query语句。 |
| modeldata | bytea | 保存的二进制模型信息。 |
| weight | real[] | 目前只适用于GD算子模型。 |
| hyperparametersnames | text[] | 涉及的超参名称。 |
| hyperparametersvalues | text[] | 超参所对应的取值。 |
| hyperparametersoids | oid[] | 超参对应的数据类型OID。 |
| coefnames | text[] | 模型参数名称。 |
| coefvalues | text[] | 模型参数对应的取值。 |
| coefoids | oid[] | 模型参数对应的数据类型OID。 |
| trainingscoresname | text[] | 度量模型性能方法的名称。 |
| trainingscoresvalue | real[] | 度量模型性能方法的数值。 |
| modeldescribe | text[] | 模型的描述信息。 |

### GS\_OPT\_MODEL

GS\_OPT\_MODEL是启用AiEngine执行计划时间预测功能时的数据表，记录机器学习模型的配置、训练结果、功能、对应系统函数、训练历史等相关信息。

**表 1** GS\_OPT\_MODEL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 数据库对象id。 |
| template\_name | name | 机器学习模型的模板名，决定训练和预测调用的函数接口，目前只实现了rlstm，方便后续扩展。 |
| model\_name | name | 模型的实例名，每个模型对应aiEngine在线学习进程中的一套参数、训练日志、模型系数。此列需为unique。 |
| datname | name | 该模型所服务的database名，每个模型只针对单个database。此参数决定训练时所使用的数据。 |
| ip | name | AiEngine端所部署的host ip地址。 |
| port | integer | AiEngine端所侦听的端口号。 |
| max\_epoch | integer | 模型每次训练的迭代次数上限。 |
| learning\_rate | real | 模型训练的学习速率，推荐缺省值1。 |
| dim\_red | real | 模型特征维度降维系数。 |
| hidden\_units | integer | 模型隐藏层神经元个数。如果训练发现模型长期无法收敛，可以适量提升本参数。 |
| batch\_size | integer | 模型每次迭代时一个batch的大小，尽量设为大于等于训练数据总量的值，加快模型的收敛速度。 |
| feature\_size | integer | [不需设置] 模型特征的长度，用于触发重新训练，模型训练后该参数自动更新。 |
| available | boolean | [不需设置]标识模型是否收敛。 |
| Is\_training | boolean | [不需设置]标识模型是否正在训练。 |
| label | “char”[] | 模型的目标任务：S：startup timeT：total timeR：rowsM：peak memory目前受模型性能限制，推荐{S, T}或{R}。 |
| max | bigint[] | [不需设置]标识模型各任务标签的最大值，用于触发重新训练。 |
| acc | real[] | [不需设置]标识模型各任务的准确率。 |
| description | text | 模型注释。 |

### GS\_PACKAGE

GS\_PACKAGE系统表记录PACKAGE内的信息。

**表 1** GS\_PACKAGE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| pkgnamespace | oid | package所属schema。 |
| pkgowner | oid | package的所属者。 |
| pkgname | name | package的名字。 |
| pkgspecsrc | text | package specification的内容。 |
| pkgbodydeclsrc | text | package body的内容。 |
| pkgbodyinitsrc | text | package init的内容。 |
| pkgacl | aclitem | 访问权限。 |
| pkgsecdef | boolean | package是否是定义者权限。 |

### GS\_POLICY\_LABEL

GS\_POLICY\_LABEL系统表记录资源标签配置信息，一个资源标签对应着一条或多条记录，每条记录标记了数据库资源所属的资源标签。需要有系统管理员或安全策略管理员权限才可以访问此系统表。

FQDN（Fully Qualified Domain Name）标识了数据库资源所属的绝对路径。

**表 1** GS\_POLICY\_LABEL表字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| labelname | name | 资源标签名称。 |
| labeltype | name | 资源标签类型，目前仅为RESOURCE。 |
| fqdnnamespace | oid | 被标识的数据库资源所属的namespace oid。 |
| fqdnid | oid | 被标识的数据库资源的oid，若数据库资源为列，则该列为所属表的oid。 |
| relcolumn | name | 列名，若被标识的数据库资源为列，该列指出列名，否则该列为空。 |
| fqdntype | name | 被标识的数据库资源的类型名称，例如：schema、table、column、view等。 |

### GS\_RECYCLEBIN

gs\_recyclebin描述了回收站对象的详细信息。

**表 1** gs\_recyclebin字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 系统列。 |
| rcybaseid | oid | 基表对象id，引用gs\_recyclebin.oid。 |
| rcydbid | oid | 当前对象所属数据库oid。 |
| rcyrelid | oid | 当前对象oid。 |
| rcyname | name | 回收站对象名称，格式“BIN$unique\_id$oid$0”，其中unique\_id为最多16字符唯一标识，oid为对象标识符。 |
| rcyoriginname | name | 原始对象名称。 |
| rcyoperation | “char” | 操作类型。d表示dropt表示truncate |
| rcytype | int | 对象类型。0表示table。1表示index。2表示toast table。3表示toast index。4表示sequence，指serial、bigserial类型自动关联的序列对象。 |
| rcyrecyclecsn | bigint | 对象drop、truncate时csn。 |
| rcyrecycletime | timestamptz | 对象drop、truncate时间。 |
| rcycreatecsn | bigint | 对象创建时csn。 |
| rcychangecsn | bigint | 对象定义改变的csn。 |
| rcynamespace | oid | 包含这个关系的名字空间的OID。 |
| rcyowner | oid | 关系所有者。 |
| rcytablespace | oid | 这个关系存储所在的表空间。如果为0，则意味着使用该数据库的缺省表空间。如果关系在磁盘上没有文件，则这个字段没有什么意义。 |
| rcyrelfilenode | oid | 回收站对象在磁盘上的文件的名称，如果没有则为0，用于TRUNCATE对象恢复时纹理文件还原。 |
| rcycanrestore | bool | 是否可以被单独闪回。 |
| rcycanpurge | bool | 是否可以被单独purge。 |
| rcyfrozenxid | xid32 | 该表中所有在这个之前的事务ID已经被一个固定的（”frozen”）事务ID替换。 |
| rcyfrozenxid64 | xid | 该表中所有在这个之前的事务ID已经被一个固定的（”frozen”）事务ID替换。 |

### GS\_TXN\_SNAPSHOT

GS\_TXN\_SNAPSHOT是“时间戳-CSN”映射表，周期性采样，并维护适当的时间范围，用于估算范围内的时间戳对应的CSN值。

**表 1** GS\_TXN\_SNAPSHOT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| snptime | timestamptz | 快照捕获时间 |
| snpxmin | bigint | 快照xmin |
| snpcsn | bigint | 快照csn |
| snpsnapshot | TEXT | 快照序列化文本 |

### GS\_UID

GS\_UID系统表存储了数据库中使用hasuids属性表的唯一标识元信息。

**表 1** GS\_UID字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的oid信息。 |
| uid\_backup | bigint | 当前可以为表分配唯一标识的最大值。 |

### GS\_WLM\_EC\_OPERATOR\_INFO

GS\_WLM\_EC\_OPERATOR\_INFO系统表存储执行EC（Extension Connector）作业结束后的算子相关的记录。当设置GUC参数enable\_resource\_record（参见：GUC参数说明->负载管理章节中的enable\_resource\_record)为on时，系统会每3分钟将GS\_WLM\_EC\_OPERATOR\_HISTORY中的记录导入此系统表，开启此功能会占用系统存储空间并对性能有一定影响。查询该系统表需要sysadmin权限。

**表 1** GS\_WLM\_EC\_OPERATOR\_INFO的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | EC语句执行使用的内部query\_id。 |
| plan\_node\_id | integer | EC算子对应的执行计划的plan node id。 |
| start\_time | timestamp with time zone | EC算子处理第一条数据的开始时间。 |
| duration | bigint | EC算子到结束时候总的执行时间(ms)。 |
| tuple\_processed | bigint | EC算子返回的元素个数。 |
| min\_peak\_memory | integer | EC算子在所有DN上的最小内存峰值(MB)。 |
| max\_peak\_memory | integer | EC算子在所有DN上的最大内存峰值(MB)。 |
| average\_peak\_memory | integer | EC算子在所有DN上的平均内存峰值(MB)。 |
| ec\_status | text | EC作业的执行状态。 |
| ec\_execute\_datanode | text | 执行EC作业的DN名称。 |
| ec\_dsn | text | EC作业所使用的DSN。 |
| ec\_username | text | EC作业访问远端集群的USERNAME（远端集群为SPARK类型时该值为空）。 |
| ec\_query | text | EC作业发送给远端集群执行的语句。 |
| ec\_libodbc\_type | text | EC作业使用的unixODBC驱动类型。 |

### GS\_WLM\_INSTANCE\_HISTORY

GS\_WLM\_INSTANCE\_HISTORY系统表存储与实例（数据库主节点或数据库节点）相关的资源使用相关信息。该系统表里每条记录都是对应时间点某实例资源使用情况，包括：内存、CPU核数、磁盘IO、进程物理IO和进程逻辑IO信息。查询该系统表需要sysadmin权限，且仅在数据库postgres下面查询时有数据。

**表 1** GS\_WLM\_INSTANCE\_HISTORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| instancename | text | 实例名称。 |
| timestamp | timestamp with time zone | 时间戳。 |
| used\_cpu | int | 实例使用CPU所占用的百分比。 |
| free\_mem | int | 实例未使用的内存大小，单位MB。 |
| used\_mem | int | 实例已使用的内存大小，单位MB。 |
| io\_await | real | 实例所使用磁盘的io\_wait值（10秒均值）。 |
| io\_util | real | 实例所使用磁盘的io\_util值（10秒均值）。 |
| disk\_read | real | 实例所使用磁盘的读速率（10秒均值），单位KB/s。 |
| disk\_write | real | 实例所使用磁盘的写速率（10秒均值），单位KB/s。 |
| process\_read | bigint | 实例对应进程从磁盘读数据的读速率(不包括从磁盘pagecache中读取的字节数，10秒均值)，单位KB/s。 |
| process\_write | bigint | 实例对应进程向磁盘写数据的写速率(不包括向磁盘pagecache中写入的字节数，10秒均值)，单位KB/s。 |
| logical\_read | bigint | 数据库主节点实例：不统计。数据库节点实例：该实例在本次统计间隙（10秒）内逻辑读字节速率，单位KB/s。 |
| logical\_write | bigint | 数据库主节点实例：不统计。数据库节点实例：该实例在本次统计间隙（10秒）内逻辑写字节速率，单位KB/s。 |
| read\_counts | bigint | 数据库主节点实例：不统计。数据库节点实例：该实例在本次统计间隙（10秒）内逻辑读操作次数之和，单位次。 |
| write\_counts | bigint | 数据库主节点实例：不统计。数据库节点实例：该实例在本次统计间隙（10秒）内逻辑写操作次数之和，单位次。 |

### GS\_WLM\_OPERATOR\_INFO

GS\_WLM\_OPERATOR\_INFO系统表显示执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表中的数据。查询该系统表需要sysadmin权限，且仅在数据库postgres下面查询时有数据。

**表 1** GS\_WLM\_OPERATOR\_INFO的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间(ms)。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库节点上的最小内存峰值(MB)。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值(MB)。 |
| average\_peak\_memory | integer | 当前算子在数据库节点平均内存峰值(MB)。 |
| memory\_skew\_percent | integer | 当前算子在数据库节点间的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库节点盘的最小数据量(MB)，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库节点上下盘的最大数据量(MB)，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库节点盘的平均数据量(MB)，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库节点间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库节点最小执行时间(ms)。 |
| max\_cpu\_time | bigint | 该算子在数据库节点上的最大执行时间(ms)。 |
| total\_cpu\_time | bigint | 该算子在数据库节点上的总执行时间(ms)。 |
| cpu\_skew\_percent | integer | 数据库节点间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Sort/SetOp/HashAgg/HashJoin spill * Spill file size large than 256MB * Broadcast size large than 100MB * Early spill * Spill times is greater than 3 * Spill on memory adaptive * Hash table conflict |

### GS\_WLM\_PLAN\_ENCODING\_TABLE

GS\_WLM\_PLAN\_ENCODING\_TABLE系统表显示计划算子级的编码信息，为机器学习模型的提供包括startup time、total time、peak memory、rows等标签值的训练、预测集。

**表 1** GS\_WLM\_PLAN\_ENCODING\_TABLE的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| parent\_node\_id | integer | 当前算子的父节点node id。 |
| startup\_time | bignit | 该算子处理第一条数据的开始时间。 |
| total\_time | bigint | 该算子到结束时候总的执行时间（ms）。 |
| rows | bigint | 当前算子执行的行数信息。 |
| peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值（MB）。 |
| encode | text | 当前计划算子的编码信息。 |

### GS\_WLM\_PLAN\_OPERATOR\_INFO

GS\_WLM\_PLAN\_OPERATOR\_INFO系统表显示执行作业结束后计划算子级的相关的记录。此数据是从内核中转储到系统表中的数据。

**表 1** GS\_WLM\_PLAN\_OPERATOR\_INFO的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datname | name | 收集计划信息所在的database名。 |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| startup\_time | bigint | 该算子处理第一条数据的开始时间。 |
| total\_time | bigint | 该算子到结束时候总的执行时间(ms)。 |
| actual\_rows | bigint | 实际执行的行数信息。 |
| max\_peak\_memory | integer | 当前算子在数据库节点上的最大内存峰值(MB)。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| parent\_node\_id | integer | 当前算子的父节点node id。 |
| left\_child\_id | integer | 当前算子的左孩子节点node id。 |
| right\_child\_id | integer | 当前算子的右孩子节点node id。 |
| operation | text | 当前算子进行的操作名称。 |
| orientation | text | 当前算子的对齐方式。 |
| strategy | text | 当前算子操作的实现方法。 |
| options | text | 当前算子操作的选择方式。 |
| condition | text | 当前算子操作的过滤条件。 |
| projection | text | 当前算子的映射关系。 |

### GS\_WLM\_SESSION\_QUERY\_INFO\_ALL

GS\_WLM\_SESSION\_QUERY\_INFO\_ALL系统表显示当前数据库实例执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。当设置GUC参数enable\_resource\_record（参见：GUC参数说明->负载管理章节中的enable\_resource\_record)为on时，系统会定时（周期为3分钟）将内核中query信息导入GS\_WLM\_SESSION\_QUERY\_INFO\_ALL系统表。查询该系统表需要sysadmin权限，且仅在数据库postgres下面查询时有数据。

**表 1** GS\_WLM\_SESSION\_QUERY\_INFO\_ALL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| dbname | text | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 语句执行的数据库实例名称。 |
| username | text | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| block\_time | bigint | 语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| finish\_time | timestamp with time zone | 语句执行的结束时间。 |
| duration | bigint | 语句实际执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句预估执行时间，单位ms。 |
| status | text | 语句执行结束状态：正常为finished，异常为aborted。 |
| abort\_info | text | 语句执行结束状态为aborted时显示异常信息。 |
| resource\_pool | text | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句估算内存大小。 |
| min\_peak\_memory | integer | 语句在数据库实例上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库实例上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句数据库实例间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库实例上的下盘信息：   * None：数据库实例均未下盘。 * All：数据库实例均下盘。 * [a:b]：数量为b个数据库实例中有a个数据库实例下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库实例上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库实例间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库实例上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库实例上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库实例上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库实例间的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库实例上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库实例上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库实例上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库实例间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库实例上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库实例上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库实例上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库实例间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Sort/SetOp/HashAgg/HashJoin spill * Spill file size large than 256MB * Broadcast size large than 100MB * Early spill * Spill times is greater than 3 * Spill on memory adaptive * Hash table conflict |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑数据库实例。 |
| cpu\_top1\_node\_name | text | cpu使用率第1的节点名称。 |
| cpu\_top2\_node\_name | text | cpu使用率第2的节点名称。 |
| cpu\_top3\_node\_name | text | cpu使用率第3的节点名称。 |
| cpu\_top4\_node\_name | text | cpu使用率第4的节点名称。 |
| cpu\_top5\_node\_name | text | cpu使用率第5的节点名称。 |
| mem\_top1\_node\_name | text | 内存使用量第1的节点名称。 |
| mem\_top2\_node\_name | text | 内存使用量第5的节点名称。 |
| mem\_top3\_node\_name | text | 内存使用量第5的节点名称。 |
| mem\_top4\_node\_name | text | 内存使用量第5的节点名称。 |
| mem\_top5\_node\_name | text | 内存使用量第5的节点名称。 |
| cpu\_top1\_value | bigint | cpu使用率。 |
| cpu\_top2\_value | bigint | cpu使用率。 |
| cpu\_top3\_value | bigint | cpu使用率。 |
| cpu\_top4\_value | bigint | cpu使用率。 |
| cpu\_top5\_value | bigint | cpu使用率。 |
| mem\_top1\_value | bigint | 内存使用量。 |
| mem\_top2\_value | bigint | 内存使用量。 |
| mem\_top3\_value | bigint | 内存使用量。 |
| mem\_top4\_value | bigint | 内存使用量。 |
| mem\_top5\_value | bigint | 内存使用量。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |
| n\_returned\_rows | bigint | Select返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行数。 |
| n\_tuples\_returned | bigint | 顺序扫描行数。 |
| n\_tuples\_inserted | bigint | 插入行数。 |
| n\_tuples\_updated | bigint | 更新行数。 |
| n\_tuples\_deleted | bigint | 删除行数。 |
| n\_blocks\_fetched | bigint | Cache加载次数。 |
| n\_blocks\_hit | bigint | Cache命中数。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| net\_send\_time | bigint | 网络上的时间花费（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 |
| is\_slow\_query | bigint | 标记是否为慢查询。取值为1时表示其为慢查询。 |

### GS\_WLM\_USER\_RESOURCE\_HISTORY

GS\_WLM\_USER\_RESOURCE\_HISTORY系统表存储与用户使用资源相关的信息。该系统表的每条记录都是对应时间点某用户的资源使用情况，包括：内存、CPU核数、存储空间、临时空间、算子落盘空间、逻辑IO流量、逻辑IO次数和逻辑IO速率信息。其中，内存、CPU、IO相关监控项仅记录用户复杂作业的资源使用情况。对于IO相关监控项，当参数enable\_logical\_io\_statistics为on时才有效；当参数enable\_user\_metric\_persistent为on时，才会开启用户监控数据保存功能。GS\_WLM\_USER\_RESOURCE\_HISTORY系统表的数据来源于PG\_TOTAL\_USER\_RESOURCE\_INFO视图。查询该系统表需要sysadmin权限，且仅在数据库postgres下面查询时有数据。

**表 1** GS\_WLM\_USER\_RESOURCE\_HISTORY

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| username | text | 用户名 |
| timestamp | timestamp with time zone | 时间戳 |
| used\_memory | integer | 正在使用的内存大小，单位MB。 |
| total\_memory | integer | 可以使用的内存大小，单位为MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 |
| used\_cpu | real | 正在使用的CPU核数。 |
| total\_cpu | integer | 该机器节点上，用户关联控制组的CPU核数总和。 |
| used\_space | bigint | 已使用的存储空间大小，单位KB。 |
| total\_space | bigint | 可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。 |
| used\_temp\_space | bigint | 已使用的临时存储空间大小，单位KB。 |
| total\_temp\_space | bigint | 可使用的临存储空间大小，单位KB，值为-1表示未限制最大临时存储空间。 |
| used\_spill\_space | bigint | 已使用的算子落盘存储空间大小，单位KB。 |
| total\_spill\_space | bigint | 可使用的算子落盘存储空间大小，单位KB，值为-1表示未限制最大算子落盘存储空间。 |
| read\_kbytes | bigint | 监控周期内，读操作的字节流量，单位KB。 |
| write\_kbytes | bigint | 监控周期内，写操作的字节流量，单位KB。 |
| read\_counts | bigint | 监控周期内，读操作的次数，单位次。 |
| write\_counts | bigint | 监控周期内，写操作的次数，单位次。 |
| read\_speed | real | 监控周期内，读操作的字节速率，单位KB/s。 |
| write\_speed | real | 监控周期内，写操作的字节速率，单位KB/s。 |

### PG\_AGGREGATE

PG\_AGGREGATE系统表存储与聚集函数有关的信息。PG\_AGGREGATE里的每条记录都是一条pg\_proc里面的记录的扩展。PG\_PROC记录承载该聚集的名称、输入和输出数据类型，以及其它一些和普通函数类似的信息。

**表 1** PG\_AGGREGATE字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| aggfnoid | regproc | PG\_PROC.proname | 此聚集函数的PG\_PROC proname。 |
| aggtransfn | regproc | PG\_PROC.proname | 转换函数。 |
| aggcollectfn | regproc | PG\_PROC.proname | 收集函数。 |
| aggfinalfn | regproc | PG\_PROC.proname | 最终处理函数（如果没有则为零）。 |
| aggsortop | oid | PG\_OPERATOR.oid | 关联排序操作符（如果没有则为零）。 |
| aggtranstype | oid | PG\_TYPE.oid | 此聚集函数的内部转换（状态）数据的数据类型。可能取值及其含义见于pg\_type.h中诸type定义，主要分为多态（isPolymorphicType）和非多态两类。 |
| agginitval | text | - | 转换状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则转换状态值从null开始。 |
| agginitcollect | text | - | 收集状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则收集状态值从null开始。 |
| aggkind | “char” | - | 此聚集函数类型： -'n' ：表示Normal Agg。 -'o' ：表示Ordered Set Agg。 |
| aggnumdirectargs | smallint | - | Ordered Set Agg类型聚集函数的直接参数（非聚集相关参数）数量。对Normal Agg类型聚集函数，该值为0。 |

### PG\_AM

PG\_AM系统表存储有关索引访问方法的信息。系统支持的每种索引访问方法都有一行。

**表 1** PG\_AM字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| amname | name | - | 访问方法的名称。 |
| amstrategies | smallint | - | 访问方法的操作符策略个数，或者如果访问方法没有一个固定的操作符策略集则为0。 |
| amsupport | smallint | - | 访问方法的支持过程个数。 |
| amcanorder | boolean | - | 这种访问方式是否支持通过索引字段值的命令扫描排序。 |
| amcanorderbyop | boolean | - | 这种访问方式是否支持通过索引字段上操作符的结果的命令扫描排序。 |
| amcanbackward | boolean | - | 访问方式是否支持向后扫描。 |
| amcanunique | boolean | - | 访问方式是否支持唯一索引。 |
| amcanmulticol | boolean | - | 访问方式是否支持多字段索引。 |
| amoptionalkey | boolean | - | 访问方式是否支持第一个索引字段上没有任何约束的扫描。 |
| amsearcharray | boolean | - | 访问方式是否支持ScalarArrayOpExpr搜索。 |
| amsearchnulls | boolean | - | 访问方式是否支持IS NULL/NOT NULL搜索。 |
| amstorage | boolean | - | 允许索引存储的数据类型与列的数据类型是否不同。 |
| amclusterable | boolean | - | 是否允许在一个这种类型的索引上聚簇。 |
| ampredlocks | boolean | - | 是否允许这种类型的一个索引管理细粒度的谓词锁定。 |
| amkeytype | oid | PG\_TYPE.oid | 存储在索引里数据的类型，如果不是一个固定的类型则为0。 |
| aminsert | regproc | PG\_PROC.proname | “插入这个行”函数。 |
| ambeginscan | regproc | PG\_PROC.proname | “准备索引扫描”函数。 |
| amgettuple | regproc | PG\_PROC.proname | “下一个有效行”函数，如果没有则为0。 |
| amgetbitmap | regproc | PG\_PROC.proname | “抓取所有的有效行” 函数，如果没有则为0。 |
| amrescan | regproc | PG\_PROC.proname | “（重新）开始索引扫描”函数。 |
| amendscan | regproc | PG\_PROC.proname | “索引扫描后清理” 函数。 |
| ammarkpos | regproc | PG\_PROC.proname | “标记当前扫描位置”函数。 |
| amrestrpos | regproc | PG\_PROC.proname | “恢复已标记的扫描位置”函数。 |
| ammerge | regproc | PG\_PROC.proname | “归并多个索引对象”函数。 |
| ambuild | regproc | PG\_PROC.proname | “建立新索引”函数。 |
| ambuildempty | regproc | PG\_PROC.proname | “建立空索引”函数。 |
| ambulkdelete | regproc | PG\_PROC.proname | 批量删除函数。 |
| amvacuumcleanup | regproc | PG\_PROC.proname | VACUUM后的清理函数。 |
| amcanreturn | regproc | PG\_PROC.proname | 检查是否索引支持唯一索引扫描的函数，如果没有则为0。 |
| amcostestimate | regproc | PG\_PROC.proname | 估计一个索引扫描开销的函数。 |
| amoptions | regproc | PG\_PROC.proname | 为一个索引分析和确认reloptions的函数。 |

### PG\_AMOP

PG\_AMOP系统表存储有关和访问方法操作符族关联的信息。如果一个操作符是一个操作符族中的成员，则在这个表中会占据一行。一个族成员是一个search操作符或一个ordering操作符。一个操作符可以在多个族中出现，但是不能在一个族中的多个搜索位置或多个排序位置中出现。

**表 1** PG\_AMOP字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐藏属性，必须明确选择）。 |
| amopfamily | oid | PG\_OPFAMILY.oid | 这个项的操作符族。 |
| amoplefttype | oid | PG\_TYPE.oid | 操作符的左输入类型。 |
| amoprighttype | oid | PG\_TYPE.oid | 操作符的右输入类型。 |
| amopstrategy | smallint | - | 操作符策略数。 |
| amoppurpose | “char” | - | 操作符目的，s为搜索或o为排序。 |
| amopopr | oid | PG\_OPERATOR.oid | 该操作符的OID。 |
| amopmethod | oid | PG\_AM.oid | 索引访问方式操作符族。 |
| amopsortfamily | oid | PG\_OPFAMILY.oid | 如果是一个排序操作符，则为这个项排序所依据的btree操作符族；如果是一个搜索操作符，则为0。 |

search操作符表明这个操作符族的一个索引可以被搜索，找到所有满足WHERE indexed\_column operator constant的行。显然，这样的操作符必须返回布尔值，并且它的左输入类型必须匹配索引的字段数据类型。

ordering操作符表明这个操作符族的一个索引可以被扫描，返回以ORDER BY indexed\_column operator constant顺序表示的行。这样的操作符可以返回任意可排序的数据类型，它的左输入类型也必须匹配索引的字段数据类型。ORDER BY的确切的语义是由amopsortfamily字段指定的，该字段必须为操作符的返回类型引用一个btree操作符族。

### PG\_AMPROC

PG\_AMPROC系统表存储有关与访问方法操作符族相关联的支持过程的信息。每个属于某个操作符族的支持过程都占有一行。

**表 1** PG\_AMPROC字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| amprocfamily | oid | PG\_OPFAMILY.oid | 该项的操作符族。 |
| amproclefttype | oid | PG\_TYPE.oid | 相关操作符的左输入数据类型。可能取值及其描述见于pg\_type.h。 |
| amprocrighttype | oid | PG\_TYPE.oid | 相关操作符的右输入数据类型。可能取值及其描述见于pg\_type.h。 |
| amprocnum | smallint | - | 支持过程编号。 |
| amproc | regproc | PG\_PROC.proname | 过程的OID。 |

amproclefttype和amprocrighttype字段的习惯解释，标识一个特定支持过程支持的操作符的左和右输入类型。对于某些访问方式，匹配支持过程本身的输入数据类型，对其他的则不这样。有一个对索引的“缺省”支持过程的概念，amproclefttype和amprocrighttype都等于索引操作符类的opcintype。

### PG\_APP\_WORKLOADGROUP\_MAPPING

PG\_APP\_WORKLOADGROUP\_MAPPING系统表提供了数据库负载映射组的信息。

**表 1** PG\_APP\_WORKLOADGROUP\_MAPPING字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| appname | name | 应用名称。 |
| workload\_gpname | name | 映射到的负载组名称。 |

### PG\_ATTRDEF

PG\_ATTRDEF系统表存储列的默认值。

**表 1** PG\_ATTRDEF字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| adrelid | oid | 该列的所属表。 |
| adnum | smallint | 该列的数目。 |
| adbin | pg\_node\_tree | 字段缺省值或生成表达式的内部表现形式。 |
| adsrc | text | 可读缺省值或生成表达式的内部表现形式。 |
| adgencol | “char” | 标识该列是否为生成列。取值为's'表示该列为生成列，取值为'\0'表示该列为普通列，默认值为'\0'。 |

### PG\_ATTRIBUTE

PG\_ATTRIBUTE系统表存储关于表字段的信息。

**表 1** PG\_ATTRIBUTE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| attrelid | oid | 此字段所属表。 |
| attname | name | 字段名。 |
| atttypid | oid | 字段类型。 |
| attstattarget | integer | 控制ANALYZE为这个字段积累的统计细节的级别。零值表示不收集统计信息。负数表示使用系统缺省的统计对象。正数值的确切信息是和数据类型相关的。对于标量数据类型，ATTSTATTARGET既是要收集的“最常用数值”的目标数目，也是要创建的柱状图的目标数量。 |
| attlen | smallint | 是本字段类型的pg\_type.typlen的拷贝。 |
| attnum | smallint | 字段编号。 |
| attndims | integer | 如果该字段是数组，则是维数，否则是0 。 |
| attcacheoff | integer | 在磁盘上的时候总是-1，但是如果加载入内存中的行描述器中，它可能会被更新以缓冲在行中字段的偏移量。 |
| atttypmod | integer | 记录创建新表时支持的类型特定的数据（比如一个varchar字段的最大长度）。它传递给类型相关的输入和长度转换函数当做第三个参数。其值对那些不需要ATTTYPMOD的类型通常为-1。 |
| attbyval | Boolean | 这个字段类型的pg\_type.typbyval的拷贝。 |
| attstorage | “char” | 这个字段类型的pg\_type.typstorage的拷贝。 |
| attalign | “char” | 这个字段类型的pg\_type.typalign的拷贝。 |
| attnotnull | Boolean | 这代表一个非空约束。可以改变这个字段以打开或者关闭这个约束。 |
| atthasdef | Boolean | 这个字段有一个缺省值，此时它对应pg\_attrdef表里实际定义此值的记录。 |
| attisdropped | Boolean | 这个字段已经被删除了，不再有效。一个已经删除的字段物理上仍然存在表中，但会被分析器忽略，因此不能再通过SQL访问。 |
| attislocal | Boolean | 这个字段是局部定义在关系中的。请注意一个字段可以同时是局部定义和继承的。 |
| attcmprmode | tinyint | 对某一列指定压缩方式。压缩方式包括：   * ATT\_CMPR\_NOCOMPRESS。 * ATT\_CMPR\_DELTA。 * ATT\_CMPR\_DICTIONARY。 * ATT\_CMPR\_PREFIX。 * ATT\_CMPR\_NUMSTR。 |
| attinhcount | integer | 这个字段所拥有的直接父表的个数。如果一个字段的父表个数非零，则它就不能被删除或重命名。 |
| attcollation | oid | 对此列定义的校对列。 |
| attacl | aclitem[] | 列级访问权限控制。 |
| attoptions | text[] | 字段属性。目前支持以下两种属性：n\_distinct，表示该字段的distinct值数量（不包含字表）n\_distinct\_inherited，表示该字段的distinct值数量（包含字表） |
| attfdwoptions | text[] | 外表字段属性。当前支持的dist\_fdw、file\_fdw、log\_fdw未使用外表字段属性。 |
| attinitdefval | bytea | 存储了此列默认的值表达式。行存表的ADD COLUMN需要使用此字段。 |
| attkvtype | tinyint | 对某一列指定key value类型。类型包括：   * 0. ATT\_KV\_UNDEFINED：默认。 * 1. ATT\_KV\_TAG：维度。 * 2. ATT\_KV\_FIELD：指标。 * 3. ATT\_KV\_TIMETAG：时间列。 |

### PG\_AUTH\_HISTORY

PG\_AUTH\_HISTORY系统表记录了角色的认证历史。需要有系统管理员权限才可以访问此系统表。

**表 1** PG\_AUTH\_HISTORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| roloid | oid | 角色标识。 |
| passwordtime | timestamp with time zone | 创建和修改密码的时间。 |
| rolpassword | text | 角色密码密文，加密方式由GUC参数password\_encryption\_type确定。 |

### PG\_AUTH\_MEMBERS

PG\_AUTH\_MEMBERS系统表存储显示角色之间的成员关系。

**表 1** PG\_AUTH\_MEMBERS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| roleid | oid | 拥有成员的角色ID。 |
| member | oid | 属于ROLEID角色的一个成员的角色ID。 |
| grantor | oid | 赋予此成员关系的角色ID。 |
| admin\_option | Boolean | 如果MEMBER可以把ROLEID角色的成员关系赋予其他角色，则为真。 |

### PG\_AUTHID

PG\_AUTHID系统表存储有关数据库认证标识符（角色）的信息。角色把“用户”的概念包含在内。一个用户实际上就是一个rolcanlogin标志被设置的角色。任何角色（不管rolcanlogin设置与否）都能够把其他角色作为成员。

Vastbase中只有一份pg\_authid，不是每个数据库有一份。需要有系统管理员权限才可以访问此系统表。

**表 1** PG\_AUTHID字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| rolname | name | 角色名称。 |
| rolsuper | boolean | 角色是否是拥有最高权限的初始系统管理员。 -t（true）：表示是。 -f（false）：表示不是。 |
| rolinherit | boolean | 角色是否自动继承其所属角色的权限。 -t（true）：表示自动继承。 -f（false）：表示不自动继承。 |
| rolcreaterole | boolean | 角色是否可以创建更多角色。 -t（true）：表示可以。 -f（false）：表示不可以。 |
| rolcreatedb | boolean | 角色是否可以创建数据库。 -t（true）：表示可以。 -f（false）：表示不可以。 |
| rolcatupdate | boolean | 角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。 -t（true）：表示可以。 -f（false）：表示不可以。 |
| rolcanlogin | boolean | 角色是否可以登录，也就是说，这个角色可以给予会话认证标识符。 -t（true）：表示可以。 -f（false）：表示不可以。 |
| rolreplication | boolean | 角色是否具有复制权限： -t（true）：表示有。 -f（false）：表示没有。 |
| rolauditadmin | boolean | 角色是否具有审计管理员权限： -t（true）：表示有。 -f（false）：表示没有。 |
| rolsystemadmin | boolean | 角色是否具有系统管理员权限： -t（true）：表示有。 -f（false）：表示没有。 |
| rolconnlimit | integer | 对于可以登录的角色，限制其最大并发连接数量。-1 表示没有限制。 |
| rolpassword | text | 口令（可能是加密的），如果没有口令，则为NULL。 |
| rolvalidbegin | timestamp with time zone | 帐户的有效开始时间，如果没有开始时间，则为NULL。 |
| rolvaliduntil | timestamp with time zone | 帐户的有效结束时间，如果没有结束时间，则为NULL。 |
| rolrespool | name | 用户所能够使用的resource pool。 |
| roluseft | boolean | 角色是否可以操作外表。 -t（true）：表示可以。 -f（false）：表示不可以。 |
| rolparentid | oid | 用户所在组用户的OID。 |
| roltabspace | text | 用户数据表的最大空间限额。 |
| rolkind | char | 特殊用户种类，包括私有用户、普通用户。 |
| rolnodegroup | oid | 该字段不支持。 |
| roltempspace | text | 用户临时表的最大空间限额，单位为KB。 |
| rolspillspace | text | 用户执行作业时下盘数据的最大空间限额，单位为KB。 |
| rolexcpdata | text | 用户可以设置的查询规则（当前未使用）。 |
| rolmonitoradmin | boolean | 角色是否具有监控管理员权限： -t（true）：表示有。 -f（false）：表示没有。 |
| roloperatoradmin | boolean | 角色是否具有运维管理员权限： -t（true）：表示有。 -f（false）：表示没有。 |
| rolpolicyadmin | boolean | 角色是否具有安全策略管理员权限： -t（true）：表示有。 -f（false）：表示没有。 |

### PG\_CAST

PG\_CAST系统表存储数据类型之间的转化关系。

**表 1** PG\_CAST字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| castsource | oid | 源数据类型的OID。 |
| casttarget | oid | 目标数据类型的OID。 |
| castfunc | oid | 转化函数的OID。如果为零表明不需要转化函数。 |
| castcontext | “char” | 源数据类型和目标数据类型间的转化方式： -'e'：表示只能进行显式转化（使用CAST或::语法）。 -'i'：表示能进行隐式转化。 -'a'：表示类型间同时支持隐式和显式转化。 |
| castmethod | “char” | 转化方法： -'f'：使用castfunc字段中指定的函数进行转化。 -'b'：类型间是二进制强制转化，不使用castfunc。 |

### PG\_CLASS

PG\_CLASS系统表存储数据库对象信息及其之间的关系。

**表 1** PG\_CLASS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| relname | name | 表、索引、视图等对象的名称。 |
| relnamespace | oid | 包含这个关系的名称空间的OID。 |
| reltype | oid | 对应这个表的行类型的数据类型（索引为零，因为索引没有pg\_type记录）。 |
| reloftype | oid | 复合类型的OID，0表示其他类型。 |
| relowner | oid | 关系所有者。 |
| relam | oid | 如果行是索引，则就是所用的访问模式（B-tree、hash等）。 |
| relfilenode | oid | 这个关系在磁盘上的文件的名称，如果没有则为0。 |
| reltablespace | oid | 这个关系存储所在的表空间。如果为零，则意味着使用该数据库的缺省表空间。如果关系在磁盘上没有文件，则这个字段没有什么意义。 |
| relpages | double precision | 以页（大小为BLCKSZ）为单位的此表在磁盘上的大小，它只是优化器用的一个近似值。 |
| reltuples | double precision | 表中行的数目，只是优化器使用的一个估计值。 |
| relallvisible | integer | 被标识为全可见的表中的页的数量。此字段是优化器用来做SQL执行优化使用的。VACUUM、ANALYZE和一些DDL语句（例如，CREATE INDEX）会引起此字段更新。 |
| reltoastrelid | oid | 与此表关联的TOAST表的OID，如果没有则为0。TOAST表在一个从属表里“离线”存储大字段。 |
| reltoastidxid | oid | 对于TOAST表是它的索引的OID，如果不是TOAST表则为0。 |
| reldeltarelid | oid | Delta表的OID。Delta表附属于列存表。用于存储数据导入过程中的甩尾数据。 |
| reldeltaidx | oid | Delta表的索引表OID。 |
| relcudescrelid | oid | CU描述表的OID。CU描述表（Desc表）附属于列存表。用于控制表目录中存储数据的可见性。 |
| relcudescidx | oid | CU描述表的索引表OID。 |
| relhasindex | boolean | 如果它是一个表而且至少有（或者最近有过）一个索引，则为真。它是由CREATE INDEX设置的，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，将会把它将清理relhasindex字段，将值设置为假。 |
| relisshared | boolean | 如果该表在openGauss中由所有数据库共享则为真。只有某些系统表（比如pg\_database）是共享的。 |
| relpersistence | “char” | -p：表示永久表。 -u：表示非日志表。 -g：表示临时表。 |
| relkind | “char” | -r：表示普通表。 -i：表示索引。 -I：表示分区表GLOBAL索引。 -S：表示序列。 -L：表示Large序列。 -v：表示视图。 -c：表示复合类型。 -t：表示TOAST表。 -f：表示外表。 -m：表示物化视图。 |
| relnatts | smallint | 关系中用户字段数目（除了系统字段以外）。在pg\_attribute里肯定有相同数目对应行。 |
| relchecks | smallint | 表里的检查约束的数目；参阅pg\_constraint表。 |
| relhasoids | boolean | 如果为关系中每行都生成一个OID则为真。 |
| relhaspkey | boolean | 如果这个表有一个（或者曾经有一个）主键，则为真。 |
| relhasrules | boolean | 如表有规则就为真。是否有规则可参考系统表PG\_REWRITE。 |
| relhastriggers | boolean | True表示表中有触发器，或者曾经有过触发器。系统表pg\_trigger中记录了表和视图的触发器。 |
| relhassubclass | boolean | 如果有（或者曾经有）任何继承的子表，为真。 |
| relcmprs | tinyint | 表示是否启用表的启用压缩特性。需要特别注意，当且仅当批量插入才会触发压缩，普通的CRUD并不能够触发压缩。 -0表示其他不支持压缩的表（主要是指系统表，不支持压缩属性的修改操作）。 -1表示表数据的压缩特性为NOCOMPRESS或者无指定关键字。 -2表示表数据的压缩特性为COMPRESS。 |
| relhasclusterkey | boolean | 是否有局部聚簇存储。 |
| relrowmovement | boolean | 针对分区表进行update操作时，是否允许行迁移。 -true：表示允许行迁移。 -false：表示不允许行迁移。 |
| parttype | “char” | 表或者索引是否具有分区表的性质。 -p：表示带有分区表性质。 -n：表示没有分区表特性。 -v：表示该表为HDFS的Value分区表。 -s：表示该表为二级分区表。 |
| relfrozenxid | xid32 | 该表中所有在这个之前的事务ID已经被一个固定的（“frozen”）事务ID替换。该字段用于跟踪此表是否需要为了防止事务ID重叠（或者允许收缩pg\_clog）而进行清理。如果该关系不是表则为零（InvalidTransactionId）。为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。 |
| relacl | aclitem[] | 访问权限。查询的回显结果为以下形式： rolename=xxxx/yyyy #赋予一个角色的权限 =xxxx/yyyy #赋予public的权限 xxxx表示赋予的权限，yyyy表示授予这个权限的角色。权限的参数说明请参见表2：权限的参数说明。 |
| relreplident | “char” | 逻辑解码中解码列的标识： -d = 默认 （主键，如果存在）。 -n = 无。 -f = 所有列。 -i = 索引的indisreplident被设置或者为默认。 |
| relfrozenxid64 | xid | 该表中所有在这个之前的事务ID已经被一个固定的（”frozen”）事务ID替换。该字段用于跟踪此表是否需要为了防止事务ID重叠（或者允许收缩pg\_clog）而进行清理。如果该关系不是表则为零（InvalidTransactionId）。 |
| relbucket | oid | pg\_hashbucket中的桶信息。 |
| relbucketkey | int2vector | 哈希分区列号。 |
| relminmxid | xid | 该表中所有在这个之前的多事务ID已经被一个事务ID替换。这用于跟踪该表是否需要为了防止多事务ID重叠或者允许收缩pg\_clog而进行清理。如果该关系不是表则为零(InvalidTransactionId)。 |

**表 2** 权限的参数说明

| **参数** | **参数说明** |
| --- | --- |
| r | SELECT（读） |
| w | UPDATE（写） |
| a | INSERT（插入） |
| d | DELETE |
| D | TRUNCATE |
| x | REFERENCES |
| t | TRIGGER |
| X | EXECUTE |
| U | USAGE |
| C | CREATE |
| c | CONNECT |
| T | TEMPORARY |
| A | ALTER |
| P | DROP |
| m | COMMENT |
| i | INDEX |
| v | VACUUM |
| \* | 给前面权限的授权选项 |

### PG\_COLLATION

PG\_COLLATION系统表描述可用的排序规则，本质上从一个SQL名称映射到操作系统本地类别。

**表 1** PG\_COLLATION字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| collname | name | - | 排序规则名（每个名称空间和编码唯一）。 |
| collnamespace | oid | PG\_NAMESPACE.oid | 包含这个排序规则的名称空间的OID。 |
| collowner | oid | PG\_AUTHID.oid | 排序规则的所有者。 |
| collencoding | integer | - | 排序规则可用的编码，兼容PostgreSQL所有的字符编码类型，如果适用于任意编码为-1。 |
| collcollate | name | - | 这个排序规则对象的LC\_COLLATE。 |
| collctype | name | - | 这个排序规则对象的LC\_CTYPE。 |

### PG\_CONSTRAINT

PG\_CONSTRAINT系统表存储表上的检查约束、主键和唯一约束。

**表 1** PG\_CONSTRAINT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| conname | name | 约束名称（不一定是唯一的）。 |
| connamespace | oid | 包含这个约束的名称空间的OID。 |
| contype | “char” | c = 检查约束。p = 主键约束。u = 唯一约束。t = 触发器约束。 |
| condeferrable | boolean | 这个约束是否可以推迟。 |
| condeferred | boolean | 缺省时这个约束是否可以推迟。 |
| convalidated | boolean | 约束是否有效。目前，只有外键和CHECK约束可将其设置为FALSE。 |
| conrelid | oid | 这个约束所在的表；如果不是表约束则为0。 |
| contypid | oid | 这个约束所在的域；如果不是一个域约束则为0。 |
| conindid | oid | 与约束关联的索引ID。 |
| confrelid | oid | 如果是外键，则为参考的表；否则为0。 |
| confupdtype | “char” | 外键更新动作代码。a = 没动作。r = 限制。c = 级联。n = 设置为null。d = 设置为缺省。 |
| confdeltype | “char” | 外键删除动作代码。a = 没动作。r = 限制。c = 级联。n = 设置为null。d = 设置为缺省。 |
| confmatchtype | “char” | 外键匹配类型。f = 全部。p = 部分。u = 未指定（在f的基础上允许匹配NULL值）。 |
| conislocal | boolean | 是否是为关系创建的本地约束。 |
| coninhcount | integer | 约束直接继承父表的数目。继承父表数非零时，不能删除或重命名该约束。 |
| connoinherit | boolean | 是否可以被继承。 |
| consoft | boolean | 是否为信息约束（Informational Constraint）。 |
| conopt | boolean | 是否使用信息约束优化执行计划。 |
| conenable | boolean | 是否启用约束。 |
| conkey | smallint[] | 如果是表约束，则是约束控制的字段列表。 |
| confkey | smallint[] | 如果是一个外键，是参考的字段的列表。 |
| conpfeqop | oid[] | 如果是一个外键，是做PK=FK比较的相等操作符ID的列表。 |
| conppeqop | oid[] | 如果是一个外键，是做PK=PK比较的相等操作符ID的列表。 |
| conffeqop | oid[] | 如果是一个外键，是做FK=FK比较的相等操作符ID的列表。由于当前不支持外键，所以值为空。 |
| conexclop | oid[] | 如果是一个排他约束，是列的排他操作符ID列表。 |
| conbin | pg\_node\_tree | 如果是检查约束，那就是其表达式的内部形式。 |
| consrc | text | 如果是检查约束，则是表达式的人类可读形式。 |
| conincluding | smallint[] | 不用做约束，但是会包含在INDEX中的属性列。 |

fig: **须知：**

* consrc在被引用的对象改变之后不会被更新，它不会跟踪字段的名称修改。与其依赖这个字段，最好还是使用pg\_get\_constraintdef()来抽取一个检查约束的定义。
* pg\_class.relchecks需要和在此表上为给定关系找到的检查约束的数目一致。

### PG\_CONVERSION

PG\_CONVERSION系统表描述编码转换信息。

**表 1** PG\_CONVERSION字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐藏属性，必须明确选择）。 |
| conname | name | - | 转换名称（在一个名称空间里是唯一的）。 |
| connamespace | oid | PG\_NAMESPACE.oid | 包含这个转换的名称空间的OID。 |
| conowner | oid | PG\_AUTHID.oid | 编码转换的属主。 |
| conforencoding | integer | - | 源编码ID。 |
| contoencoding | integer | - | 目的编码ID。 |
| conproc | regproc | PG\_PROC.proname | 转换过程。 |
| condefault | boolean | - | 如果这是缺省转换则为真。 |

### PG\_DATABASE

PG\_DATABASE系统表存储关于可用数据库的信息。

**表 1** PG\_DATABASE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| datname | name | 数据库名称。 |
| datdba | oid | 数据库所有人，通常为其创建者。 |
| encoding | integer | 数据库的字符编码方式。 |
| datcollate | name | 数据库使用的排序顺序。 |
| datctype | name | 数据库使用的字符分类。 |
| datistemplate | Boolean | 是否允许作为模板数据库。 |
| datallowconn | Boolean | 如果为假，则没有用户可以连接到这个数据库。这个字段用于保护template0数据库不被更改。 |
| datconnlimit | integer | 该数据库上允许的最大并发连接数，-1表示无限制。 |
| datlastsysoid | oid | 数据库里最后一个系统OID 。 |
| datfrozenxid | xid32 | 用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。当前版本该字段已经废弃使用，为保持前向兼容，保留此字段，新增datfrozenxid64用于记录此信息。 |
| dattablespace | oid | 数据库的缺省表空间。 |
| datcompatibility | name | 数据库兼容模式，当前支持四种兼容模式：A、B、C、PG，分别表示兼容O、MY、TD和POSTGRES。 |
| datacl | aclitem[] | 访问权限。 |
| datfrozenxid64 | xid | 用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。 |
| datminmxid | xid | 该数据库中中所有在这个之前的多事务ID已经被一个事务ID替换。这用于跟踪该数据库是否需要为了防止事务ID重叠或者允许收缩pg\_clog而进行清理。它是此数据库中所有表的pg\_class.relminmxid中的最小值。 |

### PG\_DB\_ROLE\_SETTING

PG\_DB\_ROLE\_SETTING系统表存储数据库运行时每个角色与数据绑定的配置项的默认值。

**表 1** PG\_DB\_ROLE\_SETTING字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| setdatabase | oid | 配置项所对应的数据库，如果未指定数据库，则为0。 |
| setrole | oid | 配置项所对应的角色，如果未指定角色，则为0。 |
| setconfig | text[] | 运行时配置项的默认值，配置方法参考：GUC参数说明->配置运行参数->重设参数章节中表2。 |

### PG\_DEFAULT\_ACL

PG\_DEFAULT\_ACL系统表存储为新建对象设置的初始权限。

**表 1** PG\_DEFAULT\_ACL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| defaclrole | oid | 与此权限相关的角色ID。 |
| defaclnamespace | oid | 与此权限相关的名称空间，如果没有，则为0。 |
| defaclobjtype | “char” | 此权限的对象类型。r表示表或视图。S表示序列。f表示函数。T表示类型。K表示客户端主密钥。k表示列加密密钥。 |
| defaclacl | aclitem[] | 创建该类型时所拥有的访问权限。 |

### PG\_DEPEND

PG\_DEPEND系统表记录数据库对象之间的依赖关系。这个信息允许DROP命令找出哪些其它对象必须由DROP CASCADE删除，或者是在DROP RESTRICT的情况下避免删除。

这个表的功能类似PG\_SHDEPEND（参考：系统表和系统视图->系统表->PG\_SHDEPEND章节），用于记录那些在Vastbase之间共享的对象之间的依赖性关系。

**表 1** PG\_DEPEND字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| classid | oid | PG\_CLASS.oid | 有依赖对象所在系统表的OID。 |
| objid | oid | 任意OID属性 | 指定的依赖对象的OID。 |
| objsubid | integer | - | 对于表字段，这个是该属性的字段数（objid和classid引用表本身）。对于所有其它对象类型，目前这个字段是0。 |
| refclassid | oid | PG\_CLASS.oid | 被引用对象所在的系统表的OID。 |
| refobjid | oid | 任意OID属性 | 指定的被引用对象的OID。 |
| refobjsubid | integer | - | 对于表字段，这个是该字段的字段号（refobjid和refclassid引用表本身）。对于所有其它对象类型，目前这个字段是0。 |
| deptype | “char” | - | 一个定义这个依赖关系特定语义的代码。 |

在所有情况下，一个PG\_DEPEND记录表示被引用的对象不能在有依赖的对象被删除前删除。不过，这里还有几种由deptype定义的情况：

* DEPENDENCY\_NORMAL (n)：独立创建的对象之间的一般关系。有依赖的对象可以在不影响被引用对象的情况下删除。被引用对象只有在声明了CASCADE的情况下删除，这时有依赖的对象也被删除。例子：一个表字段对其数据类型有一般依赖关系。
* DEPENDENCY\_AUTO (a)：有依赖对象可以和被引用对象分别删除，并且如果删除了被引用对象则应该被自动删除（不管是RESTRICT或CASCADE模式）。例子：一个表上面的命名约束是在该表上的自动依赖关系，因此如果删除了表，它也会被删除。
* DEPENDENCY\_INTERNAL (i)：有依赖的对象是作为被引用对象的一部分创建的，实际上只是它的内部实现的一部分。DROP有依赖对象是不能直接允许的（将告诉用户发出一条删除被引用对象的DROP）。一个对被引用对象的DROP将传播到有依赖对象，不管是否声明了CASCADE。
* DEPENDENCY\_EXTENSION (e)：依赖对象是被依赖对象extension的一个成员（参考：系统表和系统视图->系统表->PG\_EXTENSION章节）。依赖对象只可以通过在被依赖对象上DROP EXTENSION删除。函数上这个依赖类型和内部依赖一样动作，但是它为了清晰和简化gs\_dump保持分开。
* DEPENDENCY\_PIN (p)：没有依赖对象；这种类型的记录标志着系统本身依赖于被引用对象，因此这个对象决不能被删除。这种类型的记录只有在initdb的时候创建。有依赖对象的字段里是零。

### PG\_DESCRIPTION

PG\_DESCRIPTION系统表可以给每个数据库对象存储一个可选的描述（注释）。许多内置的系统对象的描述提供了PG\_DESCRIPTION的初始内容。

这个表的功能类似PG\_SHDESCRIPTION（参考：系统表和系统视图->系统表->PG\_SHDESCRIPTION章节），用于记录Vastbase范围内共享对象的注释。

**表 1** PG\_DESCRIPTION字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| objoid | oid | 任意OID属性 | 这条描述所描述的对象的OID。 |
| classoid | oid | PG\_CLASS.oid | 这个对象出现的系统表的OID。 |
| objsubid | integer | - | 对于一个表字段的注释，它是字段号（objoid和classoid指向表自身）。对于其它对象类型，它是零。 |
| description | text | - | 对该对象描述的任意文本。 |

### PG\_DIRECTORY

PG\_DIRECTORY系统表用于保存用户添加的directory对象可以通过CREATE DIRECTORY（参考：SQL语法参考->SQL语法->CREATE DIRECTORY章节）语句向该表中添加记录，目前只有系统管理员用户可以向该表中添加记录。

**表 1** PG\_DIRECTORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| dirname | name | 目录对象的名称。 |
| owner | oid | 目录对象的所有者。 |
| dirpath | text | 目录路径。 |
| diracl | aclitem[] | 访问权限。 |

### PG\_ENUM

PG\_ENUM系统表包含显示每个枚举类型值和标签的记录。给定枚举类型的内部表示实际上是PG\_ENUM里面相关行的OID。

**表 1** PG\_ENUM字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| enumtypid | oid | PG\_TYPE.oid | 拥有这个枚举值的pg\_type记录的OID。 |
| enumsortorder | real | - | 这个枚举值在它的枚举类型中的排序位置。 |
| enumlabel | name | - | 这个枚举值的文本标签。 |

PG\_ENUM行的OID跟着一个特殊规则：偶数的OID保证用和它们的枚举类型一样的排序顺序排序。也就是，如果两个偶数OID属于相同的枚举类型，那么较小的OID必须有较小enumsortorder值。奇数OID需要毫无关系的排序顺序。这个规则允许枚举比较例程在许多常见情况下避开目录查找。创建和修改枚举类型的例程只要可能就尝试分配偶数OID给枚举值。

当创建了一个枚举类型时，它的成员赋予了排序顺序位置1到n。但是随后添加的成员可能会分配enumsortorder的负值或分数值。对这些值的唯一要求是它们要正确的排序和在每个枚举类型中唯一。

### PG\_EXTENSION

PG\_EXTENSION系统表存储关于所安装扩展的信息。Vastbase默认扩展是PLPGSQL和MOT\_FDW。

**表 1** PG\_EXTENSION

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 数据库对象id。 |
| extname | name | 扩展名。 |
| extowner | oid | 扩展的所有者。 |
| extnamespace | oid | 扩展导出对象的名称空间。 |
| extrelocatable | Boolean | 标识此扩展是否可迁移到其他名称空间，true表示允许。 |
| extversion | text | 扩展的版本号。 |
| extconfig | oid[] | 扩展的配置信息。 |
| extcondition | text[] | 扩展配置信息的过滤条件。 |

### PG\_EXTENSION\_DATA\_SOURCE

PG\_EXTENSION\_DATA\_SOURCE系统表存储外部数据源对象的信息。一个外部数据源对象（Data Source）包含了外部数据库的一些口令编码等信息，主要配合Extension Connector使用。

**表 1** PG\_EXTENSION\_DATA\_SOURCE字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| srcname | name | - | 外部数据源对象的名称。 |
| srcowner | oid | PG\_AUTHID.oid | 外部数据源对象的所有者。 |
| srctype | text | - | 外部数据源对象的类型，缺省为空。 |
| srcversion | text | - | 外部数据源对象的版本，缺省为空。 |
| srcacl | aclitem[] | - | 访问权限。 |
| srcoptions | text[] | - | 外部数据源对象的指定选项，使用“keyword=value”格式的字符串。 |

### PG\_FOREIGN\_DATA\_WRAPPER

PG\_FOREIGN\_DATA\_WRAPPER系统表存储外部数据封装器定义。一个外部数据封装器是在外部服务器上驻留外部数据的机制，是可以访问的。

**表 1** PG\_FOREIGN\_DATA\_WRAPPER字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| fdwname | name | - | 外部数据封装器名。 |
| fdwowner | oid | PG\_AUTHID.oid | 外部数据封装器的所有者。 |
| fdwhandler | oid | PG\_PROC.oid | 引用一个负责为外部数据封装器提供扩展例程的处理函数。如果没有提供处理函数则为零。 |
| fdwvalidator | oid | PG\_PROC.oid | 引用一个验证器函数，这个验证器函数负责验证给予外部数据封装器的选项、外部服务器选项和使用外部数据封装器的用户映射的有效性。如果没有提供验证器函数则为零。 |
| fdwacl | aclitem[] | - | 访问权限。 |
| fdwoptions | text[] | - | 外部数据封装器指定选项，使用“keyword=value”格式的字符串。 |

### PG\_FOREIGN\_SERVER

PG\_FOREIGN\_SERVER系统表存储外部服务器定义。一个外部服务器描述了一个外部数据源，例如一个远程服务器。外部服务器通过外部数据封装器访问。

**表 1** PG\_FOREIGN\_SERVER字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| srvname | name | - | 外部服务器名。 |
| srvowner | oid | PG\_AUTHID.oid | 外部服务器的所有者。 |
| srvfdw | oid | PG\_FOREIGN\_DATA\_WRAPPER.oid | 这个外部服务器的外部数据封装器的OID。 |
| srvtype | text | - | 服务器的类型（可选）。 |
| srvversion | text | - | 服务器的版本（可选）。 |
| srvacl | aclitem[] | - | 访问权限。 |
| srvoptions | text[] | - | 外部服务器指定选项，使用“keyword=value”格式的字符串。 |

### PG\_FOREIGN\_TABLE

PG\_FOREIGN\_TABLE系统表存储外部表的辅助信息。

**表 1** PG\_FOREIGN\_TABLE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| ftrelid | oid | 外部表的ID。 |
| ftserver | oid | 外部表的所在服务器。 |
| ftwriteonly | Boolean | 外部表是否可写。取值如下：t（true）：表示可写。f（false）：表示不可写。 |
| ftoptions | text[] | 外部表的可选项，具体参考CREATE FOREIGN TABLE语法说明。 |

### PG\_HASHBUCKET

PG\_HASHBUCKET系统表存储hash bucket信息。

**表 1** PG\_HASHBUCKET字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段，必须明确选择）。 |
| bucketid | oid | 对bucketvector计算的hash值，通过hash值可以加速对bucketvector的查找。 |
| bucketcnt | integer | 包含分片的个数。 |
| bucketmapsize | integer | 所有DN上包含的分片总数。 |
| bucketref | integer | 预留字段，默认值为1。 |
| bucketvector | oidvector\_extend | 记录此行bucket信息包含的所有bucket的id，在此列上建立唯一索引，具有相同bucketid信息的表共享同一行pg\_hashbucket数据。 |

### PG\_INDEX

PG\_INDEX系统表存储索引的一部分信息，其他的信息大多数在PG\_CLASS(参考：系统表和系统视图->系统表->PG\_CLASS章节）中。

**表 1** PG\_INDEX字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| indexrelid | oid | 这个索引在pg\_class里的记录的OID。 |
| indrelid | oid | 使用这个索引的表在pg\_class里的记录的OID。 |
| indnatts | smallint | 索引中的字段数目。 |
| indisunique | Boolean | 如果为真，这是个唯一索引。如果为假，这不是唯一索引。 |
| indisprimary | Boolean | 如果为真，该索引代表该表的主键。这个字段为真的时候indisunique总是为真。如果为假，该索引不是该表的主键。 |
| indisexclusion | Boolean | 如果为真，该索引支持排他约束。如果为假，该索引不支持排他约束。 |
| indimmediate | Boolean | 如果为真，在插入数据时会立即进行唯一性检查。如果为假，在插入数据时不会进行唯一性检查。 |
| indisclustered | Boolean | 如果为真，则该表最后在这个索引上建了簇。如果为假，则该表没有再这个索引上建簇。 |
| indisusable | Boolean | 如果为真，该索引对insert/select可用。如果为假，该索引对insert/select不可用。 |
| indisvalid | Boolean | 如果为真，则该索引可以用于查询。如果为假，则该索引可能不完整，仍然必须在INSERT/UPDATE操作时进行更新，不过不能安全的用于查询。如果是唯一索引，则唯一属性也将不为真。 |
| indcheckxmin | Boolean | 如果为true，查询不能使用索引，直到pg\_index此行的xmin低于其快照的TransactionXmin，因为该表可能包含它们能看到的不兼容行断开的HOT链。如果为false，查询可以用于索引。 |
| indisready | Boolean | 如果为真，表示此索引对插入数据是可用的，否则，在插入或修改数据时忽略此索引。 |
| indkey | int2vector | 这是一个包含indnatts值的数组，这些数组值表示这个索引所建立的表字段。比如一个值为1 3的意思是第一个字段和第三个字段组成这个索引键字。这个数组里的零表明对应的索引属性是在这个表字段上的一个表达式，而不是一个简单的字段引用。 |
| indcollation | oidvector | 索引用到的各列的ID。 |
| indclass | oidvector | 对于索引键字里面的每个字段，这个字段都包含一个指向所使用的操作符类的OID，参阅pg\_opclass获取细节。 |
| indoption | int2vector | 存储列前标识位，该标识位是由索引的访问方法定义。 |
| indexprs | pg\_node\_tree | 表达式树（以nodeToString()形式表现）用于那些非简单字段引用的索引属性。它是一个列表，个数与INDKEY中的零值个数相同。如果所有索引属性都是简单的引用，则为空。 |
| indpred | pg\_node\_tree | 部分索引断言的表达式树（以nodeToString()的形式表现）。如果不是部分索引，则是空字符串。 |
| indisreplident | Boolean | 如果为真，则此索引的列成为逻辑解码的解码列。如果为假，则此索引的列不是逻辑解码的解码列。 |
| indnkeyatts | smallint | 索引中的总字段数，超出indnatts的部分不参与索引查询。 |

### PG\_INHERITS

PG\_INHERITS系统表记录关于表继承层次的信息。数据库里每个直接的子系表都有一条记录。间接的继承可以通过追溯记录链来判断。

**表 1** PG\_INHERITS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| inhrelid | oid | PG\_CLASS.oid | 子表的OID。 |
| inhparent | oid | PG\_CLASS.oid | 父表的OID。 |
| inhseqno | integer | - | 如果一个子表存在多个直系父表（多重继承），这个数字表明此继承字段的排列顺序。计数从1开始。 |

### PG\_JOB

PG\_JOB系统表存储用户创建的定时任务的任务详细信息，定时任务线程定时轮询pg\_job系统表中的时间，当任务到期会触发任务的执行，并更新pg\_job表中的任务状态。该系统表属于Shared Relation，所有创建的job记录对所有数据库可见。

**表 1** PG\_JOB字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| job\_id | bigint | 作业ID，主键，是唯一的（有唯一索引） |
| current\_postgres\_pid | bigint | 如果当前任务已被执行，那么此处记录运行此任务的gaussdb线程ID。默认为 -1，表示此任务未被执行过。 |
| log\_user | name | 创建者的UserName |
| priv\_user | name | 作业执行者的UserName |
| dbname | name | 标识作业要在哪个数据库执行的数据库名称 |
| node\_name | name | 标识当前作业是在哪个数据库主节点上创建和执行 |
| job\_status | “char” | 当前任务的执行状态，取值范围：('r', 's', 'f', 'd')，默认为's'，取值含义：Status of job step: r=running, s=successfully finished, f=job failed, d=disable当job连续执行失败16次，会将job\_status自动设置为失效状态'd'，后续不再执行该job。注：当用户将定时任务关闭（即：guc参数job\_queue\_processes为0时），由于监控job执行的线程不会启动，所以该状态不会根据job的实时状态进行设置，用户不需要关注此状态。只有当开启定时任务功能（即：guc参数job\_queue\_processes为非0时），系统才会根据当前job的实时状态刷新该字段值。 |
| start\_date | timestamp without time zone | 作业第一次开始执行时间，时间精确到毫秒。 |
| next\_run\_date | timestamp without time zone | 下次定时执行任务的时间，时间精确到毫秒。 |
| failure\_count | smallint | 失败计数，作业连续执行失败16次，不再继续执行。 |
| interval | text | 作业执行的重复时间间隔。 |
| last\_start\_date | timestamp without time zone | 上次运行开始时间，时间精确到毫秒。 |
| last\_end\_date | timestamp without time zone | 上次运行的结束时间，时间精确到毫秒。 |
| last\_suc\_date | timestamp without time zone | 上次成功运行的开始时间，时间精确到毫秒。 |
| this\_run\_date | timestamp without time zone | 正在运行任务的开始时间，时间精确到毫秒。 |
| nspname | name | 标识作业执行时的schema的名称。 |
| job\_name | text | DBE\_SCHEDULER定时任务专用，定时任务名称。 |
| end\_date | timestamp without time zone | DBE\_SCHEDULER定时任务专用，定时任务失效时间，时间精确到毫秒。 |
| enable | boolean | DBE\_SCHEDULER定时任务专用，定时任务启用状态：true：启用false：未启用 |
| failure\_msg | text | 最新一次执行任务报错信息。 |

### PG\_JOB\_PROC

PG\_JOB\_PROC系统表对应PG\_JOB（参考：系统表和系统视图->系统表->PG\_JOB章节）表中每个任务的作业内容（包括：PL/SQL代码块、匿名块）。将存储过程信息独立出来，如果放到PG\_JOB中，被加载到共享内存的时候，会占用不必要的空间，所以在使用的时候再进行查询获取。

**表 1** PG\_JOB\_PROC字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| job\_id | integer | 外键，关联pg\_job表中的job\_id。 |
| what | text | 作业内容，DBE\_SCHEDULER定时任务中的程序内容。 |
| job\_name | text | DBE\_SCHEDULER定时任务专用，定时任务或程序名称。 |

### PG\_LANGUAGE

PG\_LANGUAGE系统表登记编程语言，用户可以用这些语言或接口写函数或者存储过程。

**表 1** PG\_LANGUAGE字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| lanname | name | - | 语言的名称。 |
| lanowner | oid | PG\_AUTHID.oid | 语言的所有者。 |
| lanispl | Boolean | - | 对于内部语言而言是假（比如SQL），对于用户定义的语言则是真。目前，gs\_dump仍然使用这个东西判断哪种语言需要转储，但是这些可能在将来被其它机制取代。 |
| lanpltrusted | Boolean | - | 如果这是可信语言则为真，意味着系统相信它不会被授予任何正常SQL执行环境之外的权限。只有初始用户可以用不可信的语言创建函数。 |
| lanplcallfoid | oid | PG\_PROC.oid | 对于非内部语言，这是指向该语言处理器的引用，语言处理器是一个特殊函数，负责执行以某种语言写的所有函数。 |
| laninline | oid | PG\_PROC.oid | 这个字段引用一个负责执行“inline”匿名代码块的函数（DO块）。如果不支持内联块则为零。 |
| lanvalidator | oid | PG\_PROC.oid | 这个字段引用一个语言校验器函数，它负责检查新创建的函数的语法和有效性。如果没有提供校验器，则为零。 |
| lanacl | aclitem[] | - | 访问权限。 |

### PG\_LARGEOBJECT

PG\_LARGEOBJECT系统表保存那些标记着“大对象”的数据。一个大对象是使用其创建时分配的OID标识的。每个大对象都分解成足够小的小段或者“页面”以便以行的形式存储在PG\_LARGEOBJECT里。每页的数据定义为LOBLKSIZE。

需要有系统管理员权限才可以访问此系统表。

**表 1** PG\_LARGEOBJECT字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| loid | oid | PG\_LARGEOBJECT\_METADATA.oid | 包含本页的大对象的标识符。 |
| pageno | integer | - | 本页在其大对象数据中的页码从零开始计算。 |
| data | bytea | - | 存储在大对象中的实际数据。这些数据绝不会超过LOBLKSIZE字节，而且可能更少。 |

PG\_LARGEOBJECT的每一行保存一个大对象的一个页面，从该对象内部的字节偏移（pageno \* LOBLKSIZE）开始。这种实现允许松散的存储：页面可以丢失，而且可以比LOBLKSIZE字节少（即使它们不是对象的最后一页）。大对象内丢失的部分读做零。

### PG\_LARGEOBJECT\_METADATA

PG\_LARGEOBJECT\_METADATA系统表存储与大数据相关的元数据。实际的大对象数据存储在[PG\_LARGEOBJECT（参考：系统表和系统视图->系统表->PG\_LARGEOBJECT章节）里。

**表 1** PG\_LARGEOBJECT\_METADATA字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| lomowner | oid | PG\_AUTHID.oid | 大对象的所有者。 |
| lomacl | aclitem[] | - | 访问权限。 |

### PG\_NAMESPACE

PG\_NAMESPACE系统表存储名称空间，即存储schema相关的信息。

**表 1** PG\_NAMESPACE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| nspname | name | 名称空间的名称。 |
| nspowner | oid | 名称空间的所有者。 |
| nsptimeline | bigint | 在数据库节点上创建此命名空间时的时间线。此字段为内部使用，仅在数据库节点上有效。 |
| nspacl | aclitem[] | 访问权限。 |
| in\_redistribution | “char” | 是否处于重发布状态。 |
| nspblockchain | Boolean | 如果为真，则该模式为防篡改模式。如果为假，则此模式为非防篡改模式。 |

### PG\_OBJECT

PG\_OBJECT系统表存储限定类型对象（普通表、索引、序列、视图、存储过程和函数）的创建用户、创建时间和最后修改时间。

**表 1** PG\_OBJECT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| object\_oid | oid | 对象标识符。 |
| object\_type | “char” | 对象类型：r 表示普通表。i 表示索引。s 表示序列。l 表示Large序列。v 表示视图。p 表示存储过程和函数。 |
| creator | oid | 创建用户的标识符。 |
| ctime | timestamp with time zone | 对象的创建时间。 |
| mtime | timestamp with time zone | 对象的最后修改时间，修改行为包括ALTER操作和GRANT、REVOKE操作。 |
| createcsn | int8 | 对象创建时的CSN。 |
| changecsn | int8 | 对表或索引执行DDL操作时的CSN。 |

fig: **须知：**

* 无法记录初始化数据库（initdb）过程中所创建或修改的对象，即PG\_OBJECT无法查询到该对象记录。
* 对于升级前创建的对象，再次修改时会记录其修改时间（mtime），对表或索引执行DDL操作时会记录其所属事务的事务提交序列号（changecsn）。由于无法得知该对象创建时间，因此ctime和createcsn为空。
* ctime和mtime所记录的时间为用户当次操作所属事务的起始时间。
* 由扩容引起的对象修改时间也会被记录。
* createcsn和changecsn记录的是用户当次操作所属事务的事务提交序列号。

### PG\_OPCLASS

PG\_OPCLASS系统表定义索引访问方法操作符类。

每个操作符类为一种特定数据类型和一种特定索引访问方法定义索引字段的语义。一个操作符类本质上指定一个特定的操作符族适用于一个特定的可索引的字段数据类型。索引的字段实际可用的族中的操作符集是接受字段的数据类型作为它们的左边的输入的那个。

**表 1** PG\_OPCLASS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| opcmethod | oid | PG\_AM.oid | 操作符类所服务的索引访问方法。 |
| opcname | name | - | 这个操作符类的名称。 |
| opcnamespace | oid | PG\_NAMESPACE.oid | 这个操作符类的名称空间。 |
| opcowner | oid | PG\_AUTHID.oid | 操作符类属主。 |
| opcfamily | oid | PG\_OPFAMILY.oid | 包含该操作符类的操作符族。 |
| opcintype | oid | PG\_TYPE.oid | 操作符类索引的数据类型。 |
| opcdefault | boolean | - | 如果操作符类是opcintype的缺省，则为真。 |
| opckeytype | oid | PG\_TYPE.oid | 索引数据的类型，如果和opcintype相同则为零。 |

一个操作符类的opcmethod必须匹配包含它的操作符族的opfmethod。

### PG\_OPERATOR

PG\_OPERATOR系统表存储有关操作符的信息。

**表 1** PG\_OPERATOR字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| oprname | name | - | 操作符的名称。 |
| oprnamespace | oid | PG\_NAMESPACE.oid | 包含此操作符的名称空间的OID。 |
| oprowner | oid | PG\_AUTHID.oid | 操作符所有者。 |
| oprkind | “char” | - | b=infix =中缀（“两边”）。l=前缀（“左边”）。r=后缀（“右边”）。 |
| oprcanmerge | Boolean | - | 这个操作符是否支持合并连接。t（true）：表示支持合并连接。f（false）：表示不支持合并连接。 |
| oprcanhash | Boolean | - | 这个操作符是否支持Hash连接。t（true）：表示支持Hash连接。f（false）：表示不支持Hash连接。 |
| oprleft | oid | PG\_TYPE.oid | 左操作数的类型。 |
| oprright | oid | PG\_TYPE.oid | 右操作数的类型。 |
| oprresult | oid | PG\_TYPE.oid | 结果类型。 |
| oprcom | oid | PG\_OPERATOR.oid | 此操作符的交换符，如果存在的话。 |
| oprnegate | oid | PG\_OPERATOR.oid | 此操作符的反转器，如果存在的话。 |
| oprcode | regproc | PG\_PROC.proname | 实现这个操作符的函数。 |
| oprrest | regproc | PG\_PROC.proname | 此操作符的约束选择性计算函数。 |
| oprjoin | regproc | PG\_PROC.proname | 此操作符的连接选择性计算函数。 |

### PG\_OPFAMILY

PG\_OPFAMILY系统表定义操作符族。

每个操作符族是一个操作符和相关支持例程的集合，其中的例程实现为一个特定的索引访问方式指定的语义。另外，族中的操作符都是“兼容的”，通过由访问方式指定的方法。操作符族的概念允许交叉数据类型操作符和索引一起使用，并且合理的使用访问方式的语义的知识。

**表 1** PG\_OPFAMILY字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| opfmethod | oid | PG\_AM.oid | 操作符族使用的索引方法。 |
| opfname | name | - | 这个操作符族的名称。 |
| opfnamespace | oid | PG\_NAMESPACE.oid | 这个操作符的名称空间。 |
| opfowner | oid | PG\_AUTHID.oid | 操作符族的所有者。 |

定义一个操作符族的大多数信息不在它的PG\_OPFAMILY行里面，而是在相关的行参考：系统表和系统视图->系统表->PG\_AMOP、PG\_AMPROC、PG\_OPCLASS章节。

### PG\_PARTITION

PG\_PARTITION系统表存储数据库内所有分区表（partitioned table）、分区（table partition）、分区上toast表和分区索引（index partition）四类对象的信息。分区表索引（partitioned index）的信息不在PG\_PARTITION系统表中保存。

**表 1** PG\_PARTITION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| relname | name | 分区表、分区、分区上toast表和分区索引的名称。 |
| parttype | “char” | 对象类型：'r'：partitioned table。'p'：table partition。's'：table subpartition。'x'：index partition。't'：toast table。 |
| parentid | oid | 当对象为分区表或分区时，此字段表示分区表在PG\_CLASS中的OID。当对象为index partition时，此字段表示所属分区表索引（partitioned index）的OID。 |
| rangenum | integer | 保留字段。 |
| intervalnum | integer | 保留字段。 |
| partstrategy | “char” | 分区表分区策略，现在仅支持：'r'：范围分区。'v'：数值分区。'i'：间隔分区。'l'：list分区。'h'：hash分区。'n'：无效分区。 |
| relfilenode | oid | table partition、index partition、分区上toast表的物理存储位置。 |
| reltablespace | oid | table partition、index partition、分区上toast表所属表空间的OID。 |
| relpages | double precision | 统计信息：table partition、index partition的数据页数量。 |
| reltuples | double precision | 统计信息：table partition、index partition的元组数。 |
| relallvisible | integer | 统计信息：table partition、index partition的可见数据页数。 |
| reltoastrelid | oid | table partition所对应toast表的OID。 |
| reltoastidxid | oid | table partition所对应toast表的索引的OID。 |
| indextblid | oid | index partition对应table partition的OID。 |
| indisusable | boolean | 分区索引是否可用。 |
| reldeltarelid | oid | Delta表的OID。 |
| reldeltaidx | oid | Delta表的索引表的OID。 |
| relcudescrelid | oid | CU描述表的OID。 |
| relcudescidx | oid | CU描述表的索引表的OID。 |
| relfrozenxid | xid32 | 冻结事务ID号。为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。 |
| intspnum | integer | 间隔分区所属表空间的个数。 |
| partkey | int2vector | 分区键的列号。 |
| intervaltablespace | oidvector | 间隔分区所属的表空间，间隔分区以round-robin方式落在这些表空间内。 |
| interval | text[] | 间隔分区的间隔值。 |
| boundaries | text[] | 范围分区和间隔分区的上边界。 |
| transit | text[] | 间隔分区的跳转点。 |
| reloptions | text[] | 设置partition的存储属性，与pg\_class.reloptions的形态一样，用“keyword=value”格式的字符串来表示，目前用于在线扩容的信息搜集。 |
| relfrozenxid64 | xid | 冻结事务ID号。 |
| relminmxid | xid | 冻结多事务ID号。 |

### PG\_PLTEMPLATE

PG\_PLTEMPLATE系统表存储过程语言的“模板”信息。

**表 1** PG\_PLTEMPLATE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| tmplname | name | 这个模板所应用的语言的名称。 |
| tmpltrusted | Boolean | 如果语言被认为是可信的，则为真。 |
| tmpldbacreate | Boolean | 如果语言是由数据库所有者创建的，则为真。 |
| tmplhandler | text | 调用处理器函数的名称。 |
| tmplinline | text | 匿名块处理器的名称，若没有则为NULL。 |
| tmplvalidator | text | 校验函数的名称，如果没有则为NULL。 |
| tmpllibrary | text | 实现语言的共享库的路径。 |
| tmplacl | aclitem[] | 模板的访问权限（未使用）。 |

### PG\_PROC

PG\_PROC系统表存储函数或过程的信息。

**表 1** PG\_PROC字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| proname | name | 函数名称。 |
| pronamespace | oid | 包含该函数名称空间的OID。 |
| proowner | oid | 函数的所有者。 |
| prolang | oid | 这个函数的实现语言或调用接口。 |
| procost | real | 估算的执行成本。 |
| prorows | real | 估算的影响行的数目。 |
| provariadic | oid | 参数元素的数据类型。 |
| protransform | regproc | 此函数的简化调用方式。 |
| proisagg | boolean | 函数是聚集函数。t（true）：表示是。f（false）：表示不是。 |
| proiswindow | boolean | 函数是窗口函数。t（true）：表示是。f（false）：表示不是。 |
| prosecdef | boolean | 函数是一个安全定义器（也就是一个“setuid”函数）。t（true）：表示是。f（false）：表示不是。 |
| proleakproof | boolean | 函数没副作用。如果函数没有对参数进行防泄露处理，则会抛出错误。t（true）：表示没副作用。f（false）：表示有副作用。 |
| proisstrict | boolean | 如果任何调用参数是空，则函数返回空。这时函数实际上连调用都不调用。不是“strict”的函数必须准备处理空输入。 |
| proretset | boolean | 函数返回一个集合（也就是说，指定数据类型的多个数值）。 |
| provolatile | “char” | 告诉该函数的结果是否只依赖于它的输入参数，或者还会被外界因素影响。i：“不可变的”（immutable）函数，这样的函数对于相同的输入总是产生相同的结果。s：“稳定的”（stable）函数它是s，（对于固定输入）其结果在一次扫描里不变。v：“易变”（volatile）函数它是v，其结果可能在任何时候变化v也用于那些有副作用的函数，因此调用它们无法得到优化。 |
| pronargs | smallint | 参数数目。 |
| pronargdefaults | smallint | 有默认值的参数数目。 |
| prorettype | oid | 返回值的数据类型。 |
| proargtypes | oidvector | 一个存放函数参数的数据类型的数组。数组里只包括输入参数（包括INOUT参数）此代表该函数的调用签名（接口）。 |
| proallargtypes | oid[] | 一个包含函数参数的数据类型的数组。数组里包括所有参数的类型（包括OUT和INOUT参数），如果所有参数都是IN参数，则这个字段就会是空。请注意数组下标是以1为起点的，而因为历史原因，proargtypes的下标起点为0。 |
| proargmodes | “char”[] | 一个保存函数参数模式的数组，编码如下：i表示IN参数。o表示OUT参数。b表示INOUT参数。v表示VARIADIC参数。如果所有参数都是IN参数，则这个字段为空。请注意，下标对应的是proallargtypes的位置，而不是proargtypes。 |
| proargnames | text[] | 一个保存函数参数的名称的数组。没有名称的参数在数组里设置为空字符串。如果没有一个参数有名称，这个字段将是空。请注意，此数组的下标对应proallargtypes而不是proargtypes。 |
| proargdefaults | pg\_node\_tree | 默认值的表达式树。是PRONARGDEFAULTS元素的列表。 |
| prosrc | text | 描述函数或存储过程的定义。例如，对于解释型语言来说就是函数的源程序，或者一个链接符号，一个文件名，或者函数和存储过程创建时指定的其他任何函数体内容，具体取决于语言/调用习惯的实现。 |
| probin | text | 关于如何调用该函数的附加信息。同样，其含义也是和语言相关的。 |
| proconfig | text[] | 函数针对运行时配置变量的本地设置。 |
| proacl | aclitem[] | 访问权限。具体请参考：SQL语法参考->SQL语法->GRANT、REVOKE章节)。 |
| prodefaultargpos | int2vector | 函数具有默认值的入参的位置。 |
| fencedmode | boolean | 函数的执行模式，表示函数是在fence还是not fence模式下执行。如果是fence执行模式，函数的执行会在重新fork的进程中执行。用户创建的C函数，fencedmode字段默认值均为true，即fence模式；系统内建函数，fencedmode字段均为false，即not fence模式。 |
| proshippable | boolean | 表示该函数是否可以下推到数据库节点上执行，默认值是false。对于IMMUTABLE类型的函数，函数始终可以下推到数据库节点上执行。对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到数据库节点执行。 |
| propackage | boolean | 表示该函数是否支持重载，默认值是false。t（true）：表示支持。f（false）：表示不支持。 |
| prokind | “char” | 表示该对象为函数还是存储过程：值为'f'表示该对象为函数。值为'p'表示该对象为存储过程。 |
| proargsrc | text | 描述兼容oracle语法定义的函数或存储过程的参数输入字符串，包括参数注释。默认值为NULL。 |
| proisprivate | boolean | 描述函数是否是PACKAGE内的私有函数，默认为false。 |
| propackageid | oid | 函数所属的package oid，如果不在package内，则为0。 |
| proargtypesext | oidvector\_extend | 当函数参数较多时，用来存放函数参数的数据类型的数组。数组里只包括输入参数（包括INOUT参数）此代表该函数的调用签名（接口）。 |
| prodefaultargposext | int2vector\_extend | 当函数参数较多时，函数具有默认值的入参的位置。 |
| allargtypes | oidvector | 不区分参数类型，包含存储过程所有参数（包含入参、出参、INOUT参数）。 |
| allargtypesext | oidvector\_extend | 当函数参数较多时，用来存放函数参数的数据类型的数组。数组里包含所有参数（包含入参、出参、INOUT参数）。 |

### PG\_PUBLICATION

系统表pg\_publication包含当前数据库中创建的所有publication。

**表 1** PG\_PUBLICATION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pubname | text | publication的名称。 |
| pubowner | oid | publication的拥有者。 |
| puballtables | bool | 如果为真，这个publication自动包括数据库中的所有表，包括未来将会创建的任何表。 |
| pubinsert | bool | 如果为真，为publication中的表复制INSERT操作。 |
| pubupdate | bool | 如果为真，为publication中的表复制UPDATE操作。 |
| pubdelete | bool | 如果为真，为publication中的表复制DELETE操作。 |

### PG\_PUBLICATION\_REL

系统表PG\_PUBLICATION\_REL包含当前数据库中的表和publication之间的映射，这是一种多对多映射。

**表 1** PG\_PUBLICATION\_REL字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid |  | 行标识符（隐含属性，必须明确选择）。 |
| prpubid | oid | - | 对publication的引用。 |
| prrelid | oid | - | 对表的引用。 |

### PG\_RANGE

PG\_RANGE系统表存储关于范围类型的信息。除了PG\_TYPE（参考：系统表和系统视图->系统表->PG\_TYPE章节）里类型的记录。

**表 1** PG\_RANGE字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| rngtypid | oid | PG\_TYPE.oid | 范围类型的OID。 |
| rngsubtype | oid | PG\_TYPE.oid | 这个范围类型的元素类型（子类型）的OID。 |
| rngcollation | oid | PG\_COLLATION.oid | 用于范围比较的排序规则的OID，如果没有则为零。 |
| rngsubopc | oid | PG\_OPCLASS.oid | 用于范围比较的子类型的操作符类的OID。 |
| rngcanonical | regproc | PG\_PROC.proname | 转换范围类型为规范格式的函数名，如果没有则为0。 |
| rngsubdiff | regproc | PG\_PROC.proname | 返回两个double precision元素值的不同的函数名，如果没有则为0。 |

rngsubopc（如果元素类型是可排序的，则加上rngcollation）决定用于范围类型的排序顺序。当元素类型是离散的时使用rngcanonical。

### PG\_REPLICATION\_ORIGIN

PG\_REPLICATION\_ORIGIN系统表包含所有已创建的复制源，该表为全局共享表，即在每个节点上只有一份pg\_replication\_origin，而不是每个数据库一份。

**表 1** PG\_REPLICATION\_ORIGIN字段

名称 类型 描述 roident oid 一个集群范围内唯一的复制源标识符。 roname text 外部的由用户定义的复制源名称。

### PG\_RESOURCE\_POOL

PG\_RESOURCE\_POOL系统表提供了数据库资源池的信息。

**表 1** PG\_RESOURCE\_POOL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| respool\_name | name | 资源池名称。 |
| mem\_percent | integer | 内存配置的百分比。 |
| cpu\_affinity | bigint | CPU绑定core的数值。 |
| control\_group | name | 资源池所在的control group名称。 |
| active\_statements | integer | 资源池上最大的并发数。 |
| max\_dop | integer | 最大并发度。用作扩容的接口，表示数据重分布时，扫描并发度。 |
| memory\_limit | name | 资源池最大的内存。 |
| parentid | oid | 父资源池OID。 |
| io\_limits | integer | 每秒触发IO的次数上限。行存单位是万次/s，列存是次/s。 |
| io\_priority | name | IO利用率高达90%时，重消耗IO作业进行IO资源管控时关联的优先级等级。 |
| nodegroup | name | 表示资源池所在的逻辑openGauss的名称。 |
| is\_foreign | boolean | 表示资源池是否用于逻辑openGauss之外的用户。如果为true，表示资源池用来控制不属于当前资源池的普通用户的资源。 |
| max\_worker | integer | 只用于扩容的接口，表示扩容数据重分布时，表内并发度。 |

注：max\_dop和max\_worker用户扩容，不适用于Vastbase。

### PG\_REWRITE

PG\_REWRITE系统表存储为表和视图定义的重写规则。

**表 1** PG\_REWRITE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| rulename | name | 规则名称。 |
| ev\_class | oid | 使用这条规则的表名称。 |
| ev\_attr | smallint | 这条规则适用的字段（目前总是为零，表示整个表）。 |
| ev\_type | “char” | 规则适用的事件类型：1 = SELECT。2 = UPDATE。3 = INSERT。4 = DELETE。 |
| ev\_enabled | “char” | 用于控制复制的触发。O =“origin”和“local”模式时触发。D =禁用触发。R =“replica”时触发。A ＝任何模式是都会触发。 |
| is\_instead | boolean | 如果该规则是INSTEAD规则，则为真。 |
| ev\_qual | pg\_node\_tree | 规则的资格条件的表达式树（以nodeToString()形式存在）。 |
| ev\_action | pg\_node\_tree | 规则动作的查询树（以nodeToString()形式存在）。 |

### PG\_RLSPOLICY

PG\_RLSPOLICY系统表存储行级访问控制策略。

**表 1** PG\_RLSPOLICY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| polname | name | 行级访问控制策略的名称。 |
| polrelid | oid | 行级访问控制策略作用的表对象oid。 |
| polcmd | “char” | 行级访问控制策略影响的SQL操作。 |
| polpermissive | boolean | 行级访问控制策略的属性，t为表达式OR条件拼接，f为表达式AND条件拼接。 |
| polroles | oid[] | 行级访问控制策略影响的用户oid列表，不指定表示影响所有的用户。 |
| polqual | pg\_node\_tree | 行级访问控制策略的表达式。 |

### PG\_SECLABEL

PG\_SECLABEL系统表存储数据对象上的安全标签。

PG\_SHSECLABEL（参考：系统表和系统视图->系统表->PG\_SHSECLABEL章节）的作用类似，只是它是用于在一个Vastbase内共享的数据库对象的安全标签上的。

**表 1** PG\_SECLABEL字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| objoid | oid | 任意OID属性 | 这个安全标签所属的对象的OID。 |
| classoid | oid | PG\_CLASS.oid | 出现这个对象的系统目录的OID。 |
| objsubid | integer | - | 出现在这个对象中的列的序号。 |
| provider | text | - | 与这个标签相关的标签提供程序。 |
| label | text | - | 应用于这个对象的安全标签。 |

### PG\_SHDEPEND

PG\_SHDEPEND系统表记录数据库对象和共享对象（比如角色）之间的依赖性关系。这些信息允许Vastbase保证在企图删除这些对象之前，这些对象是没有被引用的。

PG\_DEPEND(参考：系统表和系统视图->系统表->PG\_DEPEND章节）的作用类似，只是它是用于在一个数据库内部的对象的依赖性关系的。

和其它大多数系统表不同，PG\_SHDEPEND是在Vastbase里面所有的数据库之间共享的：每个Vastbase只有一个PG\_SHDEPEND，而不是每个数据库一个。

**表 1** PG\_SHDEPEND字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| dbid | oid | PG\_DATABASE.oid | 依赖对象所在的数据库的OID，如果是共享对象，则为零。 |
| classid | oid | PG\_CLASS.oid | 依赖对象所在的系统表的OID。 |
| objid | oid | 任意OID属性 | 指定的依赖对象的OID。 |
| objsubid | integer | - | 对于一个表字段，这是字段号（objid和classid参考表本身）。对于所有其他对象类型，这个字段为零。 |
| refclassid | oid | PG\_CLASS.oid | 被引用对象所在的系统表的OID（必须是一个共享表）。 |
| refobjid | oid | 任意OID属性 | 指定的被引用对象的OID。 |
| deptype | “char” | - | 一段代码，定义了这个依赖性关系的特定语义；参阅下文。 |
| objfile | text | - | 用户定义函数库文件路径。 |

在任何情况下，一条PG\_SHDEPEND记录就表明这个被引用的对象不能在未删除依赖对象的前提下删除。不过，deptype同时还标出了几种不同的子风格：

* SHARED\_DEPENDENCY\_OWNER (o)
* 被引用的对象（必须是一个角色）是依赖对象的所有者。
* SHARED\_DEPENDENCY\_ACL (a)
* 被引用的对象（必须是一个角色）在依赖对象的ACL（访问控制列表，也就是权限列表）里提到。SHARED\_DEPENDENCY\_ACL不会在对象的所有者头上添加的，因为所有者会有一个SHARED\_DEPENDENCY\_OWNER记录。
* SHARED\_DEPENDENCY\_PIN (p)
* 没有依赖对象；这类记录标识系统自身依赖于该被依赖对象，因此这样的对象绝对不能被删除。这种类型的记录只是由initdb创建。这样的依赖对象的字段都是零。
* SHARED\_DEPENDENCY\_DBPRIV(d)
* 被引用的对象（必须是一个角色）具有依赖对象所对应的ANY权限（指定的依赖对象的OID对应的是系统表gs\_db\_privilege中一行）。

### PG\_SHDESCRIPTION

PG\_SHDESCRIPTION系统表为共享数据库对象存储可选的注释。可以使用[COMMENT命令（参考：SQL语法参考->SQL语法->COMMENT章节）操作注释的内容，使用psql的\d命令查看注释内容。

PG\_DESCRIPTION提供了类似的功能，它记录了单个数据库中对象的注释。

不同于大多数系统表，PG\_SHDESCRIPTION是在Vastbase里面所有的数据库之间共享的：每个Vastbase只有一个PG\_SHDESCRIPTION，而不是每个数据库一个。

**表 1** PG\_SHDESCRIPTION字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| objoid | oid | 任意OID属性 | 这条描述所描述的对象的OID。 |
| classoid | oid | PG\_CLASS.oid | 这个对象出现的系统表的OID。 |
| description | text | - | 作为对该对象的描述的任意文本。 |

### PG\_SHSECLABEL

PG\_SHSECLABEL系统表存储在共享数据库对象上的安全标签。安全标签可以用SECURITY LABEL命令操作。

查看安全标签的简单点的方法，参考：系统表和系统视图->系统视图->PG\_SECLABELS。

PG\_SECLABEL的作用类似，只是它是用于在单个数据库内部的对象的安全标签的。

不同于大多数的系统表，PG\_SHSECLABEL在Vastbase中的所有数据库中共享：每个Vastbase只有一个PG\_SHSECLABEL，而不是每个数据库一个。

**表 1** PG\_SHSECLABEL字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| objoid | oid | 任意OID属性 | 这个安全标签所属的对象的OID。 |
| classoid | oid | PG\_CLASS.oid | 出现这个对象的系统目录的OID。 |
| provider | text | - | 与这个标签相关的标签提供程序。 |
| label | text | - | 应用于这个对象的安全标签。 |

### PG\_STATISTIC

PG\_STATISTIC系统表存储有关该数据库中表和索引列的统计数据。默认只有系统管理员权限才可以访问此系统表，普通用户需要授权才可以访问。

**表 1** PG\_STATISTIC字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| starelid | oid | 所描述的字段所属的表或者索引。 |
| starelkind | “char” | 所属对象的类型。 |
| staattnum | smallint | 所描述的字段在表中的编号，从1开始。 |
| stainherit | Boolean | 是否统计有继承关系的对象。 |
| stanullfrac | real | 该字段中为NULL的记录的比率。 |
| stawidth | integer | 非NULL记录的平均存储宽度，以字节计。 |
| stadistinct | real | 标识全局统计信息中数据库节点上字段里唯一的非NULL数据值的数目。一个大于零的数值是独立数值的实际数目。一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。零值表示独立数值的数目未知。 |
| stakindN | smallint | 一个编码，表示这种类型的统计存储在pg\_statistic行的第n个“槽位”。n的取值范围：1～5 |
| staopN | oid | 一个用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。n的取值范围：1～5 |
| stanumbersN | real[] | 第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。n的取值范围：1～5 |
| stavaluesN | anyarray | 第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则就是NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好地办法。n的取值范围：1～5 |
| stadndistinct | real | 标识dn1上字段里唯一的非NULL数据值的数目。一个大于零的数值是独立数值的实际数目。一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。零值表示独立数值的数目未知。 |
| staextinfo | text | 统计信息的扩展信息。预留字段。 |

fig: **须知：**

PG\_STATISTIC系统表存储了统计对象的一些敏感信息，如高频值MCV。系统管理员和授权后的其他用户可以通过访问PG\_STATISTIC系统表查询到统计对象的这些敏感信息。

### PG\_STATISTIC\_EXT

PG\_STATISTIC\_EXT系统表存储有关该数据库中表的扩展统计数据，包括多列统计数据和表达式统计数据（后续支持）。收集哪些扩展统计数据是由用户指定的。需要有系统管理员权限才可以访问此系统表。

**表 1** PG\_STATISTIC\_EXT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| starelid | oid | 所描述的字段所属的表或者索引。 |
| starelkind | “char” | 所属对象的类型，'c'表示普通表，'p'表示分区表。 |
| stainherit | Boolean | 是否统计有继承关系的对象。 |
| stanullfrac | real | 该字段中为NULL的记录的比率。 |
| stawidth | integer | 非NULL记录的平均存储宽度，以字节计。 |
| stadistinct | real | 标识全局统计信息中数据库节点上字段里唯一的非NULL数据值的数目。一个大于零的数值是独立数值的实际数目。一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。零值表示独立数值的数目未知。 |
| stadndistinct | real | 标识dn1上字段里唯一的非NULL数据值的数目。一个大于零的数值是独立数值的实际数目。一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。零值表示独立数值的数目未知。 |
| stakindN | smallint | 一个编码，表示这种类型的统计存储在pg\_statistic行的第n个“槽位”。n的取值范围：1～5 |
| staopN | oid | 一个用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。n的取值范围：1～5 |
| stakey | int2vector | 所描述的字段编号的数组。 |
| stanumbersN | real[] | 第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。n的取值范围：1～5 |
| stavaluesN | anyarray | 第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则就是NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好地办法。n的取值范围：1～5 |
| staexprs | pg\_node\_tree | 扩展统计信息对应的表达式。 |

fig: **须知：**

PG\_STATISTIC\_EXT系统表存储了统计对象的一些敏感信息，如高频值MCV。系统管理员和授权后的其他用户可以通过访问PG\_STATISTIC\_EXT系统表查询到统计对象的这些敏感信息。

### PG\_SUBSCRIPTION

系统表PG\_SUBSCRIPTION包含所有现有的逻辑复制订阅。需要有系统管理员权限才可以访问此系统表。

和大部分系统表不同，pg\_subscription在数据库实例的所有数据库之间共享，即在每个节点上有只有一份pg\_replication\_origin，而不是每个数据库一份。

**表 1** PG\_SUBSCRIPTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| subdbid | oid | 订阅所在的数据库的OID。 |
| subname | text | 订阅的名称。 |
| subowner | oid | 订阅的拥有者。 |
| subenabled | bool | 如果为真，订阅被启用并且应该被复制。 |
| subconninfo | text | 到发布端数据库的连接信息。 |
| subslotname | text | 发布端数据库中复制槽的名称。空表示为NONE。 |
| subsynccommit | text | 订阅worker的synchronous\_commit设置的值。 |
| subpublications | text[] | 被订阅的publication名称的数组。这些引用的是发布者服务器上的publication。 |

### PG\_SYNONYM

PG\_SYNONYM系统表存储同义词对象名与其他数据库对象名间的映射信息。

**表 1** PG\_SYNONYM字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 数据库对象id。 |
| synname | name | 同义词名称。 |
| synnamespace | oid | 包含该同义词的名字空间的OID。 |
| synowner | oid | 同义词的所有者，通常是创建它的用户OID。 |
| synobjschema | name | 关联对象指定的模式名。 |
| synobjname | name | 关联对象名。 |

### PG\_TABLESPACE

PG\_TABLESPACE系统表存储表空间信息。

**表 1** PG\_TABLESPACE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| spcname | name | 表空间名称。 |
| spcowner | oid | 表空间的所有者，通常是创建它的人。 |
| spcacl | aclitem[] | 参考：SQL语法参考->SQL语法->GRANT、REVOKE章节。 |
| spcoptions | text[] | 表空间的选项。 |
| spcmaxsize | text | 可使用的最大磁盘空间大小，单位Byte。 |
| relative | boolean | 标识表空间指定的存储路径是否为相对路径。 |
| spcthreshold | spcthreshold | 表空间阈值。 |

### PG\_TRIGGER

PG\_TRIGGER系统表存储触发器信息。

**表 1** PG\_TRIGGER字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| tgrelid | oid | 触发器所在表的OID。 |
| tgname | name | 触发器名。 |
| tgfoid | oid | 要被触发器调用的函数。 |
| tgtype | smallint | 触发器类型。 |
| tgenabled | “char” | O =触发器在“origin”和“local”模式下触发。D =触发器被禁用。R =触发器在“replica”模式下触发。A =触发器始终触发。 |
| tgisinternal | boolean | 内部触发器标识，如果为true表示内部触发器。 |
| tgconstrrelid | oid | 完整性约束引用的表。 |
| tgconstrindid | oid | 完整性约束的索引。 |
| tgconstraint | oid | 约束触发器在pg\_constraint中的OID。 |
| tgdeferrable | boolean | 约束触发器是为DEFERRABLE类型。 |
| tginitdeferred | boolean | 约束触发器是否为INITIALLY DEFERRED类型。 |
| tgnargs | smallint | 触发器函数入参个数。 |
| tgattr | int2vector | 当触发器指定列时的列号，未指定则为空数组。 |
| tgargs | bytea | 传递给触发器的参数。 |
| tgqual | pg\_node\_tree | 表示触发器的WHEN条件，如果没有则为null。 |
| tgowner | oid | 触发器的所有者。 |

### PG\_TS\_CONFIG

PG\_TS\_CONFIG系统表包含表示文本搜索配置的记录。一个配置指定一个特定的文本搜索解析器和一个为了每个解析器的输出类型使用的字典的列表。

解析器在PG\_TS\_CONFIG记录中显示，但是字典映射的标记是由PG\_TS\_CONFIG\_MAP（参考：系统表和系统视图->系统表->PG\_TS\_CONFIG\_MAP章节）里面的辅助记录定义的。

**表 1** PG\_TS\_CONFIG字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| cfgname | name | - | 文本搜索配置名。 |
| cfgnamespace | oid | PG\_NAMESPACE.oid | 包含这个配置的名称空间的OID。 |
| cfgowner | oid | PG\_AUTHID.oid | 配置的所有者。 |
| cfgparser | oid | PG\_TS\_PARSER.oid | 这个配置的文本搜索解析器的OID。 |
| cfoptions | text[] | - | 分词相关配置选项。 |

### PG\_TS\_CONFIG\_MAP

PG\_TS\_CONFIG\_MAP系统表包含为每个文本搜索配置的解析器的每个输出符号类型，显示哪个文本搜索字典应该被咨询、以什么顺序搜索的记录。

**表 1** PG\_TS\_CONFIG\_MAP字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| mapcfg | oid | PG\_TS\_CONFIG.oid | 拥有这个映射记录的PG\_TS\_CONFIG记录的OID。 |
| maptokentype | integer | - | 由配置的解析器产生的一个符号类型值。 |
| mapseqno | integer | - | 在相同mapcfg或maptokentype值的情况下，该符号类型的顺序号。 |
| mapdict | oid | PG\_TS\_DICT.oid | 要咨询的文本搜索字典的OID。 |

### PG\_TS\_DICT

PG\_TS\_DICT系统表包含定义文本搜索字典的记录。字典取决于文本搜索模板，该模板声明所有需要的实现函数；字典本身提供模板支持的用户可设置的参数的值。

这种分工允许字典通过非权限用户创建。参数由文本字符串dictinitoption指定，参数的格式和意义取决于模板。

**表 1** PG\_TS\_DICT字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| dictname | name | - | 文本搜索字典名。 |
| dictnamespace | oid | PG\_NAMESPACE.oid | 包含这个字典的名称空间的OID。 |
| dictowner | oid | PG\_AUTHID.oid | 字典的所有者。 |
| dicttemplate | oid | PG\_TS\_TEMPLATE.oid | 这个字典的文本搜索模板的OID。 |
| dictinitoption | text | - | 该模板的初始化选项字符串。 |

### PG\_TS\_PARSER

PG\_TS\_PARSER系统表包含定义文本解析器的记录。解析器负责分裂输入文本为词位，并且为每个词位分配标记类型。新解析器必须由数据库系统管理员创建。

**表 1** PG\_TS\_PARSER字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| prsname | name | - | 文本搜索解析器名。 |
| prsnamespace | oid | PG\_NAMESPACE.oid | 包含这个解析器的名称空间的OID。 |
| prsstart | regproc | PG\_PROC.proname | 解析器的启动函数名。 |
| prstoken | regproc | PG\_PROC.proname | 解析器的下一个标记函数名。 |
| prsend | regproc | PG\_PROC.proname | 解析器的关闭函数名。 |
| prsheadline | regproc | PG\_PROC.proname | 解析器的标题函数名。 |
| prslextype | regproc | PG\_PROC.proname | 解析器的lextype函数名。 |

### PG\_TS\_TEMPLATE

PG\_TS\_TEMPLATE系统表包含定义文本搜索模板的记录。模板是文本搜索字典的类的实现框架。因为模板必须通过C语言级别的函数实现，索引新模板的创建必须由数据库系统管理员创建。

**表 1** PG\_TS\_TEMPLATE字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| tmplname | name | - | 文本搜索模板名。 |
| tmplnamespace | oid | PG\_NAMESPACE.oid | 包含这个模板的名称空间的OID。 |
| tmplinit | regproc | PG\_PROC.proname | 模板的初始化函数名。 |
| tmpllexize | regproc | PG\_PROC.proname | 模板的lexize函数名。 |

### PG\_TYPE

PG\_TYPE系统表存储数据类型的相关信息。

**表 1** PG\_TYPE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| typname | name | 数据类型名称。 |
| typnamespace | oid | 包含这个类型的名称空间的OID。 |
| typowner | oid | 该类型的所有者。 |
| typlen | smallint | 对于定长类型是该类型内部表现形式的字节数目。对于变长类型是负数。   * -1表示一种“变长”（有长度字属性的数据）。 * -2表示这是一个NULL结尾的C字符串。 |
| typbyval | boolean | 指定内部传递这个类型的数值时是传值（该值为true）还是传引用（该值为false）。如果该类型的TYPLEN不是1、2、4、8， TYPBYVAL最好为false。变长类型通常是传引用。即使TYPLEN允许传值，TYPBYVAL也可以为false。 |
| typtype | “char” | 对于基础类型是b。对于复合类型是c（比如，一个表的行类型）。对于域类型是d。对于伪类型是p。参见typrelid和typbasetype。 |
| typcategory | “char” | 是数据类型的模糊分类，可用于解析器做为数据转换的依据。 |
| typispreferred | boolean | 如果为真，则数据符合TYPCATEGORY所指定的转换规则时进行转换。 |
| typisdefined | boolean | 如果定义了类型则为真，如果是一种尚未定义的类型的占位符则为假。如果为假，则除了该类型名称，名称空间和OID之外没有可靠的信息。 |
| typdelim | “char” | 当分析数组输入时，分隔两个此类型数值的字符请注意该分隔符是与数组元素数据类型相关联的，而不是和数组数据类型关联。 |
| typrelid | oid | 如果是复合类型（请参见typtype），则这个字段指向pg\_class中定义该表的行。对于自由存在的复合类型，pg\_class记录并不表示一个表，但是总需要它来查找该类型连接的pg\_attribute记录。对于非复合类型为零。 |
| typelem | oid | 如果不为0，则它标识pg\_type里面的另外一行。当前类型可以当做一个产生类型为typelem的数组来描述。一个“真正的”数组类型是变长的（typlen= -1），但是一些定长的（typlen > 0）类型也拥有非零的typelem（比如name和point）。如果一个定长类型拥有一个typelem ，则他的内部形式必须是typelem数据类型的某个数目的个数值，不能有其他数据。变长数组类型有一个该数组子过程定义的头（文件）。 |
| typarray | oid | 如果不为0，则表示在pg\_type中有对应的类型记录。 |
| typinput | regproc | 输入转换函数（文本格式）。 |
| typoutput | regproc | 输出转换函数（文本格式）。 |
| typreceive | regproc | 输入转换函数（二进制格式），如果没有则为0。 |
| typsend | regproc | 输出转换函数（二进制格式），如果没有则为0。 |
| typmodin | regproc | 输入类型修改符函数，如果为0，则不支持。 |
| typmodout | regproc | 输出类型修改符函数，如果为0，则不支持。 |
| typanalyze | regproc | 自定义的ANALYZE函数，如果使用标准函数，则为0。 |
| typalign | “char” | 当存储此类型的数值时要求的对齐性质。它应用于磁盘存储以及该值在Vastbase内部的大多数形式。如果数值是连续存放的，比如在磁盘上以完全的裸数据的形式存放时，则先在此类型的数据前填充空白，这样它就可以按照要求的界限存储。对齐引用是该序列中第一个数据的开头。可能的值包含：   * c = char对齐，也就是不需要对齐。 * s = short对齐（在大多数机器上是2字节）。 * i = int对齐（在大多数机器上是4字节）。 * d = double对齐（在大多数机器上是8字节，但不一定是全部）。   须知：对于在系统表里使用的类型，在pg\_type里定义的尺寸和对齐必须和编译器在一个表示表的一行的结构里的布局一样。 |
| typstorage | “char” | 指明一个变长类型（那些有typlen = -1）是否准备好应付非常规值，以及对这种属性的类型的缺省策略是什么。可能的值包含：   * p：数值总是以简单方式存储。 * e：数值可以存储在一个“次要”关系中（如果该关系有这么一个，请参见pg\_class.reltoastrelid）。 * m：数值可以以内联的压缩方式存储。 * x：数值可以以内联的压缩方式或者在“次要”表里存储。   须知：m域也可以移到从属表里存储，但只是最后的解决方法（e和x域先移走）。 |
| typenotnull | boolean | 该类型是否存在NOTNULL约束。目前只用于域。 |
| typbasetype | oid | 如果这是一个衍生类型（请参见typtype），则该标识作为这个类型的基础的类型。如果不是衍生类型则为零。 |
| typtypmod | integer | 域使用typtypmod记录要作用到它们的基础类型上的typmod（如果基础类型不使用typmod则为-1）。如果这种类型不是域，则为-1。 |
| typndims | integer | 如果一个域是数组，则typndims是数组维数的数值（也就是说，typbasetype是一个数组类型；域的typelem将匹配基本类型的typelem）。非域非数组域为零。 |
| typcollation | oid | 指定类型的排序规则。取值参考：系统表和系统视图->系统表->PG\_COLLATION章节。如果为0，则表示不支持排序。 |
| typdefaultbin | pg\_node\_tree | 如果为非NULL，则它是该类型缺省表达式的nodeToString()表现形式。目前这个字段只用于域。 |
| typdefault | text | 如果某类型没有相关缺省值，则取值是NULL。   * 如果typdefaultbin为非NULL，则typdefault必须包含一个typdefaultbin代表的缺省表达式。 * 如果typdefaultbin为NULL但typdefault不是，typdefault则是该类型缺省值的外部表现形式，可以把它作为该类型的输入，转换器生成一个常量。 |
| typacl | aclitem[] | 访问权限。 |

### PG\_USER\_MAPPING

PG\_USER\_MAPPING系统表存储从本地用户到远程的映射。

需要有系统管理员权限才可以访问此系统表。普通用户可以使用视图PG\_USER\_MAPPINGS（参考：系统表和系统视图->系统视图->PG\_USER\_MAPPINGS章节）进行查询。

**表 1** PG\_USER\_MAPPING字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| oid | oid | - | 行标识符（隐含属性，必须明确选择）。 |
| umuser | oid | PG\_AUTHID.oid | 被映射的本地用户的OID，如果用户映射是公共的则为0。 |
| umserver | oid | PG\_FOREIGN\_SERVER.oid | 包含这个映射的外部服务器的OID。 |
| umoptions | text[] | - | 用户映射指定选项，使用“keyword=value”格式的字符串。 |

### PG\_USER\_STATUS

PG\_USER\_STATUS系统表提供了访问数据库用户的状态。需要有系统管理员权限才可以访问此系统表。

**表 1** PG\_USER\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段，必须明确选择）。 |
| roloid | oid | 角色的标识。 |
| failcount | integer | 尝试失败次数。 |
| locktime | timestamp with time zone | 角色被锁定的时间点。 |
| rolstatus | smallint | 角色的状态。0：正常状态。1：由于登录失败次数超过阈值被锁定了一定的时间。2：被管理员锁定。 |
| permspace | bigint | 角色已经使用的永久表存储空间大小。 |
| tempspace | bigint | 角色已经使用的临时表存储空间大小。 |
| passwordexpired | smallint | 密码是否失效。0：密码有效。1：密码失效。 |

### PG\_WORKLOAD\_GROUP

PG\_WORKLOAD\_GROUP系统表提供了数据库负载组的信息。

**表 1** PG\_WORKLOAD\_GROUP字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含属性，必须明确选择）。 |
| workload\_gpname | name | 负载组名称。 |
| respool\_oid | oid | 绑定到的资源池的id。 |
| act\_statements | integer | 负载组内最大的活跃语句数。 |

### PGXC\_CLASS

PGXC\_CLASS系统表存储每张表的复制或分布信息。PGXC\_CLASS系统表仅在分布式场景下有具体含义，Vastbase只能查询表定义。

**表 1** PGXC\_CLASS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pcrelid | oid | 表的OID。 |
| pclocatortype | “char” | 定位器类型。H：hashG：RangeL：ListM：ModuloN：Round RobinR：Replication |
| pchashalgorithm | smallint | 使用哈希算法分布元组。 |
| pchashbuckets | smallint | 哈希容器的值。 |
| pgroup | name | 节点群的名称。 |
| redistributed | “char” | 表已经完成重分布。 |
| redis\_order | integer | 重分布的顺序。该值等于0的表在本轮重分布过程中不进行重分布。 |
| pcattnum | int2vector | 用作分布键的列标号。 |
| nodeoids | oidvector\_extend | 表分布的节点OID列表。 |
| options | text | 系统内部保留字段，存储扩展状态信息。 |

### PGXC\_GROUP

PGXC\_GROUP系统表存储节点组信息。PGXC\_GROUP系统表仅在分布式场景下有具体含义，Vastbase只能查询表定义。

**表 1** PGXC\_GROUP字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段，必须明确选择）。 |
| group\_name | name | 节点组名称。 |
| in\_redistribution | “char” | 是否需要重分布。取值包括n，y，t。n：表示NodeGroup没有再进行重分布。y：表示NodeGroup是重分布过程中的源节点组。t：表示NodeGroup是重分布过程中的目的节点组。 |
| group\_members | oidvector\_extend | 节点组的节点OID列表。 |
| group\_buckets | text | 分布数据桶的集合。 |
| is\_installation | Boolean | 是否安装子集群。t（true）：表示安装。f（false）：表示不安装。 |
| group\_acl | aclitem[] | 访问权限。 |
| group\_kind | “char” | node group类型，取值包括i, n, v, e。i：表示installation node group。n：表示普通非逻辑集群node group。v：表示逻辑集群node group。e：表示弹性集群。 |
| group\_parent | oid | 如果是子node group，该字段表示父node group的OID，如果是父node group，该字段值为空。 |

### PGXC\_NODE

PGXC\_NODE系统表存储集群节点信息。PGXC\_NODE系统表仅在分布式场景下有具体含义，Vastbase只能查询表定义。

**表 1** PGXC\_NODE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| oid | oid | 行标识符（隐含字段，必须明确选择）。 |
| node\_name | name | 节点名称。 |
| node\_type | “char” | 节点类型。C：协调节点。D：数据节点。S：数据节点的备节点。 |
| node\_port | integer | 节点的端口号。 |
| node\_host | name | 节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。 |
| node\_port1 | integer | 复制节点的端口号。 |
| node\_host1 | name | 复制节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。 |
| hostis\_primary | Boolean | 表明当前节点是否发生主备切换。t（true）：表示发生。f（false）：表示不发生。 |
| nodeis\_primary | Boolean | 在replication表下，是否优选当前节点作为优先执行的节点进行非查询操作。t（true）：表示优选。f（false）：表示不优选。 |
| nodeis\_preferred | Boolean | 在replication表下，是否优选当前节点作为首选的节点进行查询。t（true）：表示优选。f（false）：表示不优选。 |
| node\_id | integer | 节点标志符。由node\_name经过hash函数计算后得到。 |
| sctp\_port | integer | 主节点使用TCP代理通信库或SCTP通信库的数据通道侦听端口。 |
| control\_port | integer | 主节点使用TCP代理通信库或SCTP通信库的控制通道侦听端口。 |
| sctp\_port1 | integer | 备节点使用TCP代理通信库或SCTP通信库的数据通道侦听端口。 |
| control\_port1 | integer | 备节点使用TCP代理通信库或SCTP通信库的控制通道侦听端口。 |
| nodeis\_central | Boolean | 表明当前节点是否为中心控制节点，只用于CN，对DN无效。t（true）：表示是。f（false）：表示不是。 |
| nodeis\_active | Boolean | 表明当前节点是否是正常状态，用于标记CN是否被剔除，对DN无效。t（true）：表示是。f（false）：表示不是。 |

### PGXC\_SLICE

PGXC\_SLICE表是针对range范围分布和list分布创建的系统表，用来记录分布具体信息，当前不支持range interval自动扩展分片，不过在系统表中预留。

PGXC\_SLICE系统表仅在分布式场景下有具体含义，Vastbase只能查询表定义。

**表 1** PGXC\_SLICE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relname | name | 表名或者分片名，通过type区分 |
| type | char | 当为’t’时relname是表名，当为’s’时relname是分片的名字 |
| strategy | char | ‘r’：为range分布表‘l’：为list分布表后续interval分片会扩展该值 |
| relid | oid | 该tuple隶属的分布表oid |
| referenceoid | oid | 所参考分布表的oid,用于slice reference建表语法 |
| sindex | int | 当为list分布表时，表示当前boundary在某个分片内的位置 |
| interval | text[] | 预留字段 |
| transitboundary | text[] | 预留字段 |
| transitno | int | 预留字段 |
| nodeoid | oid | 当relname为分片名时，表示该分片的数据存放在哪个DN上，nodeoid表示这个DN的oid |
| boundaries | text[] | 当relname为分片名时，对应该分片的边界值 |
| specified | boolean | 当前分片对应的DN是否是用户在DDL中显示指定的 |
| sliceorder | int | 用户定义分片的顺序 |

### PLAN\_TABLE\_DATA

PLAN\_TABLE\_DATA存储了用户通过执行EXPLAIN PLAN收集到的计划信息。与PLAN\_TABLE视图（参考：系统表和系统视图->系统视图->PLAN\_TABLE章节）不同的是PLAN\_TABLE\_DATA表存储了所有session和user执行EXPLAIN PLAN收集的计划信息。

**表 1** PLAN\_TABLE\_DATA字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| session\_id | text | 表示插入该条数据的会话，由服务线程启动时间戳和服务线程ID组成。受非空约束限制。 |
| user\_id | oid | 用户ID，用于标识触发插入该条数据的用户。受非空约束限制。 |
| statement\_id | varchar2(30) | 用户输入的查询标签。 |
| plan\_id | bigint | 查询标识。该标识在计划生成阶段自动产生，供内核工程师调试使用。 |
| id | int | 计划中的节点编号。 |
| operation | varchar2(30) | 操作描述。 |
| options | varchar2(255) | 操作选项。 |
| object\_name | name | 操作对应的对象名，来自于用户定义。 |
| object\_type | varchar2(30) | 对象类型。 |
| object\_owner | name | 对象所属schema，来自于用户定义。 |
| projection | varchar2(4000) | 操作输出的列信息。 |

fig: **说明：**

* PLAN\_TABLE\_DATA中包含了当前节点所有用户、所有会话的数据，仅管理员有访问权限。普通用户可以通过PLAN\_TABLE视图（参考：系统表和系统视图->系统视图->PLAN\_TABLE章节）查看属于自己的数据。
* PLAN\_TABLE\_DATA中的数据是用户通过执行EXPLAIN PLAN命令后由系统自动插入表中，因此禁止用户手动对数据进行插入或更新，否则会引起表中的数据混乱。需要对表中数据删除时，建议通过PLAN\_TABLE视图（参考：系统表和系统视图->系统视图->PLAN\_TABLE章节）。

statement\_id、object\_name、object\_owner和projection字段内容遵循用户定义的大小写存在系统库中查询到结果，用户库中无法查询。

对于此系统表查询有如下约束：

* 必须在postgres库内查询，其它库中不存数据。
* 此系统表受track\_stmt\_stat\_level控制，默认为“OFF,L0”，第一部分控制Full SQL，第二部分控制Slow SQL，具体字段记录级别见下表。
* 对于Slow SQL，当track\_stmt\_stat\_level的值为非OFF时，且SQL执行时间超过log\_min\_duration\_statement，会记录为慢SQL。
* **表 1** STATEMENT\_HISTORY字段储，其它字段内容采用大写存储。

### STATEMENT\_HISTORY

获得当前节点的执行语句的信息。查询系统表必须具有sysadmin权限。只可

| **名称** | **类型** | **描述** | **记录级别** |
| --- | --- | --- | --- |
| db\_name | name | 数据库名称。 | L0 |
| schema\_name | name | schema名称。 | L0 |
| origin\_node | integer | 节点名称。 | L0 |
| user\_name | name | 用户名。 | L0 |
| application\_name | text | 用户发起的请求的应用程序名称。 | L0 |
| client\_addr | text | 用户发起的请求的客户端地址。 | L0 |
| client\_port | integer | 用户发起的请求的客户端端口。 | L0 |
| unique\_query\_id | bigint | 归一化SQL ID。 | L0 |
| debug\_query\_id | bigint | 唯一SQL ID。 | L0 |
| query | text | 归一化SQL。 | L0 |
| start\_time | timestamp with time zone | 语句启动的时间。 | L0 |
| finish\_time | timestamp with time zone | 语句结束的时间。 | L0 |
| slow\_sql\_threshold | bigint | 语句执行时慢SQL的标准。 | L0 |
| transaction\_id | bigint | 事务ID。 | L0 |
| thread\_id | bigint | 执行线程ID。 | L0 |
| session\_id | bigint | 用户session id。 | L0 |
| n\_soft\_parse | bigint | 软解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 | L0 |
| n\_hard\_parse | bigint | 硬解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 | L0 |
| query\_plan | text | 语句执行计划。 | L1 |
| n\_returned\_rows | bigint | SELECT返回的结果集行数。 | L0 |
| n\_tuples\_fetched | bigint | 随机扫描行。 | L0 |
| n\_tuples\_returned | bigint | 顺序扫描行。 | L0 |
| n\_tuples\_inserted | bigint | 插入行。 | L0 |
| n\_tuples\_updated | bigint | 更新行。 | L0 |
| n\_tuples\_deleted | bigint | 删除行。 | L0 |
| n\_blocks\_fetched | bigint | buffer的块访问次数。 | L0 |
| n\_blocks\_hit | bigint | buffer的块命中次数。 | L0 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 | L0 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 | L0 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 | L0 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 | L0 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 | L0 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 | L0 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 | L0 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 | L0 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 | L0 |
| net\_send\_info | text | 通过物理连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 | L0 |
| net\_recv\_info | text | 通过物理连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 | L0 |
| net\_stream\_send\_info | text | 通过逻辑连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 | L0 |
| net\_stream\_recv\_info | text | 通过逻辑连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 | L0 |
| lock\_count | bigint | 加锁次数。 | L0 |
| lock\_time | bigint | 加锁耗时。 | L1 |
| lock\_wait\_count | bigint | 加锁等待次数。 | L0 |
| lock\_wait\_time | bigint | 加锁等待耗时。 | L1 |
| lock\_max\_count | bigint | 最大持锁数量。 | L0 |
| lwlock\_count | bigint | 轻量级加锁次数（预留）。 | L0 |
| lwlock\_wait\_count | bigint | 轻量级等锁次数。 | L0 |
| lwlock\_time | bigint | 轻量级加锁时间（预留）。 | L1 |
| lwlock\_wait\_time | bigint | 轻量级等锁时间。 | L1 |
| details | bytea | 语句锁事件的列表，该列表按时间书序记录事件，记录的数量受参数track\_stmt\_details\_size的影响。该字段为二进制，需要借助解析函数pg\_catalog.statement\_detail\_decode读取。  事件包括：加锁开始、加锁结束、等锁开始、等锁结束、放锁开始、放锁结束、轻量级等锁开始、轻量级等锁结束 | L2 |
| is\_slow\_sql | boolean | 该SQL是否为slow SQL。t（true）：表示是。f（false）：表示不是。 | L0 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 | L0 |

## 系统视图

### GS\_AUDITING

GS\_AUDITING视图显示对数据库相关操作的所有审计信息。需要有系统管理员或安全策略管理员权限才可以访问此视图。

**表 1** GS\_AUDITING字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| polname | name | 策略名称，需要唯一，不可重复。 |
| pol\_type | text | 审计策略类型，值为access或者privilege。   * access：表示审计DML操作。 * privilege：表示审计DDL操作。 |
| polenabled | boolean | 用来表示策略启动开关。t（true）：表示启动。f（false）：表示不启动。 |
| access\_type | bigint | DML数据库操作相关类型。例如SELECT、INSERT、DELETE等。 |
| label\_name | name | 资源标签名称。对应系统表gs\_auditing\_policy中的polname字段。 |
| priv\_object | name | 用来描述数据库资产的路径。 |
| filter\_name | text | 过滤条件的逻辑字符串。 |

### GS\_AUDITING\_ACCESS

GS\_AUDITING\_ACCESS视图显示对数据库DML相关操作的所有审计信息。需要有系统管理员或安全策略管理员权限才可以访问此视图。

**表 1** GS\_AUDITING\_ACCESS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| polname | name | 策略名称，需要唯一，不可重复。 |
| pol\_type | text | 审计策略类型，值为‘access’，表示审计DML操作。 |
| polenabled | boolean | 用来表示策略启动开关。 |
| access\_type | name | DML数据库操作相关类型。例如SELECT、INSERT、DELETE等。 |
| label\_name | name | 资源标签名称。对应系统表gs\_auditing\_policy中的polname字段。 |
| access\_object | text | 用来描述数据库资产的路径。 |
| filter\_name | text | 过滤条件的逻辑字符串。 |

### GS\_AUDITING\_PRIVILEGE

GS\_AUDITING\_PRIVILEGE视图显示对数据库DDL相关操作的所有审计信息。需要有系统管理员或安全策略管理员权限才可以访问此视图。

**表 1** GS\_AUDITING\_PRIVILEGE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| polname | name | 策略名称，需要唯一，不可重复。 |
| pol\_type | text | 审计策略类型，值为‘privilege’，表示审计DDL 操作。 |
| polenabled | boolean | 用来表示策略启动开关。 |
| access\_type | name | DDL数据库操作相关类型。例如CREATE、ALTER、DROP等。 |
| label\_name | name | 资源标签名称。对应系统表gs\_auditing\_policy中的polname字段。 |
| priv\_object | text | 带有数据库对象的全称域名。 |
| filter\_name | text | 过滤条件的逻辑字符串。 |

### GS\_DB\_PRIVILEGES

GS\_DB\_PRIVILEGES系统视图记录ANY权限的授予情况，每条记录对应一条授权信息。

**表 1** GS\_DB\_PRIVILEGES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| rolename | name | 用户名。 |
| privilege\_type | text | 用户拥有的ANY权限，取值请参考：SQL语法参考->SQL语法->GRANT章节。 |
| admin\_option | boolean | 是否具有privilege\_type列记录的ANY权限的再授权权限。   * yes：表示具有。 * no：表示不具有。 |

### GS\_FILE\_STAT

GS\_FILE\_STAT视图通过对数据文件IO的统计，反映数据的IO性能，用以发现IO操作异常等性能问题。

**表 1** GS\_FILE\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| filenum | oid | 文件标识。 |
| dbid | oid | 数据库标识。 |
| spcid | oid | 表空间标识。 |
| phyrds | bigint | 读物理文件的数目。 |
| phywrts | bigint | 写物理文件的数目。 |
| phyblkrd | bigint | 读物理文件块的数目。 |
| phyblkwrt | bigint | 写物理文件块的数目。 |
| readtim | bigint | 读文件的总时长，单位微秒。 |
| writetim | bigint | 写文件的总时长，单位微秒。 |
| avgiotim | bigint | 读写文件的平均时长，单位微秒。 |
| lstiotim | bigint | 最后一次读文件时长，单位微秒。 |
| miniotim | bigint | 读写文件的最小时长，单位微秒。 |
| maxiowtm | bigint | 读写文件的最大时长，单位微秒。 |

### GS\_GSC\_MEMORY\_DETAIL

GS\_GSC\_MEMORY\_DETAIL视图描述当前节点当前进程的全局SysCache的内存占用情况，仅在开启GSC的模式下有数据。需要注意的是，这个查询由于是以数据库内存上下文分隔的，因此会缺少一部分内存的统计，缺失的内存统计对应的内存上下文名称为GlobalSysDBCache。

**表 1** GS\_GSC\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| db\_id | integer | 数据库id。 |
| totalsize | bigint | 共享内存总大小，单位Byte。 |
| freesize | bigint | 共享内存剩余大小，单位Byte。 |
| usedsize | bigint | 共享内存使用大小，单位Byte。 |

### GS\_INSTANCE\_TIME

提供当前集节点下的各种时间消耗信息，主要分为以下类型：

* DB\_TIME：作业在多核下的有效时间花销。
* CPU\_TIME：CPU的时间花销。
* EXECUTION\_TIME：执行器内的时间花销。
* PARSE\_TIME：SQL解析的时间花销。
* PLAN\_TIME：生成Plan的时间花销。
* REWRITE\_TIME：SQL重写的时间花销。
* PL\_EXECUTION\_TIME：plpgsql（存储过程）执行的时间花销。
* PL\_COMPILATION\_TIME：plpgsql（存储过程）编译的时间花销。
* NET\_SEND\_TIME：网络上的时间花销。
* DATA\_IO\_TIME：IO的时间花销。

**表 1** GS\_INSTANCE\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 类型名称。 |
| value | bigint | 时间值（单位：微秒）。 |

### GS\_LABELS

GS\_LABELS视图显示所有已配置的资源标签信息。需要有系统管理员或安全策略管理员权限才可以访问此视图。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| labelname | name | 资源标签的名称。 |
| labeltype | name | 资源标签的类型。对应系统表[GS\_POLICY\_LABEL（参考：系统表和系统视图->系统表->GS\_POLICY\_LABEL章节）中的labeltype字段。 |
| fqdntype | name | 数据库资源的类型。如table、schema、index等。 |
| schemaname | name | 数据库资源所属的schema名称。 |
| fqdnname | name | 数据库资源名称。 |
| columnname | name | 数据库资源列名称，若标记的数据库资源不为表的列则该项为空。 |

### GS\_LSC\_MEMORY\_DETAIL

GS\_LSC\_MEMORY\_DETAIL视图统计所有的线程的本地SysCache内存使用情况，以MemoryContext节点来统计，仅在开启GSC的模式下有数据。

**表 1** GS\_LSC\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| threadid | text | 线程启动时间+线程标识（字符串信息为timestamp.sessionid）。 |
| tid | bigint | 线程标识。 |
| thrdtype | text | 线程类型。可以是系统内存在的任何线程类型，如postgresql、wlmmonitor等。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 当前上下文在整体内存上下文中的层级。 |
| parent | text | 父内存上下文名称。 |
| totalsize | bigint | 当前内存上下文的内存总数，单位Byte。 |
| freesize | bigint | 当前内存上下文中已释放的内存总数，单位Byte。 |
| usedsize | bigint | 当前内存上下文中已使用的内存总数，单位Byte。 |

### GS\_MASKING

GS\_MASKING视图显示所有已配置的动态脱敏策略信息。需要有系统管理员或安全策略管理员权限才可以访问此视图。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| polname | name | 脱敏策略名称。 |
| polenabled | boolean | 脱敏策略开关。 |
| maskaction | name | 脱敏函数。 |
| labelname | name | 脱敏函数作用的标签名称。 |
| masking\_object | text | 脱敏数据库资源对象。 |
| filter\_name | text | 过滤条件的逻辑表达式。 |

### GS\_MATVIEWS

GS\_MATVIEWS视图提供了关于数据库中每一个物化视图的信息。

**表 1** GS\_MATVIEWS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | PG\_NAMESPACE.nspname | 物化视图的模式名。 |
| matviewname | name | PG\_CLASS.relname | 物化视图名。 |
| matviewowner | name | PG\_AUTHID.Erolname | 物化视图的所有者。 |
| tablespace | name | PG\_TABLESPACE.spcname | 物化视图的表空间名（如果使用数据库默认表空间则为空）。 |
| hasindexes | boolean | - | 如果物化视图有（或者最近有过）任何索引，则此列为真。 |
| definition | text | - | 物化视图的定义（一个重构的SELECT查询）。 |

### GS\_OS\_RUN\_INFO

GS\_OS\_RUN\_INFO视图显示当前操作系统运行的状态信息。

**表 1** GS\_OS\_RUN\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| id | integer | 编号。 |
| name | text | 操作系统运行状态名称。 |
| value | numeric | 操作系统运行状态值。 |
| comments | text | 操作系统运行状态注释。 |
| cumulative | Boolean | 操作系统运行状态的值是否为累加值。 |

### GS\_REDO\_STAT

GS\_REDO\_STAT视图用于统计回话线程日志回放情况。

**表 1** GS\_REDO\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| phywrts | bigint | 日志回放过程中写数据的次数。 |
| phyblkwrt | bigint | 日志回放过程中写数据的块数。 |
| writetim | bigint | 日志回放过程中写数据所耗的总时间。 |
| avgiotim | bigint | 日志回放过程中写一次数据的平均消耗时间。 |
| lstiotim | bigint | 日志回放过程中最后一次写数据消耗的时间。 |
| miniotim | bigint | 日志回放过程中单次写数据消耗的最短时间。 |
| maxiowtm | bigint | 日志回放过程中单次写数据消耗的最长时间。 |

### GS\_SESSION\_CPU\_STATISTICS

GS\_SESSION\_CPU\_STATISTICS视图显示当前用户执行的正在运行的复杂作业的CPU使用的负载管理信息。

**表 1** GS\_SESSION\_CPU\_STATISTICS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| usename | name | 登录到该后端的用户名。 |
| pid | bigint | 后端线程ID。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| min\_cpu\_time | bigint | 语句在数据库节点上的最小CPU时间，单位为ms。 |
| max\_cpu\_time | bigint | 语句在数据库节点上的最大CPU时间，单位为ms。 |
| total\_cpu\_time | bigint | 语句在数据库节点上的CPU总时间，单位为ms。 |
| query | text | 正在执行的语句。 |
| node\_group | text | 该字段不支持。 |
| top\_cpu\_dn | text | cpu使用量信息。 |

### GS\_SESSION\_MEMORY

GS\_SESSION\_MEMORY视图统计Session级别的内存使用情况，包含执行作业在数据节点上gaussdb线程和Stream线程分配的所有内存。当GUC参数enable\_memory\_limit的值为off时，本视图不可用。

**表 1** GS\_SESSION\_MEMORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识。 |
| init\_mem | integer | 当前正在执行作业进入执行器前已分配的内存，单位MB。 |
| used\_mem | integer | 当前正在执行作业已分配的内存，单位MB。 |
| peak\_mem | integer | 当前正在执行作业已分配的内存峰值，单位MB。 |

### GS\_SESSION\_MEMORY\_CONTEXT

GS\_SESSION\_MEMORY\_CONTEXT视图统计所有的会话的内存使用情况，以MemoryContext节点来统计。该视图仅在开启线程池（enable\_thread\_pool = on）时生效。当GUC参数enable\_memory\_limit的值为off时，本视图不可用。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

**表 1** GS\_SESSION\_MEMORY\_CONTEXT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 会话启动时间+会话标识（字符串信息为timestamp.sessionid）。 |
| threadid | bigint | 会话绑定的线程标识，如果未绑定线程，该值为-1。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 当前上下文在整体内存上下文中的层级。 |
| parent | text | 父内存上下文名称。 |
| totalsize | bigint | 当前内存上下文的内存总数，单位Byte。 |
| freesize | bigint | 当前内存上下文中已释放的内存总数，单位Byte。 |
| usedsize | bigint | 当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。 |

### GS\_SESSION\_MEMORY\_DETAIL

GS\_SESSION\_MEMORY\_DETAIL统计会话的内存使用情况，以MemoryContext节点来统计。当开启线程池（enable\_thread\_pool = on）时，该视图包含所有的线程和会话的内存使用情况。当GUC参数enable\_memory\_limit的值为off时，本视图不可用。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

可通过“select \* from gs\_session\_memctx\_detail(threadid, ‘’);”将某个线程所有内存上下文信息记录到“$GAUSSLOG/pg\_log/${node\_name}/dumpmem”目录下的“threadid\_timestamp.log”文件中。其中threadid可通过下表sessid中获得。

**表 1** GS\_SESSION\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程启动时间+线程标识（字符串信息为timestamp.threadid）。 |
| sesstype | text | 线程名称。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 当前上下文在整体内存上下文中的层级。 |
| parent | text | 父内存上下文名称。 |
| totalsize | bigint | 当前内存上下文的内存总数，单位Byte。 |
| freesize | bigint | 当前内存上下文中已释放的内存总数，单位Byte。 |
| usedsize | bigint | 当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。 |

### GS\_SESSION\_MEMORY\_STATISTICS

GS\_SESSION\_MEMORY\_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理内存使用的信息。

**表 1** GS\_SESSION\_MEMORY\_STATISTICS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| usename | name | 登录到该后端的用户名。 |
| pid | bigint | 后端线程ID。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| min\_peak\_memory | integer | 语句在数据库节点上的最小内存峰值大小，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库节点上的最大内存峰值大小，单位MB。 |
| spill\_info | text | 语句在数据库节点上的下盘信息：None：数据库节点均未下盘。All：数据库节点均下盘。[a:b]：数量为b个数据库节点中有a个数据库节点下盘。 |
| query | text | 正在执行的语句。 |
| node\_group | text | 该字段不支持。 |
| top\_mem\_dn | text | mem使用量信息。 |

### GS\_SESSION\_STAT

GS\_SESSION\_STAT视图以会话线程或AutoVacuum线程为单位，统计会话状态信息。

**表 1** GS\_SESSION\_STAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程标识+线程启动时间。 |
| statid | integer | 统计编号。 |
| statname | text | 统计会话名称。 |
| statunit | text | 统计会话单位。 |
| value | bigint | 统计会话值。 |

### GS\_SESSION\_TIME

GS\_SESSION\_TIME视图用于统计会话线程的运行时间信息，及各执行阶段所消耗时间。

**表 1** GS\_SESSION\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| sessid | text | 线程标识+线程启动时间。 |
| stat\_id | integer | 统计编号。 |
| stat\_name | text | 会话类型名称。 |
| value | bigint | 会话值。 |

### GS\_SQL\_COUNT

GS\_SQL\_COUNT视图显示数据库当前节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）统计信息。

* 普通用户查询GS\_SQL\_COUNT视图仅能看到该用户当前节点的统计信息；管理员权限用户查询GS\_SQL\_COUNT视图则能看到所有用户当前节点的统计信息。
* 当Vastbase或该节点重启时，计数将清零，并重新开始计数 。
* 计数以节点收到的查询数为准，包括Vastbase内部进行的查询。

**表 1** GS\_SQL\_COUNT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| user\_name | text | 用户名。 |
| select\_count | bigint | select语句统计结果。 |
| update\_count | bigint | update语句统计结果。 |
| insert\_count | bigint | insert语句统计结果。 |
| delete\_count | bigint | delete语句统计结果。 |
| mergeinto\_count | bigint | MERGE INTO语句统计结果。 |
| ddl\_count | bigint | DDL语句的数量。 |
| dml\_count | bigint | DML语句的数量。 |
| dcl\_count | bigint | DML语句的数量。 |
| total\_select\_elapse | bigint | 总select的时间花费（单位：微秒）。 |
| avg\_select\_elapse | bigint | 平均select的时间花费（单位：微秒）。 |
| max\_select\_elapse | bigint | 最大select的时间花费（单位：微秒）。 |
| min\_select\_elapse | bigint | 最小select的时间花费（单位：微秒）。 |
| total\_update\_elapse | bigint | 总update的时间花费（单位：微秒）。 |
| avg\_update\_elapse | bigint | 平均update的时间花费（单位：微秒）。 |
| max\_update\_elapse | bigint | 最大update的时间花费（单位：微秒）。 |
| min\_update\_elapse | bigint | 最小update的时间花费（单位：微秒）。 |
| total\_insert\_elapse | bigint | 总insert的时间花费（单位：微秒）。 |
| avg\_insert\_elapse | bigint | 平均insert的时间花费（单位：微秒）。 |
| max\_insert\_elapse | bigint | 最大insert的时间花费（单位：微秒）。 |
| min\_insert\_elapse | bigint | 最小insert的时间花费（单位：微秒）。 |
| total\_delete\_elapse | bigint | 总delete的时间花费（单位：微秒）。 |
| avg\_delete\_elapse | bigint | 平均delete的时间花费（单位：微秒）。 |
| max\_delete\_elapse | bigint | 最大delete的时间花费（单位：微秒）。 |
| min\_delete\_elapse | bigint | 最小delete的时间花费（单位：微秒）。 |

### GS\_STAT\_SESSION\_CU

GS\_STAT\_SESSION\_CU视图查询Vastbase各个节点，当前运行session的CU命中情况。session退出相应的统计数据会清零。Vastbase重启后，统计数据也会清零。

**表 1** GS\_STAT\_SESSION\_CU字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| mem\_hit | integer | 内存命中次数。 |
| hdd\_sync\_read | integer | 硬盘同步读次数。 |
| hdd\_asyn\_read | integer | 硬盘异步读次数。 |

### GS\_THREAD\_MEMORY\_CONTEXT

GS\_THREAD\_MEMORY\_CONTEXT视图统计所有的线程的内存使用情况，以MemoryContext节点来统计。该视图在关闭线程池（enable\_thread\_pool = off）时等价于GS\_SESSION\_MEMORY\_DETAIL视图。当GUC参数enable\_memory\_limit的值为off时，本视图不可用。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

**表 1** GS\_THREAD\_MEMORY\_CONTEXT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| threadid | text | 线程启动时间+线程标识（字符串信息为timestamp.sessionid）。 |
| tid | bigint | 线程标识。 |
| thrdtype | text | 线程类型。 |
| contextname | text | 内存上下文名称。 |
| level | smallint | 当前上下文在整体内存上下文中的层级。 |
| parent | text | 父内存上下文名称。 |
| totalsize | bigint | 当前内存上下文的内存总数，单位Byte。 |
| freesize | bigint | 当前内存上下文中已释放的内存总数，单位Byte。 |
| usedsize | bigint | 当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。 |

### GS\_TOTAL\_MEMORY\_DETAIL

GS\_TOTAL\_MEMORY\_DETAIL视图统计当前数据库节点使用内存的信息，单位为MB。当GUC参数enable\_memory\_limit的值为off时，本视图不可用。

**表 1** GS\_TOTAL\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 节点名称。 |
| memorytype | text | 内存类型，包括以下几种：max\_process\_memory：openGauss实例所占用的内存大小。process\_used\_memory：openGauss进程所使用的内存大小。max\_dynamic\_memory：最大动态内存。dynamic\_used\_memory：已使用的动态内存。dynamic\_peak\_memory：内存的动态峰值。dynamic\_used\_shrctx：最大动态共享内存上下文。dynamic\_peak\_shrctx：共享内存上下文的动态峰值。max\_shared\_memory：最大共享内存。shared\_used\_memory：已使用的共享内存。max\_cstore\_memory：列存所允许使用的最大内存。cstore\_used\_memory：列存已使用的内存大小。max\_sctpcomm\_memory：通信库所允许使用的最大内存。sctpcomm\_used\_memory：通信库已使用的内存大小。sctpcomm\_peak\_memory：通信库的内存峰值。other\_used\_memory：其他已使用的内存大小。 |
| memorymbytes | integer | 内存类型分配内存的大小。 |

### GS\_WLM\_CGROUP\_INFO

GS\_WLM\_CGROUP\_INFO视图显示当前执行作业的控制组的信息。

**表 1** GS\_WLM\_CGROUP\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| cgoup\_name | text | 控制组的名称。 |
| priority | interger | 作业的优先级。 |
| usage\_pecent | interger | 控制组占用的百分比。 |
| shares | bigint | 控制组分配的CPU资源配额。 |
| cpuacct | bigint | CPU配额分配。 |
| cpuset | text | CPU限额分配。 |
| relpath | text | 控制组的相对路径。 |
| valid | text | 该控制组是否有效。 |
| node\_group | text | 逻辑数据库实例名称。 |

### GS\_WLM\_EC\_OPERATOR\_STATISTICS

GS\_WLM\_EC\_OPERATOR\_STATISTICS视图显示当前用户正在执行的EC（Extension Connector）作业的算子相关信息。查询该视图需要sysadmin权限。

**表 1** GS\_WLM\_EC\_OPERATOR\_STATISTICS的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | EC语句执行使用的内部query\_id。 |
| plan\_node\_id | integer | EC算子对应的执行计划的plan node id。 |
| start\_time | timestamp with time zone | EC算子处理第一条数据的开始时间。 |
| ec\_status | text | EC作业的执行状态。   * EC\_STATUS\_INIT：初始化。 * EC\_STATUS\_CONNECTED：已连接。 * EC\_STATUS\_EXECUTED：已执行。 * EC\_STATUS\_FETCHING：获取中。 * EC\_STATUS\_END：已结束。 |
| ec\_execute\_datanode | text | 执行EC作业的DN名称。 |
| ec\_dsn | text | EC作业所使用的DSN。 |
| ec\_username | text | EC作业访问远端数据库实例的USERNAME（远端集群为SPARK类型时该值为空）。 |
| ec\_query | text | EC作业发送给远端数据库实例执行的语句。 |
| ec\_libodbc\_type | text | EC作业使用的unixODBC驱动类型。  类型1：对应 libodbc.so.1。  类型2：对应 libodbc.so.2。 |
| ec\_fetch\_count | bigint | EC作业当前处理的数据条数。 |

### GS\_WLM\_OPERATOR\_HISTORY

GS\_WLM\_OPERATOR\_HISTORY视图显示的是当前用户当前数据库主节点上执行作业结束后的算子的相关记录。查询该视图需要sysadmin权限。

记录的数据同GS\_WLM\_OPERATOR\_INFO表1（详细请参考：系统表和系统视图->系统表->GS\_WLM\_OPERATOR\_INFO章节）。

### GS\_WLM\_OPERATOR\_STATISTICS

GS\_WLM\_OPERATOR\_STATISTICS视图显示当前用户正在执行的作业的算子相关信息。查询该视图需要sysadmin权限。

**表 1** GS\_WLM\_OPERATOR\_STATISTICS的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| queryid | bigint | 语句执行使用的内部query\_id。 |
| pid | bigint | 后端线程id。 |
| plan\_node\_id | integer | 查询对应的执行计划的plan node id。 |
| plan\_node\_name | text | 对应于plan\_node\_id的算子的名称。 |
| start\_time | timestamp with time zone | 该算子处理第一条数据的开始时间。 |
| duration | bigint | 该算子到结束时候总的执行时间（ms）。 |
| status | text | 当前算子的执行状态，包括finished和running。 |
| query\_dop | integer | 当前算子执行时的并行度。 |
| estimated\_rows | bigint | 优化器估算的行数信息。 |
| tuple\_processed | bigint | 当前算子返回的元素个数。 |
| min\_peak\_memory | integer | 当前算子在数据库实例上的最小内存峰值（MB）。 |
| max\_peak\_memory | integer | 当前算子在数据库实例上的最大内存峰值（MB）。 |
| average\_peak\_memory | integer | 当前算子在数据库实例上的平均内存峰值（MB）。 |
| memory\_skew\_percent | integer | 当前算子在数据库实例间的内存使用倾斜率。 |
| min\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最小数据量（MB)，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最大数据量（MB），默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库实例上下盘的平均数据量（MB），默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库实例间下盘倾斜率。 |
| min\_cpu\_time | bigint | 该算子在数据库实例上的最小执行时间（ms)。 |
| max\_cpu\_time | bigint | 该算子在数据库实例上的最大执行时间（ms)。 |
| total\_cpu\_time | bigint | 该算子在数据库实例上的总执行时间（ms）。 |
| cpu\_skew\_percent | integer | 数据库实例间执行时间的倾斜率。 |
| warning | text | 主要显示如下几类告警信息：   * Sort/SetOp/HashAgg/HashJoin spill * Spill file size large than 256MB * Broadcast size large than 100MB * Early spill * Spill times is greater than 3 * Spill on memory adaptive * Hash table conflict |

### GS\_WLM\_PLAN\_OPERATOR\_HISTORY

GS\_WLM\_PLAN\_OPERATOR\_HISTORY视图显示的是当前用户数据库主节点上执行作业结束后的执行计划算子级的相关记录。

记录的数据同GS\_WLM\_PLAN\_OPERATOR\_INFO表1（详细请参考：系统表和系统视图->系统表->GS\_WLM\_PLAN\_OPERATOR\_INFO章节）。

### GS\_WLM\_REBUILD\_USER\_RESOURCE\_POOL

该视图用于在当前连接节点上重建内存中用户的资源池信息，无输出。只是用于资源池信息缺失或者错乱时用作补救措施。查询该视图需要sysadmin权限。

**表 1** GS\_WLM\_REBUILD\_USER\_RESOURCE\_POOL的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| gs\_wlm\_rebuild\_user\_resource\_pool | boolean | 重建内存中用户资源池信息结果。t为成功，f为失败。 |

### GS\_WLM\_RESOURCE\_POOL

这是资源池上的一些统计信息。

**表 1** GS\_WLM\_RESOURCE\_POOL的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| rpoid | oid | 资源池的OID。 |
| respool | name | 资源池的名称。 |
| control\_group | name | 该字段不支持。 |
| parentid | oid | 父资源池的OID。 |
| ref\_count | integer | 关联到该资源池上的作业数量。 |
| active\_points | integer | 资源池上已经使用的点数。 |
| running\_count | integer | 正在资源池上运行的作业数量。 |
| waiting\_count | integer | 正在资源池上排队的作业数量。 |
| io\_limits | integer | 资源池的iops上限。 |
| io\_priority | integer | 资源池的io优先级。 |

### GS\_WLM\_SESSION\_INFO

GS\_WLM\_SESSION\_INFO视图显示数据库实例执行作业结束后的负载管理记录。查询该视图需要sysadmin权限。

具体的字段请参考：系统表和系统视图->系统视图->GS\_WLM\_SESSION\_HISTORY章节表1字段中的信息。

### GS\_WLM\_SESSION\_HISTORY

GS\_WLM\_SESSION\_HISTORY视图显示当前用户在数据库实例上执行作业结束后的负载管理记录。查询该视图需要sysadmin或者monitor admin权限。

**表 1** GS\_WLM\_SESSION\_HISTORY的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| dbname | text | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 语句执行的数据库实例名称。 |
| username | text | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| block\_time | bigint | 语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| finish\_time | timestamp with time zone | 语句执行的结束时间。 |
| duration | bigint | 语句实际执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句预估执行时间，单位ms。 |
| status | text | 语句执行结束状态：正常为finished，异常为aborted。 |
| abort\_info | text | 语句执行结束状态为aborted时显示异常信息。 |
| resource\_pool | text | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句估算内存大小。 |
| min\_peak\_memory | integer | 语句在数据库实例上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库实例上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句数据库实例间的内存使用倾斜率。 |
| spill\_info | text | 语句在数据库实例上的下盘信息：None：数据库实例均未下盘。All：数据库实例均下盘。[a:b]：数量为b个数据库实例中有a个数据库实例下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库实例上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库实例间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库实例上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库实例上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库实例上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库实例间的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库实例上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库实例上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库实例上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库实例间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库实例上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库实例上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库实例上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库实例间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Spill file size large than 256MBBroadcast size large than 100MBEarly spillSpill times is greater than 3Spill on memory adaptiveHash table conflict |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑数据库实例。 |
| cpu\_top1\_node\_name | text | cpu使用率第1的节点名称。 |
| cpu\_top2\_node\_name | text | cpu使用率第2的节点名称。 |
| cpu\_top3\_node\_name | text | cpu使用率第3的节点名称。 |
| cpu\_top4\_node\_name | text | cpu使用率第4的节点名称。 |
| cpu\_top5\_node\_name | text | cpu使用率第5的节点名称。 |
| mem\_top1\_node\_name | text | 内存使用量第1的节点名称。 |
| mem\_top2\_node\_name | text | 内存使用量第2的节点名称。 |
| mem\_top3\_node\_name | text | 内存使用量第3的节点名称。 |
| mem\_top4\_node\_name | text | 内存使用量第4的节点名称。 |
| mem\_top5\_node\_name | text | 内存使用量第5的节点名称。 |
| cpu\_top1\_value | bigint | cpu使用率。 |
| cpu\_top2\_value | bigint | cpu使用率。 |
| cpu\_top3\_value | bigint | cpu使用率。 |
| cpu\_top4\_value | bigint | cpu使用率。 |
| cpu\_top5\_value | bigint | cpu使用率。 |
| mem\_top1\_value | bigint | 内存使用量。 |
| mem\_top2\_value | bigint | 内存使用量。 |
| mem\_top3\_value | bigint | 内存使用量。 |
| mem\_top4\_value | bigint | 内存使用量。 |
| mem\_top5\_value | bigint | 内存使用量。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |

### GS\_WLM\_SESSION\_INFO\_ALL

GS\_WLM\_SESSION\_INFO\_ALL视图显示在数据库实例上执行作业结束后的负载管理记录。查询该视图需要sysadmin或者monitor admin权限。

**表 1** GS\_WLM\_SESSION\_INFO\_ALL的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| dbname | text | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 语句执行的CN名称。 |
| username | text | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| block\_time | bigint | 语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| finish\_time | timestamp with time zone | 语句执行的结束时间。 |
| duration | bigint | 语句实际执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句预估执行时间，单位ms。 |
| status | text | 语句执行结束状态：正常为finished，异常为aborted。 |
| abort\_info | text | 语句执行结束状态为aborted时显示异常信息。 |
| resource\_pool | text | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句估算内存大小。 |
| min\_peak\_memory | integer | 语句在所有DN上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在所有DN上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句各DN间的内存使用倾斜率。 |
| spill\_info | text | 语句在所有DN上的下盘信息：None：所有DN均未下盘。All：所有DN均下盘。[a:b]：数量为b个DN中有a个DN下盘。 |
| min\_spill\_size | integer | 若发生下盘，所有DN上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，所有DN上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，所有DN上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，DN间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在所有DN上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在所有DN上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在所有DN上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在各DN间的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在所有DN上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在所有DN上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在所有DN上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在DN间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在DN间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息以及SQL自诊断调优相关告警：Spill file size large than 256MBBroadcast size large than 100MBEarly spillSpill times is greater than 3Spill on memory adaptiveHash table conflict |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑集群。 |
| cpu\_top1\_node\_name | text | cpu使用率第1的节点名称。 |
| cpu\_top2\_node\_name | text | cpu使用率第2的节点名称。 |
| cpu\_top3\_node\_name | text | cpu使用率第3的节点名称。 |
| cpu\_top4\_node\_name | text | cpu使用率第4的节点名称。 |
| cpu\_top5\_node\_name | text | cpu使用率第5的节点名称。 |
| mem\_top1\_node\_name | text | 内存使用量第1的节点名称。 |
| mem\_top2\_node\_name | text | 内存使用量第2的节点名称。 |
| mem\_top3\_node\_name | text | 内存使用量第3的节点名称。 |
| mem\_top4\_node\_name | text | 内存使用量第4的节点名称。 |
| mem\_top5\_node\_name | text | 内存使用量第5的节点名称。 |
| cpu\_top1\_value | bigint | cpu使用率。 |
| cpu\_top2\_value | bigint | cpu使用率。 |
| cpu\_top3\_value | bigint | cpu使用率。 |
| cpu\_top4\_value | bigint | cpu使用率。 |
| cpu\_top5\_value | bigint | cpu使用率。 |
| mem\_top1\_value | bigint | 内存使用量。 |
| mem\_top2\_value | bigint | 内存使用量。 |
| mem\_top3\_value | bigint | 内存使用量。 |
| mem\_top4\_value | bigint | 内存使用量。 |
| mem\_top5\_value | bigint | 内存使用量。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |
| n\_returned\_rows | bigint | SELECT返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行。 |
| n\_tuples\_returned | bigint | 顺序扫描行。 |
| n\_tuples\_inserted | bigint | 插入行。 |
| n\_tuples\_updated | bigint | 更新行。 |
| n\_tuples\_deleted | bigint | 删除行。 |
| n\_blocks\_fetched | bigint | buffer的块访问次数。 |
| n\_blocks\_hit | bigint | buffer的块命中次数。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| net\_send\_time | bigint | 网络上的时间花费（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 |
| is\_slow\_query | bigint | 是否是慢SQL记录。 |

### GS\_WLM\_SESSION\_STATISTICS

GS\_WLM\_SESSION\_STATISTICS视图显示当前用户在数据库实例上正在执行的作业的负载管理记录。查询该视图需要sysadmin权限。

**表 1** GS\_WLM\_SESSION\_STATISTICS的字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据OID。 |
| dbname | name | 连接后端的数据库名称。 |
| schemaname | text | 模式的名称。 |
| nodename | text | 语句执行的数据库实例名称。 |
| username | name | 连接到后端的用户名。 |
| application\_name | text | 连接到后端的应用名。 |
| client\_addr | inet | 连接到后端的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| query\_band | text | 用于标示作业类型，可通过GUC参数query\_band进行设置，默认为空字符串。 |
| pid | bigint | 后端线程ID。 |
| sessionid | bigint | 会话ID。 |
| block\_time | bigint | 语句执行前的阻塞时间，单位ms。 |
| start\_time | timestamp with time zone | 语句执行的开始时间。 |
| duration | bigint | 语句已经执行的时间，单位ms。 |
| estimate\_total\_time | bigint | 语句执行预估总时间，单位ms。 |
| estimate\_left\_time | bigint | 语句执行预估剩余时间，单位ms。 |
| enqueue | text | 工作负载管理资源状态。 |
| resource\_pool | name | 用户使用的资源池。 |
| control\_group | text | 语句所使用的Cgroup。 |
| estimate\_memory | integer | 语句预估使用内存，单位MB。 |
| min\_peak\_memory | integer | 语句在数据库实例上的最小内存峰值，单位MB。 |
| max\_peak\_memory | integer | 语句在数据库实例上的最大内存峰值，单位MB。 |
| average\_peak\_memory | integer | 语句执行过程中的内存使用平均值，单位MB。 |
| memory\_skew\_percent | integer | 语句在数据库实例间的内存使用倾斜率。 |
| spill\_info | text | * 语句在数据库实例上的下盘信息：None：数据库实例均未下盘。 * All：数据库实例均下盘。 * [a:b]：数量为b个数据库实例中有a个数据库实例下盘。 |
| min\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最小数据量，单位MB，默认为0。 |
| max\_spill\_size | integer | 若发生下盘，数据库实例上下盘的最大数据量，单位MB，默认为0。 |
| average\_spill\_size | integer | 若发生下盘，数据库实例上下盘的平均数据量，单位MB，默认为0。 |
| spill\_skew\_percent | integer | 若发生下盘，数据库实例间下盘倾斜率。 |
| min\_dn\_time | bigint | 语句在数据库实例上的最小执行时间，单位ms。 |
| max\_dn\_time | bigint | 语句在数据库实例上的最大执行时间，单位ms。 |
| average\_dn\_time | bigint | 语句在数据库实例上的平均执行时间，单位ms。 |
| dntime\_skew\_percent | integer | 语句在数据库实例间的执行时间倾斜率。 |
| min\_cpu\_time | bigint | 语句在数据库实例上的最小CPU时间，单位ms。 |
| max\_cpu\_time | bigint | 语句在数据库实例上的最大CPU时间，单位ms。 |
| total\_cpu\_time | bigint | 语句在数据库实例上的CPU总时间，单位ms。 |
| cpu\_skew\_percent | integer | 语句在数据库实例间的CPU时间倾斜率。 |
| min\_peak\_iops | integer | 语句在数据库实例上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| max\_peak\_iops | integer | 语句在数据库实例上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| average\_peak\_iops | integer | 语句在数据库实例上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。 |
| iops\_skew\_percent | integer | 语句在数据库实例间的IO倾斜率。 |
| warning | text | 主要显示如下几类告警信息：Spill file size large than 256MBBroadcast size large than 100MBEarly spillSpill times is greater than 3Spill on memory adaptiveHash table conflict |
| queryid | bigint | 语句执行使用的内部query id。 |
| query | text | 正在执行的语句。 |
| query\_plan | text | 语句的执行计划。 |
| node\_group | text | 语句所属用户对应的逻辑数据库实例。 |
| top\_cpu\_dn | text | cpu使用量topN信息。 |
| top\_mem\_dn | text | 内存使用量topN信息。 |

### GS\_WLM\_USER\_INFO

用户统计信息视图。

**表 1** GS\_WLM\_USER\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| userid | oid | 用户OID。 |
| username | name | 用户名。 |
| sysadmin | boolean | 是否是管理员用户。 |
| rpoid | oid | 关联的资源池的OID。 |
| respool | name | 关联的资源池的名称。 |
| parentid | oid | 用户组的OID。 |
| totalspace | bigint | 用户的可用空间上限。 |
| spacelimit | bigint | 用户表空间限制。 |
| childcount | interger | 子用户的个数。 |
| childlist | text | 子用户列表。 |

### MPP\_TABLES

MPP\_TABLES视图显示信息如下。

**表 1** MPP\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 包含表的模式名。 |
| tablename | name | 表名。 |
| tableowner | name | 表的所有者。 |
| tablespace | name | 表所在的表空间。 |
| pgroup | name | 节点群的名称。 |
| nodeoids | oidvector\_extend | 表分布的节点OID列表。 |

### PG\_AVAILABLE\_EXTENSION\_VERSIONS

PG\_AVAILABLE\_EXTENSION\_VERSIONS视图显示数据库中某些特性的扩展版本信息。

**表 1** PG\_AVAILABLE\_EXTENSION\_VERSIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | name | 扩展名。 |
| version | text | 版本名。 |
| installed | Boolean | 如果这个扩展的版本是当前已经安装了的则为真。 |
| superuser | Boolean | 如果只允许系统管理员安装这个扩展则为真。 |
| relocatable | Boolean | 如果扩展可以重新加载到另一个模式则为真。 |
| schema | name | 扩展必须安装到的模式名，如果部分或全部可重新定位则为NULL。 |
| requires | name[] | 先决条件扩展的名称，如果没有则为NULL。 |
| comment | text | 扩展的控制文件中的评论字符串。 |

### PG\_AVAILABLE\_EXTENSIONS

PG\_AVAILABLE\_EXTENSIONS视图显示数据库中某些特性的扩展信息。

**表 1** PG\_AVAILABLE\_EXTENSIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | name | 扩展名。 |
| default\_version | text | 缺省版本的名称，如果没有指定则为NULL。 |
| installed\_version | text | 扩展当前安装版本，如果没有安装任何版本则为NULL。 |
| comment | text | 扩展的控制文件中的评论字符串。 |

### PG\_COMM\_DELAY

PG\_COMM\_DELAY视图展示单个节点的通信库时延状态。

**表 1** PG\_COMM\_DELAY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| remote\_name | text | 连接对端节点名称。 |
| remote\_host | text | 连接对端IP地址。 |
| stream\_num | integer | 当前物理连接使用的stream逻辑连接数量。 |
| min\_delay | integer | 当前物理连接一分钟内探测到的最小时延，单位微秒。说明：负数结果无效，请重新等待时延状态更新后再执行。 |
| average | integer | 当前物理连接一分钟内探测时延的平均值，单位微秒。 |
| max\_delay | integer | 当前物理连接一分钟内探测到的最大时延，单位微秒。 |

### PG\_COMM\_RECV\_STREAM

PG\_COMM\_RECV\_STREAM视图展示节点上所有的通信库接收流状态。

**表 1** PG\_COMM\_RECV\_STREAM字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| local\_tid | bigint | 使用此通信流的线程ID。 |
| remote\_name | text | 连接对端节点名称。 |
| remote\_tid | bigint | 连接对端线程ID。 |
| idx | integer | 通信对端DN在本DN内的标识编号。 |
| sid | integer | 通信流在物理连接中的标识编号。 |
| tcp\_sock | integer | 通信流所使用的tcp通信socket。 |
| state | text | 通信流当前的状态。   * UNKNOWN：表示当前逻辑连接状态未知。 * READY：表示逻辑连接已就绪。 * RUN：表示逻辑连接发送报文正常。 * HOLD：表示逻辑连接发送报文等待中。 * CLOSED：表示关闭逻辑连接。TO\_CLOSED：表示将会关闭逻辑连接。 |
| query\_id | bigint | 通信流对应的debug\_query\_id编号。 |
| pn\_id | integer | 通信流所执行查询的plan\_node\_id编号。 |
| send\_smp | integer | 通信流所执行查询send端的smpid编号。 |
| recv\_smp | integer | 通信流所执行查询recv端的smpid编号。 |
| recv\_bytes | bigint | 通信流接收的数据总量，单位Byte。 |
| time | bigint | 通信流当前生命周期使用时长，单位ms。 |
| speed | bigint | 通信流的平均接收速率，单位Byte/s。 |
| quota | bigint | 通信流当前的通信配额值，单位Byte。 |
| buff\_usize | bigint | 通信流当前缓存的数据大小，单位Byte。 |

### PG\_COMM\_SEND\_STREAM

PG\_COMM\_SEND\_STREAM视图展示节点上所有的通信库发送流状态。

**表 1** PG\_COMM\_SEND\_STREAM字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| local\_tid | bigint | 使用此通信流的线程ID。 |
| remote\_name | text | 连接对端节点名称。 |
| remote\_tid | bigint | 连接对端线程ID。 |
| idx | integer | 通信对端DN在本DN内的标识编号。 |
| sid | integer | 通信流在物理连接中的标识编号。 |
| tcp\_sock | integer | 通信流所使用的tcp通信socket。 |
| state | text | 通信流当前的状态。   * UNKNOWN：表示当前逻辑连接状态未知。 * READY：表示逻辑连接已就绪。 * RUN：表示逻辑连接发送报文正常。 * HOLD：表示逻辑连接发送报文等待中。 * CLOSED：表示关闭逻辑连接。 * TO\_CLOSED：表示将会关闭逻辑连接。 |
| query\_id | bigint | 通信流对应的debug\_query\_id编号。 |
| pn\_id | integer | 通信流所执行查询的plan\_node\_id编号。 |
| send\_smp | integer | 通信流所执行查询send端的smpid编号。 |
| recv\_smp | integer | 通信流所执行查询recv端的smpid编号。 |
| send\_bytes | bigint | 通信流发送的数据总量，单位Byte。 |
| time | bigint | 通信流当前生命周期使用时长，单位ms。 |
| speed | bigint | 通信流的平均发送速率，单位Byte/s。 |
| quota | bigint | 通信流当前的通信配额值，单位Byte。 |
| wait\_quota | bigint | 通信流等待quota值产生的额外时间开销，单位ms。 |

### PG\_COMM\_STATUS

PG\_COMM\_STATUS视图展示节点的通信库状态。

**表 1** PG\_COMM\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| rxpck\_rate | integer | 节点通信库接收速率，单位Byte/s。 |
| txpck\_rate | integer | 节点通信库发送速率，单位Byte/s。 |
| rxkbyte\_rate | bigint | bigint节点通信库接收速率，单位KByte/s。 |
| txkbyte\_rate | bigint | bigint节点通信库发送速率，单位KByte/s。 |
| buffer | bigint | cmailbox的buffer大小。 |
| memkbyte\_libcomm | bigint | libcomm进程通信内存大小，单位Byte。 |
| memkbyte\_libpq | bigint | libpq进程通信内存大小，单位Byte。 |
| used\_pm | integer | postmaster线程实时使用率。 |
| used\_sflow | integer | gs\_sender\_flow\_controller线程实时使用率。 |
| used\_rflow | integer | gs\_receiver\_flow\_controller线程实时使用率。 |
| used\_rloop | integer | 多个gs\_receivers\_loop线程中高的实时使用率。 |
| stream | integer | 当前使用的逻辑连接总数。 |

### PG\_CONTROL\_GROUP\_CONFIG

PG\_CONTROL\_GROUP\_CONFIG视图存储系统的控制组配置信息。查询该视图需要sysadmin权限。

**表 1** PG\_CONTROL\_GROUP\_CONFIG字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pg\_control\_group\_config | text | 控制组的配置信息。 |

### PG\_CURSORS

PG\_CURSORS视图列出了当前可用的游标。

**表 1** PG\_CURSORS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | text | 游标名。 |
| statement | text | 声明改游标时的查询语句。 |
| is\_holdable | Boolean | 如果该游标是持久的（就是在声明该游标的事务结束后仍然可以访问该游标）则为TRUE，否则为FALSE。 |
| is\_binary | Boolean | 如果该游标被声明为BINARY则为TRUE，否则为FALSE。 |
| is\_scrollable | Boolean | 如果该游标可以滚动（就是允许以不连续的方式检索）则为TRUE，否则为FALSE。 |
| creation\_time | timestamp with time zone | 声明该游标的时间戳。 |

### PG\_EXT\_STATS

PG\_EXT\_STATS视图提供对存储在PG\_STATISTIC\_EXT表（参考：系统表和系统视图->系统表->PG\_STATISTIC\_EXT章节）里面的扩展统计信息的访问。扩展统计信息目前包括多列统计信息。

**表 1** PG\_EXT\_STATS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | [PG\_NAMESPACE](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_NAMESPACE.html).nspname | 包含表的模式名。 |
| tablename | name | [PG\_CLASS](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_CLASS.html).relname | 表名。 |
| attname | int2vector | [PG\_STATISTIC\_EXT](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_STATISTIC_EXT.html).stakey | 统计信息扩展的多列信息。 |
| inherited | Boolean | - | 如果为真，则包含继承的子列，否则只是指定表的字段。 |
| null\_frac | real | - | 记录中字段组合为空的百分比。 |
| avg\_width | integer | - | 字段组合记录以字节记的平均宽度。 |
| n\_distinct | real | - | * 如果大于零，表示字段组合中独立数值的估计数目。 * 如果小于零，表示独立数值的数目被行数除的负数。  1. 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 2. 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个字段组合中独立数值的个数和行数相同。  * 如果等于零，表示独立数值的数目未知。 |
| n\_dndistinct | real | - | 标识dn1上字段组合中非NULL数据的唯一值的数目。如果大于零，表示独立数值的实际数目。   * 如果小于零，表示独立数值的数目被行数除的负数。（比如，一个字段组合的数值平均出现概率为两次，则可以表示为n\_dndistinct=-0.5）。 * 如果等于零，表示独立数值的数目未知。 |
| most\_common\_vals | anyarray | - | 一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值均不为NULL。 |
| most\_common\_freqs | real[] | - | 一个最常用数值组合的频率的列表，也就是说，每个出现的次数除以行数。如果most\_common\_vals是NULL，则为NULL。 |
| most\_common\_vals\_null | anyarray | - | 一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值中至少有一个值为NULL。 |
| most\_common\_freqs\_null | real[] | - | 一个最常用数值组合的频率的列表，也就是说，每个出现的次数除以行数。如果most\_common\_vals\_null是NULL，则为NULL。 |
| histogram\_bounds | anyarray | - | 直方图的边界值列表。 |

### PG\_GET\_INVALID\_BACKENDS

PG\_GET\_INVALID\_BACKENDS视图提供显示数据库主节点上连接到当前备机的后台线程信息。

**表 1** PG\_GET\_INVALID\_BACKENDS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pid | bigint | 线程ID。 |
| node\_name | text | 后台线程中连接的节点信息。 |
| dbname | name | 当前连接的数据库。 |
| backend\_start | timestamp with time zone | 后台线程启动的时间。 |
| query | text | 后台线程正在执行的查询语句。 |

### PG\_GET\_SENDERS\_CATCHUP\_TIME

PG\_GET\_SENDERS\_CATCHUP\_TIME视图显示数据库节点上当前活跃的主备发送线程的追赶信息。

**表 1** PG\_GET\_SENDERS\_CATCHUP\_TIME字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pid | bigint | 当前sender的线程ID。 |
| lwpid | integer | 当前sender的lwpid。 |
| local\_role | text | 本地的角色。 |
| peer\_role | text | 对端的角色。 |
| state | text | 当前sender的复制状态。 |
| type | text | 当前sender的类型。 |
| catchup\_start | timestamp with time zone | catchup启动的时间。 |
| catchup\_end | timestamp with time zone | catchup结束的时间。 |

### PG\_GROUP

PG\_GROUP视图查看数据库认证角色及角色之间的成员关系。

**表 1** PG\_GROUP字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| groname | name | 组的名称。 |
| grosysid | oid | 组的ID。 |
| grolist | oid[] | 一个数组，包含这个组里面所有角色的ID。 |

### PG\_GTT\_ATTACHED\_PIDS

PG\_GTT\_ATTACHED\_PIDS视图查看哪些会话正在使用全局临时表，调用pg\_get\_attached\_pid函数。

**表 1** PG\_GTT\_ATTACHED\_PIDS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | schema名称。 |
| tablename | name | 全局临时表名称。 |
| relid | oid | 全局临时表的oid。 |
| pids | bigint[] | 线程pid列表。 |

### PG\_GTT\_RELSTATS

PG\_GTT\_RELSTATS视图查看当前会话所有全局临时表基本信息，调用pg\_get\_gtt\_relstats函数。

**表 1** PG\_GTT\_RELSTATS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | schema名称。 |
| tablename | name | 全局临时表名称。 |
| relfilenode | oid | 文件对象的ID。 |
| relpages | integer | 全局临时表的磁盘页面数。 |
| reltuples | real | 全局临时表的记录数。 |
| relallvisible | integer | 被标识为全可见的页面数。 |
| relfrozenxid | xid | 该表中所有在这个之前的事务ID已经被一个固定的（frozen）事务ID替换。 |
| relminmxid | xid | 预留接口，暂未启用。 |

### PG\_GTT\_STATS

PG\_GTT\_STATS视图查看当前会话所有全局临时表单列统计信息，调用pg\_get\_gtt\_statistics函数。

**表 1** PG\_GTT\_STATS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | schema名称。 |
| tablename | name | 全局临时表名称。 |
| attname | name | 属性名称。 |
| inherited | boolean | 是否统计有继承关系的对象。 |
| null\_frac | real | 该字段中为NULL的记录的比率。 |
| avg\_width | integer | 非NULL记录的平均存储宽度，以字节计算。 |
| n\_distinct | real | 标识全局统计信息中字段里唯一的非NULL数据值的数目。 |
| most\_common\_vals | text[] | 高频值列表，按照出现的频率排序。 |
| most\_common\_freqs | real[] | 高频值的频率。 |
| histogram\_bounds | text[] | 等频直方图描述列中的数据分布（不包含高频值）。 |
| correlation | real | 相关系数。 |
| most\_common\_elems | text[] | 类型高频值列表，用于数组类型或一些其他类型。 |
| most\_common\_elem\_freqs | real[] | 类型高频值的频率。 |
| elem\_count\_histogram | real[] | 数组类型直方图。 |

### PG\_INDEXES

PG\_INDEXES视图提供对数据库中每个索引的有用信息的访问。

**表 1** PG\_INDEXES字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | PG\_NAMESPACE.nspname | 包含表和索引的模式名称。 |
| tablename | name | PG\_CLASS.relname | 此索引所服务的表的名称。 |
| indexname | name | PG\_CLASS.relname | 索引的名称。 |
| tablespace | name | PG\_TABLESPACE.nspname | 包含索引的表空间名称。 |
| indexdef | text | - | 索引定义（一个重建的CREATE INDEX命令）。 |

### PG\_LOCKS

PG\_LOCKS视图存储各打开事务所持有的锁信息。

**表 1** PG\_LOCKS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| locktype | text | - | 被锁定对象的类型：relation、extend、page、tuple、transactionid、virtualxid、object、userlock、advisory。 |
| database | oid | [PG\_DATABASE](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_DATABASE.html).oid | 被锁定对象所在数据库的OID。   * 如果被锁定的对象是共享对象，则OID为0。 * 如果是一个事务ID，则为NULL。 |
| relation | oid | [PG\_CLASS](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_CLASS.html).oid | 关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。 |
| page | integer | - | 关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。 |
| tuple | smallint | - | 页面里边的行编号，如果对象不是行，则为NULL。 |
| bucket | integer | - | 子表对应的bucket number。如果目标不是表的话，则为NULL。 |
| virtualxid | text | - | 事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。 |
| transactionid | xid | - | 事务的ID，如果对象不是一个事务ID，则为NULL。 |
| classid | oid | [PG\_CLASS](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_CLASS.html).oid | 包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。 |
| objid | oid | - | 对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。 |
| objsubid | smallint | - | * 对于表的一个字段，这是字段编号； * 对于其他对象类型，这个字段是0； * 如果这个对象不是普通数据库对象，则为NULL。 |
| virtualtransaction | text | - | 持有此锁或者在等待此锁的事务的虚拟ID。 |
| pid | bigint | - | 持有或者等待这个锁的服务器线程的逻辑ID。如果锁是被一个预备事务持有的，则为NULL。 |
| sessionid | bigint | - | 持有或者等待这个锁的会话ID。 |
| mode | text | - | 这个线程持有的或者是期望的锁模式。 |
| granted | Boolean | - | * 如果锁是持有锁，则为TRUE。 * 如果锁是等待锁，则为FALSE。 |
| fastpath | Boolean | - | * 如果通过fast-path获得锁，则为TRUE； * 如果通过主要的锁表获得，则为FALSE。 |
| locktag | text | - | 会话等待锁信息，可通过locktag\_decode()函数解析。 |
| global\_sessionid | text |  | 全局会话ID。 |

### PG\_NODE\_ENV

PG\_NODE\_ENV视图提供获取当前节点的环境变量信息。

**表 1** PG\_NODE\_ENV字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 当前节点的名称。 |
| host | text | 当前节点的主机名称。 |
| process | integer | 当前节点的进程号。 |
| port | integer | 当前节点的端口号。 |
| installpath | text | 当前节点的安装目录。 |
| datapath | text | 当前节点的数据目录。 |
| log\_directory | text | 当前节点的日志目录。 |

### PG\_OS\_THREADS

PG\_OS\_THREADS视图提供当前节点下所有线程的状态信息。

**表 1** PG\_OS\_THREADS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 当前节点的名称。 |
| pid | bigint | 当前节点进程中正在运行的线程号。 |
| lwpid | integer | 与pid对应的轻量级线程号。 |
| thread\_name | text | 与pid对应的线程名称。 |
| creation\_time | timestamp with time zone | 与pid对应的线程创建的时间。 |

### PG\_PREPARED\_STATEMENTS

PG\_PREPARED\_STATEMENTS视图显示当前会话所有可用的预备语句。

**表 1** PG\_PREPARED\_STATEMENTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | text | 预备语句的标识符。 |
| statement | text | 创建该预备语句的查询字符串。对于从SQL创建的预备语句而言是客户端提交的PREPARE语句；对于通过前/后端协议创建的预备语句而言是预备语句自身的文本。 |
| prepare\_time | timestamp with time zone | 创建该预备语句的时间戳。 |
| parameter\_types | regtype[] | 该预备语句期望的参数类型，以regtype类型的数组格式出现。与该数组元素相对应的OID可以通过把regtype转换为oid值得到。 |
| from\_sql | Boolean | * 如果该预备语句是通过PREPARE语句创建的则为true。 * 如果是通过前/后端协议创建的则为false。 |

### PG\_PREPARED\_XACTS

PG\_PREPARED\_XACTS视图显示当前准备好进行两阶段提交的事务的信息。

**表 1** PG\_PREPARED\_XACTS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| transaction | xid | - | 预备事务的数字事务标识。 |
| gid | text | - | 赋予该事务的全局事务标识。 |
| prepared | timestamp with time zone | - | 事务准备好提交的时间。 |
| owner | name | PG\_AUTHID.rolname | 执行该事务的用户的名称。 |
| database | name | PG\_DATABASE.datname | 执行该事务所在的数据库名。 |

### PG\_PUBLICATION\_TABLES

视图PG\_PUBLICATION\_TABLES提供publication与其所发布的表之间的映射信息。和底层的系统表pg\_publication\_rel不同，这个视图展开了定义为FOR ALL TABLES的publication，这样对这类publication来说，每一个合格的表都有一行。

**表 1** PG\_PUBLICATION\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pubname | name | 发布的名称。 |
| schemaname | name | 包含表的模式名称。 |
| tablename | name | 表的名称。 |

### PG\_REPLICATION\_ORIGIN\_STATUS

获取复制源的复制状态。

**表 1** PG\_REPLICATION\_ORIGIN\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| local\_id | oid | 复制源ID。 |
| external\_id | text | 复制源名称。 |
| remote\_lsn | LSN | 复制源的lsn位置。 |
| local\_lsn | LSN | 本地的lsn位置。 |

### PG\_REPLICATION\_SLOTS

PG\_REPLICATION\_SLOTS视图查看复制节点的信息。

**表 1** PG\_REPLICATION\_SLOTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| slot\_name | text | 复制节点的名称。 |
| plugin | text | 逻辑复制槽对应的输出插件名。 |
| slot\_type | text | 复制节点的类型。 |
| datoid | oid | 复制节点的数据库OID。 |
| database | name | 复制节点的数据库名称。 |
| active | Boolean | 复制节点是否为激活状态。 |
| xmin | xid | 复制节点事务标识。 |
| catalog\_xmin | xid | 逻辑复制槽对应的最早解码事务标识。 |
| restart\_lsn | text | 复制节点的Xlog文件信息。 |
| dummy\_standby | Boolean | 复制节点是否为假备。 |

### PG\_RLSPOLICIES

PG\_RLSPOLICIES视图提供查询行级访问控制策略。

**表 1** PG\_RLSPOLICIES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 行级访问控制策略作用的表对象所属模式名称。 |
| tablename | name | 行级访问控制策略作用的表对象名称。 |
| policyname | name | 行级访问控制策略名称。 |
| policypermissive | text | 行级访问控制策略属性。 |
| policyroles | name[] | 行级访问控制策略影响的用户列表，不指定表示影响所有的用户。 |
| policycmd | text | 行级访问控制策略影响的SQL操作。 |
| policyqual | text | 行级访问控制策略的表达式。 |

### PG\_ROLES

PG\_ROLES视图提供访问数据库角色的相关信息，初始化用户和具有sysadmin属性或createrole属性的用户可以查看全部角色的信息，其他用户只能查看自己的信息。

**表 1** PG\_ROLES字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| rolname | name | - | 角色名称。 |
| rolsuper | Boolean | - | 该角色是否是拥有最高权限的初始系统管理员。 |
| rolinherit | Boolean | - | 该角色是否继承角色的权限。 |
| rolcreaterole | Boolean | - | 该角色是否可以创建其他的角色。 |
| rolcreatedb | Boolean | - | 该角色是否可以创建数据库。 |
| rolcatupdate | Boolean | - | 该角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。 |
| rolcanlogin | Boolean | - | 该角色是否可以登录数据库。 |
| rolreplication | Boolean | - | 该角色是否可以复制。 |
| rolauditadmin | Boolean | - | 该角色是否为审计管理员。 |
| rolsystemadmin | Boolean | - | 该角色是否为系统管理员。 |
| rolconnlimit | integer | - | 对于可以登录的角色，这里限制了该角色允许发起的最大并发连接数。-1表示无限制。 |
| rolpassword | text | - | 不是口令，总是**\*\***\*\*。 |
| rolvalidbegin | timestamp with time zone | - | 帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。 |
| rolvaliduntil | timestamp with time zone | - | 帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。 |
| rolrespool | name | - | 用户所能够使用的resource pool。 |
| rolparentid | oid | PG\_AUTHID.rolparentid | 用户所在组用户的OID。 |
| roltabspace | text | - | 用户永久表存储空间限额。 |
| roltempspace | text | - | 用户临时表存储空间限额，单位为KB。 |
| rolspillspace | text | - | 用户算子落盘空间限额，单位为KB。 |
| rolconfig | text[] | - | 运行时配置变量的会话缺省。 |
| oid | oid | PG\_AUTHID.oid | 角色的ID。 |
| roluseft | Boolean | PG\_AUTHID.roluseft | 角色是否可以操作外表。 |
| rolkind | “char” | - | 角色类型。 |
| nodegroup | name | - | 该字段不支持。 |
| rolmonitoradmin | Boolean | - | 该角色是否为监控管理员。 |
| roloperatoradmin | Boolean | - | 该角色是否为运维管理员。 |
| rolpolicyadmin | Boolean | - | 该角色是否为安全策略管理员。 |
| rolssoadmin | Boolean | - | 该角色是否为安全管理员。 |

### PG\_RULES

PG\_RULES视图提供对查询重写规则的有用信息访问的接口。

**表 1** PG\_RULES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| schemaname | name | 包含表的模式的名称。 |
| tablename | name | 规则作用的表的名称。 |
| rulename | name | 规则的名称。 |
| definition | text | 规则定义（一个重新构造的创建命令）。 |

### PG\_RUNNING\_XACTS

PG\_RUNNING\_XACTS视图主要功能是显示当前节点运行事务的信息。

**表 1** PG\_RUNNING\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| handle | integer | 事务对应的事务管理器中的槽位句柄，该值恒为-1。 |
| gxid | xid | 事务id号。 |
| state | tinyint | 事务状态（3：prepared或者0：starting）。 |
| node | text | 节点名称。 |
| xmin | xid | 节点上当前数据涉及的最小事务号xmin。 |
| vacuum | Boolean | 标志当前事务是否是lazy vacuum事务。   * t（true）：表示是。 * f（false）：表示否。 |
| timeline | bigint | 标志数据库重启次数。 |
| prepare\_xid | xid | 处于prepared状态的事务的id号，若不在prepared状态，值为0。 |
| pid | bigint | 事务对应的线程id。 |
| next\_xid | xid | 其余节点发送给当前节点的事务id，该值恒为0。 |

### PG\_SECLABELS

PG\_SECLABELS视图提供关于安全标签的信息。

**表 1** PG\_SECLABELS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| objoid | oid | 任意OID属性 | 这个安全标签指向的对象的OID。 |
| classoid | oid | PG\_CLASS.oid | 这个对象出现的系统表的OID。 |
| objsubid | integer | - | 对于一个在表字段上的安全标签，是字段编号（引用表本身的objoid和classoid）。对于所有其他对象类型，这个字段为0。 |
| objtype | text | - | 这个标签出现的对象的类型，文本格式。 |
| objnamespace | oid | PG\_NAMESPACE.oid | 这个对象的名称空间的OID，如果适用；否则为NULL。 |
| objname | text | - | 这个标签适用的对象的名称，文本格式。 |
| provider | text | PG\_SECLABEL.provider | 与这个标签相关的标签提供者。 |
| label | text | PG\_SECLABEL.label | 适用于这个对象的安全标签。 |

### PG\_SESSION\_IOSTAT

PG\_SESSION\_IOSTAT视图显示当前用户执行作业正在运行时的IO负载管理相关信息。查询该视图需要sysadmin权限或者monitor admin权限。

以下涉及到iops，对于行存，均以万次/s为单位，对于列存，均以次/s为单位。

**表 1** PG\_SESSION\_IOSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| query\_id | bigint | 作业id。 |
| mincurriops | integer | 该作业当前io在数据库实例中的最小值。 |
| maxcurriops | integer | 该作业当前io在数据库实例中的最大值。 |
| minpeakiops | integer | 在作业运行时，作业io峰值中，数据库实例的最小值。 |
| maxpeakiops | integer | 在作业运行时，作业io峰值中，数据库实例的最大值。 |
| io\_limits | integer | 该作业所设GUC参数io\_limits。 |
| io\_priority | text | 该作业所设GUC参数io\_priority。 |
| query | text | 作业。 |
| node\_group | text | 该字段不支持。 |
| curr\_io\_limits | integer | 使用io\_priority管控io时的实时io\_limits值。 |

### PG\_SESSION\_WLMSTAT

PG\_SESSION\_WLMSTAT视图显示当前用户执行作业正在运行时的负载管理相关信息。查询该视图需要sysadmin权限。

**表 1** PG\_SESSION\_WLMSTAT字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 连接后端的数据库OID。 |
| datname | name | 连接后端的数据库名称。 |
| threadid | bigint | 后端线程ID。 |
| sessionid | bigint | 会话ID。 |
| processid | integer | 后端线程的pid。 |
| usesysid | oid | 登录后端的用户OID。 |
| appname | text | 连接到后端的应用名。 |
| usename | name | 登录到该后端的用户名。 |
| priority | bigint | 语句所在Cgroups的优先级。 |
| attribute | text | 语句的属性：   * Ordinary：语句发送到数据库后被解析前的默认属性。 * Simple：简单语句。 * Complicated：复杂语句。 * Internal：数据库内部语句。Unknown：未知。 |
| block\_time | bigint | 语句当前为止的pending的时间，单位s。 |
| elapsed\_time | bigint | 语句当前为止的实际执行时间，单位s。 |
| total\_cpu\_time | bigint | 语句在上一时间周期内的数据库实例上CPU使用的总时间，单位s。 |
| cpu\_skew\_percent | integer | 语句在上一时间周期内的数据库实例上CPU使用的倾斜率。 |
| statement\_mem | integer | 语句执行使用的statement\_mem，预留字段。 |
| active\_points | integer | 语句占用的资源池并发点数。 |
| dop\_value | integer | 语句的从资源池中获取的dop值。 |
| control\_group | text | 该字段不支持。 |
| status | text | 语句当前的状态，包括：   * pending：执行前状态。 * running：执行进行状态。 * finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。）aborted：执行异常终止。 * active：非以上四种状态外的正常状态。 * unknown：未知状态。 |
| enqueue | text | 该字段不支持。 |
| resource\_pool | name | 语句当前所在的资源池。 |
| query | text | 该后端的最新查询。如果state状态是active（活的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| is\_plana | boolean | 该字段不支持。 |
| node\_group | text | 该字段不支持。 |

### PG\_SETTINGS

PG\_SETTINGS视图显示数据库运行时参数的相关信息。

**表 1** PG\_SETTINGS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| name | text | 参数名称。 |
| setting | text | 参数当前值。 |
| unit | text | 参数的隐式结构。 |
| category | text | 参数的逻辑组。 |
| short\_desc | text | 参数的简单描述。 |
| extra\_desc | text | 参数的详细描述。 |
| context | text | 设置参数值的上下文，包括internal、postmaster、sighup、backend、superuser、user。 |
| vartype | text | 参数类型，包括bool、enum、integer、real、string。 |
| source | text | 参数的赋值方式。 |
| min\_val | text | 参数最小值。如果参数类型不是数值型，那么该字段值为null。 |
| max\_val | text | 参数最大值。如果参数类型不是数值型，那么该字段值为null。 |
| enumvals | text[] | enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。 |
| boot\_val | text | 数据库启动时参数默认值。 |
| reset\_val | text | 数据库重置时参数默认值。 |
| sourcefile | text | 设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。 |
| sourceline | integer | 设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。 |

### PG\_SHADOW

PG\_SHADOW视图显示了所有在PG\_AUTHID中标记了rolcanlogin的角色的属性。

这个视图的名称来自于该视图是不可读的，因为它包含口令。PG\_USER（参考：系统表和系统视图->系统视图->PG\_USER章节）是一个在PG\_SHADOW上公开可读的视图，只是把口令域填成了空白。

**表 1** PG\_SHADOW字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| usename | name | .rolname | 用户名。 |
| usesysid | oid | .oid | 用户的ID。 |
| usecreatedb | Boolean | - | 用户可以创建数据库。 |
| usesuper | Boolean | - | 用户是系统管理员。 |
| usecatupd | Boolean | - | 用户可以更新视图。即使是系统管理员，如果这个字段不是真，也不能更新视图。 |
| userepl | Boolean | - | 用户可以初始化流复制和使系统处于或不处于备份模式。 |
| passwd | text | - | 获取加密的口令是如何存储的信息。 |
| valbegin | timestamp with time zone | - | 帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。 |
| valuntil | timestamp with time zone | - | 帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。 |
| respool | name | - | 用户使用的资源池。 |
| parent | oid | - | 父资源池。 |
| spacelimit | text | - | 永久表存储空间限额。 |
| tempspacelimit | text | - | 临时表存储空间限额。 |
| spillspacelimit | text | - | 算子落盘空间限额。 |
| usemonitoradmin | boolean | - | 用户是监控管理员。 |
| useoperatoradmin | boolean | - | 用户是运维管理员。 |
| usepolicyadmin | boolean | - | 用户是安全策略管理员。 |
| useconfig | text[] | - | 运行时配置变量的会话默认值。 |

### PG\_STAT\_ACTIVITY

PG\_STAT\_ACTIVITY视图显示和当前用户查询相关的信息，字段保存的是上一次执行的信息。

**表 1** PG\_STAT\_ACTIVITY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| datname | name | 用户会话在后台连接到的数据库名称。 |
| pid | bigint | 后台线程ID。 |
| sessionid | bigint | 会话ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| usename | name | 登录该后台的用户名。 |
| application\_name | text | 连接到该后台的应用名。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| backend\_start | timestamp with time zone | 该过程开始的时间，即当客户端连接服务器时。 |
| xact\_start | timestamp with time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestamp with time zone | 开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。 |
| state\_change | timestamp with time zone | 上次状态改变的时间。 |
| waiting | boolean | 如果后台当前正等待锁则为true。 |
| enqueue | text | 该字段不支持。 |
| state | text | 该后台当前总体状态。可能值是：   * active：后台正在执行一个查询。 * idle：后台正在等待一个新的客户端命令。idle in transaction：后台在事务中，但事务中没有语句在执行。 * idle in transaction (aborted)：后台在事务中，但事务中有语句执行失败。 * fastpath function call：后台正在执行一个fast-path函数。 * disabled：如果后台禁用track\_activities，则报告这个状态。   说明：普通用户只能查看到自己帐户所对应的会话状态。即其他帐户的state信息为空。例如以judy用户连接数据库后，在pg\_stat\_activity中查看到的普通用户joe及初始用户vastbase的state信息为空：  SELECT datname, usename, usesysid, state,pid FROM pg\_stat\_activity; datname | usename | usesysid | state | pid ———-+———+———-+——–+—————– postgres | vastbase | 10 | | 139968752121616 postgres | vastbase | 10 | | 139968903116560 db\_tpcc | judy | 16398 | active | 139968391403280 postgres | vastbase | 10 | | 139968643069712 postgres | vastbase | 10 | | 139968680818448 postgres | joe | 16390 | | 139968563377936 (6 rows)` |
| resource\_pool | name | 用户使用的资源池。 |
| query\_id | bigint | 查询语句的ID。 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| connection\_info | text | json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息。 |
| unique\_sql\_id | bigint | 语句的unique sql id。 |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

### PG\_STAT\_ACTIVITY\_NG

PG\_STAT\_ACTIVITY\_NG视图显示在当前用户所属的逻辑数据库实例下，所有查询的相关信息。

**表 1** PG\_STAT\_ACTIVITY\_NG字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 用户会话在后台连接到的数据库OID。 |
| datname | name | 用户会话在后台连接到的数据库名称。 |
| pid | bigint | 后台线程ID。 |
| sessionid | bigint | 会话ID。 |
| usesysid | oid | 登录该后台的用户OID。 |
| usename | name | 登录该后台的用户名。 |
| application\_name | text | 连接到该后台的应用名。 |
| client\_addr | inet | 连接到该后台的客户端的IP地址。如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。 |
| client\_hostname | text | 客户端的主机名，这个字段是通过client\_addr的反向DNS查找得到。这个字段只有在启动log\_hostname且使用IP连接时才非空。 |
| client\_port | integer | 客户端用于与后台通讯的TCP端口号，如果使用Unix套接字，则为-1。 |
| backend\_start | timestamp with time zone | 该过程开始的时间，即当客户端连接服务器时。 |
| xact\_start | timestamp with time zone | 启动当前事务的时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query\_start列。 |
| query\_start | timestamp with time zone | 开始当前活跃查询的时间，如果state的值不是active，则这个值是上一个查询的开始时间。 |
| state\_change | timestamp with time zone | 上次状态改变的时间。 |
| waiting | boolean | 如果后台当前正等待锁则为true。否则为false。 |
| enqueue | text | 语句当前排队状态。可能值是：   * waiting in queue：表示语句在排队中。 * 空：表示语句正在运行。 |
| state | text | 该后台当前总体状态。可能值是：active：后台正在执行一个查询。idle：后台正在等待一个新的客户端命令。idle in transaction：后台在事务中，但事务中没有语句在执行。idle in transaction (aborted)：后台在事务中，但事务中有语句执行失败。fastpath function call：后台正在执行一个fast-path函数。disabled：如果后台禁用track\_activities，则报告这个状态。说明：普通用户只能查看到自己帐户所对应的会话状态。即其他帐户的state信息为空。例如以judy用户连接数据库后，在pg\_stat\_activity中查看到的普通用户joe及初始用户vastbase的state信息为空：  SELECT datname, usename, usesysid, state,pid FROM pg\_stat\_activity\_ng; datname | usename | usesysid | state | pid ———-+———+———-+——–+—————– postgres | vastbase | 10 | | 139968752121616 postgres | vastbase | 10 | | 139968903116560 db\_tpcds | judy | 16398 | active | 139968391403280 postgres | vastbase | 10 | | 139968643069712 postgres | vastbase | 10 | | 139968680818448 postgres | joe | 16390 | | 139968563377936 (6 rows) |
| resource\_pool | name | 用户使用的资源池。 |
| query\_id | bigint | 查询语句的ID。 |
| query | text | 该后台的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。 |
| node\_group | text | 语句所属用户对应的逻辑数据库实例。 |

### PG\_STAT\_ALL\_INDEXES

PG\_STAT\_ALL\_INDEXES视图将包含当前数据库中的每个索引行，显示访问特定索引的统计。

索引可以通过简单的索引扫描或“位图”索引扫描进行使用。位图扫描中几个索引的输出可以通过AND或者OR规则进行组合， 因此当使用位图扫描的时候，很难将独立堆行抓取与特定索引进行组合，因此，一个位图扫描增加pg\_stat\_all\_indexes.idx\_tup\_read使用索引计数，并且增加pg\_stat\_all\_tables.idx\_tup\_fetch表计数，但不影响pg\_stat\_all\_indexes.idx\_tup\_fetch。

**表 1** PG\_STAT\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引的模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

### PG\_STAT\_ALL\_TABLES

PG\_STAT\_ALL\_TABLES视图将包含当前数据库中每个表的一行（包括TOAST表），显示访问特定表的统计信息。

**表 1** PG\_STAT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次清理该表的时间。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理该表的时间。 |
| last\_analyze | timestamp with time zone | 上次分析该表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析该表的时间。 |
| vacuum\_count | bigint | 这个表被清理的次数。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |
| last\_data\_changed | timestamp with time zone | 记录这个表上一次数据发生变化的时间（引起数据变化的操作包括INSERT/UPDATE/DELETE、EXCHANGE/TRUNCATE/DROP partition），该列数据仅在本地数据库主节点记录。 |

### PG\_STAT\_BAD\_BLOCK

PG\_STAT\_BAD\_BLOCK视图显示自节点启动后，读取数据时出现Page/CU校验失败的统计信息。

**表 1** PG\_STAT\_BAD\_BLOCK字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 节点名。 |
| databaseid | integer | 数据库OID。 |
| tablespaceid | integer | 表空间OID。 |
| relfilenode | integer | 文件对象ID。 |
| bucketid | smallint | 一致性hash bucket ID。 |
| forknum | integer | 文件类型。取值如下：  0，数据主文件。  1，FSM文件。  2，VM文件。  3，BCM文件。  大于4，列存表每个字段的数据文件。 |
| error\_count | integer | 出现校验失败的次数。 |
| first\_time | timestamp with time zone | 第一次出现时间。 |
| last\_time | timestamp with time zone | 最近一次出现时间。 |

### PG\_STAT\_BGWRITER

PG\_STAT\_BGWRITER视图显示关于后端写进程活动的统计信息。

**表 1** PG\_STAT\_BGWRITER字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| checkpoints\_timed | bigint | 执行的定期检查点数。 |
| checkpoints\_req | bigint | 执行的需求检查点数。 |
| checkpoint\_write\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被写入到磁盘，以毫秒为单位。 |
| checkpoint\_sync\_time | double precision | 花费在检查点处理部分的时间总量，其中文件被同步到磁盘，以毫秒为单位。 |
| buffers\_checkpoint | bigint | 检查点写缓冲区数量。 |
| buffers\_clean | bigint | 后端写进程写缓冲区数量。 |
| maxwritten\_clean | bigint | 后端写进程停止清理扫描时间数，因为它写了太多缓冲区。 |
| buffers\_backend | bigint | 通过后端直接写缓冲区数。 |
| buffers\_backend\_fsync | bigint | 后端不得不执行自己的fsync调用的时间数（通常后端写进程处理这些即使后端确实自己写）。 |
| buffers\_alloc | bigint | 分配的缓冲区数量。 |
| stats\_reset | timestamp with time zone | 这些统计被重置的时间。 |

### PG\_STAT\_DATABASE

PG\_STAT\_DATABASE视图将包含Vastbase中每个数据库的数据库统计信息。

**表 1** PG\_STAT\_DATABASE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 数据库的OID。 |
| datname | name | 这个数据库的名称。 |
| numbackends | integer | 当前连接到该数据库的后端数。这是在返回一个反映目前状态值的视图中唯一的列；自上次重置所有其他列返回累积值。 |
| xact\_commit | bigint | 此数据库中已经提交的事务数。 |
| xact\_rollback | bigint | 此数据库中已经回滚的事务数。 |
| blks\_read | bigint | 在这个数据库中读取的磁盘块的数量。 |
| blks\_hit | bigint | 高速缓存中已经发现的磁盘块的次数， 这样读取是不必要的（这只包括Vastbase缓冲区高速缓存，没有操作系统的文件系统缓存）。 |
| tup\_returned | bigint | 通过数据库查询返回的行数。 |
| tup\_fetched | bigint | 通过数据库查询抓取的行数。 |
| tup\_inserted | bigint | 通过数据库查询插入的行数。 |
| tup\_updated | bigint | 通过数据库查询更新的行数。 |
| tup\_deleted | bigint | 通过数据库查询删除的行数。 |
| conflicts | bigint | 由于数据库恢复冲突取消的查询数量（只在备用服务器发生的冲突）。请参见[PG\_STAT\_DATABASE\_CONFLICTS](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_STAT_DATABASE_CONFLICTS.html)获取更多信息。 |
| temp\_files | bigint | 通过数据库查询创建的临时文件数量。计算所有临时文件， 不论为什么创建临时文件（比如排序或者哈希）， 而且不管log\_temp\_files设置。 |
| temp\_bytes | bigint | 通过数据库查询写入临时文件的数据总量。计算所有临时文件，不论为什么创建临时文件，而且不管log\_temp\_files设置。 |
| deadlocks | bigint | 在该数据库中检索的死锁数。 |
| blk\_read\_time | double precision | 通过数据库后端读取数据文件块花费的时间，以毫秒计算。 |
| blk\_write\_time | double precision | 通过数据库后端写入数据文件块花费的时间，以毫秒计算。 |
| stats\_reset | timestamp with time zone | 重置当前状态统计的时间。 |

### PG\_STAT\_DATABASE\_CONFLICTS

PG\_STAT\_DATABASE\_CONFLICTS视图显示数据库冲突状态的统计信息。

**表 1** PG\_STAT\_DATABASE\_CONFLICTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| datid | oid | 数据库标识。 |
| datname | name | 数据库名称。 |
| confl\_tablespace | bigint | 冲突的表空间的数目。 |
| confl\_lock | bigint | 冲突的锁数目。 |
| confl\_snapshot | bigint | 冲突的快照数目。 |
| confl\_bufferpin | bigint | 冲突的缓冲区数目。 |
| confl\_deadlock | bigint | 冲突的死锁数目。 |

### PG\_STAT\_REPLICATION

PG\_STAT\_REPLICATION视图用于描述日志同步状态信息，例如发起端发送日志位置，接收端接收日志位置等。

**表 1** PG\_STAT\_REPLICATION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pid | bigint | 线程的PID。 |
| usesysid | oid | 用户系统ID。 |
| usename | name | 用户名。 |
| application\_name | text | 程序名称。 |
| client\_addr | inet | 客户端地址。 |
| client\_hostname | text | 客户端名。 |
| client\_port | integer | 客户端端口。 |
| backend\_start | timestamp with time zone | 程序启动时间。 |
| state | text | 日志复制的状态：   * 追赶状态。 * 一致的流状态。 |
| sender\_sent\_location | text | 发送端发送日志位置。 |
| receiver\_write\_location | text | 接收端write日志位置。 |
| receiver\_flush\_location | text | 接收端flush日志位置。 |
| receiver\_replay\_location | text | 接收端replay日志位置。 |
| sync\_priority | integer | 同步复制的优先级（0表示异步）。 |
| sync\_state | text | 同步状态：   * 异步复制。 * 同步复制。 * 潜在同步者。 |

### PG\_STAT\_SUBSCRIPTION

获取订阅的详细同步信息。

**表 1** PG\_STAT\_SUBSCRIPTION字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| subid | oid | 订阅的OID。 |
| subname | name | 订阅的名称。 |
| pid | thread\_id | 后台apply线程的thread id。 |
| received\_lsn | LSN | 从发布端接收到的最近的lsn。 |
| last\_msg\_send\_time | timestamp | 最近发布端发送消息的时间。 |
| last\_msg\_receipt\_time | timestamp | 最新订阅端收到消息的时间。 |
| latest\_end\_lsn | LSN | 最近一次收到保活消息时发布端的lsn。 |
| latest\_end\_time | timestamp | 最近一次收到保活消息的时间。 |

### PG\_STAT\_SYS\_INDEXES

PG\_STAT\_SYS\_INDEXES视图显示pg\_catalog、information\_schema模式中所有系统表的索引状态信息。

**表 1** PG\_STAT\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引的模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

### PG\_STAT\_SYS\_TABLES

PG\_STAT\_SYS\_TABLES视图显示pg\_catalog、information\_schema模式的所有命名空间中系统表的统计信息。

**表 1** PG\_STAT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次该表是手动清理的时间（不计算VACUUM FULL）。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的时间。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表的时间。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的时间。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被手动分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |
| last\_data\_changed | timestamp with time zone | 这个表数据最近修改时间。 |

### PG\_STAT\_USER\_FUNCTIONS

PG\_STAT\_USER\_FUNCTIONS视图显示命名空间中用户自定义函数（函数语言为非内部语言）的状态信息。

**表 1** PG\_STAT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcid | oid | 函数标识。 |
| schemaname | name | 模式的名称。 |
| funcname | name | 函数名称。 |
| calls | bigint | 函数被调用的次数。 |
| total\_time | double precision | 函数的总执行时长。 |
| self\_time | double precision | 当前线程调用函数的总的时长。 |

### PG\_STAT\_USER\_INDEXES

PG\_STAT\_USER\_INDEXES视图显示数据库中用户自定义普通表和toast表的索引状态信息。

**表 1** PG\_STAT\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 这个索引的表的OID。 |
| indexrelid | oid | 索引的OID。 |
| schemaname | name | 索引的模式名。 |
| relname | name | 索引的表名。 |
| indexrelname | name | 索引名。 |
| idx\_scan | bigint | 索引上开始的索引扫描数。 |
| idx\_tup\_read | bigint | 通过索引上扫描返回的索引项数。 |
| idx\_tup\_fetch | bigint | 通过使用索引的简单索引扫描抓取的活表行数。 |

### PG\_STAT\_USER\_TABLES

PG\_STAT\_USER\_TABLES视图显示所有命名空间中用户自定义普通表和toast表的状态信息。

**表 1** PG\_STAT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（即没有更新所需的单独索引）。 |
| n\_live\_tup | bigint | 估计活跃行数。 |
| n\_dead\_tup | bigint | 估计死行数。 |
| last\_vacuum | timestamp with time zone | 最后一次该表是手动清理的（不计算VACUUM FULL）。 |
| last\_autovacuum | timestamp with time zone | 上次被autovacuum守护进程清理的表。 |
| last\_analyze | timestamp with time zone | 上次手动分析这个表。 |
| last\_autoanalyze | timestamp with time zone | 上次被autovacuum守护进程分析的表。 |
| vacuum\_count | bigint | 这个表被手动清理的次数（不计算VACUUM FULL）。 |
| autovacuum\_count | bigint | 这个表被autovacuum清理的次数。 |
| analyze\_count | bigint | 这个表被手动分析的次数。 |
| autoanalyze\_count | bigint | 这个表被autovacuum守护进程分析的次数。 |
| last\_data\_changed | timestamp with time zone | 这个表数据最近修改时间。 |

### PG\_STAT\_XACT\_ALL\_TABLES

PG\_STAT\_XACT\_ALL\_TABLES视图显示命名空间中所有普通表和toast表的事务状态信息。

**表 1** PG\_STAT\_XACT\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

### PG\_STAT\_XACT\_SYS\_TABLES

PG\_STAT\_XACT\_SYS\_TABLES视图显示命名空间中系统表的事务状态信息。

**表 1** PG\_STAT\_XACT\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

### PG\_STAT\_XACT\_USER\_FUNCTIONS

PG\_STAT\_XACT\_USER\_FUNCTIONS视图包含每个函数的执行的统计信息。

**表 1** PG\_STAT\_XACT\_USER\_FUNCTIONS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| funcid | oid | 函数标识。 |
| schemaname | name | 模式的名称。 |
| funcname | name | 函数名称。 |
| calls | bigint | 函数被调用的次数。 |
| total\_time | double precision | 函数的总执行时长。 |
| self\_time | double precision | 当前线程调用函数的总的时长。 |

### PG\_STAT\_XACT\_USER\_TABLES

PG\_STAT\_XACT\_USER\_TABLES视图显示命名空间中用户表的事务状态信息。

**表 1** PG\_STAT\_XACT\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表的OID。 |
| schemaname | name | 该表的模式名。 |
| relname | name | 表名。 |
| seq\_scan | bigint | 该表发起的顺序扫描数。 |
| seq\_tup\_read | bigint | 顺序扫描抓取的活跃行数。 |
| idx\_scan | bigint | 该表发起的索引扫描数。 |
| idx\_tup\_fetch | bigint | 索引扫描抓取的活跃行数。 |
| n\_tup\_ins | bigint | 插入行数。 |
| n\_tup\_upd | bigint | 更新行数。 |
| n\_tup\_del | bigint | 删除行数。 |
| n\_tup\_hot\_upd | bigint | HOT更新行数（比如没有更新所需的单独索引）。 |

### PG\_STATIO\_ALL\_INDEXES

PG\_STATIO\_ALL\_INDEXES视图将包含当前数据库中的每个索引行，显示特定索引的I/O的统计。

**表 1** PG\_STATIO\_ALL\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

### PG\_STATIO\_ALL\_SEQUENCES

PG\_STATIO\_ALL\_SEQUENCES视图包含当前数据库中每个序列的I/O的统计信息。

**表 1** PG\_STATIO\_ALL\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

### PG\_STATIO\_ALL\_TABLES

PG\_STATIO\_ALL\_TABLES视图将包含当前数据库中每个表（包括TOAST表）的I/O统计信息。

**表 1** PG\_STATIO\_ALL\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

### PG\_STATIO\_SYS\_INDEXES

PG\_STATIO\_SYS\_INDEXES视图显示命名空间中所有系统表索引的IO状态信息。

**表 1** PG\_STATIO\_SYS\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

### PG\_STATIO\_SYS\_SEQUENCES

PG\_STATIO\_SYS\_SEQUENCES视图显示命名空间中所有序列的IO状态信息。

**表 1** PG\_STATIO\_SYS\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

### PG\_STATIO\_SYS\_TABLES

PG\_STATIO\_SYS\_TABLES视图显示命名空间中所有系统表的IO状态信息。

**表 1** PG\_STATIO\_SYS\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

### PG\_STATIO\_USER\_INDEXES

PG\_STATIO\_USER\_INDEXES视图显示命名空间中所有用户关系表索引的IO状态信息。

**表 1** PG\_STATIO\_USER\_INDEXES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 索引的表的OID。 |
| indexrelid | oid | 该索引的OID。 |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | bigint | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 索引命中缓存数。 |

### PG\_STATIO\_USER\_SEQUENCES

PG\_STATIO\_USER\_SEQUENCES视图显示命名空间中所有用户关系表类型为序列的IO状态信息。

**表 1** PG\_STATIO\_USER\_SEQUENCES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 序列OID。 |
| schemaname | name | 序列中模式名。 |
| relname | name | 序列名。 |
| blks\_read | bigint | 从序列中读取的磁盘块数。 |
| blks\_hit | bigint | 序列中缓存命中数。 |

### PG\_STATIO\_USER\_TABLES

PG\_STATIO\_USER\_TABLES视图显示命名空间中所有用户关系表的IO状态信息。

**表 1** PG\_STATIO\_USER\_TABLES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 表OID。 |
| schemaname | name | 该表模式名。 |
| relname | name | 表名。 |
| heap\_blks\_read | bigint | 从该表中读取的磁盘块数。 |
| heap\_blks\_hit | bigint | 该表缓存命中数。 |
| idx\_blks\_read | bigint | 从表中所有索引读取的磁盘块数。 |
| idx\_blks\_hit | bigint | 表中所有索引命中缓存数。 |
| toast\_blks\_read | bigint | 该表的TOAST表读取的磁盘块数（如果存在）。 |
| toast\_blks\_hit | bigint | 该表的TOAST表命中缓冲区数（如果存在）。 |
| tidx\_blks\_read | bigint | 该表的TOAST表索引读取的磁盘块数（如果存在）。 |
| tidx\_blks\_hit | bigint | 该表的TOAST表索引命中缓冲区数（如果存在）。 |

### PG\_STATS

PG\_STATS视图提供对存储在pg\_statistic表里面的单列统计信息的访问。该视图记录的统计信息更新时间间隔由参数autovacuum\_naptime设置。

**表 1** PG\_STATS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | PG\_NAMESPACE.nspname | 包含表的模式名。 |
| tablename | name | PG\_CLASS.relname | 表名。 |
| attname | name | PG\_ATTRIBUTE.attname | 字段的名称。 |
| inherited | Boolean | - | 如果为真，则包含继承的子列，否则只是指定表的字段。 |
| null\_frac | real | - | 记录中字段为空的百分比。 |
| avg\_width | integer | - | 字段记录以字节记的平均宽度。 |
| n\_distinct | real | - | * 如果大于零，表示字段中独立数值的估计数目。 * 如果小于零，表示独立数值的数目被行数除的负数。  1. 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 2. 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。 |
| n\_dndistinct | real | - | 标识dn1上字段中非NULL数据的唯一值的数目。   * 如果大于零，表示独立数值的实际数目。如果小于零，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n\_dndistinct=-0.5）。 * 如果等于零，表示独立数值的数目未知。 |
| most\_common\_vals | anyarray | - | 一个字段里最常用数值的列表。如果里面的字段数值是最常见的，则为NULL。 |
| most\_common\_freqs | real[] | - | 一个最常用数值的频率的列表，也就是说，每个出现的次数除以行数。如果most\_common\_vals是NULL，则为NULL。 |
| histogram\_bounds | anyarray | - | 一个数值的列表，它把字段的数值分成几组大致相同热门的组。如果在most\_common\_vals里有数值，则在这个饼图的计算中省略。如果字段数据类型没有<操作符或者most\_common\_vals列表代表了整个分布性，则这个字段为NULL。 |
| correlation | real | - | 统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。 |
| most\_common\_elems | anyarray | - | 一个最常用的非空元素的列表。 |
| most\_common\_elem\_freqs | real[] | - | 一个最常用元素的频率的列表。 |
| elem\_count\_histogram | real[] | - | 对于独立的非空元素的统计直方图。 |

### PG\_TABLES

PG\_TABLES视图提供了对数据库中每个表访问的有用信息。

**表 1** PG\_TABLES字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | PG\_NAMESPACE.nspname | 包含表的模式名。 |
| tablename | name | PG\_CLASS.relname | 表名。 |
| tableowner | name | pg\_get\_userbyid(PG\_CLASS.relowner) | 表的所有者。 |
| tablespace | name | PG\_TABLESPACE.spcname | 包含表的表空间，默认为NULL。 |
| hasindexes | Boolean | PG\_CLASS.relhasindex | 如果表上有索引（或者最近拥有）则为TRUE，否则为FALSE。 |
| hasrules | Boolean | PG\_CLASS.relhasruls | 如果表上有规则，则为TRUE，否则为FALSE。 |
| hastriggers | Boolean | PG\_CLASS.RELHASTRIGGERS | 如果表上有触发器，则为TRUE，否则为FALSE。 |
| tablecreator | name | pg\_get\_userbyid(po.creator) | 创建表的名称。 |
| created | timestamp with time zone | pg\_object.ctime | 对象的创建时间。 |
| last\_ddl\_time | timestamp with time zone | pg\_object.mtime | 对象的最后修改时间。 |

### PG\_TDE\_INFO

PG\_TDE\_INFO视图提供了Vastbase加密信息。

**表 1** PG\_TDE\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| is\_encrypt | Boolean | 是否加密Vastbase。   * f：非加密Vastbase。 * t：加密Vastbase。 |
| g\_tde\_algo | text | 加密算法。  SM4-CTR-128。  AES-CTR-128。 |
| remain | text | 保留字段。 |

### PG\_THREAD\_WAIT\_STATUS

通过PG\_THREAD\_WAIT\_STATUS视图可以检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。

**表 1** PG\_THREAD\_WAIT\_STATUS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 当前节点的名称。 |
| db\_name | text | 数据库名称。 |
| thread\_name | text | 线程名称。 |
| query\_id | bigint | 查询ID，对应debug\_query\_id。 |
| tid | bigint | 当前线程的线程号。 |
| sessionid | bigint | 当前会话ID。 |
| lwtid | integer | 当前线程的轻量级线程号。 |
| psessionid | bigint | 父会话ID。 |
| tlevel | integer | streaming线程的层级。 |
| smpid | integer | 并行线程的ID。 |
| wait\_status | text | 当前线程的等待状态。等待状态的详细信息请参见表2。 |
| wait\_event | text | 如果wait\_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则是空。 |
| locktag | text | 当前线程正在等待锁的信息。 |
| lockmode | text | 当前线程正等待获取的锁模式。包含表级锁、行级锁、页级锁下的各模式。 |
| block\_sessionid | bigint | 阻塞当前线程获取锁的会话标识。 |
| global\_sessionid | text | 全局会话ID。 |

wait\_status列的等待状态有以下状态。

**表 2** 等待状态列表

| **wait\_status值** | **含义** |
| --- | --- |
| none | 没在等任意事件。 |
| acquire lock | 等待加锁，要么加锁成功，要么加锁等待超时。 |
| acquire lwlock | 等待获取轻量级锁。 |
| wait io | 等待IO完成。 |
| wait cmd | 等待完成读取网络通信包。 |
| wait pooler get conn | 等待pooler完成获取连接。 |
| wait pooler abort conn | 等待pooler完成终止连接。 |
| wait pooler clean conn | 等待pooler完成清理连接。 |
| pooler create conn:[nodename], total N | 等待pooler建立连接，当前正在与nodename指定节点建立连接，且仍有N个连接等待建立。 |
| get conn | 获取到其他节点的连接。 |
| set cmd: [nodename] | 在连接上执行SET/RESET/TRANSACTION BLOCK LEVEL PARA SET/SESSION LEVEL PARA SET，当前正在nodename指定节点上执行。 |
| cancel query | 取消某连接上正在执行的SQL语句。 |
| stop query | 停止某连接上正在执行的查询。 |
| wait node: [nodename](plevel), total N, [phase] | 等待接收与某节点的连接上的数据，当前正在等待nodename节点plevel线程的数据，且仍有N个连接的数据待返回。如果状态包含phase信息，则可能的阶段状态有：   * begin：表示处于事务开始阶段。 * commit：表示处于事务提交阶段。 * rollback：表示处于事务回滚阶段。 |
| wait transaction sync: xid | 等待xid指定事务同步。 |
| wait wal sync | 等待特定LSN的wal log完成到备机的同步。 |
| wait data sync | 等待完成数据页到备机的同步。 |
| wait data sync queue | 等待把行存的数据页或列存的CU放入同步队列。 |
| flush data: [nodename](plevel), [phase] | 等待向网络中nodename指定节点的plevel对应线程发送数据。如果状态包含phase信息，则可能的阶段状态为wait quota，即当前通信流正在等待quota值。 |
| stream get conn: [nodename], total N | 初始化stream flow时，等待与nodename节点的consumer对象建立连接，且当前有N个待建连对象。 |
| wait producer \ready: [nodename](plevel), total N | 初始化stream flow时，等待每个producer都准备好，当前正在等待nodename节点plevel对应线程的producer对象准备好，且仍有N个producer对象处于等待状态。 |
| synchronize quit | stream plan结束时，等待stream线程组内的线程统一退出。 |
| wait stream nodegroup destroy | stream plan结束时，等待销毁stream node group。 |
| wait active statement | 等待作业执行，正在资源负载管控中。 |
| analyze: [relname], [phase] | 当前正在对表relname执行analyze。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的analyze分析操作。 |
| vacuum: [relname], [phase] | 当前正在对表relname执行vacuum。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的vacuum清理操作。 |
| vacuum full: [relname] | 当前正在对表relname执行vacuum full清理。 |
| create index | 当前正在创建索引。 |
| HashJoin - [ build hash | write file ] | 当前是HashJoin算子，主要关注耗时的执行阶段。   * build hash：表示当前HashJoin算子正在建立哈希表。 * write file：表示当前HashJoin算子正将数据写入磁盘。 |
| HashAgg - [ build hash | write file ] | 当前是HashAgg算子，主要关注耗时的执行阶段。   * build hash：表示当前HashAgg算子正在建立哈希表。 * write file：表示当前HashAgg算子正在将数据写入磁盘。 |
| HashSetop - [build hash | write file ] | 当前是HashSetop算子，主要关注耗时的执行阶段。   * build hash：表示当前HashSetop算子正在建立哈希表。 * write file：表示当前HashSetop算子正在将数据写入磁盘。 |
| Sort | Sort - [fetch tuple | write file] | 当前是Sort算子做排序，fetch tuple表示Sort算子正在获取tuple，write file表示Sort算子正在将数据写入磁盘。 |
| Material | Material - write file | 当前是Material算子，write file表示Material算子正在将数据写入磁盘。 |
| NestLoop | 当前是NestLoop算子。 |
| wait memory | 等待内存获取。 |
| wait sync consumer next step | Stream算子等待消费者执行。 |
| wait sync producer next step | Stream算子等待生产者执行。 |

**表 3** 轻量级锁等待事件列表

| **wait\_event类型** | **类型描述** |
| --- | --- |
| ShmemIndexLock | 用于保护共享内存中的主索引哈希表。 |
| OidGenLock | 用于避免不同线程产生相同的OID。 |
| XidGenLock | 用于避免两个事务获得相同的xid。 |
| ProcArrayLock | 用于避免并发访问或修改ProcArray共享数组。 |
| SInvalReadLock | 用于避免与清理失效消息并发执行。 |
| SInvalWriteLock | 用于避免与其它写失效消息、清理失效消息并发执行。 |
| WALInsertLock | 用于避免与其它WAL插入操作并发执行。 |
| WALWriteLock | 用于避免并发WAL写盘。 |
| ControlFileLock | 用于避免pg\_control文件的读写并发、写写并发。 |
| CheckpointLock | 用于避免多个checkpoint并发执行。 |
| CLogControlLock | 用于避免并发访问或者修改Clog控制数据结构。 |
| SubtransControlLock | 用于避免并发访问或者修改子事务控制数据结构。 |
| MultiXactGenLock | 用于串行分配唯一MultiXact id。 |
| MultiXactOffsetControlLock | 用于避免对pg\_multixact/offset的写写并发和读写并发。 |
| MultiXactMemberControlLock | 用于避免对pg\_multixact/members的写写并发和读写并发。 |
| RelCacheInitLock | 用于失效消息场景对init文件进行操作时加锁。 |
| CheckpointerCommLock | 用于向checkpointer发起文件刷盘请求场景，需要串行的向请求队列插入请求结构。 |
| TwoPhaseStateLock | 用于避免并发访问或者修改两阶段信息共享数组。 |
| TablespaceCreateLock | 用于确定tablespace是否已经存在。 |
| BtreeVacuumLock | 用于防止vacuum清理B-tree中还在使用的页面。 |
| AutovacuumLock | 用于串行化访问autovacuum worker数组。 |
| AutovacuumScheduleLock | 用于串行化分配需要vacuum的table。 |
| AutoanalyzeLock | 用于获取和释放允许执行Autoanalyze的任务资源。 |
| SyncScanLock | 用于确定heap扫描时某个relfilenode的起始位置。 |
| NodeTableLock | 用于保护存放数据库节点信息的共享结构。 |
| PoolerLock | 用于保证两个线程不会同时从连接池里取到相同的连接。 |
| RelationMappingLock | 用于等待更新系统表到存储位置之间映射的文件。 |
| AsyncCtlLock | 用于避免并发访问或者修改共享通知状态。 |
| AsyncQueueLock | 用于避免并发访问或者修改共享通知信息队列。 |
| SerializableXactHashLock | 用于避免对于可串行事务共享结构的写写并发和读写并发。 |
| SerializableFinishedListLock | 用于避免对于已完成可串行事务共享链表的写写并发和读写并发。 |
| SerializablePredicateLockListLock | 用于保护对于可串行事务持有的锁链表。 |
| OldSerXidLock | 用于保护记录冲突可串行事务的结构。 |
| FileStatLock | 用于保护存储统计文件信息的数据结构。 |
| SyncRepLock | 用于在主备复制时保护xlog同步信息。 |
| DataSyncRepLock | 用于在主备复制时保护数据页同步信息。 |
| CStoreColspaceCacheLock | 用于保护列存表的CU空间分配。 |
| CStoreCUCacheSweepLock | 用于列存CU Cache循环淘汰。 |
| MetaCacheSweepLock | 用于元数据循环淘汰。 |
| ExtensionConnectorLibLock | 用于初始化ODBC连接场景，在加载与卸载特定动态库时进行加锁。 |
| SearchServerLibLock | 用于GPU加速场景初始化加载特定动态库时，对读文件操作进行加锁。 |
| LsnXlogChkFileLock | 用于串行更新特定结构中记录的主备机的xlog flush位置点。 |
| ReplicationSlotAllocationLock | 用于主备复制时保护主机端的流复制槽的分配。 |
| ReplicationSlotControlLock | 用于主备复制时避免并发更新流复制槽状态。 |
| ResourcePoolHashLock | 用于避免并发访问或者修改资源池哈希表。 |
| WorkloadStatHashLock | 用于避免并发访问或者修改包含数据库主节点的SQL请求构成的哈希表。 |
| WorkloadIoStatHashLock | 用于避免并发访问或者修改用于统计当前数据库节点的IO信息的哈希表。 |
| WorkloadCGroupHashLock | 用于避免并发访问或者修改Cgroup信息构成的哈希表。 |
| OBSGetPathLock | 用于避免对obs路径的写写并发和读写并发。 |
| WorkloadUserInfoLock | 用于避免并发访问或修改负载管理的用户信息哈希表。 |
| WorkloadRecordLock | 用于避免并发访问或修改在内存自适应管理时对数据库主节点收到请求构成的哈希表。 |
| WorkloadIOUtilLock | 用于保护记录iostat，CPU等负载信息的结构。 |
| WorkloadNodeGroupLock | 用于避免并发访问或者修改内存中的nodegroup信息构成的哈希表。 |
| JobShmemLock | 用于定时任务功能中保护定时读取的全局变量。 |
| OBSRuntimeLock | 用于获取环境变量，如GASSHOME。 |
| LLVMDumpIRLock | 用于导出动态生成函数所对应的汇编语言。 |
| LLVMParseIRLock | 用于在查询开始处从IR文件中编译并解析已写好的IR函数。 |
| CriticalCacheBuildLock | 用于从共享或者本地缓存初始化文件中加载cache的场景。 |
| WaitCountHashLock | 用于保护用户语句计数功能场景中的共享结构。 |
| BufMappingLock | 用于保护对共享缓冲映射表的操作。 |
| LockMgrLock | 用于保护常规锁结构信息。 |
| PredicateLockMgrLock | 用于保护可串行事务锁结构信息。 |
| OperatorRealTLock | 用于避免并发访问或者修改记录算子级实时数据的全局结构。 |
| OperatorHistLock | 用于避免并发访问或者修改记录算子级历史数据的全局结构。 |
| SessionRealTLock | 用于避免并发访问或者修改记录query级实时数据的全局结构。 |
| SessionHistLock | 用于避免并发访问或者修改记录query级历史数据的全局结构。 |
| CacheSlotMappingLock | 用于保护CU Cache全局信息。 |
| BarrierLock | 用于保证当前只有一个线程在创建Barrier。 |
| dummyServerInfoCacheLock | 用于保护缓存加速Vastbase连接信息的全局哈希表。 |
| RPNumberLock | 用于加速Vastbase的数据库节点对正在执行计划的任务线程的计数。 |
| CBMParseXlogLock | Cbm解析xlog时的保护锁。 |
| RelfilenodeReuseLock | 避免错误地取消已重用的列属性文件的链接。 |
| RcvWriteLock | 防止并发调用WalDataRcvWrite。 |
| PercentileLock | 用于保护全局PercentileBuffer。 |
| CSNBufMappingLock | 保护csn页面。 |
| UniqueSQLMappingLock | 用于保护uniquesql hash table。 |
| DelayDDLLock | 防止并发ddl。 |
| CLOG Ctl | 用于避免并发访问或者修改Clog控制数据结构。 |
| Async Ctl | 保护Async buffer。 |
| MultiXactOffset Ctl | 保护MultiXact offet的slru buffer。 |
| MultiXactMember Ctl | 保护MultiXact member的slrubuffer。 |
| OldSerXid SLRU Ctl | 保护old xids的slru buffer。 |
| ReplicationSlotLock | 用于保护ReplicationSlot。 |
| PGPROCLock | 用于保护pgproc。 |
| MetaCacheLock | 用于保护MetaCache。 |
| DataCacheLock | 用于保护datacache。 |
| InstrUserLock | 用于保护InstrUserHTAB。 |
| BadBlockStatHashLock | 用于保护global\_bad\_block\_stat hash表。 |
| BufFreelistLock | 用于保证共享缓冲区空闲列表操作的原子性。 |
| CUSlotListLock | 用于控制列存缓冲区槽位的并发操作。 |
| AddinShmemInitLock | 保护共享内存对象的初始化。 |
| AlterPortLock | 保护协调节点更改注册端口号的操作。 |
| FdwPartitionCaheLock | HDFS分区表缓冲区的管理锁。 |
| DfsConnectorCacheLock | DFSConnector缓冲区的管理锁。 |
| DfsSpaceCacheLock | HDFS表空间管理缓冲区的管理锁。 |
| FullBuildXlogCopyStartPtrLock | 用于保护全量Build中Xlog拷贝的操作。 |
| DfsUserLoginLock | 用于HDFS用户登录以及认证。 |
| LogicalReplicationSlotPersistentDataLock | 用于保护逻辑复制过程中复制槽位的数据。 |
| WorkloadSessionInfoLock | 保护负载管理session info内存hash表访问。 |
| InstrWorkloadLock | 保护负载管理统计信息的内存hash表访问。 |
| PgfdwLock | 用于管理实例向Foreign server建立连接。 |
| InstanceTimeLock | 用于获取实例中会话的时间信息。 |
| XlogRemoveSegLock | 保护Xlog段文件的回收操作。 |
| DnUsedSpaceHashLock | 用于更新会话对应的空间使用信息。 |
| CsnMinLock | 用于计算CSNmin。 |
| GPCCommitLock | 用于保护全局Plan Cache hash表的添加操作。 |
| GPCClearLock | 用于保护全局Plan Cache hash表的清除操作。 |
| GPCTimelineLock | 用于保护全局Plan Cache hash表检查Timeline的操作。 |
| TsTagsCacheLock | 用于时序tag缓存管理。 |
| InstanceRealTLock | 用于保护共享实例统计信息hash表的更新操作。 |
| CLogBufMappingLock | 用于提交日志缓存管理。 |
| GPCMappingLock | 用于全局Plan Cache缓存管理。 |
| GPCPrepareMappingLock | 用于全局Plan Cache缓存管理。 |
| BufferIOLock | 保护共享缓冲区页面的IO操作。 |
| BufferContentLock | 保护共享缓冲区页面内容的读取、修改。 |
| CSNLOG Ctl | 用于CSN日志管理。 |
| DoubleWriteLock | 用于双写的管理操作。 |
| RowPageReplicationLock | 用于管理行存储的数据页复制。 |
| extension | 其他轻量锁。 |

**表 4** IO等待事件列表

| **wait\_event类型** | **类型描述** |
| --- | --- |
| BufFileRead | 从临时文件中读取数据到指定buffer。 |
| BufFileWrite | 向临时文件中写入指定buffer中的内容。 |
| ControlFileRead | 读取pg\_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。 |
| ControlFileSync | 将pg\_control文件持久化到磁盘。数据库初始化时发生。 |
| ControlFileSyncUpdate | 将pg\_control文件持久化到磁盘。主要在数据库启动、执行checkpoint和主备校验过程中发生。 |
| ControlFileWrite | 写入pg\_control文件。数据库初始化时发生。 |
| ControlFileWriteUpdate | 更新pg\_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。 |
| CopyFileRead | copy文件时读取文件内容。 |
| CopyFileWrite | copy文件时写入文件内容。 |
| DataFileExtend | 扩展文件时向文件写入内容。 |
| DataFileFlush | 将表数据文件持久化到磁盘。 |
| DataFileImmediateSync | 将表数据文件立即持久化到磁盘。 |
| DataFilePrefetch | 异步读取表数据文件。 |
| DataFileRead | 同步读取表数据文件。 |
| DataFileSync | 将表数据文件的修改持久化到磁盘。 |
| DataFileTruncate | 表数据文件truncate。 |
| DataFileWrite | 向表数据文件写入内容。 |
| LockFileAddToDataDirRead | 读取“postmaster.pid”文件。 |
| LockFileAddToDataDirSync | 将“postmaster.pid”内容持久化到磁盘。 |
| LockFileAddToDataDirWrite | 将pid信息写到“postmaster.pid”文件。 |
| LockFileCreateRead | 读取LockFile文件“%s.lock”。 |
| LockFileCreateSync | 将LockFile文件“%s.lock”内容持久化到磁盘。 |
| LockFileCreateWRITE | 将pid信息写到LockFile文件“%s.lock”。 |
| RelationMapRead | 读取系统表到存储位置之间的映射文件。 |
| RelationMapSync | 将系统表到存储位置之间的映射文件持久化到磁盘。 |
| RelationMapWrite | 写入系统表到存储位置之间的映射文件。 |
| ReplicationSlotRead | 读取流复制槽文件。重新启动时发生。 |
| ReplicationSlotRestoreSync | 将流复制槽文件持久化到磁盘。重新启动时发生。 |
| ReplicationSlotSync | checkpoint时将流复制槽临时文件持久化到磁盘。 |
| ReplicationSlotWrite | checkpoint时写流复制槽临时文件。 |
| SLRUFlushSync | 将pg\_clog、pg\_subtrans和pg\_multixact文件持久化到磁盘。主要在执行checkpoint和数据库停机时发生。 |
| SLRURead | 读取pg\_clog、pg\_subtrans和pg\_multixact文件。 |
| SLRUSync | 将脏页写入文件pg\_clog、pg\_subtrans和pg\_multixact并持久化到磁盘。主要在执行checkpoint和数据库停机时发生。 |
| SLRUWrite | 写入pg\_clog、pg\_subtrans和pg\_multixact文件。 |
| TimelineHistoryRead | 读取timeline history文件。在数据库启动时发生。 |
| TimelineHistorySync | 将timeline history文件持久化到磁盘。在数据库启动时发生。 |
| TimelineHistoryWrite | 写入timeline history文件。在数据库启动时发生。 |
| TwophaseFileRead | 读取pg\_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。 |
| TwophaseFileSync | 将pg\_twophase文件持久化到磁盘。在两阶段事务提交、两阶段事务恢复时发生。 |
| TwophaseFileWrite | 写入pg\_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。 |
| WALBootstrapSync | 将初始化的WAL文件持久化到磁盘。在数据库初始化发生。 |
| WALBootstrapWrite | 写入初始化的WAL文件。在数据库初始化发生。 |
| WALCopyRead | 读取已存在的WAL文件并进行复制时产生的读操作。在执行归档恢复完后发生。 |
| WALCopySync | 将复制的WAL文件持久化到磁盘。在执行归档恢复完后发生。 |
| WALCopyWrite | 读取已存在WAL文件并进行复制时产生的写操作。在执行归档恢复完后发生。 |
| WALInitSync | 将新初始化的WAL文件持久化磁盘。在日志回收或写日志时发生。 |
| WALInitWrite | 将新创建的WAL文件初始化为0。在日志回收或写日志时发生。 |
| WALRead | 从xlog日志读取数据。两阶段文件redo相关的操作产生。 |
| WALSyncMethodAssign | 将当前打开的所有WAL文件持久化到磁盘。 |
| WALWrite | 写入WAL文件。 |
| WALBufferAccess | WAL Buffer访问（出于性能考虑，内核代码里只统计访问次数，未统计其访问耗时）。 |
| WALBufferFull | WAL Buffer满时，写wal文件相关的处理。 |
| DoubleWriteFileRead | 双写 文件读取。 |
| DoubleWriteFileSync | 双写 文件强制刷盘。 |
| DoubleWriteFileWrite | 双写 文件写入。 |
| PredoProcessPending | 并行日志回放中当前记录回放等待其它记录回放完成。 |
| PredoApply | 并行日志回放中等待当前工作线程等待其他线程回放至本线程LSN。 |
| DisableConnectFileRead | HA锁分片逻辑文件读取。 |
| DisableConnectFileSync | HA锁分片逻辑文件强制刷盘。 |
| DisableConnectFileWrite | HA锁分片逻辑文件写入。 |

当wait\_status值为acquire lock（事务锁）时对应的wait\_event等待事件类型与描述信息如下。

**表 5** 事务锁等待事件列表

| **wait\_event类型** | **类型描述** |
| --- | --- |
| relation | 对表加锁。 |
| extend | 对表扩展空间时加锁。 |
| partition | 对分区表加锁。 |
| partition\_seq | 对分区表的分区加锁。 |
| page | 对表页面加锁。 |
| tuple | 对页面上的tuple加锁。 |
| transactionid | 对事务ID加锁。 |
| virtualxid | 对虚拟事务ID加锁。 |
| object | 加对象锁。 |
| cstore\_freespace | 对列存空闲空间加锁。 |
| userlock | 加用户锁。 |
| advisory | 加advisory锁。 |

### PG\_TIMEZONE\_ABBREVS

PG\_TIMEZONE\_ABBREVS视图提供了显示了所有可用的时区信息。

**表 1** PG\_TIMEZONE\_ABBREVS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| abbrev | text | 时区名缩写。 |
| utc\_offset | interval | 相对于UTC的偏移量。 |
| is\_dst | Boolean | 如果当前正处于夏令时范围则为TRUE，否则为FALSE。 |

### PG\_TIMEZONE\_NAMES

PG\_TIMEZONE\_NAMES视图提供了显示了所有能够被SET TIMEZONE识别的时区名及其缩写、UTC偏移量、是否夏时制。

**表 1** PG\_TIMEZONE\_NAMES字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| abbrev | text | 时区名缩写。 |
| utc\_offset | interval | 相对于UTC的偏移量。 |
| is\_dst | Boolean | 如果当前正处于夏令时范围则为TRUE，否则为FALSE。 |

### PG\_TOTAL\_MEMORY\_DETAIL

PG\_TOTAL\_MEMORY\_DETAIL视图显示某个数据库节点内存使用情况。

**表 1** PG\_TOTAL\_MEMORY\_DETAIL字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| nodename | text | 节点名称。 |
| memorytype | text | 内存的名称。 |
| memorymbytes | integer | 内存使用的大小，单位为MB。 |

### PG\_TOTAL\_USER\_RESOURCE\_INFO

PG\_TOTAL\_USER\_RESOURCE\_INFO视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在参数use\_workload\_manager（参考：GUC参数说明->负载管理章节）为on时才有效。其中，IO相关监控项在参数enable\_logical\_io\_statistics为on时才有效。

**表 1** PG\_TOTAL\_USER\_RESOURCE\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| username | name | 用户名。 |
| used\_memory | integer | 正在使用的内存大小，单位MB。 |
| total\_memory | integer | 可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 |
| used\_cpu | double precision | 正在使用的CPU核数（仅统计复杂作业CPU使用 情况，且该值为相关控制组的CPU使用统计值）。 |
| total\_cpu | integer | 在该机器节点上，用户关联控制组的CPU核数总和。 |
| used\_space | bigint | 已使用的永久表存储空间大小，单位KB。 |
| total\_space | bigint | 可使用的永久表存储空间大小，单位KB，值为-1表示未限制最大存储空间。 |
| used\_temp\_space | bigint | 已使用的临时空间大小，单位KB。 |
| total\_temp\_space | bigint | 可使用的临时空间总大小，单位KB，值为-1表示未限制。 |
| used\_spill\_space | bigint | 已使用的算子落盘空间大小，单位KB。 |
| total\_spill\_space | bigint | 可使用的算子落盘空间总大小，单位KB，值为-1表示未限制。 |
| read\_kbytes | bigint | 数据库主节点：过去5秒内，该用户在数据库节点上复杂作业read的字节总数（单位KB）。  数据库节点：实例启动至当前时间为止，该用户复杂作业read的字节总数（单位KB）。 |
| write\_kbytes | bigint | 数据库主节点：过去5秒内，该用户在数据库节点上复杂作业write的字节总数（单位KB）。  数据库节点：实例启动至当前时间为止，该用户复杂作业write的字节总数（单位KB）。 |
| read\_counts | bigint | 数据库主节点：过去5秒内，该用户在数据库节点上复杂作业read的次数之和（单位次）。  数据库节点：实例启动至当前时间为止，该用户复杂作业read的次数之和（单位次）。 |
| write\_counts | bigint | 数据库主节点：过去5秒内，该用户在数据库节点上复杂作业write的次数之和（单位次）。  数据库节点：实例启动至当前时间为止，该用户复杂作业write的次数之和（单位次）。 |
| read\_speed | double precision | 数据库主节点：过去5秒内，该用户在单个数据库节点上复杂作业read平均速率（单位KB/s）。  数据库节点：过去5秒内，该用户在该数据库节点上复杂作业read平均速率（单位KB/s）。 |
| write\_speed | double precision | 数据库主节点：过去5秒内，该用户在单个数据库节点上复杂作业write平均速率（单位KB/s）。  数据库节点：过去5秒内，该用户在该数据库节点上复杂作业write平均速率（单位KB/s）。 |

### PG\_TOTAL\_USER\_RESOURCE\_INFO\_OID

PG\_TOTAL\_USER\_RESOURCE\_INFO\_OID视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在参数use\_workload\_manager（参考：GUC参数说明->负载管理章节）为on时才有效。

**表 1** PG\_TOTAL\_USER\_RESOURCE\_INFO\_OID字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| userid | oid | 用户ID。 |
| used\_memory | integer | 正在使用的内存大小，单位MB。 |
| total\_memory | integer | 可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。 |
| used\_cpu | double precision | 正在使用的CPU核数。 |
| total\_cpu | integer | 在该机器节点上，用户关联控制组的CPU核数总和。 |
| used\_space | bigint | 已使用的存储空间大小，单位KB。 |
| total\_space | bigint | 可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。 |
| used\_temp\_space | bigint | 已使用的临时空间大小，单位KB。 |
| total\_temp\_space | bigint | 可使用的临时空间总大小，单位KB，值为-1表示未限制。 |
| used\_spill\_space | bigint | 已使用的下盘空间大小。单位KB。 |
| total\_spill\_space | bigint | 可使用的下盘空间总大小，单位KB，值为-1表示未限制。 |
| read\_kbytes | bigint | 读磁盘数据量，单位KB。 |
| write\_kbytes | bigint | 写磁盘数据量，单位KB。 |
| read\_counts | bigint | 读磁盘次数。 |
| write\_counts | bigint | 写磁盘次数。 |
| read\_speed | double precision | 读磁盘速率，单位B/ms。 |
| write\_speed | double precision | 写磁盘速率，单位B/ms。 |

### PG\_USER

PG\_USER视图提供了访问数据库用户的信息，默认只有初始化用户和具有sysadmin属性的用户可以查看，其余用户需要赋权后才可以查看。

**表 1** PG \_USER字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| usename | name | 用户名。 |
| usesysid | oid | 此用户的ID。 |
| usecreatedb | boolean | 用户是否可以创建数据库。 |
| usesuper | boolean | 用户是否是拥有最高权限的初始系统管理员。 |
| usecatupd | boolean | 用户是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。 |
| userepl | boolean | 用户是否可以复制数据流。 |
| passwd | text | 密文存储后的用户口令，始终为\*\*\*\*\*\*\*\*。 |
| valbegin | timestamp with time zone | 帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。 |
| valuntil | timestamp with time zone | 帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。 |
| respool | name | 用户所在的资源池。 |
| parent | oid | 父用户OID。 |
| spacelimit | text | 永久表存储空间限额。 |
| tempspacelimit | text | 临时表存储空间限额。 |
| spillspacelimit | text | 算子落盘空间限额。 |
| useconfig | text[] | 运行时配置参数的会话缺省。 |
| nodegroup | name | 用户关联的逻辑Vastbase名称，如果该用户没有管理逻辑Vastbase，则该字段为空。 |
| usemonitoradmin | boolean | 用户是否是监控管理员。  t（true）：表示是。  f（false）：表示否。 |
| useoperatoradmin | boolean | 用户是否是运维管理员。  t（true）：表示是。  f（false）：表示否。 |
| usepolicyadmin | boolean | 用户是否是安全策略管理员。  t（true）：表示是。  f（false）：表示否。 |

### \_PG\_USER\_MAPPINGS

存储从本地用户到远程的映射。该视图只有sysadmin权限可以查看。

**表 1** \_PG\_USER\_MAPPINGS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| umid | oid | [PG\_USER\_MAPPING](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_USER_MAPPING.html).oid | 用户映射的OID。 |
| srvid | oid | [PG\_FOREIGN\_SERVER](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_FOREIGN_SERVER.html).oid | 包含这个映射的外部服务器的OID。 |
| srvname | name | [PG\_FOREIGN\_SERVER](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_FOREIGN_SERVER.html).srvname | 外部服务器的名称。 |
| umuser | oid | [PG\_AUTHID](https://www.opengauss.org/zh/docs/3.0.0/docs/Developerguide/PG_AUTHID.html).oid | 被映射的本地角色的OID，如果用户映射是公共的则为0。 |
| usename | name | - | 被映射的本地用户的名称。 |
| umoptions | text[ ] | - | 如果当前用户是外部服务器的所有者，则为用户映射指定选项，使用“keyword=value”字符串，否则为null。 |

### PG\_VARIABLE\_INFO

PG\_VARIABLE\_INFO视图用于查询Vastbase中当前节点的xid、oid的状态。

**表 1** PG\_VARIABLE\_INFO字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| node\_name | text | 节点名称。 |
| next\_oid | oid | 该节点下一次生成的oid。 |
| next\_xid | xid | 该节点下一次生成的事务号。 |
| oldest\_xid | xid | 该节点最老的事务号。 |
| xid\_vac\_limit | xid | 强制autovacuum的临界点。 |
| oldest\_xid\_db | oid | 该节点datafrozenxid最小的数据库oid。 |
| last\_extend\_csn\_logpage | xid | 最后一次扩展csnlog的页面号。 |
| start\_extend\_csn\_logpage | xid | csnlog扩展的起始页面号。 |
| next\_commit\_seqno | xid | 该节点下次生成的csn号。 |
| latest\_completed\_xid | xid | 该节点提交或者回滚后节点上的最新事务号。 |
| startup\_max\_xid | xid | 该节点关机前的最后一个事务号。 |

### PG\_VIEWS

PG\_VIEWS视图提供访问数据库中每个视图的有用信息。

**表 1** PG\_VIEWS字段

| **名称** | **类型** | **引用** | **描述** |
| --- | --- | --- | --- |
| schemaname | name | PG\_NAMESPACE.nspname | 包含视图的模式名。 |
| viewname | name | PG\_CLASS.relname | 视图名。 |
| viewowner | name | PG\_AUTHID.Erolname | 视图的所有者。 |
| definition | text | - | 视图的定义。 |

### PG\_WLM\_STATISTICS

PG\_WLM\_STATISTICS视图显示作业结束后或已被处理异常后的负载管理相关信息。查询该视图需要sysadmin权限。

**表 1** PG\_WLM\_STATISTICS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| statement | text | 执行了异常处理的语句。 |
| block\_time | bigint | 语句执行前的阻塞时间。 |
| elapsed\_time | bigint | 语句的实际执行时间。 |
| total\_cpu\_time | bigint | 语句执行异常处理时数据库实例上CPU使用的总时间。 |
| qualification\_time | bigint | 语句检查倾斜率的时间周期。 |
| cpu\_skew\_percent | integer | 语句在执行异常处理时数据库实例上CPU使用的倾斜率。 |
| control\_group | text | 语句执行异常处理时所使用的Cgroups。 |
| status | text | 语句执行异常处理后的状态，包括： pending：执行前预备状态。 Running：执行进行状态。 Finished：执行正常结束。 Abort：执行异常终止。 |
| action | text | 语句执行的异常处理动作，包括：abort：执行终止操作。adjust：执行Cgroups调整操作，目前只有降级操作。finish：正常结束。 |

### PGXC\_PREPARED\_XACTS

PGXC\_PREPARED\_XACTS视图显示当前处于prepared阶段的两阶段事务。只有system admin和monitor admin用户有权限查看。

**表 1** PGXC\_PREPARED\_XACTS字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| pgxc\_prepared\_xact | text | 查看当前处于prepared阶段的两阶段事务。 |

### PLAN\_TABLE

PLAN\_TABLE显示用户通过执行EXPLAIN PLAN收集到的计划信息。计划信息的生命周期是session级别，session退出后相应的数据将被清除。同时不同session和不同user间的数据是相互隔离的。

**表 1** PLAN\_TABLE字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| statement\_id | varchar2(30) | 用户输入的查询标签。 |
| plan\_id | bigint | 查询标识。 |
| id | int | 查询生成的计划中的每一个执行算子的编号。 |
| operation | varchar2(30) | 计划中算子的操作描述。 |
| options | varchar2(255) | 操作选项。 |
| object\_name | name | 操作对应的对象名，非查询中使用到的对象别名。来自于用户定义。 |
| object\_type | varchar2(30) | 对象类型。 |
| object\_owner | name | 对象所属schema，来自于用户定义。 |
| projection | varchar2(4000) | 操作输出的列信息。 |

fig: **说明：**

* object\_type取值范围为PG\_CLASS（参考：系统表和系统视图->系统表->PG\_CLASS章节）中定义的relkind类型（TABLE普通表，INDEX索引，SEQUENCE序列，VIEW视图，COMPOSITE TYPE复合类型，TOASTVALUE TOAST表）和计划使用到的rtekind(SUBQUERY,JOIN,FUNCTION,VALUES,CTE,REMOTE\_QUERY)。
* object\_owner对于RTE来说是计划中使用的对象描述，非用户定义的类型不存在object\_owner。
* statement\_id、object\_name、object\_owner、projection字段内容遵循用户定义的大小写存储，其它字段内容采用大写存储。
* 支持用户对PLAN\_TABLE进行SELECT和DELETE操作，不支持其它DML操作。

### STATEMENT\_HISTORY

获得当前节点的执行语句的信息。查询视图必须具有sysadmin权限或者monitor admin权限。只可在系统库中查询到结果，用户库中无法查询。

**表 1** STATEMENT\_HISTORY字段

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| dbname | name | 数据库名称。 |
| schemaname | name | schema名称。 |
| origin\_node | integer | 节点名称。 |
| user\_name | name | 用户名。 |
| application\_name | text | 用户发起的请求的应用程序名称。 |
| client\_addr | text | 用户发起的请求的客户端地址。 |
| client\_port | integer | 用户发起的请求的客户端端口。 |
| unique\_query\_id | bigint | 归一化SQL ID。 |
| debug\_query\_id | bigint | 唯一SQL ID。 |
| query | text | 归一化SQL。 |
| start\_time | timestamp with time zone | 语句启动的时间。 |
| finish\_time | timestamp with time zone | 语句结束的时间。 |
| slow\_sql\_threshold | bigint | 语句执行时慢SQL的标准。 |
| transaction\_id | bigint | 事务ID。 |
| thread\_id | bigint | 执行线程ID。 |
| session\_id | bigint | 用户session id。 |
| n\_soft\_parse | bigint | 软解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 |
| n\_hard\_parse | bigint | 硬解析次数，n\_soft\_parse + n\_hard\_parse可能大于n\_calls，因为子查询未计入n\_calls。 |
| query\_plan | text | 语句执行计划。 |
| n\_returned\_rows | bigint | SELECT返回的结果集行数。 |
| n\_tuples\_fetched | bigint | 随机扫描行。 |
| n\_tuples\_returned | bigint | 顺序扫描行。 |
| n\_tuples\_inserted | bigint | 插入行。 |
| n\_tuples\_updated | bigint | 更新行。 |
| n\_tuples\_deleted | bigint | 删除行。 |
| n\_blocks\_fetched | bigint | buffer的块访问次数。 |
| n\_blocks\_hit | bigint | buffer的块命中次数。 |
| db\_time | bigint | 有效的DB时间花费，多线程将累加（单位：微秒）。 |
| cpu\_time | bigint | CPU时间（单位：微秒）。 |
| execution\_time | bigint | 执行器内执行时间（单位：微秒）。 |
| parse\_time | bigint | SQL解析时间（单位：微秒）。 |
| plan\_time | bigint | SQL生成计划时间（单位：微秒）。 |
| rewrite\_time | bigint | SQL重写时间（单位：微秒）。 |
| pl\_execution\_time | bigint | plpgsql上的执行时间（单位：微秒）。 |
| pl\_compilation\_time | bigint | plpgsql上的编译时间（单位：微秒）。 |
| data\_io\_time | bigint | IO上的时间花费（单位：微秒）。 |
| net\_send\_info | text | 通过物理连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_recv\_info | text | 通过物理连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_send\_info | text | 通过逻辑连接发送消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| net\_stream\_recv\_info | text | 通过逻辑连接接收消息的网络状态，包含时间（微秒）、调用次数、吞吐量（字节）。通过该字段可以分析SQL在分布式系统下的网络开销，单机模式下不支持该字段。  例如：{“time”:xxx, “n\_calls”:xxx, “size”:xxx}。 |
| lock\_count | bigint | 加锁次数。 |
| lock\_time | bigint | 加锁耗时。 |
| lock\_wait\_count | bigint | 加锁等待次数。 |
| lock\_wait\_time | bigint | 加锁等待耗时。 |
| lock\_max\_count | bigint | 最大持锁数量。 |
| lwlock\_count | bigint | 轻量级加锁次数（预留）。 |
| lwlock\_wait\_count | bigint | 轻量级等锁次数。 |
| lwlock\_time | bigint | 轻量级加锁时间（预留）。 |
| lwlock\_wait\_time | bigint | 轻量级加锁时间。 |
| details | bytea | 语句锁事件的列表，该列表按时间书序记录事件，记录的数量受参数track\_stmt\_details\_size的影响。 事件包括： 加锁开始、加锁结束、等锁开始、等锁结束、放锁开始、放锁结束、轻量级等锁开始、轻量级等锁结束 |
| is\_slow\_sql | boolean | 该SQL是否为slow SQL |
| trace\_id | text | 驱动传入的trace id，与应用的一次请求相关联。 |

### SUMMARY\_STATIO\_ALL\_INDEXES

SUMMARY\_STATIO\_ALL\_INDEXES视图包含含Vastbase内汇聚的数据库中的每个索引行， 显示特定索引的I/O的统计。

| **名称** | **类型** | **描述** |
| --- | --- | --- |
| relid | oid | 该索引的表ID。 |
| indexrelid | oid | 索引ID |
| schemaname | name | 该索引的模式名。 |
| relname | name | 该索引的表名。 |
| indexrelname | name | 索引名称。 |
| idx\_blks\_read | numeric | 从索引中读取的磁盘块数。 |
| idx\_blks\_hit | numeric | 索引命中缓存数。 |

# 物化视图

物化视图是一种特殊的物理表，物化视图是相对普通视图而言的。普通视图是虚拟表，应用的局限性较大，任何对视图的查询实际上都是转换为对SQL语句的查询，性能并没有实际上提高。物化视图实际上就是存储SQL执行语句的结果，起到缓存的效果。

目前Ustore引擎不支持创建、使用物化视图。

## 全量物化视图

### 概述

全量物化视图仅支持对已创建的物化视图进行全量更新，而不支持进行增量更新。创建全量物化视图语法和CREATE TABLE AS语法类似。

### 使用

**语法格式**

* 创建全量物化视图

CREATE MATERIALIZED VIEW [ view\_name ] AS { query\_block };

* 全量刷新物化视图

REFRESH MATERIALIZED VIEW [ view\_name ];

* 删除物化视图

DROP MATERIALIZED VIEW [ view\_name ];

* 查询物化视图

SELECT \* FROM [ view\_name ];

**示例**

--准备数据。  
vastbase=# CREATE TABLE t1(c1 int, c2 int);  
vastbase=# INSERT INTO t1 VALUES(1, 1);  
vastbase=# INSERT INTO t1 VALUES(2, 2);  
  
--创建全量物化视图。  
vastbase=# CREATE MATERIALIZED VIEW mv AS select count(\*) from t1;  
CREATE MATERIALIZED VIEW  
  
--查询物化视图结果。  
vastbase=# SELECT \* FROM mv;  
 count   
-------  
 2  
(1 row)  
  
--向物化视图中基表插入数据。  
vastbase=# INSERT INTO t1 VALUES(3, 3);  
  
--对全量物化视图做全量刷新。  
vastbase=# REFRESH MATERIALIZED VIEW mv;  
REFRESH MATERIALIZED VIEW  
  
--查询物化视图结果。  
vastbase=# SELECT \* FROM mv;  
 count   
-------  
 3  
(1 row)  
  
--删除物化视图。  
vastbase=# DROP MATERIALIZED VIEW mv;  
DROP MATERIALIZED VIEW

### 支持和约束

**支持场景**

* 通常全量物化视图所支持的查询范围与CREATE TABLE AS语句一致。
* 全量物化视图上支持创建索引。
* 支持analyze、explain。

**不支持场景**

物化视图不支持增删改操作，只支持查询语句。

**约束**

全量物化视图的刷新、删除过程中会给基表加高级别锁，若物化视图的定义涉及多张表，需要注意业务逻辑，避免死锁产生。

## 增量物化视图

### 概述

增量物化视图可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据刷新。与全量创建物化视图的不同在于目前增量物化视图所支持场景较小。目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

### 使用

**语法格式**

* 创建增量物化视图

CREATE INCREMENTAL MATERIALIZED VIEW [ view\_name ] AS { query\_block };

* 全量刷新物化视图

REFRESH MATERIALIZED VIEW [ view\_name ];

* 增量刷新物化视图

REFRESH INCREMENTAL MATERIALIZED VIEW [ view\_name ];

* 删除物化视图

DROP MATERIALIZED VIEW [ view\_name ];

* 查询物化视图

SELECT \* FROM [ view\_name ];

**示例**

--准备数据。  
vastbase=# CREATE TABLE t1(c1 int, c2 int);  
vastbase=# INSERT INTO t1 VALUES(1, 1);  
vastbase=# INSERT INTO t1 VALUES(2, 2);  
  
--创建增量物化视图。  
vastbase=# CREATE INCREMENTAL MATERIALIZED VIEW mv AS SELECT \* FROM t1;  
CREATE MATERIALIZED VIEW  
  
--插入数据。  
vastbase=# INSERT INTO t1 VALUES(3, 3);  
INSERT 0 1  
  
--增量刷新物化视图。  
vastbase=# REFRESH INCREMENTAL MATERIALIZED VIEW mv;  
REFRESH MATERIALIZED VIEW  
  
--查询物化视图结果。  
vastbase=# SELECT \* FROM mv;  
 c1 | c2   
----+----  
 1 | 1  
 2 | 2  
 3 | 3  
(3 rows)  
  
--插入数据。  
vastbase=# INSERT INTO t1 VALUES(4, 4);  
INSERT 0 1  
  
--全量刷新物化视图。  
vastbase=# REFRESH MATERIALIZED VIEW mv;  
REFRESH MATERIALIZED VIEW  
  
--查询物化视图结果。  
vastbase=# select \* from mv;  
 c1 | c2   
----+----  
 1 | 1  
 2 | 2  
 3 | 3  
 4 | 4  
(4 rows)  
  
--删除物化视图。  
vastbase=# DROP MATERIALIZED VIEW mv;  
DROP MATERIALIZED VIEW

### 支持和约束

**支持场景**

* 单表查询语句。
* 多个单表查询的UNION ALL。
* 物化视图上支持创建索引。
* 物化视图支持Analyze操作。

**不支持场景**

* 物化视图中不支持多表Join连接计划以及subquery计划。
* 除少部分ALTER操作外，不支持对物化视图中基表执行绝大多数DDL操作。
* 物化视图不支持增删改操作，只支持查询语句。
* 不支持用临时表/hashbucket/unlog/分区表创建物化视图。
* 不支持物化视图嵌套创建（即物化视图上创建物化视图）。
* 仅支持行存表，不支持列存表。
* 不支持UNLOGGED类型的物化视图，不支持WITH语法。

**约束**

* 物化视图定义如果为UNION ALL，则其中每个子查询需使用不同的基表。
* 增量物化视图的创建、全量刷新、删除过程中会给基表加高级别锁，若物化视图的定义为UNION ALL，需要注意业务逻辑，避免死锁产生。

# PL/SQL

## 存储过程

### 动态语句

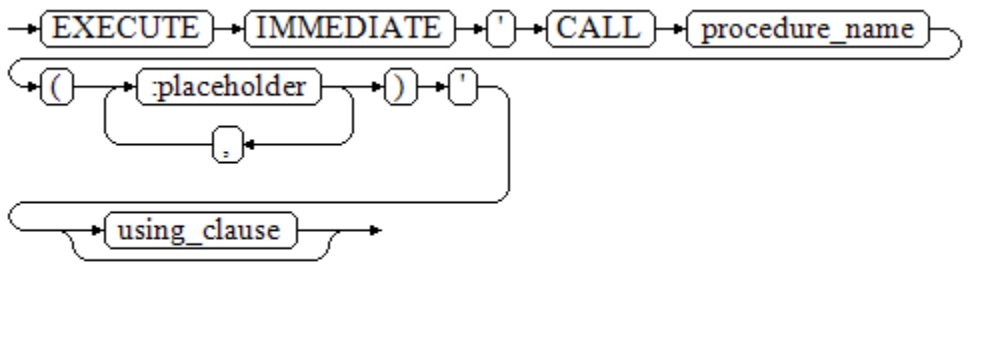
#### 动态调用存储过程

**功能描述**

动态调用存储过程就是在程序运行的时候调用存储过程。使用EXECUTE IMMEDIATE…CALL语句执行。

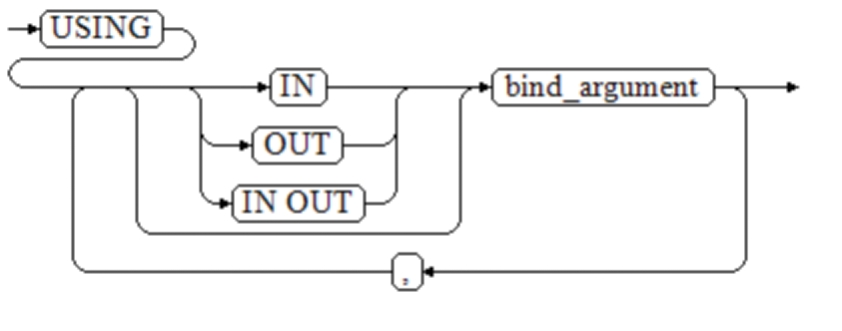
**语法格式**

语法请参见下图。

call\_procedure::=

using\_clause子句的语法参见下图。

using\_clause::=



**参数说明**

* CALL procedure\_name: 调用存储过程。
* [:placeholder1，:placeholder2，…] ：存储过程参数占位符列表。占位符个数与参数个数相同。
* USING [IN|OUT|IN OUT] bind\_argument: 用于指定存放传递给存储过程参数值的变量。bind\_argument前的修饰符与对应参数的修饰符一致。

**注意事项**

动态调用存储过程必须使用匿名的语句块将存储过程或语句块包在里面，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

#### 动态调用匿名块

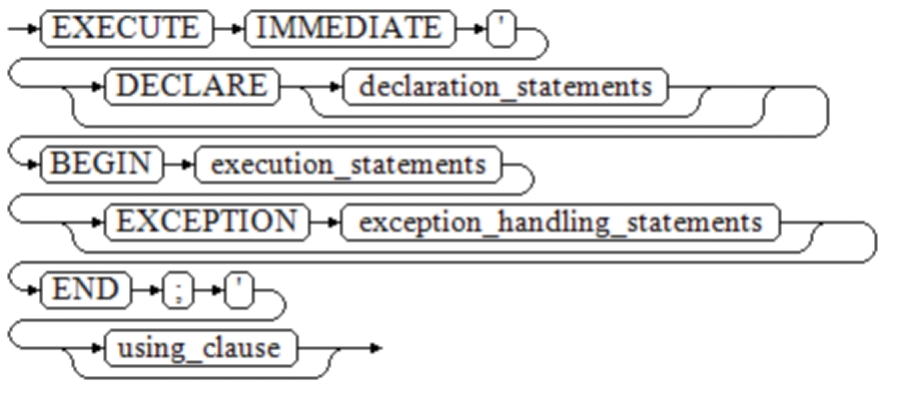
**功能描述**

动态调用匿名块是指在动态语句中执行匿名块，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

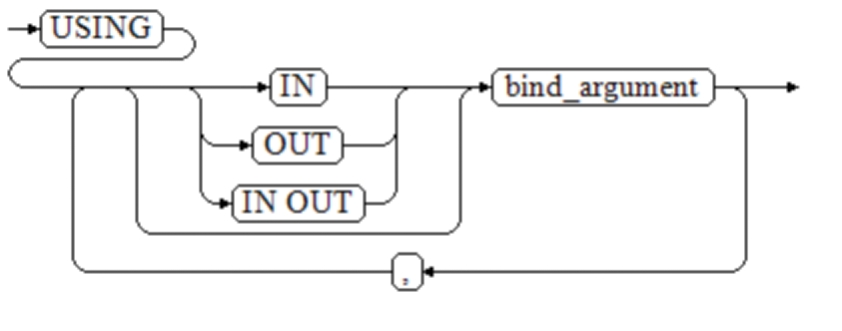
**语法格式**

语法请参见下图。

call\_anonymous\_block::=



using\_clause子句的语法参见下图。

using\_clause::=

对以上语法格式的解释如下：

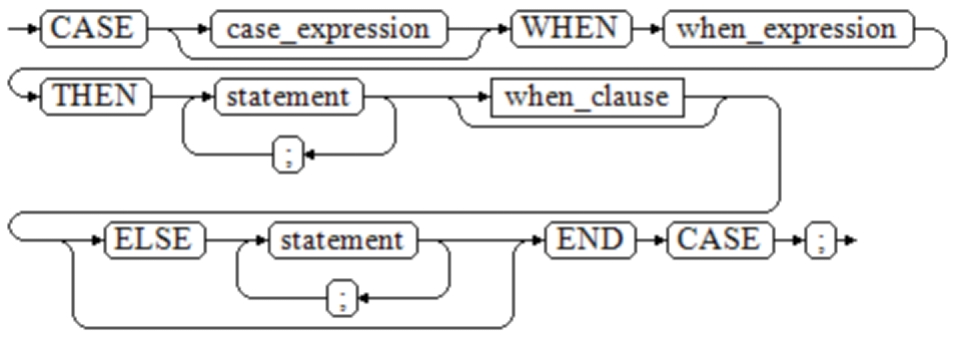
* 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。
* USING [IN|OUT|IN OUT] bind\_argument，用于指定存放传递给存储过程参数值的变量。bind\_argument前的修饰符与对应参数的修饰符一致。
* 匿名块中间的输入输出参数使用占位符来指明，要求占位符个数与参数个数相同，并且占位符所对应参数的顺序和USING中参数的顺序一致。
* 目前Vastbase在动态语句调用匿名块时，EXCEPTION语句中暂不支持使用占位符进行输入输出参数的传递。

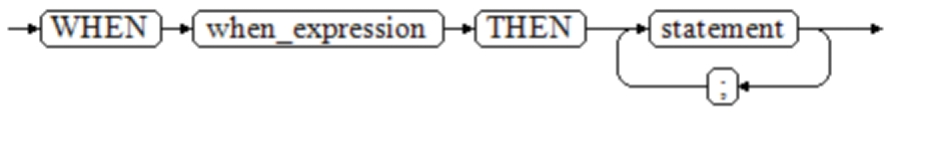
#### 分支语句

**语法格式**

分支语句的语法请参见下图。

case\_when::=



 when\_clause子句的语法图参见下图。

when\_clause::=

**参数说明**

* case\_expression：变量或表达式。
* when\_expression：常量或者条件表达式。
* statement：执行语句。

**示例**

1、创建并调用存储过程。

CREATE OR REPLACE PROCEDURE proc\_case\_branch(pi\_result in integer, pi\_return out integer)   
AS   
 BEGIN   
 CASE pi\_result   
 WHEN 1 THEN   
 pi\_return := 111;   
 WHEN 2 THEN   
 pi\_return := 222;   
 WHEN 3 THEN   
 pi\_return := 333;   
 WHEN 6 THEN   
 pi\_return := 444;   
 WHEN 7 THEN   
 pi\_return := 555;   
 WHEN 8 THEN   
 pi\_return := 666;   
 WHEN 9 THEN   
 pi\_return := 777;   
 WHEN 10 THEN   
 pi\_return := 888;   
 ELSE   
 pi\_return := 999;   
 END CASE;   
 raise info 'pi\_return : %',pi\_return ;   
END;   
/   
  
CALL proc\_case\_branch(3,0);

当结果显示如下信息，则表示调用成功。

pi\_return   
-------------  
 333  
(1 row)

2、删除存储过程

DROP PROCEDURE proc\_case\_branch;

#### 执行动态查询语句

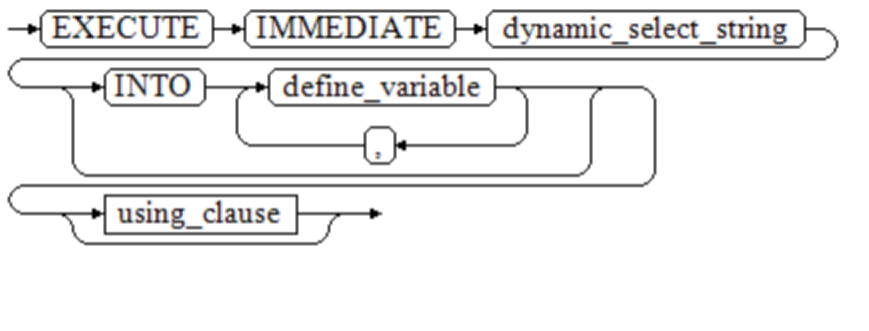
**功能描述**

动态查询就是在程序运行的时候来进行查询。Vastbase提供两种方式：使用EXECUTE IMMEDIATE、OPEN FOR实现动态查询。前者通过动态执行SELECT语句，后者结合了游标的使用。当需要将查询的结果保存在一个数据集用于提取时，可使用OPEN FOR实现动态查询。

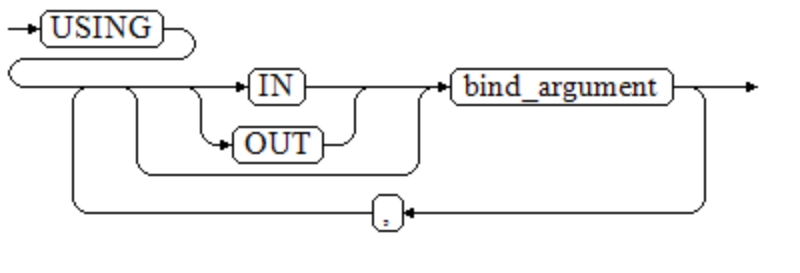
**EXECUTE IMMEDIATE**

语法图请参见下图。

EXECUTE IMMEDIATE dynamic\_select\_clause::=



using\_clause子句的语法图参见下图。

using\_clause::=

对以上语法格式的解释如下：

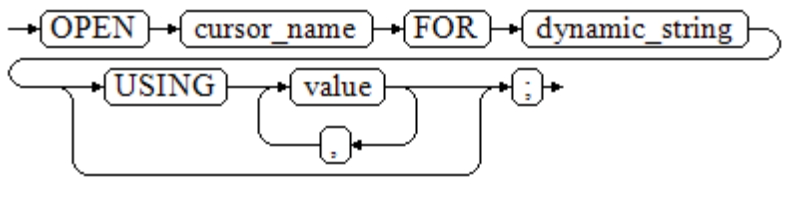
* define\_variable：用于指定存放单行查询结果的变量。
* USING IN bind\_argument：用于指定存放传递给动态SQL值的变量，即在dynamic\_select\_string中存在占位符时使用。
* USING OUT bind\_argument：用于指定存放动态SQL返回值的变量。
* 查询语句中，into和out不能同时存在；
* 占位符命名以“:”开始，后面可跟数字、字符或字符串，与USING子句的bind\_argument一一对应；
* bind\_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象，即不支持使用bind\_argument为动态SQL语句传递模式对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic\_select\_clause；
* 动态PL/SQL块允许出现重复的占位符，即相同占位符只能与USING子句的一个bind\_argument按位置对应。

**OPEN FOR**

动态查询语句还可以使用OPEN FOR打开动态游标来执行。

语法参见下图。

open\_for::=



参数说明：

* cursor\_name：要打开的游标名。
* dynamic\_string：动态查询语句。
* USING value：在dynamic\_string中存在占位符时使用。

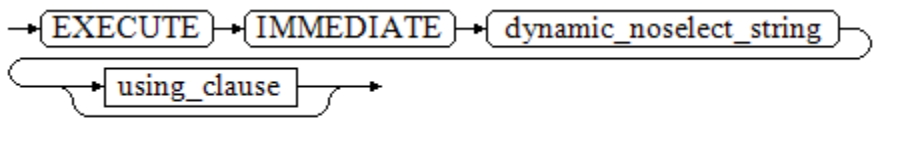
游标的使用请参考：PL/SQL->游标章节。

#### 执行动态非查询语句

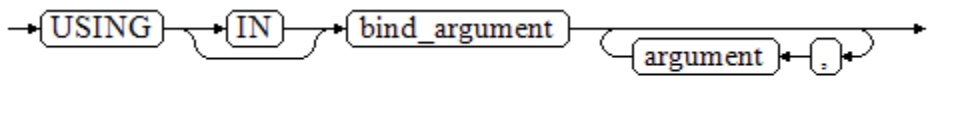
**功能描述**

动态非查询就是在程序运行的时候来进行非查询操作。

**语法格式**

语法格式请参见下图。

noselect::=

using\_clause子句的语法参见图下图。

using\_clause::=

**参数说明**

USING IN bind\_argument：用于指定存放传递给动态SQL值的变量，在dynamic\_noselect\_string中存在占位符时使用，即动态SQL语句执行时，bind\_argument将替换相对应的占位符。

bind\_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic\_select\_clause。另外，动态语句允许出现重复的占位符，相同占位符只能与唯一一个bind\_argument按位置一一对应。

**示例**

1、创建测试表。

CREATE TABLE sections\_t1   
(   
 section NUMBER(4) ,   
 section\_name VARCHAR2(30),   
 manager\_id NUMBER(6),   
 place\_id NUMBER(4)   
);

2、声明变量。

DECLARE   
 section NUMBER(4) := 280;   
 section\_name VARCHAR2(30) := 'Info support';   
 manager\_id NUMBER(6) := 103;   
 place\_id NUMBER(4) := 1400;   
 new\_colname VARCHAR2(10) := 'sec\_name';   
BEGIN

3、执行查询 。

EXECUTE IMMEDIATE 'insert into sections\_t1 values(:1, :2, :3, :4)'   
USING section, section\_name, manager\_id,place\_id;

4、执行查询（重复占位符） 。

EXECUTE IMMEDIATE 'insert into sections\_t1 values(:1, :2, :3, :1)'   
USING section, section\_name, manager\_id;

5、执行ALTER语句（建议采用“||”拼接数据库对象构造DDL语句） 。

EXECUTE IMMEDIATE 'alter table sections\_t1 rename section\_name to ' || new\_colname;   
END;   
/

6、查询数据 。

SELECT \* FROM sections\_t1;

7、删除表 。

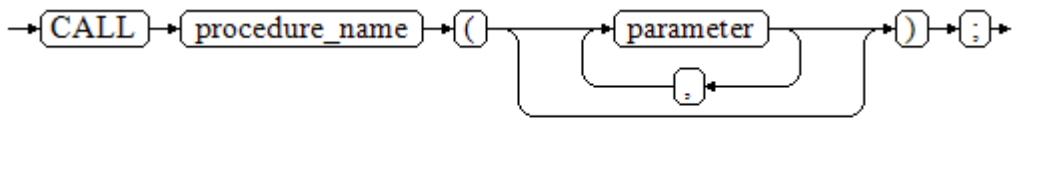
DROP TABLE sections\_t1;

### 基本语句

#### 调用语句

**语法**

调用一个语句的语法请参见下图。

call\_clause::=

对以上语法格式的解释如下：

* procedure\_name：存储过程名。
* parameter：存储过程的参数，可以没有或者有多个参数。

**示例**

1、创建表并插入测试数据。

create table staffs(  
section\_id int,  
salary int  
);  
insert into staffs values(1,1000);  
insert into staffs values(2,1000);  
insert into staffs values(3,1000);  
insert into staffs values(4,1000);  
insert into staffs values(5,1000);  
insert into staffs values(6,1000);  
insert into staffs values(7,1000);  
insert into staffs values(8,1000);

2、创建存储过程proc\_staffs 。

CREATE OR REPLACE PROCEDURE proc\_staffs   
(   
section NUMBER(6),   
salary\_sum out NUMBER(8,2),   
staffs\_count out INTEGER   
)   
IS   
BEGIN   
SELECT sum(salary), count(\*) INTO salary\_sum, staffs\_count FROM staffs where section\_id = section;   
END;   
/

3、调用存储过程proc\_return。

CALL proc\_staffs(2,8,6);

结果显示如下：

salary\_sum | staffs\_count   
------------+--------------  
 1000 | 1  
(1 row)

4、清除存储过程 。

DROP PROCEDURE proc\_staffs;

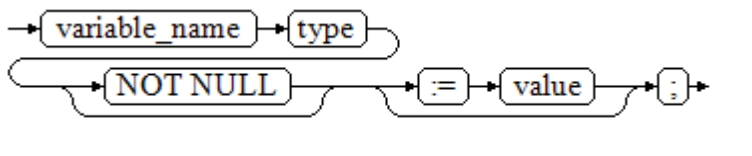
#### 定义变量

介绍PL/SQL中变量的声明，以及该变量在代码中的作用域。

**变量声明**

变量声明语法请参见下图。

declare\_variable::=



对以上语法格式的解释如下：

* variable\_name：变量名。
* type：变量类型。
* value：该变量的初始值（如果不给定初始值，则初始为NULL）。value也可以是表达式。

变量类型除了支持基本类型，还可以使用%TYPE和%ROWTYPE去声明一些与其他表字段或表结构本身相关的变量。

* %TYPE属性

%TYPE主要用于声明某个与其他变量类型（例如，表中某列的类型）相同的变量。假如我们想定义一个my\_name变量，它的变量类型与employee的firstname类型相同，我们可以通过如下定义：

my\_name employee.firstname%TYPE

这样定义可以带来两个好处，首先，我们不用预先知道employee 表的firstname类型具体是什么。其次，即使之后firstname类型有了变化，我们也不需要再次修改my\_name的类型。

* %ROWTYPE属性

%ROWTYPE属性主要用于对一组数据的类型声明，用于存储表中的一行数据或从游标匹配的结果。假如，我们需要一组数据，该组数据的字段名称与字段类型都与employee表相同。我们可以通过如下定义：

my\_employee employee%ROWTYPE

**变量作用域**

变量的作用域表示变量在代码块中的可访问性和可用性。只有在它的作用域内，变量才有效。

* 变量必须在declare部分声明，即必须建立BEGIN-END块。块结构也强制变量必须先声明后使用，即变量在过程内有不同作用域、不同的生存期。
* 同一变量可以在不同的作用域内定义多次，内层的定义会覆盖外层的定义。
* 在外部块定义的变量，可以在嵌套块中使用。但外部块不能访问嵌套块中的变量。

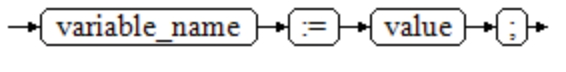
**示例**

DECLARE   
 emp\_id INTEGER := 7788; --定义变量并赋值   
BEGIN   
 emp\_id := 5\*7784; --变量赋值   
END;   
/

#### 赋值语句

**语法**

给变量赋值的语法请参见下图。

assignment\_value::=

对以上语法格式的解释如下：

* variable\_name：变量名。
* value：可以是值或表达式。值value的类型需要和变量variable\_name的类型兼容才能正确赋值。

**示例**

DECLARE   
 emp\_id INTEGER := 7788;--赋值   
BEGIN   
 emp\_id := 5;--赋值   
 emp\_id := 5\*7784;   
END;   
/

### 控制语句

#### GOTO语句

**功能描述**

GOTO语句可以实现从GOTO位置到目标语句的无条件跳转。GOTO语句会改变原本的执行逻辑，因此应该慎重使用，或者也可以使用EXCEPTION处理特殊场景。当执行GOTO语句时，目标Label必须是唯一的。

**语法格式**

label declaration ::=

labelGOTOgoto statement ::=

**注意事项**

* 不支持有多个相同的GOTO labels目标场景，无论是否在同一个block中。

BEGIN  
 GOTO pos1;   
 <<pos1>>  
 SELECT \* FROM ...  
 <<pos1>>  
 UPDATE t1 SET ...  
 END;  
/

* 不支持GOTO跳转到IF语句、CASE语句、LOOP语句中。

BEGIN  
 GOTO pos1;   
 IF valid THEN  
 <<pos1>>  
 SELECT \* FROM ...  
 END IF;  
 END;  
/

* 不支持GOTO语句从一个IF子句跳转到另一个IF子句，或从一个CASE语句的WHEN子句跳转到另一个WHEN子句。

BEGIN   
 IF valid THEN  
 GOTO pos1;  
 SELECT \* FROM ...  
 ELSE  
 <<pos1>>  
 UPDATE t1 SET ...  
 END IF;  
 END;  
/

* 不支持从外部块跳转到内部的BEGIN-END块。

BEGIN  
 GOTO pos1;   
 BEGIN  
 <<pos1>>  
 UPDATE t1 SET ...  
 END;  
 END;  
/

* 不支持从异常处理部分跳转到当前的BEGIN-END块。但可以跳转到上层BEGIN-END块。

BEGIN  
 <<pos1>>  
 UPDATE t1 SET ...  
 EXCEPTION  
 WHEN condition THEN  
 GOTO pos1;  
 END;  
/

* 如果从GOTO到一个不包含执行语句的位置，需要添加NULL语句。

DECLARE  
 done BOOLEAN;  
 BEGIN  
 FOR i IN 1..50 LOOP  
 IF done THEN  
 GOTO end\_loop;  
 END IF;  
 <<end\_loop>> -- not allowed unless an executable statement follows  
 NULL; -- add NULL statement to avoid error  
 END LOOP; -- raises an error without the previous NULL  
 END;  
/

**示例**

1、创建测试表test。

CREATE TABLE test(id,int);

2、创建匿名块进行测试，执行过程将从GOTO语句位置跳转到'pos1'标记位置执行。

BEGIN  
GOTO pos1;  
INSERT INTO test(id) VALUES(1);  
<<pos1>>  
INSERT INTO test(id) VALUES(2);  
END;  
/

3、验证结果。

SELECT \* FROM test;

当结果返回如下信息，则表示跳转成功。

id   
----  
 2  
(1 row)

#### 错误捕获语句

**功能描述**

缺省时，当PL/SQL函数执行过程中发生错误时退出函数执行，并且周围的事务也会回滚。可以用一个带有EXCEPTION子句的BEGIN块捕获错误并且从中恢复。

**语法格式**

[<<label>>]  
[DECLARE  
 declarations]  
BEGIN  
 statements  
EXCEPTION  
 WHEN condition [OR condition ...] THEN  
 handler\_statements  
 [WHEN condition [OR condition ...] THEN  
 handler\_statements  
 ...]  
END;

**参数说明**

* condition：可以是SQL标准错误码编号说明的任意值。特殊的条件名OTHERS匹配除了QUERY\_CANCELED之外的所有错误类型。
* handler\_statements：捕获到错误语句后执行的操作。

**注意事项**

* 如果没有发生错误，这种形式的块儿只是简单地执行所有语句，然后转到END之后的下一个语句。但是如果在执行的语句内部发生了一个错误，则这个语句将会回滚，然后转到EXCEPTION列表。寻找匹配错误的第一个条件。若找到匹配，则执行对应的handler\_statements，然后转到END之后的下一个语句。如果没有找到匹配，则会向事务的外层报告错误，和没有EXCEPTION子句一样。错误码可以捕获同一类的其他错误码。也就是说该错误可以被一个包围块用EXCEPTION捕获，如果没有包围块，则进行退出函数处理。
* 如果在选中的handler\_statements里发生了新错误，则不能被这个EXCEPTION子句捕获，而是向事务的外层报告错误。一个外层的EXCEPTION子句可以捕获它。
* 如果一个错误被EXCEPTION捕获，PL/SQL函数的局部变量保持错误发生时的原值，但是所有该块中想写入数据库中的状态都回滚。

**示例**

* 示例1：当控制到达给y赋值的地方时，会有一个division\_by\_zero错误失败。这个错误将被EXCEPTION子句捕获。而在RETURN语句里返回的数值将是x的增量值。

1、创建测试表并插入数据。

CREATE TABLE mytab(id INT,firstname VARCHAR(20),lastname VARCHAR(20)) ;  
INSERT INTO mytab(firstname, lastname) VALUES('Tom', 'Jones');

2、创建函数并调用。

CREATE FUNCTION fun\_exp() RETURNS INT  
AS $$  
DECLARE  
 x INT :=0;  
 y INT;  
BEGIN  
 UPDATE mytab SET firstname = 'Joe' WHERE lastname = 'Jones';  
 x := x + 1;  
 y := x / 0;  
EXCEPTION  
 WHEN division\_by\_zero THEN  
 RAISE NOTICE 'caught division\_by\_zero';  
 RETURN x;  
END;$$  
LANGUAGE plpgsql;  
  
call fun\_exp();

当结果显示如下信息，则表示捕获到错误语句。

NOTICE: caught division\_by\_zero  
 fun\_exp   
----------------  
 1  
(1 row)

3、查询表数据验证结果。

select \* from mytab;

结果显示如下：

id | firstname | lastname   
----+-----------+----------  
 | Tom | Jones  
(1 row)

4、删除测试表和函数。

DROP FUNCTION fun\_exp();  
DROP TABLE mytab;

* 进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大的多。因此，不必要的时候不要使用EXCEPTION。
* 在下列场景中，无法捕获处理异常，整个存储过程回滚：节点故障、网络故障引起的存储过程参与节点线程退出以及COPY FROM操作中源数据与目标表的表结构不一致造成的异常。
* 示例2：UPDATE/INSERT异常，这个例子根据使用异常处理器执行恰当的UPDATE或INSERT 。

1、创建测试表。

CREATE TABLE db (a INT, b TEXT);

2、创建测试函数。

CREATE FUNCTION merge\_db(key INT, data TEXT) RETURNS VOID AS  
$$  
BEGIN  
 LOOP   
 UPDATE db SET b = data WHERE a = key; --第一次尝试更新key  
 IF found THEN  
 RETURN;  
 END IF;   
 BEGIN --不存在，所以尝试插入key，如果其他人同时插入相同的key，我们可能得到唯一key失败。  
 INSERT INTO db(a,b) VALUES (key, data);  
 RETURN;  
 EXCEPTION WHEN unique\_violation THEN  
 END; --什么也不做，并且循环尝试再次更新。  
 END LOOP;  
END;  
$$  
LANGUAGE plpgsql;

3、调用函数并查询数据进行对比。

SELECT merge\_db(1, 'david');  
SELECT \* FROM db;

结果显示如下：

a | b   
---+-------  
 1 | david  
(1 row)

再次调用并查询。

SELECT merge\_db(1, 'dennis');  
SELECT \* FROM db;

结果显示如下：

a | b   
---+--------  
 1 | dennis  
(1 row)

4、删除FUNCTION和TABLE。

DROP FUNCTION merge\_db;  
DROP TABLE db;

#### 返回语句

**RETURN**

**语法格式**

返回语句的语法格式请参见下图。

return\_clause::=



对以上语法的解释如下：

用于将控制从存储过程或函数返回给调用者。

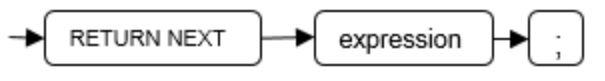
**示例**

请参见章节“调用语句”的示例。

**RETURN NEXT及RETURN QUERY**

**语法格式**

创建函数时需要指定返回值SETOF datatype。

return\_next\_clause::=

return\_query\_clause::=

**参数说明**

* RETURN NEXT：可用于标量和复合数据类型。
* RETURN QUERY：有一种变体RETURN QUERY EXECUTE，后面还可以增加动态查询，通过USING向查询插入参数。

当需要函数返回一个集合时，使用RETURN NEXT或者RETURN QUERY向结果集追加结果，然后继续执行函数的下一条语句。随着后续的RETURN NEXT或RETURN QUERY命令的执行，结果集中会有多个结果。函数执行完成后会一起返回所有结果。

**示例**

1、创建测试表并插入数据。

CREATE TABLE t1(a int);   
INSERT INTO t1 VALUES(1),(10);

2、创建函数测试RETURN NEXT 。

CREATE OR REPLACE FUNCTION fun\_for\_return\_next() RETURNS SETOF t1 AS $$   
DECLARE   
 r t1%ROWTYPE;   
BEGIN   
 FOR r IN select \* from t1   
 LOOP   
 RETURN NEXT r;   
 END LOOP;   
 RETURN;   
END;   
$$ LANGUAGE PLPgsql;

3、调用函数。

call fun\_for\_return\_next();

结果显示如下：

a   
---   
 1   
 10   
(2 rows)

4、创建函数测试RETURN QUERY 。

CREATE OR REPLACE FUNCTION fun\_for\_return\_query() RETURNS SETOF t1 AS $$   
DECLARE   
 r t1%ROWTYPE;   
BEGIN   
 RETURN QUERY select \* from t1;   
END;   
$$   
language plpgsql;

5、调用函数

call fun\_for\_return\_query();

结果显示如下：

a   
---   
 1   
 10   
(2 rows)

#### 空语句

**功能描述**

在PL/SQL程序中，可以用NULL语句来说明“不用做任何事情”，相当于一个占位符，可以使某些语句变得有意义，提高程序的可读性。

**语法格式**

DECLARE   
 …   
BEGIN   
 …   
 IF v\_num IS NULL THEN   
 NULL; -- 不需要处理任何数据。   
 END IF;   
END;

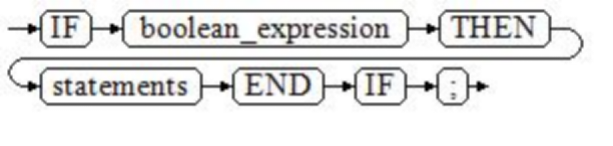
#### 条件语句

**功能描述**

条件语句的主要作用判断参数或者语句是否满足已给定的条件，根据判定结果执行相应的操作。

**语法格式**

Vastbase有五种形式的IF：

* IF\_THEN

IF\_THEN::=

IF\_THEN语句是IF的最简单形式。如果条件为真，statements将被执行。否则，将忽略它们的结果使该IF\_THEN语句执行结束。

* IF\_THEN\_ELSE

IF\_THEN\_ELSE::=

IF\_THEN\_ELSE语句增加了ELSE的分支，可以声明在条件为假的时候执行的语句。

* IF\_THEN\_ELSE IF

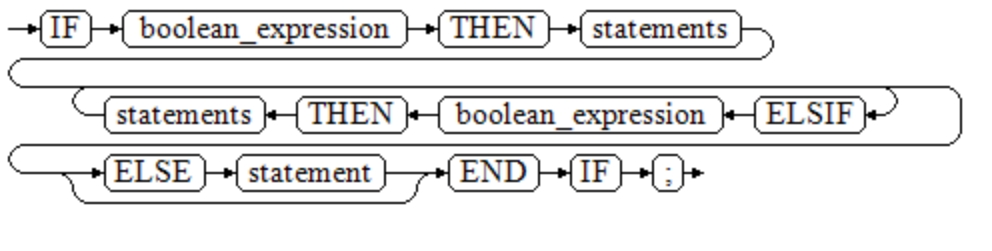
IF语句可以嵌套，嵌套方式如下：

IF sex = 'm' THEN   
 pretty\_sex := 'man';   
ELSE   
 IF sex = 'f' THEN   
 pretty\_sex := 'woman';   
 END IF;   
END IF;

这种形式实际上就是在一个IF语句的ELSE部分嵌套了另一个IF语句。因此需要一个END IF语句给每个嵌套的IF，另外还需要一个END IF语句结束父IF-ELSE。如果有多个选项，可使用下面的形式。

* IF\_THEN\_ELSIF\_ELSE

IF\_THEN\_ELSIF\_ELSE::=



* IF\_THEN\_ELSEIF\_ELSE

ELSEIF是ELSIF的别名。

**示例**

1、创建存储过程。

CREATE OR REPLACE PROCEDURE proc\_control\_structure(i in integer)   
AS   
 BEGIN   
 IF i > 0 THEN   
 raise info 'i:% is greater than 0. ',i;   
 ELSIF i < 0 THEN   
 raise info 'i:% is smaller than 0. ',i;   
 ELSE   
 raise info 'i:% is equal to 0. ',i;   
 END IF;   
 RETURN;   
 END;   
/

2、调用存储过程。

CALL proc\_control\_structure(3);

结果显示如下：

INFO: i:3 is greater than 0.   
-----------------------------  
 proc\_control\_structure   
  
(1 row)

3、删除存储过程

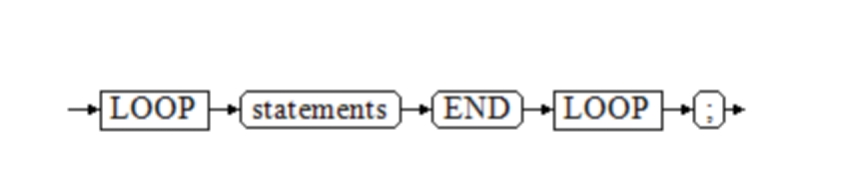
DROP PROCEDURE proc\_control\_structure;

#### 循环语句

* 简单LOOP语句

**语法图**

loop::=



**示例**

创建存储过程 proc\_loop并调用。

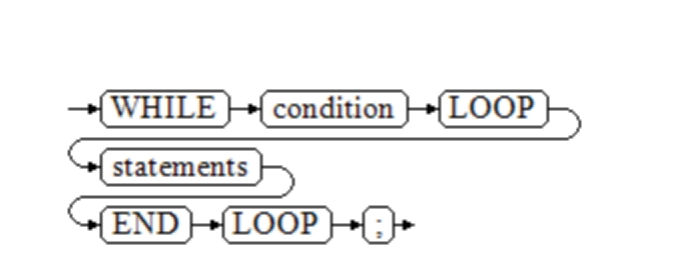
CREATE OR REPLACE PROCEDURE proc\_loop(i in integer, count out integer)   
AS   
 BEGIN   
 count:=0;   
 LOOP   
 IF count > i THEN   
 raise info 'count is %. ', count;   
 EXIT;   
 ELSE   
 count:=count+1;   
 END IF;   
 END LOOP;   
 END;   
/   
CALL proc\_loop(10,5);

当结果显示如下信息，则表示loop验证成功。

INFO: count is 11.   
 count   
----------------------  
 11  
(1 row)

该循环必须要结合EXIT使用，否则将陷入死循环。

* WHILE\_LOOP语句

**语法图**

while\_loop::=

只要条件表达式为真，WHILE语句就会不停的在一系列语句上进行循环，在每次进入循环体的时候进行条件判断。

**示例**

创建存储过程并调用。

CREATE TABLE integertable(c1 integer) ;   
CREATE OR REPLACE PROCEDURE proc\_while\_loop(maxval in integer)   
AS   
 DECLARE   
 i int :=1;   
 BEGIN   
 WHILE i < maxval LOOP   
 INSERT INTO integertable VALUES(i);   
 i:=i+1;   
 END LOOP;   
 END;   
/   
  
CALL proc\_while\_loop(10);

查询表数据验证结果。

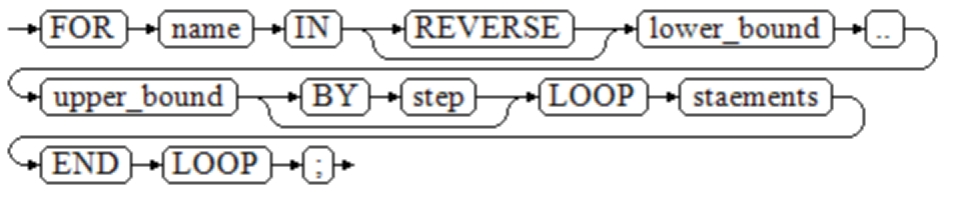
select \* from integertable;

当结果显示如下信息，则表示循环插入成功。

c1   
-------  
 1  
 2  
 3  
 4  
 5  
 6  
 7  
 8  
 9  
(9 rows)

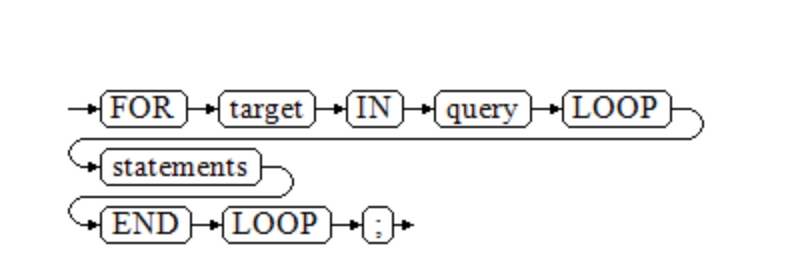
* FOR\_LOOP（integer变量）语句

**语法图**

for\_loop::=

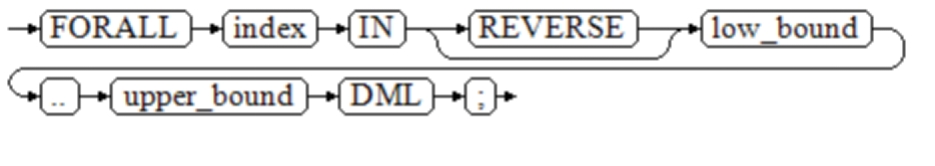
* 变量name会自动定义为integer类型并且只在此循环里存在。变量name介于lower\_bound和upper\_bound之间。
* 当使用REVERSE关键字时，lower\_bound必须大于等于upper\_bound，否则循环体不会被执行。
* FOR\_LOOP查询语句

**语法图**

for\_loop\_query::=

变量target会自动定义，类型和query的查询结果的类型一致，并且只在此循环中有效。target的取值就是query的查询结果。

* FORALL批量查询语句

**语法图**

forall::=

变量index会自动定义为integer类型并且只在此循环里存在。index的取值介于low\_bound和upper\_bound之间。

**示例**

1、创建测试表并插入数据。

CREATE TABLE hdfs\_t1 (   
 title NUMBER(6),   
 did VARCHAR2(20),   
 data\_peroid VARCHAR2(25),   
 kind VARCHAR2(25),   
 interval VARCHAR2(20),   
 time DATE,   
 isModified VARCHAR2(10)   
);   
  
INSERT INTO hdfs\_t1 VALUES( 8, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', to\_date('21-06-1999', 'dd-mm-yyyy'), 'SH\_CLERK' );

2、创建存储过程并调用。

CREATE OR REPLACE PROCEDURE proc\_forall()   
AS   
BEGIN   
 FORALL i IN 100..120   
 update hdfs\_t1 set title = title + 100\*i;   
END;   
/   
  
CALL proc\_forall();

3、查询存储过程调用结果

SELECT \* FROM hdfs\_t1 ;

当结果显示如下信息，则表示循环调用成功。

title | did | data\_peroid | kind | interval | time | ismodified   
--------+--------+-------------+----------+--------------+---------------------+------------  
 231008 | Donald | OConnell | DOCONNEL | 650.507.9833 | 1999-06-21 00:00:00 | SH\_CLERK  
(1 row)

4、删除存储过程和表

DROP PROCEDURE proc\_forall;   
DROP TABLE hdfs\_t1;

### 锁操作

Vastbase提供了多种锁模式用于控制对表中数据的并发访问。这些模式可以用在MVCC（多版本并发控制）无法给出期望行为的场合。同样，大多数Vastbase命令自动施加恰当的锁，以保证被引用的表在命令的执行过程中不会以一种不兼容的方式被删除或者修改。比如，在存在其他并发操作的时候，ALTER TABLE是不能在同一个表上执行的。

### 游标

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。

游标的使用分为显式游标和隐式游标。对于不同的SQL语句，游标的使用情况不同，详细信息请参见下表

| **SQL语句** | **游标** |
| --- | --- |
| 非查询语句 | 隐式的 |
| 结果是单行的查询语句 | 隐式的或显式的 |
| 结果是多行的查询语句 | 显式的 |

#### 显式游标

**功能描述**

显式游标主要用于对查询语句的处理，尤其是在查询结果为多条记录的情况下。

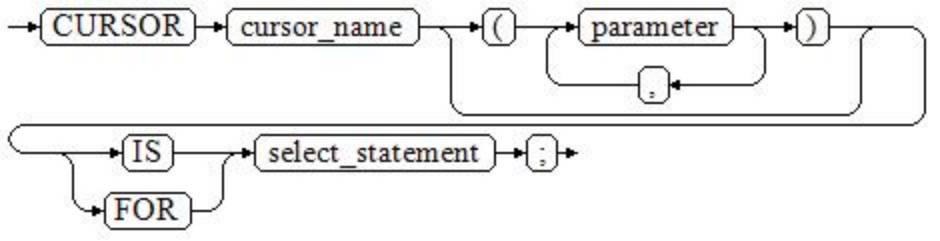
**处理步骤**

显式游标处理需六个PL/SQL步骤（定义静态游标->打开静态游标->提取游标数据->对该记录进行处理->继续处理->关闭游标）如下所示：

1、定义静态游标：就是定义一个游标名，以及与其相对应的SELECT语句。

定义静态游标的语法图，请参见下图：

static\_cursor\_define::=

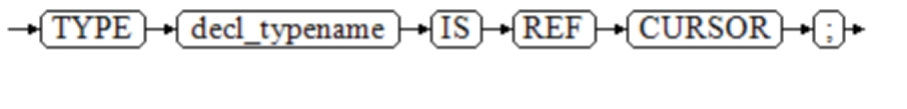


参数说明：

* cursor\_name：定义的游标名。
* parameter：游标参数，只能为输入参数，其格式为 parameter\_name datatype。
* select\_statement：查询语句。

根据执行计划的不同，系统会自动判断该游标是否可以用于以倒序的方式检索数据行。

定义动态游标：指ref游标，可以通过一组静态的SQL语句动态的打开游标。首先定义ref游标类型，然后定义该游标类型的游标变量，在打开游标时通过OPEN FOR动态绑定SELECT语句。

定义动态游标的语法图，请参见下图。

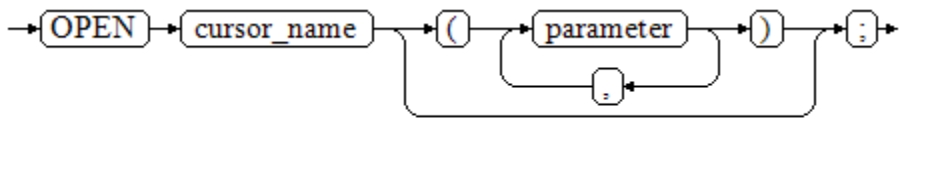
cursor\_typename::=

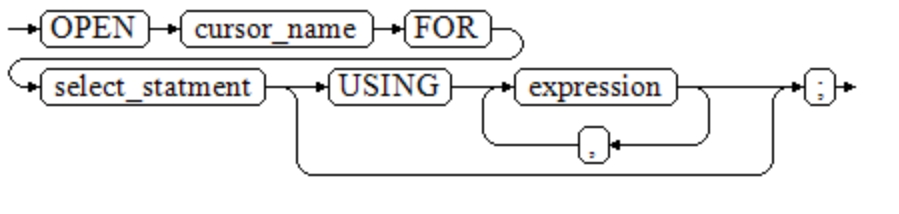
dynamic\_cursor\_define::=



2、打开静态游标：就是执行游标所对应的SELECT语句，将其查询结果放入工作区，并且指针指向工作区的首部，标识游标结果集合。如果游标查询语句中带有FOR UPDATE选项，OPEN语句还将锁定数据库表中游标结果集合对应的数据行。

打开静态游标的语法图，请参见下图

open\_static\_cursor::=

打开动态游标：可以通过OPEN FOR语句打开动态游标，动态绑定SQL语句。

打开动态游标的语法图，请参见下图。

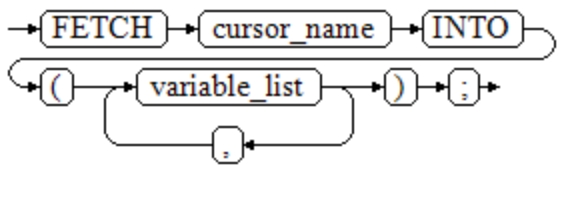
open\_dynamic\_cursor::=

PL/SQL程序不能用OPEN语句重复打开一个游标。

3、提取游标数据：检索结果集合中的数据行，放入指定的输出变量中。

提取游标数据的语法图，请参见下图。

fetch\_cursor::=



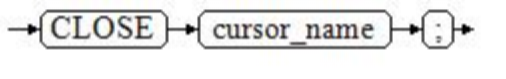
4、对该记录进行处理。

5、继续处理，直到活动集合中没有记录。

6、关闭游标：当提取和处理完游标结果集合数据后，应及时关闭游标，以释放该游标所占用的系统资源，并使该游标的工作区变成无效，不能再使用FETCH语句获取其中数据。关闭后的游标可以使用OPEN语句重新打开。

关闭游标的语法图，请参见下图。

close\_cursor::=



**属性**

游标的属性用于控制程序流程或者了解程序的状态。当运行DML语句时，PL/SQL打开一个内建游标并处理结果，游标是维护查询结果的内存中的一个区域，游标在运行DML语句时打开，完成后关闭。显式游标的属性为：

* %FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
* %NOTFOUND布尔型属性：与%FOUND相反。
* %ISOPEN布尔型属性：当游标已打开时返回TRUE。
* %ROWCOUNT数值型属性：返回已从游标中读取的记录数。

#### 隐式游标

**功能描述**

对于非查询语句，如修改、删除操作，则由系统自动地为这些操作设置游标并创建其工作区，这些由系统隐含创建的游标称为隐式游标，隐式游标的名称为SQL，这是由系统定义的。

对于隐式游标的操作，如定义、打开、取值及关闭操作，都由系统自动地完成，无需用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。在隐式游标的工作区中，所存放的数据是最新处理的一条SQL语句所包含的数据，与用户自定义的显式游标无关。

**语法格式**

格式调用为： SQL%

INSERT，UPDATE，DELETE，SELECT语句中不必明确定义游标。

**属性**

隐式游标属性为：

* SQL%FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
* SQL%NOTFOUND布尔型属性：与%FOUND相反。
* SQL%ROWCOUNT数值型属性：返回已从游标中读取得记录数。
* SQL%ISOPEN布尔型属性：取值总是FALSE。SQL语句执行完毕立即关闭隐式游标。

**示例**

1、创建测试表staffs。

create table staffs(  
section\_id int,  
salary int  
);  
insert into staffs values(1,1000);  
insert into staffs values(2,1000);  
insert into staffs values(3,1000);  
insert into staffs values(4,1000);  
insert into staffs values(5,1000);  
insert into staffs values(6,1000);  
insert into staffs values(7,1000);  
insert into staffs values(8,1000);

2、创建测试表sections。

create table sections(  
section\_id int,  
salary int  
);  
insert into sections values(1,1000);  
insert into sections values(2,1000);  
insert into sections values(3,1000);  
insert into sections values(4,1000);  
insert into sections values(5,1000);  
insert into sections values(6,1000);  
insert into sections values(7,1000);

3、删除员工表staffs中某部门的所有员工，如果该部门中已没有员工，则在部门表section中删除该部门。

CREATE OR REPLACE PROCEDURE proc\_cursor1()   
AS   
 DECLARE   
 V\_DEPTNO NUMBER(4) := 100;   
 BEGIN   
 DELETE FROM staffs WHERE section\_ID = V\_DEPTNO;   
 --根据游标状态做进一步处理   
 IF SQL%NOTFOUND THEN   
 DELETE FROM sections WHERE section\_ID = V\_DEPTNO;   
 END IF;   
 END;   
/   
CALL proc\_cursor1();

4、删除存储过程和表 。

DROP PROCEDURE proc\_cursor1;   
DROP TABLE sections;  
DROP TABLE staffs;

#### 游标循环

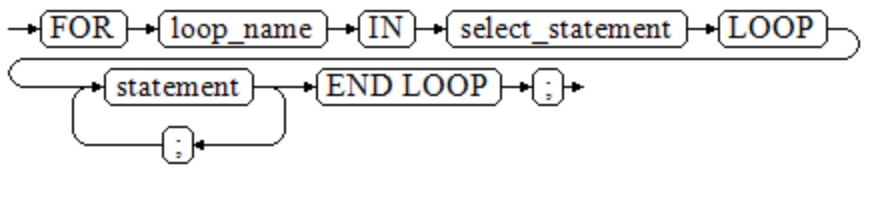
**功能描述**

游标在WHILE语句、LOOP语句中的使用称为游标循环，一般这种循环都需要使用OPEN、FETCH和CLOSE语句。下面要介绍的一种循环不需要这些操作，可以简化游标循环的操作，这种循环方式适用于静态游标的循环，不用执行静态游标的四个步骤。

**语法**

FOR AS循环的语法请参见下图。

FOR\_AS\_loop::=



**注意事项**

* 不能在该循环语句中对查询的表进行更新操作。
* 变量loop\_name会自动定义且只在此循环中有效，类型和select\_statement的查询结果类型一致。loop\_name的取值就是select\_statement的查询结果。
* 游标的属性中%FOUND、%NOTFOUND、%ROWCOUNT在Vastbase数据库中都是访问同一个内部变量，事务和匿名块不支持多个游标同时访问。

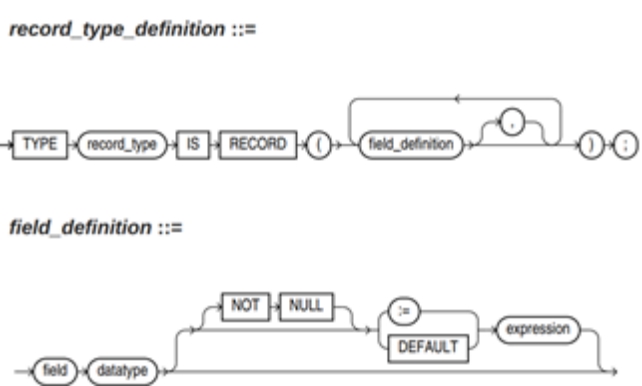
### record

**record类型的变量**

创建一个record变量的方式：定义一个record类型 ，然后使用该类型来声明一个变量。

**语法**

record类型的语法参见下图。



对以上语法格式的解释如下：

* record\_type：声明的类型名称。
* field：record类型中的成员名称。
* datatype：record类型中成员的类型。
* expression：设置默认值的表达式。

在Vastbase中：

* record类型变量的赋值支持：
* 在函数或存储过程的声明阶段，声明一个record类型，并且可以在该类型中定义成员变量。
* 一个record变量到另一个record变量的赋值。
* SELECT INTO和FETCH向一个record类型的变量中赋值。
* 将一个NULL值赋值给一个record变量。
* 不支持INSERT和UPDATE语句使用record变量进行插入数据和更新数据。
* 如果成员有复合类型，在声明阶段不支持指定默认值，该行为同声明阶段的变量一样。

**示例**

1、创建测试表。

CREATE TABLE emp\_rec(empno numeric(4,0),ename character varying(10), job character varying(9), mgr numeric(4,0),hiredate timestamp(0) without time zone,sal numeric(7,2),comm numeric(7,2),deptno numeric(2,0));

2、查看表定义。

\d emp\_rec

结果返回如下：

Table "public.emp\_rec"   
 Column | Type | Modifiers   
----------+--------------------------------+-----------   
 empno | numeric(4,0) | not null   
 ename | character varying(10) |   
 job | character varying(9) |   
 mgr | numeric(4,0) |   
 hiredate | timestamp(0) without time zone |   
 sal | numeric(7,2) |   
 comm | numeric(7,2) |   
 deptno | numeric(2,0) |

3、演示在存储过程中对数组进行操作。

CREATE OR REPLACE FUNCTION regress\_record(p\_w VARCHAR2)   
RETURNS   
VARCHAR2 AS $$   
DECLARE   
  
type rec\_type is record (name varchar2(100), epno int); -- 声明一个record类型.   
employer rec\_type;   
  
type rec\_type1 is record (name emp\_rec.ename%type, epno int not null :=10); --使用%type声明record类型   
employer1 rec\_type1;   
  
 type rec\_type2 is record ( --声明带有默认值的record类型   
 name varchar2 not null := 'SCOTT',   
 epno int not null :=10);   
 employer2 rec\_type2;   
 CURSOR C1 IS select ename,empno from emp\_rec order by 1 limit 1;   
  
BEGIN   
  
 employer.name := 'WARD'; --对一个record类型的变量的成员赋值。   
 employer.epno = 18;   
 raise info 'employer name: % , epno:%', employer.name, employer.epno;   
   
 employer1 := employer; --将一个record类型的变量赋值给另一个变量。   
 raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;   
  
 employer1 := NULL; --将一个record类型变量赋值为NULL。   
 raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;   
  
 raise info 'employer2 name: % ,epno: %', employer2.name, employer2.epno; --获取record变量的默认值。  
  
 for employer in select ename,empno from emp\_rec order by 1 limit 1 --在for循环中使用record变量   
 loop   
 raise info 'employer name: % , epno: %', employer.name, employer.epno;   
 end loop;   
  
 select ename,empno into employer2 from emp\_rec order by 1 limit 1; --在select into 中使用record变量。  
 raise info 'employer name: % , epno: %', employer2.name, employer2.epno;   
  
 OPEN C1; --在cursor中使用record变量。   
 FETCH C1 INTO employer2;   
 raise info 'employer name: % , epno: %', employer2.name, employer2.epno;   
 CLOSE C1;   
 RETURN employer.name;   
END;   
$$   
LANGUAGE plpgsql;

4、调用该存储过程。

CALL regress\_record('abc');

结果显示如下：

INFO: employer name: WARD , epno:18  
INFO: employer1 name: WARD , epno: 18  
INFO: employer1 name: <NULL> , epno: <NULL>  
INFO: employer2 name: SCOTT ,epno: 10  
INFO: employer name: <NULL> , epno: <NULL>  
INFO: employer name: <NULL> , epno: <NULL>  
  
 regress\_record   
  
(1 row)

5、删除存储过程。

DROP PROCEDURE regress\_record;

### Retry管理

**功能描述**

Retry是数据库在SQL或存储过程（包含匿名块）执行失败时，在数据库内部进行重新执行的过程，以提高执行成功率和用户体验。数据库内部通过检查发生错误时的错误码及Retry相关配置，决定是否进行重试。

**注意事项**

失败时回滚之前执行的语句，并重新执行存储过程进行Retry。

**示例**

CREATE OR REPLACE PROCEDURE retry\_basic ( IN x INT)   
AS   
BEGIN  
 INSERT INTO t1 (a) VALUES (x);  
 INSERT INTO t1 (a) VALUES (x+1);  
END;  
/  
  
CALL retry\_basic(1);

### 事务管理

**功能描述**

存储过程本身就处于一个事务中，开始调用最外围存储过程时会自动开启一个事务，在调用结束时自动提交或者发生异常时回滚。除了系统自动的事务控制外，也可以使用COMMIT/ROLLBACK来控制存储过程中的事务。在存储过程中调用COMMIT/ROLLBACK命令，将提交/回滚当前事务并自动开启一个新的事务，后续的所有操作都会在此新事务中运行。

保存点SAVEPOINT是事务中的一个特殊记号，它允许将那些在它建立后执行的命令全部回滚，把事务的状态恢复到保存点所在的时刻。存储过程中允许使用保存点来进行事务管理，当前支持保存点的创建、回滚和释放操作。存储过程中使用回滚保存点只是回退当前事务的修改，而不会改变存储过程的执行流程，也不会回退存储过程中的局部变量值等。

**语法格式**

* 定义保存点。

SAVEPOINT savepoint\_name;

* 回滚保存点。

ROLLBACK TO [SAVEPOINT] savepoint\_name;

* 释放保存点。

RELEASE [SAVEPOINT] savepoint\_name;

**使用场景**

支持调用的上下文环境：

* 支持在PLSQL的存储过程内使用COMMIT/ROLLBACK/SAVEPOINT。
* 支持含有EXCEPTION的存储过程使用COMMIT/ROLLBACK/SAVEPOINT。
* 支持在存储过程的EXCEPTION语句内使用COMMIT/ROLLBACK/SAVEPOINT。
* 支持在事务块里调用含有COMMIT/ROLLBACK/SAVEPOINT的存储过程，即通过/BEGIN/START/END等开启控制的外部事务。
* 支持在子事务中调用含有SAVEPOINT的存储过程，即存储过程中使用外部定义的SAVEPOINT，回退事务状态到存储过程外定义的SAVEPOINT位置。
* 支持存储过程外部对存储过程内定义的SAVEPOINT可见，即存储过程外可以将事务修改回滚到存储过程中定义SAVEPOINT的位置。
* 支持多数PLSQL的上下文和语句内调用COMMIT/ROLLBACK/SAVEPOINT，包括常用的IF/FOR/CURSOR LOOP/WHILE。
* 支持存储过程返回值与简单表达式计算中调用含有COMMIT/ROLLBACK/SAVEPOINT的存储过程或者函数。

支持提交/回滚的内容：

* 支持DDL在COMMIT/ROLLBACK后的提交/回滚。
* 支持DML的COMMIT/ROLLBACK后的提交。
* 支持存储过程内GUC参数的回滚提交。

**注意事项**

不支持调用的上下文环境：

* 不支持除PLSQL的其他存储过程中调用COMMIT/ROLLBACK/SAVEPOINT，例如PLJAVA、PLPYTHON等。
* 不支持函数中调用COMMIT/ROLLBACK/SAVEPOINT，包括函数调用含有COMMIT/ROLLBACK/SAVEPOINT的存储过程。
* 不支持事务块中调用了SAVEPOINT后，调用含有COMMIT/ROLLBACK的存储过程。
* 不支持TRIGGER中调用含有COMMIT/ROLLBACK/SAVEPOINT语句的存储过程。
* 不支持EXECUTE语句中调用COMMIT/ROLLBACK/SAVEPOINT语句。
* 不支持在CURSOR语句中打开一个含有COMMIT/ROLLBACK/SAVEPOINT的存储过程。
* 不支持带有IMMUTABLE以及SHIPPABLE的存储过程调用COMMIT/ROLLBACK/SAVEPOINT，或调用带有COMMIT/ROLLBACK/SAVEPOINT语句的存储过程。
* 不支持SQL中调用含有COMMIT/ROLLBACK/SAVEPOINT语句的存储过程，除了SELECT PROC以及CALL PROC。
* 存储过程头带有GUC参数设置的不允许调用COMMIT/ROLLBACK/SAVEPOINT语句。
* 不支持CURSOR/EXECUTE语句，以及各类表达式内调用COMMIT/ROLLBACK/SAVEPOINT。
* 自治事务和存储过程事务是两个独立的事务，不能互相使用对方事务中定义的保存点。

不支持提交回滚的内容：

* 不支持存储过程内声明变量以及传入变量的提交/回滚。
* 不支持存储过程内必须重启生效的GUC参数的提交/回滚。

**示例**

* 示例1：支持在PLSQL的存储过程内使用COMMIT/ROLLBACK。

CREATE TABLE EXAMPLE1(COL1 INT);  
CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE()  
AS  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1(COL1) VALUES (i);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 END LOOP;  
END;  
/

* 示例2：
* 支持含有EXCEPTION的存储过程使用COMMIT/ROLLBACK。
* 支持在存储过程的EXCEPTION语句内使用COMMIT/ROLLBACK。
* 支持DDL在COMMIT/ROLLBACK后的提交/回滚。

CREATE OR REPLACE PROCEDURE TEST\_COMMIT\_INSERT\_EXCEPTION\_ROLLBACK()  
AS  
BEGIN  
 DROP TABLE IF EXISTS TEST\_COMMIT;   
 CREATE TABLE TEST\_COMMIT(A INT, B INT);  
 INSERT INTO TEST\_COMMIT SELECT 1, 1;  
 COMMIT;  
 CREATE TABLE TEST\_ROLLBACK(A INT, B INT);  
 RAISE EXCEPTION 'RAISE EXCEPTION AFTER COMMIT';  
 EXCEPTION  
 WHEN OTHERS THEN  
 INSERT INTO TEST\_COMMIT SELECT 2, 2;  
 ROLLBACK;  
END;  
/

* 示例3：支持在事务块里调用含有COMMIT/ROLLBACK的存储过程，即通过/BEGIN/START/END等开启控制的外部事务。

BEGIN;  
 CALL TEST\_COMMIT\_INSERT\_EXCEPTION\_ROLLBACK();  
END;

* 示例4：支持多数PLSQL的上下文和语句内调用COMMIT/ROLLBACK，包括常用的IF/FOR/CURSOR LOOP/WHILE。

CREATE OR REPLACE PROCEDURE TEST\_COMMIT2()  
IS  
BEGIN  
 DROP TABLE IF EXISTS TEST\_COMMIT;  
 CREATE TABLE TEST\_COMMIT(A INT);  
 FOR I IN REVERSE 3..0 LOOP  
 INSERT INTO TEST\_COMMIT SELECT I;  
 COMMIT;  
 END LOOP;  
 FOR I IN REVERSE 2..4 LOOP  
 UPDATE TEST\_COMMIT SET A=I;  
 COMMIT;  
 END LOOP;  
EXCEPTION  
WHEN OTHERS THEN   
 INSERT INTO TEST\_COMMIT SELECT 4;  
 COMMIT;  
END;  
/

* 示例5：支持存储过程返回值与简单表达式计算。

CREATE OR REPLACE PROCEDURE exec\_func3(RET\_NUM OUT INT)   
AS   
BEGIN   
 RET\_NUM := 1+1;   
COMMIT;   
END;   
/   
CREATE OR REPLACE PROCEDURE exec\_func4(ADD\_NUM IN INT)   
AS   
SUM\_NUM INT;   
BEGIN   
SUM\_NUM := ADD\_NUM + exec\_func3();   
COMMIT;   
END;   
/

* 示例6：支持存储过程内GUC参数的回滚提交。

1、查询参数值。

SHOW explain\_perf\_mode;  
SHOW enable\_force\_vector\_engine;

2、创建存储过程修改参数。

CREATE OR REPLACE PROCEDURE GUC\_ROLLBACK()  
AS  
BEGIN  
 SET enable\_force\_vector\_engine = on;  
 COMMIT;  
 SET explain\_perf\_mode TO pretty;  
 ROLLBACK;  
END;  
/

3、调用存储过程并查询。

call GUC\_ROLLBACK();  
SHOW explain\_perf\_mode;  
SHOW enable\_force\_vector\_engine;

结果显示如下：

explain\_perf\_mode   
-------------------  
 normal  
(1 row)

enable\_force\_vector\_engine   
----------------------------  
 on  
(1 row)

4、将参数改回。

SET enable\_force\_vector\_engine = off;

* 示例7：函数（Function）中不允许调用commit/rollback语句

CREATE OR REPLACE FUNCTION FUNCTION\_EXAMPLE1() RETURN INT  
AS  
EXP INT;  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1(col1) VALUES (i);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 END LOOP;  
 SELECT COUNT(\*) FROM EXAMPLE1 INTO EXP;  
 RETURN EXP;  
END;  
/

* 示例8：函数（Fucntion）中不允许调用带有commit/rollback语句的存储过程。

CREATE OR REPLACE FUNCTION FUNCTION\_EXAMPLE2() RETURN INT  
AS  
EXP INT;  
BEGIN  
 --transaction\_example为存储过程，带有commit/rollback语句  
 CALL transaction\_example();  
 SELECT COUNT(\*) FROM EXAMPLE1 INTO EXP;  
 RETURN EXP;  
END;  
/

* 示例9：不允许Trigger的存储过程包含commit/rollback语句，或调用带有commit/rollback语句的存储过程。

CREATE OR REPLACE FUNCTION FUNCTION\_TRI\_EXAMPLE2() RETURN TRIGGER  
AS  
EXP INT;  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1(col1) VALUES (i);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 END LOOP;  
 SELECT COUNT(\*) FROM EXAMPLE1 INTO EXP;  
   
END;  
/  
   
CREATE TRIGGER TRIGGER\_EXAMPLE AFTER DELETE ON EXAMPLE1   
FOR EACH ROW EXECUTE PROCEDURE FUNCTION\_TRI\_EXAMPLE2();  
   
DELETE FROM EXAMPLE1;

* 示例10：不支持带有IMMUABLE以及SHIPPABLE的存储过程调用commit/rollback，或调用带有commit/rollback语句的存储过程。

CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE1()  
IMMUTABLE  
AS  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1 (col1) VALUES (i);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 END LOOP;  
END;  
/

* 示例11：不支持出现在SQL中的调用（除了Select Procedure）。

CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE3()  
AS  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1 (col1) VALUES (i);  
 IF i % 2 = 0 THEN  
 EXECUTE IMMEDIATE 'COMMIT';  
 ELSE  
 EXECUTE IMMEDIATE 'ROLLBACK';  
 END IF;  
 END LOOP;  
END;  
/

* 示例12：存储过程头带有GUC参数设置的不允许调用commit/rollback语句。

CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE4()  
SET ARRAY\_NULLS TO "ON"  
AS  
BEGIN  
 FOR i IN 0..20 LOOP  
 INSERT INTO EXAMPLE1 (col1) VALUES (i);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 END LOOP;  
END;  
/

* 示例13：游标open的对象不允许为带有commit/rollback语句的存储过程。

CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE5(INTIN IN INT, INTOUT OUT INT)  
AS  
BEGIN  
INTOUT := INTIN + 1;  
COMMIT;  
END;  
/  
   
CREATE OR REPLACE PROCEDURE TRANSACTION\_EXAMPLE6()  
AS  
CURSOR CURSOR1(EXPIN INT)  
IS SELECT TRANSACTION\_EXAMPLE5(EXPIN);  
INTEXP INT;  
BEGIN  
 FOR i IN 0..20 LOOP  
 OPEN CURSOR1(i);  
 FETCH CURSOR1 INTO INTEXP;  
 INSERT INTO EXAMPLE1(COL1) VALUES (INTEXP);  
 IF i % 2 = 0 THEN  
 COMMIT;  
 ELSE  
 ROLLBACK;  
 END IF;  
 CLOSE CURSOR1;  
 END LOOP;  
END;   
/

* 示例14：不支持CURSOR/EXECUTE语句，以及各类表达式内调用COMMIT/ROLLBACK。

CREATE OR REPLACE PROCEDURE exec\_func1()  
AS  
BEGIN  
 CREATE TABLE TEST\_exec(A INT);  
COMMIT;  
END;  
/  
CREATE OR REPLACE PROCEDURE exec\_func2()  
AS  
BEGIN  
EXECUTE exec\_func1();  
COMMIT;  
END;  
/

* 示例15：存储过程使用保存点回退事务部分修改。

CREATE OR REPLACE PROCEDURE STP\_SAVEPOINT\_EXAMPLE1()  
AS  
BEGIN  
 INSERT INTO EXAMPLE1 VALUES(1);  
 SAVEPOINT s1;  
 INSERT INTO EXAMPLE1 VALUES(2);  
 ROLLBACK TO s1; -- 回退插入记录2  
 INSERT INTO EXAMPLE1 VALUES(3);  
END;  
/

* 示例16：存储过程中使用保存点回退到存储过程外部定义的保存点。

CREATE OR REPLACE PROCEDURE STP\_SAVEPOINT\_EXAMPLE2()  
AS  
BEGIN  
 INSERT INTO EXAMPLE1 VALUES(2);  
 ROLLBACK TO s1; -- 回退插入记录2  
 INSERT INTO EXAMPLE1 VALUES(3);  
END;  
/  
   
BEGIN;  
INSERT INTO EXAMPLE1 VALUES(1);  
SAVEPOINT s1;  
CALL STP\_SAVEPOINT\_EXAMPLE2();  
SELECT \* FROM EXAMPLE1;  
COMMIT;

* 示例17：存储过程外部回退到存储过程中定义的保存点。

CREATE OR REPLACE PROCEDURE STP\_SAVEPOINT\_EXAMPLE3()  
AS  
BEGIN  
 INSERT INTO EXAMPLE1 VALUES(1);  
 SAVEPOINT s1;  
 INSERT INTO EXAMPLE1 VALUES(2);  
END;  
/  
   
BEGIN;  
INSERT INTO EXAMPLE1 VALUES(3);  
CALL STP\_SAVEPOINT\_EXAMPLE3();  
ROLLBACK TO SAVEPOINT s1; --回退存储过程中插入记录2  
SELECT \* FROM EXAMPLE1;  
COMMIT;

### 数据类型

数据类型是一组值的集合以及定义在这个值集上的一组操作。Vastbase数据库是由表的集合组成的，而各表中的列定义了该表，每一列都属于一种数据类型，Vastbase根据数据类型有相应函数对其内容进行操作，例如Vastbase可对数值型数据进行加、减、乘、除操作。

**数据类型转换**

数据库中允许有些数据类型进行隐式类型转换（赋值、函数调用的参数等），有些数据类型间不允许进行隐式数据类型转换，可尝试使用Vastbase提供的类型转换函数，例如CAST进行数据类型强转。

Vastbase数据库常见的隐式类型转换，请参见下表。

隐式类型转换表

| **原始数据类型** | **目标数据类型** | **备注** |
| --- | --- | --- |
| CHAR | VARCHAR2 | - |
| CHAR | NUMBER | 原数据必须由数字组成。 |
| CHAR | DATE | 原数据不能超出合法日期范围。 |
| CHAR | RAW | - |
| CHAR | CLOB | - |
| VARCHAR2 | CHAR | - |
| VARCHAR2 | NUMBER | 原数据必须由数字组成。 |
| VARCHAR2 | DATE | 原数据不能超出合法日期范围。 |
| VARCHAR2 | CLOB | - |
| NUMBER | CHAR | - |
| NUMBER | VARCHAR2 | - |
| DATE | CHAR | - |
| DATE | VARCHAR2 | - |
| RAW | CHAR | - |
| RAW | VARCHAR2 | - |
| CLOB | CHAR | - |
| CLOB | VARCHAR2 | - |
| CLOB | NUMBER | 原数据必须由数字组成。 |
| INT4 | CHAR | - |

Vastbase支持的DATE的效限范围是：公元前4713年到公元294276年。

### 数组

**数组类型的使用**

在使用数组之前，需要自定义一个数组类型。

在存储过程中紧跟AS关键字后面定义数组类型。定义方法为：

TYPE array\_type IS VARRAY(size) OF data\_type;

其中：

* array\_type：要定义的数组类型名。
* VARRAY：表示要定义的数组类型。
* size：取值为正整数，表示可以容纳的成员的最大数量。
* data\_type：要创建的数组中成员的类型。
* 在Vastbase中，数组会自动增长，访问越界会返回一个NULL，不会报错。
* 在存储过程中定义的数组类型，其作用域仅在该存储过程中。
* 建议选择上述定义方法的一种来自定义数组类型，当同时使用两种方法定义同名的数组类型时，Vastbase会优先选择存储过程中定义的数组类型来声明数组变量。

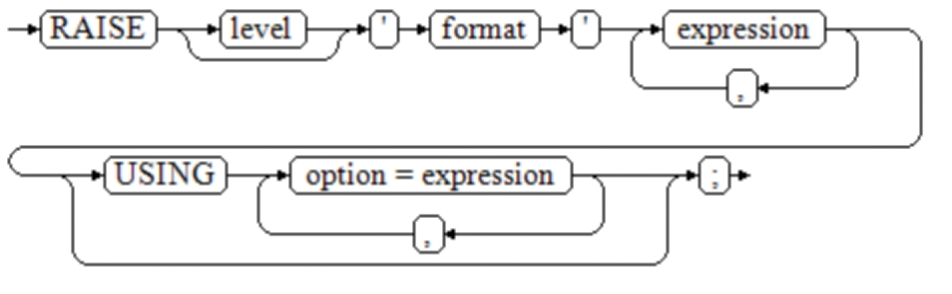
Vastbase支持使用圆括号来访问数组元素，且还支持一些特有的函数，如extend，count，first，last来访问数组的内容。

存储过程中如果有DML语句（SELECT、UPDATE、INSERT、DELETE），DML语句只能使用中括号来访问数组元素，使用小括号默认识别为数组访问，若数组不存在，则识别为函数表达式。

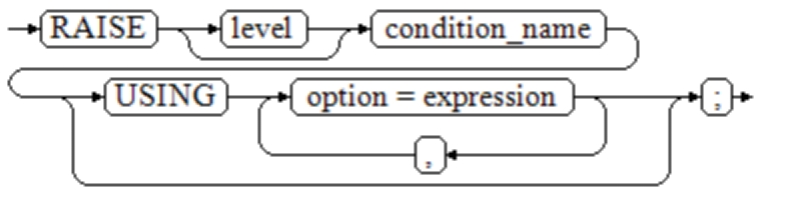
### 调试

**语法格式**

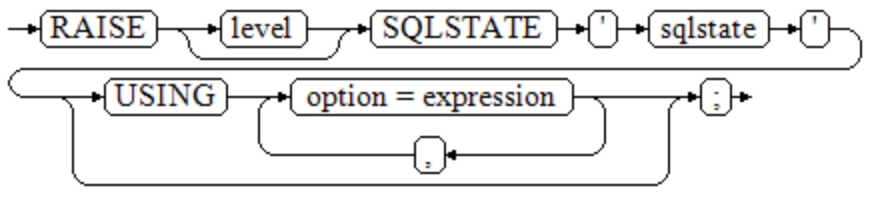
RAISE有以下五种语法格式：

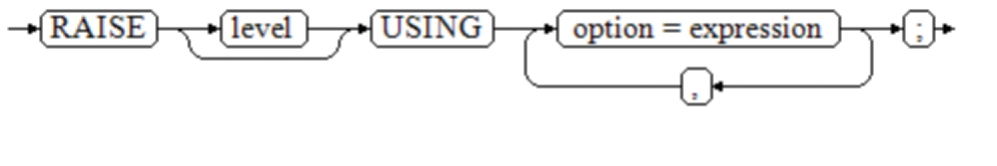
raise\_format::=

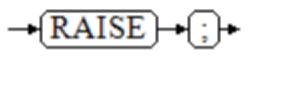
raise\_condition::=



raise\_sqlstate::=

 raise\_option::=

raise::=



**参数说明：**

* level选项用于指定错误级别，有DEBUG，LOG，INFO，NOTICE，WARNING以及EXCEPTION（默认值）。EXCEPTION抛出一个正常终止当前事务的异常，其他的仅产生不同异常级别的信息。特殊级别的错误信息是否报告到客户端、写到服务器日志由log\_min\_messages和client\_min\_messages这两个配置参数控制。
* format：格式字符串，指定要报告的错误消息文本。格式字符串后可跟表达式，用于向消息文本中插入。在格式字符串中，%由format后面跟着的参数的值替换，%%用于打印出%。例如：

--v\_job\_id 将替换字符串中的 %：   
RAISE NOTICE 'Calling cs\_create\_job(%)',v\_job\_id;

* option = expression：向错误报告中添加另外的信息。关键字option可以是MESSAGE、DETAIL、HINT以及ERRCODE，并且每一个expression可以是任意的字符串。
* MESSAGE，指定错误消息文本，这个选项不能用于在USING前包含一个格式字符串的RAISE语句中。
* DETAIL，说明错误的详细信息。
* HINT，用于打印出提示信息。
* ERRCODE，向报告中指定错误码（SQLSTATE）。可以使用条件名称或者直接用五位字符的SQLSTATE错误码。
* condition\_name：错误码对应的条件名。
* sqlstate：错误码。
* 如果在RAISE EXCEPTION命令中既没有指定条件名也没有指定SQLSTATE，默认用RAISE EXCEPTION (P0001)。如果没有指定消息文本，默认用条件名或者SQLSTATE作为消息文本。
* 图1-5[raise](#raise)所示的语法不接任何参数。这种形式仅用于一个BEGIN块中的EXCEPTION语句，它使得错误重新被处理。
* 当由SQLSTATE指定了错误码，则不局限于已定义的错误码，可以选择任意包含五个数字或者大写的ASCII字母的错误码，而不是00000。建议避免使用以三个0结尾的错误码，因为这种错误码是类别码，会被整个种类捕获。
* 兼容O模式下，SQLCODE等于SQLSTATE。

**示例**

* 示例1：终止事务时，给出错误和提示信息：

CREATE OR REPLACE PROCEDURE proc\_raise1(user\_id in integer)   
AS   
BEGIN   
RAISE EXCEPTION 'Noexistence ID --> %',user\_id USING HINT = 'Please check your user ID';   
END;   
/   
  
call proc\_raise1(300011);

结果显示如下：

ERROR: Noexistence ID --> 300011   
HINT: Please check your user ID

示例2：两种设置SQLSTATE的方式：

CREATE OR REPLACE PROCEDURE proc\_raise2(user\_id in integer)   
AS   
BEGIN   
RAISE 'Duplicate user ID: %',user\_id USING ERRCODE = 'unique\_violation';   
END;   
/   
  
\set VERBOSITY verbose   
  
call proc\_raise2(300011);

结果显示如下：

ERROR: Duplicate user ID: 300011   
SQLSTATE: 23505

如果主要的参数是条件名或者是SQLSTATE，可以使用：

RAISE division\_by\_zero;  
RAISE SQLSTATE '22012';

例如：

CREATE OR REPLACE PROCEDURE division(div in integer, dividend in integer)   
AS   
DECLARE   
res int;   
 BEGIN   
 IF dividend=0 THEN   
 RAISE division\_by\_zero;   
 RETURN;   
 ELSE   
 res := div/dividend;   
 RAISE INFO 'division result: %', res;   
 RETURN;   
 END IF;   
 END;   
/   
  
call division(3,0);

结果显示如下

ERROR: division\_by\_zero  
SQLSTATE: 22012

或者另一种方式：

RAISE unique\_violation USING MESSAGE = 'Duplicate user ID: ' || user\_id;

### 声明语法

**结构**

PL/SQL块中可以包含子块，子块可以位于PL/SQL中任何部分。PL/SQL块的结构如下：

* 声明部分：声明PL/SQL用到的变量、类型、游标、局部的存储过程和函数。

DECLARE

不涉及变量声明时声明部分可以没有。

* 对匿名块来说，没有变量声明部分时，可以省去DECLARE关键字。
* 对存储过程来说，没有DECLARE， AS相当于DECLARE。即便没有变量声明的部分，关键字AS也必须保留。
* 执行部分：过程及SQL语句，程序的主要部分。必选。

BEGIN

* 执行异常部分：错误处理。可选。

EXCEPTION

* 结束

END;   
/

禁止在PL/SQL块中使用连续的Tab，连续的Tab可能会造成在使用vsql工具带“-r”参数执行PL/SQL块时出现异常。

**分类**

PL/SQL块可以分为以下几类：

* 匿名块：动态构造，只能执行一次。语法请参考[图1](#匿名块语法)。
* 子程序：存储在数据库中的存储过程、函数、操作符和高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

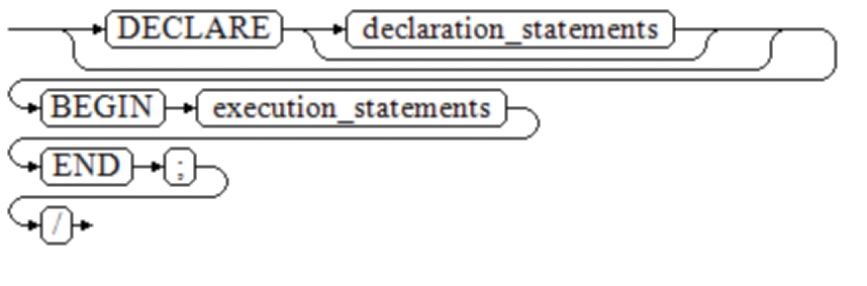
**匿名块**

匿名块（Anonymous Block）一般用于不频繁执行的脚本或不重复进行的活动。它们在一个会话中执行，并不被存储。

**语法**

匿名块的语法参见图1。

图1. anonymous\_block::=



对以上语法图的解释如下：

* 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。输入“/”按回车执行。

最后的结束符“/”必须独占一行，不能直接跟在END后面。

* 声明部分包括变量定义、类型、游标定义等。
* 最简单的匿名块不执行任何命令。但一定要在任意实施块里至少有一个语句，甚至是一个NULL语句。

**子程序**

存储在数据库中的存储过程、函数、操作符和高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

### PL/pgSQL语言函数

PL/pgsql是一种可载入的过程语言。

用PL/pgsql创建的函数可以被用在任何可以使用内建函数的地方。例如，可以创建复杂条件的计算函数并且后面用它们来定义操作符或把它们用于索引表达式。

SQL被大多数数据库用作查询语言。它是可移植的并且容易学习。但是每一个SQL语句必须由数据库服务器单独执行。

这意味着客户端应用必须发送每一个查询到数据库服务器、等待它被处理、接收并处理结果、做一些计算，然后发送更多查询给服务器。如果客户端和数据库服务器不在同一台机器上，则会引起进程间通信并且将带来网络负担。

通过PL/pgsql，可以将一整块计算和一系列查询分组在数据库服务器内部，这样就有了一种过程语言的能力并且使SQL更易用，同时能节省客户端/服务器通信开销。

* 客户端和服务器之间的额外往返通信被消除。
* 客户端不需要的中间结果不必被整理或者在服务器和客户端之间传送。
* 多轮的查询解析可以被避免。

PL/pgsql可以使用SQL中所有的数据类型、操作符和函数。

应用PL/pgsql创建函数的语法为CREATE FUNCTION。PL/pgsql是一种可载入的过程语言。其应用方法与存储过程相似，只是存储过程无返回值，函数有返回值。

# 并发控制

## 事务管理

### 管理事务

事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。Vastbase数据库支持的事务控制命令有启动、设置、提交、回滚事务。Vastbase数据库支持的事务隔离级别有读已提交和可重复读。

**事务控制**

以下是数据库支持的事务命令：

* 启动事务
* 用户可以使用START TRANSACTION和BEGIN语法启动事务,具体参考：SQL语法参考->SQL语法->START TRANSACTION和BEGIN章节。
* 设置事务
* 用户可以使用SET TRANSACTION或者SET LOCAL TRANSACTION语法设置事务特性，详细操作请参考：SQL语法参考->SQL语法->START TRANSACTION章节。
* 提交事务
* 用户可以使用CvastbaseIT或者END完成提交事务的功能，即提交事务的所有操作，详细操作请参考：SQL语法参考->SQL语法->COMMIT | END章节。
* 回滚事务
* 回滚是在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的操作全部撤销，详细操作请参考：SQL语法参考->SQL语法->ROLLBACK章节。

数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，如果其中有一个语句失败，那么整个请求都将会被回滚。

支持在存储过程中使用COMMIT和ROLLBACK。

未执行事务控制语句进行事务控制时，事务默认自动提交，不支持修改AUTOCOMMIT参数。

**事务隔离级别**

事务隔离级别，它决定多个事务并发操作同一个对象时的处理方式。

fig: **说明：**   
在事务中第一个数据修改语句（SELECT、INSERT、DELETE、UPDATE、FETCH、COPY）执行之后，事务隔离级别就不能再次设置。

* **READ COMMITTED**：读已提交隔离级别，事务只能读到已提交的数据而不会读到未提交的数据，这是缺省值。
* 实际上，SELECT查询会查看到在查询开始运行的瞬间该数据库的一个快照。不过，SELECT能查看到其自身所在事务中先前更新的执行结果。即使先前更新尚未提交。请注意，在同一个事务里两个相邻的SELECT命令可能会查看到不同的快照，因为其它事务会在第一个SELECT执行期间提交。
* 因为在读已提交模式里，每个新的命令都是从一个新的快照开始的，而这个快照包含所有到该时刻为止已提交的事务，因此同一事务中后面的命令将看到任何已提交的其它事务的效果。这里关心的问题是在单个命令里是否看到数据库里绝对一致的视图。
* 读已提交模式提供的部分事务隔离对于许多应用而言是足够的，并且这个模式速度快，使用简单。不过，对于做复杂查询和更新的应用，可能需要保证数据库有比读已提交模式更加严格的一致性视图。
* **REPEATABLE READ**：事务可重复读隔离级别，事务只能读到事务开始之前已提交的数据，不能读到未提交的数据以及事务执行期间其它并发事务提交的修改（但是，查询能查看到自身所在事务中先前更新的执行结果，即使先前更新尚未提交）。这个级别和读已提交是不一样的，因为可重复读事务中的查询看到的是事务开始时的快照，不是该事务内部当前查询开始时的快照，就是说，单个事务内部的select命令总是查看到同样的数据，查看不到自身事务开始之后其他并发事务修改后提交的数据。使用该级别的应用必须准备好重试事务，因为可能会发生串行化失败。

### 自治事务

**功能描述**

自治事务为在主事务执行过程中，以独立的事务上下文执行指定的某一类SQL语句，在定义函数/存储过程中，在函数/存储过程的声明部分增加指令：PRAGMA AUTONOMOUS\_TRANSACTION，来控制对当前的存储过程中的内部事务，能够独立提交，其commit和rollback不受主事务commit和rollback的影响。该执行方式与普通方式执行函数/存储过程的结果相同。

典型使用场景是，用一个表记录主事务执行过程中的操作信息，并且在主事务失败回退的时候，表中记录的操作信息不予回退。

**功能特性**

* 支持输入参数
* 支持嵌套调用
* 支持在触发器中使用自治事务
* 支持匿名块使用自治事务

**注意事项**

* 不支持集合作为返回值
* 不支持组合类型作为返回类型
* 不支持out参数

**示例**

* 创建测试表。

create table at\_tb2(id int, val varchar(64));

* 支持声明为自治事务的存储过程。

1、创建存储过程

create or replace procedure at\_test3(i int)  
AS  
DECLARE  
 PRAGMA AUTONOMOUS\_TRANSACTION;  
BEGIN  
 insert into at\_tb2 values(1, 'before s1');  
 insert into at\_tb2 values(2, 'after s1');  
 if i > 10 then  
 rollback;  
 else  
 commit;  
 end if;  
end;  
/

如果数据库未开启默认自动提交，则需要手动提交。

commit;

2、调用存储过程。

call at\_test3(6);

3、查询结果。

select \* from at\_tb2;

当结果显示如下信息，则说明写入成功。

id | val  
----+-----------  
 1 | before s1  
 2 | after s1  
(2 rows)

* 支持plpgsql块内的sql语句使用函数/存储过程参数和定义的变量

1、创建存储过程。

create or replace procedure at\_proc(i int)  
as  
declare  
strval varchar;  
pragma autonomous\_transaction;  
begin  
strval:= 'begin:insert by autonomous transaction procedure<at\_proc>';  
insert into at\_tb2 values(i,strval);  
commit;  
end;  
/

如果数据库未开启默认自动提交，则需要手动提交。

commit;

2、调用存储过程。

call at\_proc(1);

3、查询结果。

select \* from at\_tb2;

当结果显示如下信息，则说明验证成功。

id | val  
----+-----------  
 1 | before s1  
 2 | after s1  
 3 | begin:insert by autonomous transaction procedure<at\_proc>  
(3 rows)

* 支持在触发器函数内声明自治事务

1、创建源表及触发表。

create table test\_trigger\_src\_tbl(id1 int, id2 int, id3 int);  
create table test\_trigger\_des\_tbl(id1 int, id2 int, id3 int);

2、创建触发器函数。

create or replace function tri\_insert\_func() returns trigger as  
$$  
declare  
pragma autonomous\_transaction;  
begin  
insert into test\_trigger\_des\_tbl values(new.id1, new.id2, new.id3);  
return new;  
end  
$$ language plpgsql;

3、创建insert触发器。

create trigger insert\_trigger  
before insert on test\_trigger\_src\_tbl  
for each row  
execute procedure tri\_insert\_func();

4、执行insert触发事件。

insert into test\_trigger\_src\_tbl values(100,200,300);

5、检查触发结果。

select \* from test\_trigger\_src\_tbl;

结果显示如下：

id1 | id2 | id3   
-----+-----+-----  
 100 | 200 | 300  
(1 row)

6、查看触发操作是否生效。

select \* from test\_trigger\_des\_tbl;

结果显示如下：

id1 | id2 | id3   
-----+-----+-----  
 100 | 200 | 300  
(1 row)

* 支持在匿名块内声明自治事务

1、清空测试表数据。

truncate at\_tb2;

2、匿名块声明自治事务。

declare  
pragma autonomous\_transaction;  
strval varchar;  
begin strval:='insert by autonomous transaction by anonymous block';  
insert into at\_tb2 values(1,strval);  
commit;  
end;  
/

3、查询结果。

select \* from at\_tb2;

结果显示如下：

id | val  
1 | insert by autonomous transaction by anonymous block  
(1 row)

* 支持自治事务嵌套使用

1、创建嵌套自治事务的存储过程。

create or replace procedure at\_proc\_inner(i int)  
as  
declare  
pragma autonomous\_transaction;  
begin  
insert into at\_tb2 values(1,'insert by autonomous transaction procedure<at\_proc\_inner>');  
if i>5 then  
rollback;  
else  
commit;  
end if;  
end;  
/

create or replace procedure at\_proc\_outer(i int)  
as  
declare  
pragma autonomous\_transaction;  
begin  
insert into at\_tb2 values(1,'insert by autonomous transaction procedure<at\_proc\_outer>');  
at\_proc\_inner(i);  
if i>10 then rollback;  
else  
commit;  
end if;  
end;  
/

2、调用存储过程。

* 调用存储过程（入参为11）

truncate at\_tb2;  
call at\_proc\_outer(11);

验证结果：

select \* from at\_tb2;

结果显示如下：

----+-----  
(0 rows)

* 调用存储过程（入参为8）

truncate at\_tb2;  
call at\_proc\_outer(8);

验证结果：

select \* from at\_tb2;

结果返回如下：

id | val   
----+-----------------------------------------------------------  
 1 | insert by autonomous transaction procedure<at\_proc\_outer>

* 调用存储过程（入参为2）

truncate at\_tb2;  
call at\_proc\_outer(2);

验证结果：

select \* from at\_tb2;

结果返回如下：

id | val   
----+-----------------------------------------------------------  
 1 | insert by autonomous transaction procedure<at\_proc\_outer>  
 1 | insert by autonomous transaction procedure<at\_proc\_inner>  
(2 rows)

# 插件管理

## PostGIS插件

**功能描述**

PostGIS是在对象关系型数据库PostgreSQL上增加了存储管理空间数据的能力的开源GIS数据库，它在PostgreSQL的基础上增加了表达地理信息的空间数据类型和操作这些类型的函数。

Vastbase G100 V2.2版本可使用PostGIS的如下模块：

* 支持矢量分析的postgis模块。
* 支持计算几何的sfcgal模块。
* 支持栅格分析的raster模块。
* 支持拓扑分析的topology模块。
* 支持扩展组件如下：
* Fuzzystrmatch扩展，是一个判断字符串之间相似性和距离的一个插件，主要是提供Soundex、Levenshtein、Metaphone和Double Metaphone四个功能，开源PostgreSQL的contrib中包含此扩展。PostGIS中的postgis\_tiger\_geocoder依赖此扩展。
* postgis\_tiger\_geocoder扩展，TIGER指的是拓扑集成地理编码和参考，这个扩展提供了TIGER数据的地理编码支持。注意，此扩展启用前，需要先启用fuzzystrmatch（字符串模糊查询），以及可选的address\_standardizer、address\_standardizer\_data\_us。
* address\_standardizer扩展， TIGER数据地址规则化，是基于PAGC标准的地名标准化的扩展。
* address\_standardizer\_data\_us扩展，TIGER地址规则化示例数据集，包含了基于PAGC标准的地名标准化的美国样例数据。

**注意事项**

* Vastbase创建函数不支持添加WINDOW关键字，所以不支持函数ST\_ClusterDBSCAN和ST\_ClusterKMeans。
* Vastbase不支持BRIN 索引，所以不支持geog\_brin\_inclusion\_add\_value和geom2d\_brin\_inclusion\_add\_value等相关函数。
* Vastbase不支持WITH ORDINALITY功能，所以topology.ValidateTopology提供的功能与PostGIS 3.2版本不一致。
* 不支持PostGIS 3.2版本中的ST\_AsGeoJson函数。

**示例**

**示例1：**安装PostGIS插件。

1、获取postgis插件。

2、解压得到postgis文件夹，将其中的内容复制到GAUSSHOME中（GAUSSHOME为数据库的安装路径）。

cp -r postgis/\* $GAUSSHOME

3、yum安装相关依赖（如果不能使用yum可下载对应版本的安装包进行安装）。

yum install -y gmp gmp-devel （版本6.0.0）  
yum install -y mpfr mpfr-devel （版本3.1.1）  
yum install -y boost boost-devel （版本1.53.0）

3.启动数据库并设置guc参数behavior\_compat\_options（PGDATA为数据库的实例路径）。

vb\_ctl start  
vb\_guc reload -D $PGDATA -c " behavior\_compat\_options = 'bind\_procedure\_searchpath, display\_leading\_zero' "

4.重启并连接数据库（可能需要指定路径和端口）。

vb\_ctl restart  
vsql vastbase -r

5、创建插件。

create extension postgis;  
create extension postgis\_sfcgal;  
create extension postgis\_raster;  
create extension postgis\_topology;

6、查询测试。

SELECT Box3D(ST\_GeomFromEWKT('LINESTRING(1 2 3, 3 4 5, 5 6 5)'));

结果返回如下：

Box3d  
---------  
BOX3D(1 2 3,5 6 5)

7、删除插件

drop extension postgis cascade;

**示例2：**创建address\_standardizer\_data\_us扩展。

1、创建扩展（前提条件：已创建PostGIS插件）。

create extension address\_standardizer\_data\_us;

2、查询是否安装成功。

\dx

**示例3：**创建fuzzystrmatch和postgis\_tiger\_geocoder扩展。

1、创建扩展（前提条件：已创建PostGIS插件）。

create extension fuzzystrmatch;  
create extension postgis\_tiger\_geocoder;

2、查看已创建扩展。

\dx

**示例4：**创建address\_standardizer扩展

1、创建扩展（前提条件：已创建PostGIS插件）。

create extension address\_standardizer;

2、查询address\_standardizer信息。

\dx address\_standardizer

## pg\_zhtrgm插件

**功能描述**

创建扩展pg\_zhtrgm，其主要作用是做相似度匹配，可以简化全文索引的使用方式，并且该扩展不仅支持对英文字符全文检索，还允许对中文字符进行全文检索。

**注意事项**

* 用户需要在当前数据库中具有create权限，即可在当前数据库创建插件pg\_zhtrgm。
* 若未使用安装程序安装Vastbase，为自己初始化实例，initdb时指定的 -E 和--locale参数对应的编码规则必须一致，否则将无法使用pg\_zhtrgm插件。

**操作说明**

此类全文索引仅支持模糊查询，不支持等值匹配。对应的操作符如下所示：

| **操作符** | **返回类型** | **描述** |
| --- | --- | --- |
| text ％ text | boolean | 如果其参数的相似度大于pg\_trgm.similarity\_threshold设置的当前相似度阈值，则返回true。 |
| text <％ text | boolean | 如果第一个参数的子集与第二个参数的有序集合的相似程度大于系统设定值，则返回true。当前的相似性阈值通过pg\_trgm.word\_similarity\_threshold参数设定。 |
| text ％> text | boolean | <％运算符的换向器。 |
| text <-> text | real | 返回参数之间的“距离”，即1减去similarity()值。 |
| text <<-> text | real | 返回参数之间的“距离”，即1减去word\_similarity()值。 |
| text <->> text | real | <<->运算符的换向器。 |

pg\_zhgrtm支持使用函数。对应的函数如下所示

| **函数** | **返回类型** | **描述** |
| --- | --- | --- |
| similarity(text,text) | real | 返回一个数字指示两个参数有多相似。该结果的范围是0（指示两个字符串完全不相似）到1（指示两个字符串完全一样）。 |
| show\_trgm(text) | text | 返回一个给定字符串中所有的trigram组成的一个数组（实际上除了调试很少有用）。 |
| show\_limit() | real | 返回%操作符使用的当前相似性阈值。 |
| set\_limit(real) | real | 设置%操作符使用的当前相似性阈值。 |

**示例**

1、创建插件

create extension pg\_zhtrgm;

2、创建中文数据表如下：

create table t\_full\_text(id int, info text);  
insert into t\_full\_text values (1, ' Vastbase是一种特性非常齐全的自由软件的对象-关系型数据库管理系统（ORDBMS），Vastbase支持大部分的SQL标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多版本并发控制等。同样，PostgreSQL也可以用许多方法扩展，例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、修改和分发PostgreSQL。');

3、在此表上创建以pg\_zhgrtm提供的gin\_zhtrgm\_ops、gist\_zhtrgm\_ops操作符类所支持的gin索引或gist索引，该索引创建不再需要借助to\_tsvector函数，同样地，使用该索引进行查询时也不再需要借助to\_tsquery函数。

create index idx\_full\_text1 on t\_full\_text using gin(info gin\_zhtrgm\_ops);  
create index idx\_full\_text2 on t\_full\_text using gist(info gist\_zhtrgm\_ops);

4、可以进行单字、词语、条件、短句的全文搜索：

* 使用单字搜索。

select \* from t\_full\_text where info like '%函%';

结果显示如下：

id |   
 info   
----+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
----------------------  
 1 | Vastbase是一种特性非常齐全的自由软件的对象-关系型数据库管理系统（ORDBMS），Vastbase支持大部分的SQL标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多  
版本并发控制等。同样，PostgreSQL也可以用许多方法扩展，例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、  
改和分发PostgreSQL。  
(1 row)

* 使用词语搜索。

select \* from t\_full\_text where info like '%视图%';

结果显示如下：

id |   
 info   
----+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
----------------------  
 1 | Vastbase是一种特性非常齐全的自由软件的对象-关系型数据库管理系统（ORDBMS），Vastbase支持大部分的SQL标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多  
版本并发控制等。同样，PostgreSQL也可以用许多方法扩展，例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、  
改和分发PostgreSQL。  
(1 row)

* not like条件搜索。

select \* from t\_full\_text where info like '%查询%' and info not like '%外键%';

结果显示如下：

id | info   
----+------  
(0 rows)

* 使用语句搜索

select \* from t\_full\_text where info like '%Vastbase是一种特性非常齐全的自由软件的对象%';

结果显示如下：

id |   
 info   
----+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
----------------------  
 1 | Vastbase是一种特性非常齐全的自由软件的对象-关系型数据库管理系统（ORDBMS），Vastbase支持大部分的SQL标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多  
版本并发控制等。同样，PostgreSQL也可以用许多方法扩展，例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、  
改和分发PostgreSQL。  
(1 row)

## PGroonga插件

**功能描述**

PGroonga是一个全文检索插件，Vastbase G100已经内置了 PGroonga扩展，您只需要创建扩展，即可使用PGroonga支持的语言进行超高速全文检索。

**语法格式**

1、使用pgroonga()创建单字段全文索引。

CREATE INDEX pgroonga\_index ON memos USING pgroonga(content1）;

* 使用&@操作符进行全文检索。

SELECT \* FROM memos WHERE content &@ '全文搜索关键字';

* 使用&@~操作符执行含pgroonga查询语法的全文检索。

SELECT \* FROM memos WHERE content2 &@~ 'PGroonga OR PostgreSQL';

* 使用like操作符（数据库本身的一种查询语法，区分大小写）执行全文检索。

SELECT \* FROM memos WHERE content2 LIKE '%全文搜索关键字%';

* 使用ilike操作符（数据库本身的一种查询语法，不区分大小写）执行全文检索。

SELECT \* FROM memos WHERE content2 ILIKE '%全文搜索关键字%';

2、使用pgroonga()创建多字段全文索引。

CREATE INDEX pgroonga\_index\_1 ON memos USING pgroonga(content1, content2);

* 使用&@操作符进行多字段全文检索。

SELECT \* FROM memos WHERE content1 &@ '全文搜索关键字' and content2 &@ '全文搜索关键字';

* 使用like操作符（数据库本身的一种查询语法，区分大小写）执行多字段全文检索。

SELECT \* FROM memos WHERE content1 like '%全文搜索关键字%' and content2 like '%全文搜索关键字%';

* 使用ilike操作符（数据库本身的一种查询语法，不区分大小写）执行多字段全文检索。

SELECT \* FROM memos WHERE content1 ilike '%全文搜索关键字%' and content2 ilike '%全文搜索关键字%';

使用pgroonga\_vacuum()函数清理临时数据。

select pgroonga\_vacuum();

**示例**

1、创建插件。

修改postgresql.conf中配置enable\_pgroonga=true。

CREATE EXTENSION pgroonga;

2、关闭全表扫描，为了在少量数据下体现索引效果。

set enable\_seqscan=off;

3、创建表，并插入数据。

CREATE TABLE memos (  
id integer,  
content text  
);  
INSERT INTO memos VALUES (1, 'PostgreSQL is a relational database management system.');  
INSERT INTO memos VALUES (2, 'Groonga is a fast full text search engine that supports all languages.');  
INSERT INTO memos VALUES (3, 'PGroonga is a PostgreSQL extension that uses Groonga as index.');  
INSERT INTO memos VALUES (4, 'There is groonga command.');

4、创建全文索引。

CREATE INDEX idxA ON memos USING pgroonga (content pgroonga\_text\_full\_text\_search\_ops\_v2);

查看执行计划。

explain select \* from memos where content LIKE '%groonga%';

结果显示如下：

QUERY PLAN   
-------------------------------------------------------------------  
 Index Scan using idxa on memos (cost=0.00..4.01 rows=1 width=36)  
 Index Cond: (content ~~ '%groonga%'::text)  
(2 rows)

5、查看表中内容。

select \* from memos where content LIKE '%groonga%';

结果显示如下：

id | content   
----+---------------------------  
 4 | There is groonga command.  
(1 row)

使用ILIKE查询。

select \* from memos where content ILIKE '%roonga%';

结果显示如下：

id | content   
----+---------  
(0 rows)

6、查询验证。

* 使用&@操作符。

SELECT \* FROM memos WHERE content &@ 'languages';

结果显示如下：

id | content  
----+------------------------------------------------------------------------  
2 | Groonga is a fast full text search engine that supports all languages.  
(1 row)

* 使用&@~操作符。

SELECT \* FROM memos WHERE content &@~ 'PGroonga OR PostgreSQL';

结果显示如下：

id | content  
----+----------------------------------------------------------------  
3 | PGroonga is a PostgreSQL extension that uses Groonga as index.  
1 | PostgreSQL is a relational database management system.  
(2 rows)

* 使用like操作符。

SELECT \* FROM memos WHERE content LIKE '%groonga%';

结果显示如下：

id | content  
----+---------------------------  
4 | There is groonga command.  
(1 row)

* 使用ilike操作符。

SELECT \* FROM memos WHERE content ILIKE '%groonga%';

结果显示如下：

id | content   
----+------------------------------------------------------------------------  
 2 | Groonga is a fast full text search engine that supports all languages.  
 3 | PGroonga is a PostgreSQL extension that uses Groonga as index.  
 4 | There is groonga command.  
(3 rows)

## uuid-ossp插件

**功能描述**

uuid-ossp模块提供了一些函数用来生成通用唯一识别码（UUID），兼容uuid-ossp1.1，使用该模块提供的函数，采用几种标准算法之一产生通用唯一标识符（UUID）。

兼容的uuid-ossp模块的函数分为如下两类：

* 用于产生uuid的函数：

| **函数** | **描述** |
| --- | --- |
| uuid\_generate\_v1() | 这个函数产生一个版本 1 的 UUID。这涉及到计算机的 MAC 地址和一个时间戳。注：这种 UUID 会泄露产生该标识符的计算机标识以及产生的时间，因此它不适合某些对安全性很敏感的应用。 |
| uuid\_generate\_v1mc() | 这个函数产生一个版本 1 的 UUID，但是使用一个随机广播 MAC 地址而不是该计算机真实的 MAC 地址。 |
| uuid\_generate\_v3(namespace uuid, name text) | 这个函数使用指定的输入名称在给定名字的空间中产生一个版本 3 的 UUID。该名字空间应该是由uuid\_ns\_\*()函数产生的特殊常量之一（理论上它可以是任意 UUID）。名称是选择的名字空间中的一个标识符。例如：vastbase=# SELECT uuid\_generate\_v3(uuid\_ns\_url(), '<http://www.postgresql.org>'); 名称参数将使用 MD5 进行哈希，因此从产生的 UUID 中得不到明文。采用这种方法的 UUID 生成没有随机性并且不涉及依赖于环境的元素，因此是可以重现的。 |
| uuid\_generate\_v4() | 这个函数产生一个版本 4 的 UUID，它完全从随机数产生。 |
| uuid\_generate\_v5(namespace uuid, name text) | 这个函数产生一个版本 5 的 UUID，它和版本 3 的 UUID 相似，但是采用的是 SHA-1 作为哈希方法。注：版本 5 比版本 3 更好，因为 SHA-1 比 MD5 更安全。 |

* 返回 uuid常量的函数：

| **函数** | **描述** |
| --- | --- |
| uuid\_nil() | 一个"nil" UUID 常量，它不作为一个真正的 UUID 发生。 |
| uuid\_ns\_dns() | 为 UUID 指定 DNS 名字空间的常量。 |
| uuid\_ns\_url() | 为 UUID 指定 URL 名字空间的常量。 |
| uuid\_ns\_oid() | 为 UUID 指定 ISO 对象标识符（OID） 名字空间的常量（这属于 ASN.1 OID，它与PostgreSQL使用的 OID 无关）。 |
| uuid\_ns\_x500() | 为 UUID 指定 X.500 可识别名（DN）名字空间的常量。Constant designating the X.500 distinguished name (DN) namespace for UUIDs。 |

**注意事项**

* Vastbase G100 V2.2版本兼容uuid-ossp插件目前仅兼容ossp，对bsd和e2fs不做硬性要求。
* Vastbase G100 V2.2已经内置了uuid-ossp扩展，您只需要创建扩展即可使用。

**示例**

示例一：

1、创建uuid-ossp扩展。

create extension "uuid-ossp";

2、使用产生uuid的函数。

* 测试uuid\_generate\_v1()。

select uuid\_generate\_v1();

结果显示如下：

uuid\_generate\_v1  
--------------------------------------  
 bdf064b8-baef-11ec-8838-000c2948c478  
 (1 row)

* 测试uuid\_generate\_v1mc()。

select uuid\_generate\_v1mc();

结果显示如下：

uuid\_generate\_v1mc  
--------------------------------------  
 ce8693b0-baef-11ec-8839-83581df53531  
(1 row)

* 测试uuid\_generate\_v3(namespace uuid, name text)。

SELECT uuid\_generate\_v3(uuid\_ns\_url(),'http://www.postgresql.org');

结果显示如下：

uuid\_generate\_v3  
--------------------------------------  
 cf16fe52-3365-3a1f-8572-288d8d2aaa46  
(1 row)

* 测试uuid\_generate\_v4()。

select uuid\_generate\_v4();

结果显示如下：

uuid\_generate\_v4  
--------------------------------------  
 3cad2542-a2a5-43ed-9df4-0dacb2f1d52f  
(1 row)

* 测试uuid\_generate\_v5(namespace uuid, name text)。

SELECT uuid\_generate\_v5(uuid\_ns\_url(),'http://www.postgresql.org');

结果显示如下：

uuid\_generate\_v5  
--------------------------------------  
 e1ee1ad4-cd4e-5889-962a-4f605a68d94e  
(1 row)

示例二：返回uuid常量的函数

* 测试uuid\_nil()函数。

select uuid\_nil();

结果显示如下：

uuid\_nil  
--------------------------------------  
 00000000-0000-0000-0000-000000000000  
(1 row)

* 测试uuid\_ns\_dns()函数。

select uuid\_ns\_dns();

结果显示如下：

uuid\_ns\_dns  
--------------------------------------  
 6ba7b810-9dad-11d1-80b4-00c04fd430c8  
(1 row)

* 测试uuid\_ns\_url()函数。

select uuid\_ns\_url();

结果显示如下：

uuid\_ns\_url  
--------------------------------------  
 6ba7b811-9dad-11d1-80b4-00c04fd430c8  
(1 row)

* 测试uuid\_ns\_oid()函数。

select uuid\_ns\_oid();

结果显示如下：

uuid\_ns\_oid  
--------------------------------------  
 6ba7b812-9dad-11d1-80b4-00c04fd430c8  
(1 row)

* 测试uuid\_ns\_x500()函数。

select uuid\_ns\_x500();

结果显示如下：

uuid\_ns\_x500  
--------------------------------------  
 6ba7b814-9dad-11d1-80b4-00c04fd430c8  
(1 row)

## wal2json插件

**功能描述**

wal2json是逻辑解码插件，使用该插件可以访问INSERT和UPDATE生成的元组，解析WAL中的内容。wal2json插件会在每个事务中生成一个JSON对象。JSON对象总提供了所有新/旧元组，额外选项还可以包括事务时间戳，限定架构，数据类型，事务ID等属性。

wal2json支持的参数列表如下：

| **选项** | **描述** |
| --- | --- |
| include-xids | 添加xid 信息，默认false |
| include-timestamp | 添加 timestamp信息，默认false |
| include-schemas | 添加schemas信息，默认true |
| include-types | 添加type信息，默认true |
| include-typmod | 添加 typmod 信息，默认true |
| include-type-oids | 添加 type oids信息，默认false |
| include-domain-data-type | 将域名替换成数据类型名，默认false |
| include-column-positions | 添加 column position 信息，默认false |
| include-origin | 添加origin信息，默认false |
| include-not-null | 添加not nul1信息，默认false |
| include-default | 添加默认值表达式，默认为false |
| include-pk | 添加主键信息，默认为false |
| pretty-print | 更好的输出ison结构，默认为false |
| include-lsn | 添加nextlsn信息，默认为false |
| write-in-chunks | 在每次更改后写入，而不是在每次更改集之后写入。只在format-version为1的时候有用，默认为 false |
| filter-origins | 排除指定origin的更改，默认为空不过滤，多个 origin用逗号分隔 |
| filter-tables | 排除指定table的更改，默认为空不过滤，多个table用逗号分隔 |
| add\_tables | 仅包含指定table的更改，默认值是所有的表，与filter-tables值具有相同的规则 |
| format-version | 格式化方式，可选值为1和2，默认为1 |
| actions | 解析的actions，默认是所有的action（insert，update，delete） |

**注意事项**

* GUC参数wal\_level需要设置为logical。
* 设置合适的max\_wal\_senders和max\_replication\_slots值。
* 暂不支持truncate和message操作。

**使用流程**

使用逻辑复制槽和解码插件解析数据库中wal日志记录的操作流程如下：

1、修改配置文件相关参数。

2、创建逻辑复制槽。

3、执行SQL语句。

4、解析wal内容。

该流程中涉及的函数请参见：SQL语法参考->函数和操作符->逻辑复制函数章节。

**示例**

1、修改相关参数（修改后需重启数据库实例）。

ALTER SYSTEM SET wal\_level to logical;  
ALTER SYSTEM SET max\_wal\_senders to 10;  
ALTER SYSTEM SET max\_replication\_slots to 10;

2、创建逻辑复制槽。

select 'init' from pg\_create\_logical\_replication\_slot('regression\_slot\_1098463','wal2json');

3、执行SQL语句。

create table tb\_1098463(col1 int ,col2 bigint default '1',col3 text default 'a',col4 int ) ;  
​  
insert into tb\_1098463 values(generate\_series(1,3),1,'a');  
insert into tb\_1098463(col1) values(4);  
insert into tb\_1098463 values(5,4,'a');  
insert into tb\_1098463 select \* from tb\_1098463;

4、解析wal内容。

select data from pg\_logical\_slot\_peek\_changes('regression\_slot\_1098463',null,null,'actions','insert');

返回结果为：

                                      data  
---------------------------------------------------------------------------------------------------------------------------  
 {"change":[]}  
 {"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntype  
s":["integer","bigint","text","integer"],"columnvalues":[1,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098  
463","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnvalues":[2,1,"a  
",null]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":  
["integer","bigint","text","integer"],"columnvalues":[3,1,"a",null]}]}  
 {"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntype  
s":["integer","bigint","text","integer"],"columnvalues":[4,1,"a",null]}]}  
 {"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntype  
s":["integer","bigint","text","integer"],"columnvalues":[5,4,"a",null]}]}  
 {"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntype  
s":["integer","bigint","text","integer"],"columnvalues":[1,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098  
463","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnvalues":[2,1,"a  
",null]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":  
["integer","bigint","text","integer"],"columnvalues":[3,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098463  
","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnvalues":[4,1,"a",n  
ull]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":["i  
nteger","bigint","text","integer"],"columnvalues":[5,4,"a",null]}]}  
(5 rows)  
​

5、解析wal内容添加xid（include-xids选项测试）。

select data from pg\_logical\_slot\_peek\_changes('regression\_slot\_1098463',null,null,'include-xids','true');

返回结果为：

                              data  
---------------------------------------------------------------------------------------------------------------------------  
​  
 {"xid":15504,"change":[]}  
 {"xid":15505,"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"]  
,"columntypes":["integer","bigint","text","integer"],"columnvalues":[1,1,"a",null]},{"kind":"insert","schema":"public","tab  
le":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnval  
ues":[2,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"c  
olumntypes":["integer","bigint","text","integer"],"columnvalues":[3,1,"a",null]}]}  
 {"xid":15506,"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"]  
,"columntypes":["integer","bigint","text","integer"],"columnvalues":[4,1,"a",null]}]}  
 {"xid":15507,"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"]  
,"columntypes":["integer","bigint","text","integer"],"columnvalues":[5,4,"a",null]}]}  
 {"xid":15508,"change":[{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"]  
,"columntypes":["integer","bigint","text","integer"],"columnvalues":[1,1,"a",null]},{"kind":"insert","schema":"public","tab  
le":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnval  
ues":[2,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"c  
olumntypes":["integer","bigint","text","integer"],"columnvalues":[3,1,"a",null]},{"kind":"insert","schema":"public","table"  
:"tb\_1098463","columnnames":["col1","col2","col3","col4"],"columntypes":["integer","bigint","text","integer"],"columnvalues  
":[4,1,"a",null]},{"kind":"insert","schema":"public","table":"tb\_1098463","columnnames":["col1","col2","col3","col4"],"colu  
mntypes":["integer","bigint","text","integer"],"columnvalues":[5,4,"a",null]}]}  
(5 rows)

6、删除逻辑复制槽。

select 'stop' from pg\_drop\_replication\_slot('regression\_slot\_1098463');  
drop table tb\_1098463;

# 外部数据封装器

## Oracle\_FDW

**功能描述**

支持创建外部数据封装器oracle\_fdw，连接Oracle数据库，并能在外部表上进行查询、插入、更新和删除操作。

**编译Oracle\_FDW**

编译oracle\_fdw需要安装Oracle的开发库和头文件。

1、选择合适的运行环境和版本，下载Basic Package和SDK Package并安装。另外SQLPlus Package是Oracle的客户端工具，也可以根据需要安装，用于连接Oracle Server进行测试。

2、安装好开发包后，就可以开始编译oracle\_fdw了。编译时需要在执行configure时，加入 –enable-oracle-fdw选项。后续按照正常的Vstbase安装方式安装即可。（Vastbase的编译参考章节"字符安装"）

3、编译完成后，编译产物为oracle\_fdw.so，位于安装目录的 lib/postgresql/下。oracle\_fdw相关的sql文件和control文件，位于安装目录的 share/postgresql/extension/下。

4、如果编译安装时，没有加入 –enable-oracle-fdw选项，可以在Vastbase安装完成后，再次编译oracle\_fdw，然后手动将编译产物 oracle\_fdw.so放到对应的安装目录 lib/postgresql/ ，将 oracle\_fdw–1.0–1.1.sql、oracle\_fdw–1.1.sql、oracle\_fdw.control放到对应的安装目录 share/postgresql/extension/即可。

**使用Oracle\_FDW**

* 使用oracle\_fdw需要连接Oracle，Oracle server请自行安装。
* 加载oracle\_fdw扩展：

CREATE EXTENSION oracle\_fdw

* 创建服务器对象：

CREATE SERVER

* 创建用户映射：

CREATE USER MAPPING

* 创建外表： 外表的表结构需要与Oracle数据库中的表结构保持一致。注意Oracle server侧的表的第一个字段必须具有唯一性约束（如PRIMARY KEY、UNIQUE等）。

CREATE FOREIGN TABLE

* 对外表做正常的操作，如INSERT、UPDATE、DELETE、SELECT、EXPLAIN、ANALYZE、COPY等。
* 删除外表：

DROP FOREIGN TABLE

* 删除用户映射：

DROP USER MAPPING

* 删除服务器对象：

DROP SERVER

* 删除扩展：

DROP EXTENSION oracle\_fdw

**注意事项**

* 两个Oracle外表间的SELECT JOIN不支持下推到Oracle server执行，会被分成两条SQL语句传递到Oracle执行，然后在Vastbase处汇总处理结果。
* 不支持IMPORT FOREIGN SCHEMA语法。
* 不支持对外表进行CREATE TRIGGER操作。

**示例**

1、创建扩展。

create extension oracle\_fdw;

2、创建FDW Server用于识别外部数据源。

create server ora\_fdw\_server foreign data wrapper oracle\_fdw  
options(dbserver '172.16.103.104:1521/orcl');

3、新建用户并授权。

create user use\_ora password 'Bigdata@123';  
grant usage on foreign server ora\_fdw\_server to use\_ora;

4、创建USER MAPPING用于映射Oracle数据库用户。

create user mapping for use\_ora server ora\_fdw\_server options(user 'system',password 'root');

5、在oracle创建表。

create table system.emp\_fdw(empno int,ename varchar(30));  
insert into emp\_fdw values(1,'foo');  
insert into emp\_fdw values(2,'bar');

6、在vastbase创建外部表，切换到用户use\_ora。

create foreign table emp\_fdw\_ora(empno int,ename varchar(30))  
server ora\_fdw\_server  
options(schema 'SYSTEM',table 'EMP\_FDW');  
\c - use\_ora

7、插入外部表数据。

insert into emp\_fdw\_ora values(3,'bar3');

8、查询外部表数据。

select \* from emp\_fdw\_ora;

## DBLINK

**功能描述**

当用户需要跨越本地数据库，访问远程数据库的数据时，可以通过DBLINK像访问本地数据库一样访问远程数据库表中的数据。

**语法格式**

* 打开一个到远程数据库的持久连接。

SELECT dblink\_connect(dblink\_name,connstr);

* 关闭一个到远程数据库的持久连接。

SELECT dblink\_disconnect();

* 在远程数据库执行查询。

SELECT \* FROM dblink(dblink\_name, sql\_statement);

* 在远程数据库执行命令。

SELECT dblink\_exec(dblink\_name, sql\_statement);

* 返回所有打开的命名dblink连接的名称。

SELECT dblink\_get\_connections();

* 发送一个异步查询到远程数据库。

SELECT dblink\_send\_query(dblink\_name, sql\_statement);

* 检查连接是否正在忙于一个异步查询。

SELECT dblink\_is\_busy(dblink\_name);

* 创建private dblink或public dblink。

CREATE [public] DATABASE LINK ${dblink\_name} CONNECT TO ${username} IDENTIFIED BY '${password}' USING ${fdw\_name} (  
url 'jdbc:oracle:thin:@//${ip}:${port}/${dbname}',  
jarfile '${driver\_path}'  
);

* 远程Oracle函数和存储过程。

${schema\_name}.${function\_name | procedure\_name}@${dblink\_name}[(parameter,[...])]

**参数说明**

* dblink\_name：连接名称，可自定义。
* connstr：连接信息，例如：hostaddr=172.16.103.92 port=6036 dbname=vastbase user=vastbase password='Bigdata@123'
* sql\_statement：执行的语句。
* ${schema\_name}：目标函数或存储过程所在的模式名（或者package名称）。可不指定，不指定时默认访问database link中Oracle用户对应的模式。
* ${function\_name}：函数名称。
* ${procedure\_name}：存储过程名称。
* parameter：函数或存储过程中的参数，只考虑传入参数，不考虑out输出参数。

**示例**

* **示例1：**常用函数操作

1、创建扩展。

create extension dblink;

2、先执行dblink\_connect保持连接。

SELECT dblink\_connect('mycoon','hostaddr=172.16.103.92 port=6036 dbname=vastbase user=lst password=Bigdata@123');

3、执行BEGIN命令。

SELECT dblink\_exec('mycoon', 'BEGIN');

4、执行数据操作。

SELECT dblink\_exec('mycoon', 'create table people(id int,info varchar(10))');  
SELECT dblink\_exec('mycoon', 'insert into people values(1,''foo'')');  
SELECT dblink\_exec('mycoon', 'insert into people values(2,''foo'')');  
SELECT dblink\_exec('mycoon', 'update people set info=''bar'' where id=1');

5、执行事务提交。

SELECT dblink\_exec('mycoon', 'COMMIT');

6、执行查询。

select \* from dblink('mycoon','select \* from people') as testTable (id int,info varchar(10));

7、解除连接。

SELECT dblink\_disconnect('mycoon')

* **示例2：**private dblink和public dblink

1、修改postgresql.conf参数，将jdbc\_fdw配置到shared\_preload\_libraries参数中。

shared\_preload\_libraries = 'jdbc\_fdw'

2、创建插件。

create extension jdbc\_fdw;

3、创建dblink。

（1）创建私有private dblink。

create database link oradb1 connect to SYSTEM identified by 'root'   
using jdbc\_fdw(  
url 'jdbc:oracle:thin:@//172.16.103.120:1521/utf8',  
jarfile '/home/vastbase2/ojdbc7.jar'  
);

（2）创建公共public dblink。

vastbase=# create public database link oradb2 connect to SYSTEM identified by 'root'   
using jdbc\_fdw(  
url 'jdbc:oracle:thin:@//172.16.103.104:1521/orcl',  
jarfile '/home/vastbase2/ojdbc7.jar'  
);

4、dblink使用查询。

select \* from fdw\_t1@oradb1;  
select \* from fdw\_t1@oradb2;

5、ALTER SERVER。

ALTER SERVER oradb1 VERSION '8.4';  
ALTER PUBLIC SERVER oradb2 VERSION '8.5';

6、删除DBLINK。

DROP DATABASE LINK oradb1;  
DROP PUBLIC DATABASE LINK oradb2;

## JDBC\_FDW

**功能描述**  
支持基于JDBC驱动的FDW，可以通过不同数据源的JDBC驱动在外部表访问数据。

**前置条件**

1、安装JDK1.8，并配置好JAVA\_HOME环境变量。

2、数据库环境变量LD\_LIBRARY\_PATH动态库路径加入JAVA\_HOME配置。

通过如下命令配置环境变量：

x86：export LD\_LIBRARY\_PATH=$GAUSSHOME/jre/lib/amd64/server/:$LD\_LIBRARY\_PATH  
ARM：export LD\_LIBRARY\_PATH=$GAUSSHOME/jre/lib/aarch64/server/:$LD\_LIBRARY\_PATH

3、修改数据库配置参数：在shared\_preload\_libraries参数配置中加上jdbc\_fdw，然后重启数据库。

4、驱动包ojdbc7.jar放置到$GAUSSHOME/lib/postgresql/下并执行如下命令授权

chmod 777 $GAUSSHOME/lib/postgresql/ojdbc7.jar

**注意事项**

* 不同的用户可以创建同名的private dblink，且用户只能访问自己创建的DBLINK。
* 同一用户可以创建同名的private dblink和，在访问时，优先访问private dblink。
* 为支持不同用户创建同名dblink，需禁用GRANT FOREIGN SERVER，不再允许将server权限给予其他用户，以防一个用户能访问多个同名server，造成混乱。
* 不支持访问远程Oracle包变量和常量。（注：数据也不支持）
* 不支持将远程函数和存储过程创建为同义词方式使用。
* 创建private dblink或public dblink时，必须提前创建jdbc\_fdw，并具备DBLINK相关权限。
* 支持dblink访问Oracle中数据库的Sequence序列的值。
* 支持通过NEXTVAL获取下一个序列值。
* 支持通过CURRVAL获取当前序列值。
* 支持dblink与synonym结合访问Oracle的存储过程。
* 支持远程调用无参的存储过程。
* 支持调用带输入参数的存储过程。
* 支持调用带out参数的存储过程。
* 传入参数和out参数，支持的数据类型如下：

|  |  |
| --- | --- |
| **数据类型名称** | **数据类型** |
| 数字类型 | NUMBER |
| 变长字符串类型 | VARCHAR2，NVARCHAR2 VARCHAR，NVARCHAR |
| 定长字符串类型 | CHAR，NCHAR |
| 单精度精度二进制浮点类型 | BINARY\_FLOAT |
| 双精度二进制浮点类型 | BINARY\_DOUBLE |
| 二进制数据类型 | RAW |
| 二进制大对象类型 | BLOB |
| 字符大对象类型 | CLOB |
| XML数据类型 | XMLType |
| 日期时间类型 | DATE |
| 时间戳数据类型 | TIMESTAMP， TIMESTAMP WITH TIME ZONE |
| 本地时间戳数据类型 | TIMESTAMP WITH LOCAL TIME ZONE |

* 支持dblink与synonym结合访问Oracle的自定义函数。
* 支持远程调用无参的自定义函数。
* 支持远程调用带输入参数的自定义函数。
* 支持远程调用out参数的自定义函数。
* 支持调用的函数return是%rowtype行数据类型。
* 传入参数和out参数和return函数返回，支持的数据类型如下：

|  |  |
| --- | --- |
| **数据类型名称** | **数据类型** |
| 数字类型 | NUMBER |
| 变长字符串类型 | VARCHAR2，NVARCHAR2 VARCHAR，NVARCHAR |
| 定长字符串类型 | CHAR，NCHAR |
| 单精度精度二进制浮点类型 | BINARY\_FLOAT |
| 双精度二进制浮点类型 | BINARY\_DOUBLE |
| 二进制数据类型 | RAW |
| 二进制大对象类型 | BLOB |
| 字符大对象类型 | CLOB |
| XML数据类型 | XMLType |
| 日期时间类型 | DATE |
| 时间戳数据类型 | TIMESTAMP， TIMESTAMP WITH TIME ZONE |
| 本地时间戳数据类型 | TIMESTAMP WITH LOCAL TIME ZONE |

* 支持dblink与synonym结合访问Oracle的包

**示例**

**示例1：**创建jdbc\_fdw扩展。

1、加载jdbc\_fdw扩展。

create extension jdbc\_fdw;

2、在创建测试表。

create table emp\_fdw(empno int,ename varchar(30));  
insert into emp\_fdw values(1,'foo');  
insert into emp\_fdw values(2,'bar');

3、创建连接。

create server ora\_jdbc foreign data wrapper jdbc\_fdw options(  
drivername 'oracle.jdbc.driver.OracleDriver',  
url 'jdbc:oracle:thin:@//172.16.103.104:1521/orcl',  
querytimeout '100',  
jarfile '/home/vastbase/bin/ojdbc7.jar',  
maxheapsize '200'  
);

4、新建用户并授权。

create user use\_ora\_jdbc password 'Bigdata@123';  
grant usage on foreign server ora\_jdbc to use\_ora\_jdbc;

5、创建到oracle的映射。

create user mapping for use\_ora\_jdbc server ora\_jdbc options(username 'system',password 'root');

6、创建需要访问的oracle中对应表的结构。

create foreign table emp\_fdw\_ora(empno int,ename varchar(30))  
server ora\_jdbc options(table\_name 'EMP\_FDW');

7、将权限赋予emp\_fdw\_ora。

grant all on emp\_fdw\_ora to use\_ora\_jdbc;

8、查看外部表。

\c - use\_ora\_jdbc  
select \* from emp\_fdw\_ora;

当结果显示如下信息，则表示外部表访问成功。

id | info  
----+-----------  
1 | bar  
2 | foo  
(2 row)

**示例2：**远程访问Oracle函数。

1、修改postgresql.conf参数，将jdbc\_fdw配置到shared\_preload\_libraries参数中。

2、驱动包ojdbc7.jar放置到$GAUSSHOME/lib/postgresql/下并修改权限。

chmod 777 $GAUSSHOME/lib/postgresql/ojdbc7.jar

3、在oracle中创建对应的表和存储过程。

create or replace function ora\_func01(param1 in binary\_float,param2 in out binary\_float ,param3 out binary\_float)  
return binary\_float  
as  
v\_num1 binary\_float;  
begin  
v\_num1 :=param1\*param2;  
param2:=2;  
param3:=3;  
return v\_num1;  
end;  
/

4、在Vastbase安装插件。

create extension jdbc\_fdw;

5、在vastbase中创建dblink。

CREATE public DATABASE LINK dblink\_104 CONNECT TO SYSTEM IDENTIFIED BY 'root' USING jdbc\_fdw(  
url 'jdbc:oracle:thin:@//172.16.103.104:1521/orcl',  
jarfile '/home/lmh/local/vastbase/lib/postgresql/ojdbc7.jar'  
);

6、在vastbase中创建同义词进行访问Oracle的带参数的函数。

create public SYNONYM syn\_f01 for ora\_func01@dblink\_104;

7、在Vastbase直接调用函数。

call ora\_func01@dblink\_104(11,2,3);

8、在Vastbase通过同义词调用函数。

call sync\_f01(11,2,3);

## MySQL\_FDW

**功能描述**

支持创建外部数据封装器mysql\_fdw，连接MySQL数据库，并能在外部表上进行查询、插入、更新和删除操作。

**编译MySQL\_FDW**

1、编译mysql\_fdw需要安装MariaDB的开发库和头文件，推荐使用MariaDB的官方源进行安装。

2、配置好源后，使用yum install MariaDB-devel MariaDB-shared安装相关开发库。另外MariaDB-client是MariaDB的客户端工具，也可以根据需要安装，用于连接MariaDB进行测试。

3、安装好开发包后，就可以开始编译mysql\_fdw了。编译时需要在执行configure时，加入–enable-mysql-fdw选项。后续按照正常的Vastbase安装方式安装即可。（Vastbase的安装参考章节“字符安装”）

4、编译完成后，编译产物为mysql\_fdw.so，位于安装目录的lib/postgresql/ 下。mysql\_fdw相关的sql文件和control文件，位于安装目录的share/postgresql/extension/下。

5、如果编译安装时，没有加入–enable-mysql-fdw选项，可以在Vastbase安装完成后，再次编译mysql\_fdw，然后手动将编译产物mysql\_fdw.so放到对应的安装目录lib/postgresql/，将mysql\_fdw–1.0–1.1.sql、mysql\_fdw–1.1.sql、mysql\_fdw–1.0.sql、mysql\_fdw.control放到对应的安装目录share/postgresql/extension/ ，然后重启数据库即可。

**使用MySQL\_FDW**

* 使用mysql\_fdw需要连接MariaDB或者MySQL Server，MariaDB或MySQL Server请自行安装。
* 加载mysql\_fdw扩展。

CREATE EXTENSION mysql\_fdw;

* 创建服务器对象。

CREATE SERVER

* 创建用户映射。

CREATE USER MAPPING

* 创建外表：外表的表结构需要与MySQL/MariaDB侧的表结构保持一致。注意MySQL/MariaDB侧的表的第一个字段必须具有唯一性约束（如PRIMARY KEY、UNIQUE等）。

CREATE FOREIGN TABLE

* 对外表做正常的操作，如INSERT、UPDATE、DELETE、SELECT、EXPLAIN、ANALYZE、COPY等。
* 删除外表。

DROP FOREIGN TABLE

* 删除用户映射。

DROP USER MAPPING

* 删除服务器对象：

DROP SERVER

* 删除扩展。

DROP EXTENSION mysql\_fdw

**注意事项**

* 两个mysql外表间的SELECT JOIN不支持下推到MariaDB/MySQL Server执行，会被分成两条SQL语句传递到MariaDB/MySQL Server执行，然后在Vastbase处汇总处理结果。
* 不支持IMPORT FOREIGN SCHEMA语法。
* 不支持对外表进行CREATE TRIGGER操作。

**示例**

1、创建扩展。

create extension mysql\_fdw;

2、创建FDW Server用于识别外部数据源。

create server mysql\_fdw\_server foreign data wrapper mysql\_fdw options(HOST '172.16.103.108',PORT '20001');

3、新建用户并授权。

create user use\_mysql password 'Bigdata@123';  
grant usage on foreign server mysql\_fdw\_server to use\_mysql;

4、创建USER MAPPING用于映射MySQL数据库用户。

create user mapping for use\_mysql server mysql\_fdw\_server options(username 'root',password 'root');

5、在MySQL创建表。

create table test.emp\_fdw(empno int,ename varchar(30));  
insert into emp\_fdw values(1,'foo');  
insert into emp\_fdw values(2,'bar');

6、在vastbase创建外部表，切换到用户use\_mysql。

create foreign table emp\_fdw\_mysql(empno int,ename varchar(30)) server mysql\_fdw\_server options(DBNAME 'test',table\_name 'emp\_fdw');  
\c - use\_mysql

7、查询外部表数据。

select \* from emp\_fdw\_mysql;

## PostgreSQL\_FDW

**功能描述**

postgres\_fdw功能提供了外部数据封装器postgres\_fdw插件，它可以被用来访问存储在外部Vastbase服务器中的数据。

**语法格式**

1、加载postgres\_fdw扩展

CREATE EXTENSION [ IF NOT EXISTS ] extension\_name  
 [ WITH ] [ SCHEMA schema\_name ]  
 [ VERSION version ]  
 [ FROM old\_version ]

2、创建一个新的外部服务器

CREATE SERVER server\_name  
FOREIGN DATA WRAPPER fdw\_name  
 OPTIONS ( { option\_name ' value ' } [, ...] ) ;

3、创建一个用户到一个外部服务器的新映射

CREATE USER MAPPING FOR { user\_name | USER | CURRENT\_USER | PUBLIC }  
 SERVER server\_name  
[ OPTIONS ( option 'value' [ , ... ] ) ]

4、创建外表

CREATE FOREIGN TABLE [ IF NOT EXISTS ] table\_name ( [  
column\_name type\_name [ OPTIONS ( option 'value' [, ... ] ) ] [ COLLATE collation ] [ column\_constraint [ ... ] ]  
 [, ... ]  
] )  
 SERVER server\_name  
[ OPTIONS ( option 'value' [, ... ] ) ]

这里column\_constraint 可以是:

[ CONSTRAINT constraint\_name ]  
{ NOT NULL |  
NULL |  
DEFAULT default\_expr }

**参数说明**

* server\_name：server的名称。取值范围：长度必须小于等于63。
* fdw\_name：指定外部数据封装器的名称。取值范围：oracle\_fdw，mysql\_fdw，postgres\_fdw，mot\_fdw。
* CREATE SERVER中OPTIONS参数包括：
* host（必选）：要连接的主机名，本功能填写远端服务器的IP地址。
* port（必选）：主机服务器的端口号，本功能填写远端服务器上集群的端口号。
* dbname（必选）：数据库名，本功能填写远端服务器上的数据库名。
* hostaddr(非必选)：与之链接的主机的IP地址，是标准的IPv4地址格式，比如，172.28.40.9。如果机器支持IPv6，那么也可以使用IPv6的地址。如果声明了一个非空的字符串，那么使用TCP/IP通讯机制。
* connect\_timeout(非必选)：链接的最大等待时间，以秒计（用十进制整数字符串书写），0或者不声明表示无穷。不建议把链接超时的值设置得小于2秒。
* options(非必选)：添加命令行选项以在运行时发送到服务器。
* keepalives(非必选)：控制客户端侧的TCP保持激活是否使用。缺省值是1，意思为打开，但是如果不想要保持激活，你可以更改为0，意思为关闭。通过Unix域套接字做的链接忽略这个参数。
* keepalives\_idle(非必选)：在TCP应该发送一个保持激活的信息给服务器之后，控制不活动的秒数。0值表示使用系统缺省。通过Unix域套接字做的链接或者如果禁用了保持激活则忽略这个参数。
* keepalives\_interval(非必选)：在TCP保持激活信息没有被应该传播的服务器承认之后，控制秒数。0值表示使用系统缺省。通过Unix域套接字做的链接或者如果禁用了保持激活则忽略这个参数。
* keepalives\_count(非必选)：添加命令行选项以在运行时发送到服务器。例如，设置为-c comm\_debug\_mode=off设置guc参数comm\_debug\_mode参数的会话的值为off。
* use\_remote\_estimate(非必选)：控制postgres\_fdw是否发出EXPLAIN命令以获取运行消耗估算。默认值为false。
* fdw\_startup\_cost(非必选)：执行一个外表扫描时的启动耗时估算。这个值通常包含建立连接、远端对请求的分析和生成计划的耗时。默认值为100。
* fdw\_typle\_cost(非必选)：在远端服务器上对每一个元组进行扫描时的额外消耗。这个值通常表示数据在server间传输的额外消耗。默认值为0.01。
* user\_name：要映射到外部服务器的一个现有用户的名称。 CURRENT\_USER和USER匹配当前用户的名称。 当PUBLIC被指定时，一个所谓的公共映射会被创建，当没有特定用户的映射可用时将会使用它。
* server\_name：将为其创建用户映射的现有服务器的名称。
* CREATE USER MAPPING中OPTIONS参数包括：
* user(必选)：远端服务器上Vastbase的用户名，属于普通用户。
* password（必选）：远端服务器上Vastbase用户对应的密码
* table\_name：外表的表名。取值范围：字符串，要符合标识符的命名规范。
* column\_name：外表中的字段名。取值范围：字符串，要符合标识符的命名规范。
* type\_name：字段的数据类型。
* SERVER server\_name：外表的server名称。
* OPTIONS 参数包括：
* schema\_name（必选）：远端server的schema名称。如果不指定的话，将使用外表自身的schema名称作为远端的schema名称。
* table\_name（必选）：远端server的表名。如果不指定的话，将使用外表自身的表名作为远端的表名。
* column\_name（非必选）：远端server的表的列名。如果不指定的话，将使用外表自身的列名作为远端的的表的列名。

**注意事项**

* 两个postgres\_fdw外表间的SELECT JOIN不支持下推到远端Vastbase执行，会被分成两条SQL语句传递到远端Vastbase执行，然后在本地汇总处理结果。
* 不支持IMPORT FOREIGN SCHEMA语法。
* 不支持对外表进行CREATE TRIGGER操作。

**示例**

1、在本地服务器创建用户。

create user hzy with sysadmin password '123456Aa';  
grant all on database postgres to hzy;

2、远端服务器修改配置文件pg\_hba.conf的配置参数，新增下列参数。

host all all 0.0.0.0/0 sha256

3、远端服务器修改配置文件postgres.conf中的listen\_address 参数，值为本地服务器的IP地址。

4、远端服务器中创建用户。

create user hzy with sysadmin password '123456Aa';  
grant all on database postgres to hzy;

5、远端服务器使用hzy用户创建测试表。

create table student( student\_no int4 primary key, student\_name varchar(30), age int2 );   
insert into student values(1,'xiaoming',12);

6、在本地服务器中，使用新建的用户登录系统。

7、加载postgres\_fdw扩展。

create extension postgres\_fdw ;

8、创建外部服务器。

create server server\_to\_200 foreign data wrapper postgres\_fdw options (host '172.16.103.82',port '64322',dbname 'postgres');

9、创建一个用户到外部服务器的映射。

create user MAPPING FOR hzy SERVER server\_to\_200 OPTIONS (user 'hzy',password '123456Aa');

10、创建外表。

create foreign table local\_foreign\_table\_student( student\_no int4 , student\_name varchar(30), age int2) server server\_to\_200 options(schema\_name 'public', table\_name 'student');

11、访问外表。

select \* from local\_foreign\_table\_student;  
insert into local\_foreign\_table\_student values (2,'xiaohong',11);  
update local\_foreign\_table\_student set student\_name = 'xiaoqing' where student\_no = 1;  
delete from local\_foreign\_table\_student where age = 11;  
explain select \* from local\_foreign\_table\_student ;

12、查看外表结构。

\d+ local\_foreign\_table\_student;

# AI特性

## AI4DB数据库自治运维

**功能描述**

AI4DB主要用于对数据库进行自治运维和管理，从而帮助数据库运维人员减少运维工作量。在实现上，DBMind的AI4DB框架具有监控和服务化的性质，同时也提供即时AI工具包，提供开箱即用的AI运维功能（如索引推荐）。AI4DB的监控平台以开源的Prometheus为主，DBMind提供监控数据生产者exporter, 可与Prometheus平台完成对接。DBMind的AI4DB服务架构如下图所示：

**组件说明**

* DBMind Service: DBMind后台服务，可用于定期离线计算，包括慢SQL根因分析、时序预测等；
* Prometheus-server: Prometheus 监控指标存储的服务器；
* metadatabase: DBMind在离线计算结束后，将计算结果存储在此处，支持vastbase、SQLite等数据库；
* client: 用户读取DBMind离线计算结果的客户端，目前仅实现命令行客户端；若采用vastbase等数据库存储计算DBMind计算结果，则用户可以自行配置Grafana等可视化工具对该结果进行可视化；
* vastbase-exporter: 用户从vastbase数据库节点上采集监控指标，供DBMind服务进行计算；
* node-exporter: Prometheus官方提供的exporter, 可用于监控该节点的系统指标，如CPU和内存使用情况；
* reprocessing-exporter: 用于对Prometheus采集到的指标进行二次加工处理，例如计算CPU使用率等。

**环境配置**

DBMind外置AI功能需要运行在Python 3.6版本及以上，需要的第三方依赖包记录在AI功能根目录（$GAUSSHOME/bin/dbmind）的requirements.txt文件中（包括requirements-x86.txt与requirements-arrch64.txt，用户可根据自己平台类型选择）中，可以通过pip install命令安装依赖，如：

pip install requirements-x86.txt

如果用户没有安装齐全所需的依赖，则当用户执行gs\_dbmind命令时，会再次提醒用户安装第三方依赖。需要注意，该文件提供了DBMind所需的第三方依赖，若用户环境存在第三方包冲突等情况，可由用户根据实际情况进行处理。

### DBMIND模式说明

**功能描述**

用户可通过gs\_dbmind命令调用AI4DB的全部功能，该命令可实现下列基本功能：

* 服务功能：service子命令，包括创建并初始化配置目录、启动后台服务、关闭后台服务等；
* 调用组件：component子命令，AI4DB功能（如索引推荐、参数调优等）可通过该模式进行即时调用；
* 设置参数：set子命令，通过该命令，可以一键修改配置目录中的配置文件值；当然，用户也可以通过文本编辑器进行手动修改；

**语法格式**

用户可以通过 --help 选项获得模式的帮助信息，例如：

gs\_dbmind --help

usage: [-h] [--version] {service,set,component} ...  
  
vastbase DBMind: An autonomous platform for vastbase  
  
optional arguments:  
 -h, --help show this help message and exit  
 --version show program's version number and exit  
  
available subcommands:  
 {service,set,component}  
 type '<subcommand> -h' for help on a specific subcommand  
 service send a command to DBMind to change the status of the service  
 set set a parameter  
 component pass command line arguments to each sub-component.

**参数说明**

| **参数** | **参数说明** |
| --- | --- |
| -h, --help | 帮助命令 |
| --version | 版本号 |
| service | 服务功能相关的子命令 |
| component | 调用组件的子命令 |
| set | 修改配置文件的子命令 |

#### SERVICE子命令

**功能描述**

该子命令可用于对配置目录进行初始化，同时也可以实现启动和停止后台任务。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

gs\_dbmind service --help

usage: service [-h] -c DIRECTORY [--only-run {slow\_query\_diagnosis,forecast}] [--interactive | --initialize] {setup,start,stop}  
  
positional arguments:  
 {setup,start,stop}  
 perform an action for service  
  
optional arguments:  
 -h, --help show this help message and exit  
 -c DIRECTORY, --conf DIRECTORY  
 set the directory of configuration files  
 --only-run {slow\_query\_diagnosis,forecast}  
 explicitly set a certain task running in the backend  
 --interactive configure and initialize with interactive mode  
 --initialize initialize and check configurations after configuring.

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| action | 动作参数 | setup：初始化配置目录。start：服务启动。stop：服务停止。 |
| -c，–conf | 配置文件目录地址 | - |
| --initialize | 配置参数初始化 | - |
| --interactive | 交互式输入配置参数 | - |
| --only-run | 选择只运行的模块 | forecast：预测模块。slow\_query\_diagnosis：慢SQL根因分析模块。 |
| -h, --help | 帮助命令 | - |

**配置目录初始化**

用户可通过 gs\_dbmind service setup 子命令进行配置目录的初始化。该配置文件中可包括DBMind的配置文件、日志等内容。该目录中的部分文件说明：

* dbmind.conf：DBMind的参数配置文件，用户可通过 gs\_dbmind set 命令进行修改，也可通过文本编辑器进行手动修改。
* dynamic\_config.db：DBMind服务保存在本地节点的元信息，主要包括算法的超参数、监控阈值等；该文件为DBMind服务元信息，不可由用户直接配置。
* metric\_map.conf：监控指标映射表，可用于适配到不同采集平台中。例如，在DBMind中，监控到的系统cpu使用率名为os\_cpu\_usage, 而用户自行实现的指标采集工具将cpu使用率命名为my\_cpu\_usage\_rate. 则在该种情况下，如果想要DBMind代表cpu使用率的指标名为my\_cpu\_usage\_rate, 则需要修改该配置选项。即添加“os\_cpu\_usage = my\_cpu\_usage\_rate”配置项进行映射。对于普通用户，建议直接使用DBMind配套的采集组件和方案，则不涉及修改该配置文件。
* logs: 该目录中用于存储DBMind服务产生的日志。

用户可通过两种方式进行配置目录的初始化，一种为交互式，另一种为非交互式。例如，待初始化的配置目录名为confpath, 则分别通过下述方法进行配置：

* 交互式模式

gs\_dbmind service setup -c confpath --interactive

执行完毕上述命令后，用户可通过命令行终端对配置项进行交互式配置。

* 非交互式模式

非交互式模式总共分为三个步骤，即启动配置，修改配置项，初始化配置。其中第二个步骤需要用户通过文本编辑器手动编辑配置文件。具体步骤如下：

1、启动配置，执行下述命令：

gs\_dbmind service setup -c confpath

2、执行完上述命令后，会在confpath目录下生成dbmind.conf配置文件，用户需要利用文本编辑器进行手动修改。需要修改的配置部分为“指标数据源数据库信息区【TSDB】”、“预测结果存储数据库信息区【METADATABASE】”和“自监控参数配置区【SELF-MONITORING】”相关参数的说明如下：

[TSDB] # 时序数据库相关信息，即指标数据源信息  
name = prometheus # 时序数据库类型. 当前仅支持选择'prometheus'.  
host = # 时序数据库IP地址.  
port = # 时序数据库端口号.  
   
[METADATABASE] # 元数据库相关信息，及预测结果存储方式  
dbtype = # 元数据库类型. 可选择: sqlite, vastbase, postgresql。若该配置项名为vastbase, 需关注Python驱动事宜，可参考下文中关于Python驱动的提示。  
host = # 元数据库地址.  
port = # 元数据库端口号.  
username = # 元数据库用户信息.  
password = # 元数据库密码，将会被加密存储  
database = # 元数据库库名. 如果选择的数据库类型为SQLite, 则无需填写上述配置项，仅需填写此处。DBMind会根据database配置项的内容，在配置目录中生成对应文件名的SQLite数据库文件。  
   
[SELF-MONITORING] # 自监控参数配置  
detection\_interval = 600 # 单位秒. 对vastbase进行健康检查的时间间隔（仅慢SQL根因诊断使用）.  
last\_detection\_time = 600 # 单位秒. 用于健康检查的监控数据长度（仅慢SQL根因诊断使用）.  
forecasting\_future\_time = 3600 # 单位秒，时序预测的长度（供时序预测特性使用），同时也是预测功能调度周期  
# 待时序预测进行预测的关键指标项（仅供时序预测特性使用）  
golden\_kpi = os\_cpu\_usage, os\_mem\_usage, gaussdb\_qps\_by\_instance  
   
[LOG] # 日志相关信息  
maxbytes = 10485760 # 默认值为 10Mb, 单个日志文件的最大大小.如果 maxbytes 为零，则文件无限增长（建议该值不要设置太小，默认即可）.  
backupcount = 1 # 日志文件最大数量.  
level = INFO # 日志级别，也可配置为 DEBUG, INFO, WARNING, ERROR模式。

3、待用户手动修改完上述参数后，需要执行下述命令进行配置项的初始化。在该阶段中，DBMind会初步检查配置项的正确性、加密配置项中出现的明文密码、同时初始化用于存储结果数据的元数据库。

gs\_dbmind service setup --initialize -c confpath

4、完成配置目录初始化过程，可基于该配置目录启动DBMind后台服务。

1、配置文件注释信息用于在交互模式下对用户进行提示有特殊含义，请勿手动修改或删除。  
2、需要确保配置项的值与注释信息之间通过空格符分割，否则系统会将注释信息识别为配置项的值。  
3、配置项中的特殊字符，如果需要转移，则通过转义符“百分号”（%）来转义，例如，用户配置的密码为 “password%”, 则应通过“百分号”进行转义，即 “password%%“。

**启动服务**

当用户完成配置目录的初始化后，可基于此配置目录启动DBMind后台服务。例如配置目录为confpath，则启动命令如下：

gs\_dbmind service start -c confpath

当执行上述命令后，会提示服务已启动。在未指定任何附加参数时，该命令默认会启动所有的后台任务。如果用户只想启动某一个后台任务，需要添加参数 –only-run. 例如，用户只想启动慢SQL根因分析服务，则为：

gs\_dbmind service start -c confpath --only-run slow\_query\_diagnosis

**关闭服务**

关闭服务与启动服务类似，其命令行结构更加简单，只需指定配置目录的地址即可。例如配置目录为confpath，则为：

gs\_dbmind service stop -c confpath

DBMind服务会在后台执行完正在运行的任务后自行退出。

* [METADATABASE]中的元数据库用户需要具有在该数据库下的创表和数据插入更新权限，否则工具执行会出现异常。
* 当前不支持同一配置文件下分开启动多个服务。
* 工具提供了requirement.txt文件，用户可以通过该文件安装所需的第三方依赖。

#### COMPONENT子命令

**功能描述**

该子命令可以用于启动DBMind的组件，包括可用于监控指标的exporter，以及AI功能等。该命令可以将用户通过命令行传入的命令转发给对应的子组件，故不同的子组件命令需参考其功能的对应说明。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

gs\_dbmind component --help

usage: component [-h] COMPONENT\_NAME ...  
  
positional arguments:  
 COMPONENT\_NAME choice a component to start. ['extract\_log', 'forecast', 'index\_advisor', 'vastbase\_exporter', 'reprocessing\_exporter', 'slow\_query\_diagnosis', 'sqldiag', 'xtuner']  
 ARGS arguments for the component to start  
  
optional arguments:  
 -h, --help show this help message and exit

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| COMPONENT\_NAME | 子组件名 | extract\_log, forecast, index\_advisor, vastbase\_exporter, reprocessing\_exporter, slow\_query\_diagnosis, sqldiag, xtuner |
| ARGS | 子组件的参数 | 参考子组件的命令说明 |
| -h, --help | 帮助命令 | - |

#### SET子命令

该命令用于修改配置文件dbmind.conf中的参数值，与用户手动修改配置文件dbmind.conf一般无差异。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

gs\_dbmind set --help

usage: set [-h] -c DIRECTORY section option target  
  
positional arguments:  
 section which section (case sensitive) to set  
 option which option to set  
 target the parameter target to set  
  
optional arguments:  
 -h, --help show this help message and exit  
 -c DIRECTORY, --conf DIRECTORY  
 set the directory of configuration files

**参数说明**

| **参数** | **参数说明** |
| --- | --- |
| -h, --help | 帮助命令 |
| -c，--conf | 配置文件目录confpath |
| section | 设置区 |
| option | 设置项 |
| target | 设置值 |

**注意事项**

修改密码项时必须通过该命令才可以实现修改，否则通过用户手动修改的配置文件无法实现对明文密码的加密。

**示例**

修改配置目录confpath中的配置文件dbmind.conf中TSDB配置部分，host参数的值，并将其设置为 127.0.0.1。

gs\_dbmind set TSDB host 127.0.0.1 -c confpath

修改密码。

gs\_dbmind set METADATABASE password xxxxxx -c confpath

### DBMIND支持的组件

#### Prometheus Exporter组件

**功能描述**

Prometheus是业内非常流行的开源监控系统，同时本身也是一款时序数据库。Prometheus的采集端称之为exporter，用来收集被监控模块的指标项。为了与Prometheus平台完成对接，AI工具分别实现了两款exporter，分别是用来采集数据库指标的vastbase-exporter，以及对采集到的指标进行二次加工的reprocessing-exporter。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

* reprocessing-exporter的使用帮助详情：

gs\_dbmind component reprocessing\_exporter --help

usage: [-h] [--disable-https] [--ssl-keyfile SSL\_KEYFILE]  
 [--ssl-certfile SSL\_CERTFILE]  
 [--web.listen-address WEB.LISTEN\_ADDRESS]  
 [--web.listen-port WEB.LISTEN\_PORT]  
 [--collector.config COLLECTOR.CONFIG] [--log.filepath LOG.FILEPATH]  
 [--log.level {debug,info,warn,error,fatal}] [--version]  
 prometheus\_host prometheus\_port  
  
Reprocessing Exporter: A re-processing module for metrics stored in the  
Prometheus server.  
  
positional arguments:  
 prometheus\_host from which host to pull data  
 prometheus\_port the port to connect to the Prometheus host  
  
optional arguments:  
 -h, --help show this help message and exit  
 --disable-https disable Https schema  
 --ssl-keyfile SSL\_KEYFILE  
 set the path of ssl key file  
 --ssl-certfile SSL\_CERTFILE  
 set the path of ssl certificate file  
 --web.listen-address WEB.LISTEN\_ADDRESS  
 address on which to expose metrics and web interface  
 --web.listen-port WEB.LISTEN\_PORT  
 listen port to expose metrics and web interface  
 --collector.config COLLECTOR.CONFIG  
 according to the content of the yaml file for metric  
 collection  
 --log.filepath LOG.FILEPATH  
 the path to log  
 --log.level {debug,info,warn,error,fatal}  
 only log messages with the given severity or above.  
 Valid levels: [debug, info, warn, error, fatal]  
 --version show program's version number and exit

* vastbase-exporter的使用帮助详情：

gs\_dbmind component vastbase\_exporter --help

usage: [-h] --url URL [--config CONFIG] [--constant-labels CONSTANT\_LABELS]  
 [--web.listen-address WEB.LISTEN\_ADDRESS]  
 [--web.listen-port WEB.LISTEN\_PORT]  
 [--web.telemetry-path WEB.TELEMETRY\_PATH] [--disable-cache]  
 [--disable-settings-metrics] [--disable-statement-history-metrics]  
 [--disable-https] [--ssl-keyfile SSL\_KEYFILE]  
 [--ssl-certfile SSL\_CERTFILE] [--parallel PARALLEL]  
 [--log.filepath LOG.FILEPATH]  
 [--log.level {debug,info,warn,error,fatal}] [--version]  
  
vastbase Exporter (DBMind): Monitoring for vastbase.  
  
optional arguments:  
 -h, --help show this help message and exit  
 --url URL vastbase database target url.  
 --config CONFIG path to config dir or file.  
 --constant-labels CONSTANT\_LABELS  
 a list of label=value separated by comma(,).  
 --web.listen-address WEB.LISTEN\_ADDRESS  
 address on which to expose metrics and web interface  
 --web.listen-port WEB.LISTEN\_PORT  
 listen port to expose metrics and web interface  
 --web.telemetry-path WEB.TELEMETRY\_PATH  
 path under which to expose metrics.  
 --disable-cache force not using cache.  
 --disable-settings-metrics  
 not collect pg\_settings.yml metrics.  
 --disable-statement-history-metrics  
 not collect statement-history metrics (including slow  
 queries).  
 --disable-https disable Https schema  
 --ssl-keyfile SSL\_KEYFILE  
 set the path of ssl key file  
 --ssl-certfile SSL\_CERTFILE  
 set the path of ssl certificate file  
 --parallel PARALLEL not collect pg\_settings.yml metrics.  
 --log.filepath LOG.FILEPATH  
 the path to log  
 --log.level {debug,info,warn,error,fatal}  
 only log messages with the given severity or above.  
 Valid levels: [debug, info, warn, error, fatal]  
 --version show program's version number and exit

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| --url | vastbase server 的连接地址，例如 postgres://user:pwd@host:port/dbname | - |
| --constant-labels | 常量标签，将采集到的指标项中强行添加该标签列表 | 1024-65535 |
| -h, --help | 帮助选项 | - |
| --disable-https | 禁用Https协议 | - |
| --ssl-keyfile | Https协议使用的证书私钥文件路径 | - |
| --ssl-certfile | Https协议使用的证书文件路径 | - |
| --web.listen-address | 该exporter服务的绑定IP | - |
| --web.listen-port | 该exporter服务的侦听端口 | 1024-65535 |
| --web.telemetry-path | 该exporter采集指标的URI地址，默认为 /metrics | - |
| --config | 显性指定的待采集指标配置文件路径 | - |
| --log.filepath | 日志文件保存路径，默认保存在当前目录下 | - |
| --log.level | 日志文件的打印级别，默认为INFO级别 | debug, info, warn, error, fatal |
| --version | 显示版本信息 | - |
| --disable-cache | 禁止使用缓存 | - |
| --disable-settings-metrics | 禁止采集pg\_settings表的值 | - |
| --disable-statement-history-metrics | 禁止收集statement\_history表中的慢SQL信息 | - |
| --parallel | 连接到vastbase的数据库连接池的大小 | 正整数 |

**注意事项**

* Prometheus 和exporter部署在内网环境中，不对外部暴露接口，仅供内部监控平台使用。如果用户希望提高安全性，可自行修改Prometheus的TLS配置选项，但仍不建议对外部直接暴露访问接口。
* 提供的exporter组件默认采用Https通信协议，因此需要用户默认提供ssl证书和秘钥文件，并通过--ssl-keyfile以及--ssl-certfile提供。若用户不希望使用Https协议，则可以通过--disable-https选项禁用该模式。
* vastbase-exporter中连接数据库的用户需要monitor admin或以上权限，否则会出现部分指标无法采集的情况。
* vastbase-exporter会从dbe\_perf.statement\_history中抽样慢SQL信息，dbe\_perf.statement\_history视图慢SQL记录与GUC参数log\_min\_duration\_statement和track\_stmnt\_stat\_level相关，其中log\_min\_duration\_statement是慢SQL阈值，单位毫秒，具体值由用户设置；track\_stmnt\_stat\_level是SQL记录级别，默认为'OFF,L0'，即只记录慢SQL信息，级别为L0，用户在详细了解参数意义与作用情况下谨慎修改。
* vastbase-exporter采集数据库相关信息，主要包括部分系统表和视图中的数据（具体参见代码中vastbase\_exporter中的配置文件），node-exporter采集系统指标信息，主要与系统磁盘、CPU等相关，reprocessing\_exporter基于prometheus-server中的某些指标（具体参见代码中reprocessing\_exporter中的配置文件）进行二次加工，最终提供加工后的数据供用户使用。
* prometheus-server在拉取exporter数据时有超时机制，超时时间由scrape\_timeout（默认10s）控制，因此当exporter采集数据量较大时，用户可根据实际情况增大scrape\_timeout以防止超时报错，另外需要注意的是scrape\_interval（采集间隔）不能比scrape\_timeout小，否则会出现异常。
* 如果数据库时区设置和系统不相同，可能会出现时间相关指标时间与系统时间不一致的情况，因此需要将用户时区与系统保持同步。
* 当使用https通信时，工具会检测证书与密钥文件权限以及证书有效期，如果文件权限大于600则会出现报警，证书有效期小于90天会出现报警。
* 当存在指标重复采集时，vastbase-exporter会出现异常，异常信息会打印到日志中。
* vastbase-exporter的--config、--disable-settings-metrics、--disable-statement-history-metrics三个参数需要注意，其存在以下几种情况：
* 用户不指定其中任何参数，则工具会同时对yamls目录下的三个配置文件中的指标进行采集。
* 用户显式指定--config，则工具不会采集yamls目录下default.yml中的指标，而会采集用户指定配置文件中的指标，同时pg\_settings.yml和statements.yml正常采集，此时用户需要注意指定的配置文件中的指标和pg\_settings.yml、statements.yml中的指标不能存在重复采集的现象。
* 用户显式指定--disable-settings-metrics，则工具不会采集yamls目录下pg\_settings.yml中的指标，用户显式指定--disable-statement-history-metrics，则工具不会采集yamls目录下statements.yml（慢SQL相关）中的指标。

**环境部署**

1、通过命令行参数启动对应的exporter进程，同时在本地机器创建侦听端口号。

2、在Promethues的服务器端修改配置文件prometheus.yml，将启动的exporter信息添加进去，例如：

scrape\_configs:  
 ...  
   
 - job\_name: 'vastbase\_exporter'  
 static\_configs:  
 - targets: ['127.0.0.1:9187']  
 ...

由于vastbase默认模式下的通信加密协议与PostgreSQL不兼容，故导致通过PyPI源安装的基于PostgreSQL编译的Python驱动psycopg2-binary默认无法连接至vastbase数据库。  
因此，需要用户自行编译psycopg2或修改GUC参数进行适配。也可通过vastbase官方网站下载基于vastbase编译的psycopg2（官方网站仅提供部分Python版本的编译包，一般为Python3.6版本，需要用户鉴别）。

* 官方vastbase Python驱动下载地址为：  
  [https://openGauss.org/zh/download.html](https://opengauss.org/zh/download.html)
* 关于Python驱动的适配问题，可参考vastbase官方操作指南：  
  <https://mp.weixin.qq.com/s/2TobUQKtw0N9sBpMZJr6zw>

**使用指导**

用户可通过gs\_dbmind命令启动对应的exporter。下面演示了一个完整的Prometheus监控平台的搭建过程。

1、部署vastbase-exporter：启动vastbase-exporter，采用默认侦听端口号9187，侦听地址为192.168.1.100，不采用https协议，则命令可以为：

gs\_dbmind component vastbase\_exporter --url postgresql://user:password@ip:port/dbname --web.listen-address 192.168.1.100 --disable-https

2、部署reprocessing-exporter：启动reprocessing-exporter，采用默认侦听端口号8181，侦听地址为192.168.1.101，Prometheus-server IP与端口号为192.168.1.100:9090，采用https协议，则命令可以为：

gs\_dbmind component reprocessing\_exporter 192.168.1.100 9090 --web.listen-address 192.168.1.101 --ssl-keyfile server.key --ssl-certfile server.crt

3、部署node-exporter：一般地，对于Prometheus监控平台都需要部署node-exporter用于监控Linux操作系统，后文提到的部分AI功能也需要依赖node-exporter采集Linux系统指标，故也需要用户来部署；只不过，该node-exporter为Prometheus自带组件，需要用户自行下载安装，下载地址为 [https://prometheus.io/download/#node\_exporter](https://prometheus.io/download/" \l "node_exporter)，使用方法详见：[https://prometheus.io/docs/guides/node-exporter/#installing-and-running-the-node-exporter](https://prometheus.io/docs/guides/node-exporter/" \l "installing-and-running-the-node-exporter) 。用户可解压压缩包后直接运行该node-exporter进程，其默认端口号为9100，启动命令行为：

./node\_exporter

4、配置Prometheus-server，修改配置文件prometheus.yml，添加下述内容：

scrape\_configs:  
 ...  
 - job\_name: 'vastbase\_exporter'  
 static\_configs:  
 - targets: ['192.168.1.100:9187']  
 - job\_name: 'reprocessing\_exporter'  
 scheme: https  
 tls\_config:  
 ca\_file: xxx.crt  
 static\_configs:  
 - targets: ['127.0.0.1:8181']  
 - job\_name: 'node\_exporter'  
 static\_configs:  
 - targets: ['127.0.0.1:9100']  
 ...

**常见问题**

* 提示需要用户提供--ssl-keyfile与--ssl-certfile选项：
* 上述exporter默认采用Https模式通信，因此需要用户指定证书及其私钥文件的路径。相反，如果用户只想采用Http模式，则需要显性给定--disable-https选项，从而禁用Https协议。
* 提示用户需要输入PEM密码（Enter PEM pass phrase）：
* 如果用户采用Https模式，并给定了证书及其秘钥文件的路径，且该秘钥文件是经过加密的，则需要用户输入该加密私钥证书文件的密码。

### DBMIND的AI子功能

#### X-TUNER：参数调优与诊断

**功能描述**

X-Tuner是一款数据库集成的参数调优工具，通过结合深度强化学习和全局搜索算法等AI技术，实现在无需人工干预的情况下，获取最佳数据库参数配置。本功能不强制与数据库环境部署到一起，支持独立部署，脱离数据库安装环境独立运行。

**原理简介**

调优程序是一个独立于数据库内核之外的工具，需要提供数据库及其所在实例的用户名和登录密码信息，以便控制数据库执行benchmark进行性能测试；在启动调优程序前，要求用户测试环境交互正常，能够正常跑通benchmark测试脚本、能够正常连接数据库。

如果需要调优的参数中，包含重启数据库后才能使修改生效的参数，那么在调优过程中数据库将会重启多次。如果用户的数据库正在执行作业，请慎用train与tune模式。

调优程序X-Tuner包含三种运行模式，分别是:

* recommend：通过用户指定的用户名等信息登录到数据库环境中，获取当前正在运行的workload特征信息，根据上述特征信息生成参数推荐报告。报告当前数据库中不合理的参数配置和潜在风险等；输出根据当前正在运行的workload行为和特征；输出推荐的参数配置。该模式是秒级的，不涉及数据库的重启操作，其他模式可能需要反复重启数据库。
* train：通过用户提供的benchmark信息，不断地进行参数修改和benchmark的执行。通过反复的迭代过程，训练强化学习模型，以便用户在后面通过tune模式加载该模型进行调优。
* tune：使用优化算法进行数据库参数的调优，当前支持两大类算法，一种是深度强化学习，另一种是全局搜索算法（全局优化算法）。深度强化学习模式要求先运行train模式，生成训练后的调优模型，而使用全局搜索算法则不需要提前进行训练，可以直接进行搜索调优。

如果在tune模式下，使用深度强化学习算法，要求必须有一个训练好的模型，且要求训练该模型时的参数与进行调优时的参数列表（包括max与min）必须一致。

X-Tuner结构图如下所示：

* DB侧：通过DB\_Agent模块对数据库实例进行抽象，通过该模块可以获取数据库内部的状态信息、当前数据库参数、以及设置数据库参数等。DB侧包括登录数据库环境使用的SSH连接。
* 算法侧：用于调优的算法包，包括全局搜索算法（如贝叶斯优化、粒子群算法等）和深度强化学习（如DDPG）。
* X-Tuner主体逻辑模块：通过Enviroment模块进行封装，每一个step就是一次调优过程。整个调优过程通过多个step进行迭代。
* benchmark：由用户指定的benchmark性能测试脚本，用于运行benchmark作业，通过跑分结果反映数据库系统性能优劣。

应确保benchmark脚本跑分结果越大表示性能越好。 例如TPCH这种衡量SQL语句整体执行时长的benchmark，可以通过取总体执行时间的相反数作为benchmark的输出分数。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

gs\_dbmind component xtuner --help

usage: [-h] [--db-name DB\_NAME] [--db-user DB\_USER] [--port PORT]  
 [--host HOST] [--host-user HOST\_USER]  
 [--host-ssh-port HOST\_SSH\_PORT] [-f DB\_CONFIG\_FILE]  
 [-x TUNER\_CONFIG\_FILE] [-v]  
 {train,tune,recommend}  
  
X-Tuner: a self-tuning tool integrated by vastbase.  
  
positional arguments:  
 {train,tune,recommend}  
 Train a reinforcement learning model or tune database  
 by model. And also can recommend best\_knobs according  
 to your workload.  
  
optional arguments:  
 -h, --help show this help message and exit  
 -f DB\_CONFIG\_FILE, --db-config-file DB\_CONFIG\_FILE  
 You can pass a path of configuration file otherwise  
 you should enter database information by command  
 arguments manually. Please see the template file  
 share/server.json.template.  
 -x TUNER\_CONFIG\_FILE, --tuner-config-file TUNER\_CONFIG\_FILE  
 This is the path of the core configuration file of the  
 X-Tuner. You can specify the path of the new  
 configuration file. The default path is xtuner.conf.  
 You can modify the configuration file to control the  
 tuning process.  
 -v, --version show program's version number and exit  
  
Database Connection Information:  
 --db-name DB\_NAME The name of database where your workload running on.  
 --db-user DB\_USER Use this user to login your database. Note that the  
 user must have sufficient permissions.  
 --port PORT Use this port to connect with the database.  
 --host HOST The IP address of your database installation host.  
 --host-user HOST\_USER  
 The login user of your database installation host.  
 --host-ssh-port HOST\_SSH\_PORT  
 The SSH port of your database installation host.

**参数说明**

* 命令行参数

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| mode | 指定调优程序运行的模式 | train，tune，recommend |
| --tuner-config-file, -x | X-Tuner的核心参数配置文件路径，默认路径为安装目录下的xtuner.conf。如果需要修改配置文件的加载路径，则可以通过选项-x命令行选项来指定。 |  |
| --db-config-file, -f | 调优程序的用于登录到数据库宿主机上的连接信息配置文件路径，若通过该文件配置数据库连接信息，则下述数据库连接信息可省略 | - |
| --db-name | 指定需要调优的数据库名 | - |
| --db-user | 指定以何用户身份登陆到调优的数据库上 | - |
| --port | 数据库的侦听端口 | - |
| --host | 数据库实例的宿主机IP | - |
| --host-user | 指定以何用户身份登陆到数据库实例的宿主机上，要求改用户名的环境变量中可以找到gsql、vb\_ctl等数据库运维工具。 | - |
| --host-ssh-port | 数据库实例所在宿主机的SSH端口号，可选，默认为22 | - |
| --help, -h | 返回帮助信息 | - |
| --version, -v | 返回当前工具版本号 | - |

* 配置文件中的参数详解

| **参数名** | **参数说明** | **取值范围** |
| --- | --- | --- |
| logfile | 生成的日志存放路径 | - |
| output\_tuning\_result | 可选，调优结果的保存路径 | - |
| verbose | 是否打印详情 | on, off |
| recorder\_file | 调优中间信息的记录日志存放路径 | - |
| tune\_strategy | 调优模式下采取哪种策略 | rl, gop, auto |
| drop\_cache | 是否在每一个迭代轮次中进行drop cache，drop cache可以使benchmark跑分结果更加稳定。若启动该参数，则需要将登录的系统用户加入到 /etc/sudoers 列表中，同时为其增加 NOPASSWD 权限（由于该权限可能过高，建议临时启用该权限，调优结束后关闭）。 | on, off |
| used\_mem\_penalty\_term | 数据库使用总内存的惩罚系数，用于防止通过无限量占用内存而换取的性能表现。该数值越大，惩罚力度越大。 | 建议0 ~ 1 |
| rl\_algorithm | 选择何种RL算法 | ddpg |
| rl\_model\_path | RL模型保存或读取路径，包括保存目录名与文件名前缀。在train 模式下该路径用于保存模型，在tune模式下则用于读取模型文件 | - |
| rl\_steps | 深度强化学习算法迭代的步数 | - |
| max\_episode\_steps | 每个回合的最大迭代步数 | - |
| test\_episode | 使用RL算法进行调优模式的回合数 | - |
| gop\_algorithm | 采取何种全局搜索算法 | bayes, pso |
| max\_iterations | 全局搜索算法的最大迭代轮次（并非确定数值，可能会根据实际情况多跑若干轮） | - |
| particle\_nums | PSO算法下的粒子数量 | - |
| benchmark\_script | 使用何种benchmark驱动脚本，该选项指定加载benchmark路径下同名文件，默认支持TPC-C、TPC-H等典型benchmark | tpcc, tpch, tpcds, sysbench … |
| benchmark\_path | benchmark 脚本的存储路径，若没有配置该选项，则使用benchmark驱动脚本中的配置 | - |
| benchmark\_cmd | 启动benchmark 脚本的命令，若没有配置该选项，则使用benchmark驱动脚本中的配置 | - |
| benchmark\_period | 仅对 period benchmark有效，表明整个benchmark的测试周期是多少，单位是秒 | - |
| scenario | 用户指定的当前workload所属的类型 | tp, ap, htap |
| tuning\_list | 准备调优的参数列表文件，可参考 share/knobs.json.template 文件 | - |

**注意事项**

* 数据库状态正常、客户端能够正常连接、且要求数据库内导入数据，以便调优程序可以执行benchmark测试调优效果。
* 使用本工具需要指定登录到数据库的用户身份，要求该登录到数据库上的用户具有足够的权限，以便可以获得充足的数据库状态信息。
* 使用登录到数据库宿主机上的Linux用户，需要将$GAUSSHOME/bin添加到PATH环境变量中，即能够直接运行gsql、gs\_guc、vb\_ctl等数据库运维工具。
* 本工具支持以三种模式运行，其中tune和train模式要求用户配置好benchmark运行环境，并导入数据，本工具将会通过迭代运行benchmark来判断修改后的参数是否有性能提升。
* recommend模式建议在数据库正在执行workload的过程中执行，以便获得更准确的实时workload信息。
* 本工具默认带有TPC-C、TPC-H、TPC-DS以及sysbench的benchmark运行脚本样例，如果用户使用上述benchmark对数据库系统进行压力测试，则可以对上述配置文件进行适度修改或配置。如果需要适配用户自己的业务场景，需要您参照benchmark目录中的template.py文件编写驱动您自定义benchmark的脚本文件。

**Benchmark的选择与配置**

Benchmark的驱动脚本存放路径为X-Tuner目录（即$GAUSSHOME/bin/dbmind/components/xtuner，下同）的子目录benchmark中。X-Tuner自带常用的benchmark驱动脚本，例如基于时间周期的探测脚本（默认）、TPC-C、TPC-H等。X-Tuner通过调用benchmark/\_init\_\_.py文件中 get\_benchmark\_instance()命令来加载不同的benchmark驱动脚本，获取benchmark驱动实例。其中，benchmark驱动脚本的格式说明如下：

* 驱动脚本文件名：表示benchmark的名字，该名字用于表示驱动脚本的唯一性，可通过在X-Tuner的配置文件中的配置项benchmark\_script来指定选择加载哪个benchmark驱动脚本。
* 驱动脚本内容三要素：path变量、cmd变量以及run函数。

下面分别介绍驱动脚本的内容三要素：

* path变量：表示benchmark脚本的存放地址，可以直接在驱动脚本中修改，也可以通过配置文件的benchmark\_path配置项来指定。
* cmd变量：表示执行benchmark脚本需要运行的命令，可以直接在驱动脚本中修改，也可以通过配置文件的benchmark\_cmd配置项来指定。cmd中的文本允许使用占位符，用于获取某些运行cmd命令时的必要信息，使用示例参见TPC-H驱动脚本示例。这些占位符包括：
* {host}：数据库宿主机的IP地址
* {port}：数据库实例的侦听端口号
* {user}：登录数据库系统上的用户名
* {password}：与登录数据库系统上的用户相匹配的密码
* {db}：正在进行调优的数据库名
* run函数：该函数的函数签名如下所示。

def run(remote\_server, local\_host) -> float:

其中，返回数据类型为float，表示benchmark执行后的评估分数值，要求该值越大表示性能越好，例如使用TPC-C跑分结果tpmC即可作为返回值，TPC-H的全部SQL语句执行总时间的相反数（取相反数后可保证返回值越大则性能越好）也可作为返回值。

remote\_server变量是X-Tuner程序传递给脚本使用的远端主机（数据库宿主机）的shell命令接口，local\_host变量是X-Tuner程序传递给脚本使用的本地主机（运行X-Tuner脚本的主机）的shell命令接口。上述shell命令接口提供的方法包括：

* 在主机上执行shell命令

exec\_command\_sync(command, timeout)

参数列表：command 必选，数据类型可以是str, 以及元素为str类型的list或tuple；timeout 可选，表示命令执行的超时时长，单位是秒。  
返回值：返回二元组 (stdout, stderr)，stdout表示标准输出流结果，stderr表示标准错误流结果，数据类型均为str.

* 最近一条shell命令执行后的退出状态码(exit status code)。

exit\_status

说明：一般情况，退出状态码为0表示执行正常，非0表示存在错误。

**Benchmark驱动脚本示例说明**

* TPC-C 驱动脚本

from tuner.exceptions import ExecutionError  
   
# WARN: You need to download the benchmark-sql test tool to the system,  
# replace the PostgreSQL JDBC driver with the vastbase driver,  
# and configure the benchmark-sql configuration file.  
# The program starts the test by running the following command:  
path = '/path/to/benchmarksql/run' # TPC-C测试脚本benchmark-sql 的存放路径  
cmd = "./runBenchmark.sh props.gs" # 自定义一个名为 props.gs 的benchmark-sql测试配置文件  
   
   
def run(remote\_server, local\_host):  
 # 切换到 TPC-C 脚本目录下，清除历史错误日志，然后运行测试命令。  
 # 此处最好等待几秒钟，因为benchmark-sql 测试脚本生成最终测试报告是通过一个shell脚本实现的，整个过程会有延迟，  
 # 为了保证能够获取到最终的tpmC数值报告，我们这里选择等待3秒钟。  
 stdout, stderr = remote\_server.exec\_command\_sync(['cd %s' % path, 'rm -rf benchmarksql-error.log', cmd, 'sleep 3'])  
 # 如果标准错误流中有数据，则报异常退出。  
 if len(stderr) > 0:  
 raise ExecutionError(stderr)  
   
 # 寻找最终tpmC结果  
 tpmC = None  
 split\_string = stdout.split() # 对标准输出流结果进行分词。  
 for i, st in enumerate(split\_string):  
 # 在5.0版本的benchmark-sql中，tpmC最终测试结果数值在 '(NewOrders)'关键字的后两位，正常情况下，找到该字段后直接返回即可。  
 if "(NewOrders)" in st:  
 tpmC = split\_string[i + 2]  
 break  
 stdout, stderr = remote\_server.exec\_command\_sync(  
 "cat %s/benchmarksql-error.log" % path)  
 nb\_err = stdout.count("ERROR:") # 判断整个benchmark运行过程中，是否有报错，记录报错的错误数  
 return float(tpmC) - 10 \* nb\_err # 这里将报错的错误数作为一个惩罚项，惩罚系数为10，越高的惩罚系数表示越看中报错的数量.

* TPC-H驱动脚本

import time  
   
from tuner.exceptions import ExecutionError  
   
# WARN: You need to import data into the database and SQL statements in the following path will be executed.  
# The program automatically collects the total execution duration of these SQL statements.  
path = '/path/to/tpch/queries' # 存放TPC-H测试用的SQL脚本目录  
cmd = "gsql -U {user} -W {password} -d {db} -p {port} -f {file}" # 需要运行TPC-H测试脚本的命令，一般使用'gsql -f 脚本文件' 来运行  
   
   
def run(remote\_server, local\_host):  
 # 遍历当前目录下所有的测试用例文件名  
 find\_file\_cmd = "find . -type f -name '\*.sql'"  
 stdout, stderr = remote\_server.exec\_command\_sync(['cd %s' % path, find\_file\_cmd])  
 if len(stderr) > 0:  
 raise ExecutionError(stderr)  
 files = stdout.strip().split('n')  
 time\_start = time.time()  
 for file in files:  
 # 使用 file 变量替换 {file}，然后执行该命令行。  
 perform\_cmd = cmd.format(file=file)  
 stdout, stderr = remote\_server.exec\_command\_sync(['cd %s' % path, perform\_cmd])  
 if len(stderr) > 0:  
 print(stderr)  
 # 代价为全部测试用例的执行总时长  
 cost = time.time() - time\_start  
 # 取相反数，适配run 函数的定义：返回结果越大表示性能越好。  
 return - cost

**示例**

X-Tuner支持三种模式，分别是获取参数诊断报告的recommend模式、训练强化学习模型的train模式、以及使用算法进行调优的tune模式。上述三种模式可以通过命令行参数来区别，通过配置文件来指定具体的细节。

1、配置数据库连接信息

* 通过命令行指定：分别传递 --db-name --db-user --port --host --host-user 参数，可选 --host-ssh-port 参数。

gs\_dbmind component xtuner recommend --db-name postgres --db-user omm --port 5678 --host 192.168.1.100 --host-user omm

* 通过JSON格式的连接信息配置文件指定：JSON配置文件的示例如下，并假设文件名为 connection.json，可通过 -f connection.json 传递。

{  
 "db\_name": "postgres", # 数据库名  
 "db\_user": "dba", # 登录到数据库上的用户名  
 "host": "127.0.0.1", # 数据库宿主机的IP地址  
 "host\_user": "dba", # 登录到数据库宿主机的用户名  
 "port": 5432, # 数据库的侦听端口号  
 "ssh\_port": 22 # 数据库宿主机的SSH侦听端口号  
}

为了防止密码泄露，配置文件和命令行参数中默认都不包含密码信息，用户在输入上述连接信息后，程序会采用交互式的方式要求用户输数据库密码以及操作系统登录用户的密码。

2、recommend模式使用示例

对recommend模式生效的配置项为scenario，若为auto，则自动检测workload类型。

执行下述命令，获取诊断结果：

gs\_dbmind component xtuner recommend -f connection.json

recommend模式生成的报告示意图

在上述报告中，推荐了该环境上的数据库参数配置，并进行了风险提示。报告同时生成了当前workload的特征信息，其中有几个特征是比较有参考意义的：

* temp\_file\_size：产生的临时文件数量，如果该结果大于0，则表明系统使用了临时文件。使用过多的临时文件会导致性能不佳，如果可能的话，需要提高work\_mem参数的配置。
* cache\_hit\_rate：shared\_buffer的缓存命中率，表明当前workload使用缓存的效率。
* read\_write\_ratio：数据库作业的读写比例。
* search\_modify\_ratio：数据库作业的查询与修改数据的比例。
* ap\_index：表明当前workload的AP指数，取值范围是0到10，该数值越大，表明越偏向于数据分析与检索。
* workload\_type：根据数据库统计信息，推测当前负载类型，分为AP、TP以及HTAP三种类型。
* checkpoint\_avg\_sync\_time：数据库在checkpoint时，平均每次同步刷新数据到磁盘的时长，单位是毫秒。
* load\_average：平均每个CPU核心在1分钟、5分钟以及15分钟内的负载。一般地，该数值在1左右表明当前硬件比较匹配workload、在3左右表明运行当前作业压力比较大，大于5则表示当前硬件环境运行该workload压力过大（此时一般建议减少负载或升级硬件）。

由于某些系统表会一直记录统计信息，这可能会对负载特征识别造成干扰，因此建议最好先清空某些系统表的统计信息，运行一段时间的workload后再使用recommend模式进行诊断，以便获得更准确的结果。清除统计信息的方法为：

select pg\_stat\_reset\_shared('bgwriter');  
select pg\_stat\_reset();

recommend模式会读取数据库中的pg\_stat\_database以及pg\_stat\_bgwriter等系统表中的信息，需要登录到数据库上的用户具有足够的权限（建议为管理员权限，可通过alter user username sysadmin；授予username相应的权限）。

3、train 模式使用示例

该模式是用来训练深度强化学习模型的，与该模式有关的配置项为：

* rl\_algorithm：用于训练强化学习模型的算法，当前支持设置为ddpg.
* rl\_model\_path：训练后生成的强化学习模型保存路径。
* rl\_steps：训练过程的最大迭代步数。
* max\_episode\_steps：每个回合的最大步数。
* scenario：明确指定的workload类型，如果为auto则为自动判断。在不同模式下，推荐的调优参数列表也不一样。
* tuning\_list：用户指定需要调哪些参数，如果不指定，则根据workload类型自动推荐应该调的参数列表。如需指定，则tuning\_list表示调优列表文件的路径。一个调优列表配置文件的文件内容示例如下：

{  
 "work\_mem": {  
 "default": 65536,  
 "min": 65536,  
 "max": 655360,  
 "type": "int",  
 "restart": false  
 },  
 "shared\_buffers": {  
 "default": 32000,  
 "min": 16000,  
 "max": 64000,  
 "type": "int",  
 "restart": true  
 },  
 "random\_page\_cost": {  
 "default": 4.0,  
 "min": 1.0,  
 "max": 4.0,  
 "type": "float",  
 "restart": false  
 },  
 "enable\_nestloop": {  
 "default": true,  
 "type": "bool",  
 "restart": false  
 }  
}

待上述配置项配置完成后，可以通过下述命令启动训练：

gs\_dbmind component xtuner train -f connection.json

训练完成后，会在配置项rl\_model\_path指定的目录中生成模型文件。

3、tune模式使用示例

tune模式支持多种算法，包括基于强化学习（Reinforcement Learning, RL）的DDPG算法、基于全局搜索算法（Global OPtimization algorithm, GOP）算法的贝叶斯优化算法（Bayesian Optimization）以及粒子群算法（Particle Swarm Optimization, PSO）。

与tune模式相关的配置项为：

* tune\_strategy：指定选择哪种算法进行调优，支持rl（使用强化学习模型进行调优）、gop（使用全局搜索算法）以及auto（自动选择）。若该参数设置为rl，则rl相关的配置项生效。除前文提到过的train模式下生效的配置项外，test\_episode配置项也生效，该配置项表明调优过程的最大回合数，该参数直接影响了调优过程的执行时间（一般地，数值越大越耗时）。
* gop\_algorithm：选择何种全局搜索算法，支持bayes以及pso。
* max\_iterations：最大迭代轮次，数值越高搜索时间越长，效果往往越好。
* particle\_nums：在PSO算法上生效，表示粒子数。
* scenario与tuning\_list见上文train模式中的描述。

待上述配置项配置完成后，可以通过下述命令启动调优：

gs\_dbmind component xtuner tune -f connection.json

在使用tune和train模式前，用户需要先导入benchmark所需数据并检查benchmark能否正常跑通。调优过程结束后，调优程序会自动恢复调优前的数据库参数配置。

**常见问题**

* 数据库实例连接失败：请检查数据库实例的情况，是否数据库实例出现了问题或安全权限配置（pg\_hba.conf文件中的配置项）不正确。
* 重启失败：请检查数据库实例健康情况，确保数据库实例工作正常。
* 跑TPC-C作业时发现性能越来越慢：TPC-C等高并发场景下的压力测试，往往伴随着大量的数据修改。由于每一次测试并非是幂等的（TPC-C数据库数据量的增加、没有进行vacuum full清理掉失效元组、数据库没有触发checkpoint、没有进行drop cache等），因此一般建议TPC-C等伴随着较多数据写入的benchmark应该每隔一段时间（视具体并发量、执行时长的不同而异）重新导入一次数据，比较简单的方法是备份$PGDATA目录。
* TPC-C 跑作业时，TPC-C驱动脚本报异常 “TypeError: float() argument must be a string or a number, not 'NoneType'”（不能将None转换为float类型）：这是因为没有获取到TPC-C的压测返回结果，造成该问题的原因比较多，请首先手动检测是否能够跑通TPC-C并能够获取返回结果。若无上述问题，则建议将 TPC-C 驱动脚本中的命令列表中的 “sleep” 命令延迟时间设得更大一些。

#### INDEX-ADVISOR索引推荐

**单query索引推荐**

单query索引推荐功能支持用户在数据库中直接进行操作，本功能基于查询语句的语义信息和数据库的统计信息，对用户输入的单条查询语句生成推荐的索引。本功能涉及的函数接口如下。

| **函数名** | **参数** | **功能** |
| --- | --- | --- |
| gs\_index\_advise | SQL语句字符串 | 针对单条查询语句生成推荐索引。 |

仅支持单条SELECT类型的语句，不支持其他类型的SQL语句。

暂不支持列存表、段页式表、普通视图、物化视图、全局临时表以及密态数据库。

**单query索引示例**

* 使用上述函数，获取针对该query生成的推荐索引，推荐结果由索引的表名和列名组成。

select "table", "column" from gs\_index\_advise('SELECT c\_discount from bmsql\_customer where c\_w\_id = 10');

table | column   
----------------+----------  
 bmsql\_customer | c\_w\_id  
(1 row)

上述结果表明：应当在 bmsql\_customer 的 c\_w\_id 列上创建索引，例如可以通过下述SQL语句创建索引：

CREATE INDEX idx on bmsql\_customer(c\_w\_id);

* 某些SQL语句，也可能被推荐创建联合索引，例如：

select "table", "column" from gs\_index\_advise('select name, age, sex from t1 where age >= 18 and age < 35 and sex = ''f'';');

table | column  
-------+------------  
 t1 | age, sex  
(1 row)

上述语句表明应该在表 t1 上创建一个联合索引 (age, sex)， 则可以通过下述命令创建：

CREATE INDEX idx1 on t1(age, sex);

* 针对分区表可推荐具体索引类型，例如：

select "table", "column", "indextype" from gs\_index\_advise('select name, age, sex from range\_table where age = 20;');

table | column | indextype  
-------+--------+-----------  
 t1 | age | global  
(1 row)

系统函数gs\_index\_advise()的参数是文本型，如果参数中存在如单引号（'）等特殊字符，可以使用单引号（'）进行转义，可参考上述示例。

**虚拟索引**

虚拟索引功能支持用户在数据库中直接进行操作，将模拟真实索引的建立，避免真实索引创建所需的时间和空间开销，用户基于虚拟索引，可通过优化器评估该索引对指定查询语句的代价影响。

涉及的系统函数接口如下表所示：

| **函数名** | **参数** | **功能** |
| --- | --- | --- |
| hypopg\_create\_index | 创建索引语句的字符串 | 创建虚拟索引。 |
| hypopg\_display\_index | 无 | 显示所有创建的虚拟索引信息。 |
| hypopg\_drop\_index | 索引的oid | 删除指定的虚拟索引。 |
| hypopg\_reset\_index | 无 | 清除所有虚拟索引。 |
| hypopg\_estimate\_size | 索引的oid | 估计指定索引创建所需的空间大小。 |

涉及的GUC参数如下：

| **参数名** | **功能** | **默认值** |
| --- | --- | --- |
| enable\_hypo\_index | 是否开启虚拟索引功能 | off |

**示例**

1、创建测试表并插入数据。

CREATE TABLE bmsql\_customer (c\_w\_id int,name varchar,c\_discount int);  
INSERT INTO bmsql\_customer values(1,a,11);  
INSERT INTO bmsql\_customer values(2,b,12);  
INSERT INTO bmsql\_customer values(3,c,13);

使用函数hypopg\_create\_index创建虚拟索引。例如：

select \* from hypopg\_create\_index('create index on bmsql\_customer(c\_w\_id)');

indexrelid | indexname   
------------+------------------------------------  
 24040 | <24040>btree\_bmsql\_customer\_c\_w\_id  
(1 row)

2、开启GUC参数enable\_hypo\_index，该参数控制数据库的优化器进行EXPLAIN时是否考虑创建的虚拟索引。通过对特定的查询语句执行explain，用户可根据优化器给出的执行计划评估该索引是否能够提升该查询语句的执行效率。

开启GUC参数前，执行EXPLAIN + 查询语句：

explain SELECT c\_discount from bmsql\_customer where c\_w\_id = 3;

QUERY PLAN   
----------------------------------------------------------------------  
 Seq Scan on bmsql\_customer (cost=0.00..52963.06 rows=31224 width=4)  
 Filter: (c\_w\_id = 3)  
(2 rows)

开启GUC参数后，执行EXPLAIN + 查询语句：

set enable\_hypo\_index = on;  
explain SELECT c\_discount from bmsql\_customer where c\_w\_id = 3;

QUERY PLAN   
------------------------------------------------------------------------------------------------------------------  
 [Bypass]  
 Index Scan using <329726>btree\_bmsql\_customer\_c\_w\_id on bmsql\_customer (cost=0.00..39678.69 rows=31224 width=4)  
 Index Cond: (c\_w\_id = 3)  
(3 rows)

通过对比两个执行计划可以观察到，该索引预计会降低指定查询语句的执行代价，用户可考虑创建对应的真实索引。

3、使用函数hypopg\_display\_index展示所有创建过的虚拟索引（可选）。例如：

select \* from hypopg\_display\_index();

indexname | indexrelid | table | column   
  
------------------------------------+------------+----------------+----------  
 <24040>btree\_bmsql\_customer\_c\_w\_id | 24040 | bmsql\_customer | (c\_w\_id)  
 (2 rows)

4、使用函数hypopg\_estimate\_size估计虚拟索引创建所需的空间大小（单位：字节）（可选）。例如：

select \* from hypopg\_estimate\_size(329730);

hypopg\_estimate\_size   
----------------------  
 15687680  
(1 row)

5、删除虚拟索引。

使用函数hypopg\_drop\_index删除指定oid的虚拟索引。例如：

select \* from hypopg\_drop\_index(329726);

hypopg\_drop\_index   
-------------------  
 t  
(1 row)

使用函数hypopg\_reset\_index一次性清除所有创建的虚拟索引。例如：

select \* from hypopg\_reset\_index();

hypopg\_reset\_index   
--------------------  
   
(1 row)

* 执行EXPLAIN ANALYZE不会涉及虚拟索引功能。
* 创建的虚拟索引是数据库实例级别的，各个会话（session）之间可共享设置，关闭会话后虚拟索引仍可存在，但是重启数据库后将被清空。
* 本功能暂不支持视图、物化视图、列存表。

**WORKLOAD级别索引推荐**

对于workload级别的索引推荐，用户可通过运行数据库外的脚本使用此功能，本功能将包含有多条DML语句的workload作为输入，最终生成一批可对整体workload的执行表现进行优化的索引。同时，本功能提供从日志中抽取业务数据SQL流水的功能。

**前提条件**

* 数据库状态正常、客户端能够正常连接。
* 当前执行用户下安装有gsql工具，该工具路径已被加入到PATH环境变量中。
* 若使用本功能提供的业务数据抽取功能，需提前将要收集的节点的GUC参数按如下设置：
* log\_min\_duration\_statement = 0
* log\_statement= 'all'

业务数据抽取完毕建议将上述GUC参数复原，否则容易导致日志文件膨胀。

**业务数据抽取脚本使用步骤**

1、按前提条件中要求设置相关GUC参数。

2、执行根据日志抽取SQL语句的功能，命令如下：

gs\_dbmind component extract\_log [l LOG\_DIRECTORY] [f OUTPUT\_FILE] [p LOG\_LINE\_PREFIX] [-d DATABASE] [-U USERNAME][--start\_time] [--sql\_amount] [--statement] [--json] [--max\_reserved\_period] [--max\_template\_num]

其中的输入参数依次为：

* LOG\_DIRECTORY：pg\_log的存放目录。
* OUTPUT\_PATH：输出SQL流水文件文件的保存路径，即抽取出的业务数据存放的文件路径。
* LOG\_LINE\_PREFIX：指定每条日志信息的前缀格式。
* DATABASE：（可选）数据库名称，不指定默认所有数据库。
* USERNAME：（可选）用户名称，不指定默认所有用户。
* start\_time：（可选）日志收集的开始时间，不指定默认所有文件。
* sql\_amount：（可选）收集SQL数量的最大值，不指定默认收集所有SQL。
* statement：（可选）表示收集pg\_log日志中statement标识开头的SQL，不指定默认不收集。
* json：（可选）指定收集日志的文件存储格式为SQL归一化后的json，不指定默认格式每条SQL占一行。
* max\_reserved\_period：（可选）指定json模式下，增量收集日志中保留的模板的最大的更新时长，不指定默认都保留，单位/天。
* max\_template\_num：（可选）指定json模式下保留的最大模板数量，不指定默认都保留。

使用示例：

gs\_dbmind component extract\_log $GAUSSLOG/pg\_log/dn\_6001 sql\_log.txt '%m %c %d %p %a %x %n %e' -d postgres -U omm --start\_time '2021-07-06 00:00:00' --statement

若指定-d/-U参数，日志打印每条日志信息的前缀格式需包含%d、%u，若需要抽取事务，必须指定%p，详见log\_line\_prefix参数。max\_template\_num参数设置建议不超5000条，避免workload索引推荐执行时间过长。

3、将[1](#步骤1)中设置的GUC参数还原为设置前的值。

**索引推荐脚本使用步骤**

1、准备好包含有多条DML语句的文件作为输入的workload，文件中每条语句占据一行。用户可从数据库的离线日志中获得历史的业务语句。

2、运行本功能，命令如下：

gs\_dbmind component index\_advisor [p PORT] [d DATABASE] [f FILE] [--h HOST] [-U USERNAME] [-W PASSWORD][--schema SCHEMA]  
[--max\_index\_num MAX\_INDEX\_NUM][--max\_index\_storage MAX\_INDEX\_STORAGE] [--multi\_iter\_mode] [--multi\_node] [--json] [--driver] [--show\_detail]

其中的输入参数依次为：

* PORT：连接数据库的端口号。
* DATABASE：连接数据库的名字。
* FILE：包含workload语句的文件路径。
* HOST：（可选）连接数据库的主机号。
* USERNAME：（可选）连接数据库的用户名。
* PASSWORD：（可选）连接数据库用户的密码。
* SCHEMA：模式名称。
* MAX\_INDEX\_NUM：（可选）最大的索引推荐数目。
* MAX\_INDEX\_STORAGE：（可选）最大的索引集合空间大小。
* multi\_node：（可选）指定当前是否为分布式数据库实例。
* multi\_iter\_mode：（可选）算法模式，可通过是否设置该参数来切换算法。
* json：（可选）指定workload语句的文件路径格式为SQL归一化后的json，默认格式每条SQL占一行。
* driver：（可选）指定是否使用python驱动器连接数据库，默认gsql连接。
* show\_detail：（可选）是否显示当前推荐索引集合的详细优化信息。

例如：

gs\_dbmind component index\_advisor 6001 postgres tpcc\_log.txt --schema public --max\_index\_num 10 --multi\_iter\_mode

推荐结果为一批索引，以多个创建索引语句的格式显示在屏幕上，结果示例。

create index ind0 on public.bmsql\_stock(s\_i\_id,s\_w\_id);  
create index ind1 on public.bmsql\_customer(c\_w\_id,c\_id,c\_d\_id);  
create index ind2 on public.bmsql\_order\_line(ol\_w\_id,ol\_o\_id,ol\_d\_id);  
create index ind3 on public.bmsql\_item(i\_id);  
create index ind4 on public.bmsql\_oorder(o\_w\_id,o\_id,o\_d\_id);  
create index ind5 on public.bmsql\_new\_order(no\_w\_id,no\_d\_id,no\_o\_id);  
create index ind6 on public.bmsql\_customer(c\_w\_id,c\_d\_id,c\_last,c\_first);  
create index ind7 on public.bmsql\_new\_order(no\_w\_id);  
create index ind8 on public.bmsql\_oorder(o\_w\_id,o\_c\_id,o\_d\_id);  
create index ind9 on public.bmsql\_district(d\_w\_id);

multi\_node参数需严格按照当前数据库架构进行指定，否则推荐结果不全，甚至导致无推荐结果。

#### AI4DB：慢SQL根因分析

**功能描述**

慢SQL一直是数据运维中的痛点问题，如何有效诊断慢SQL根因是当前一大难题，工具结合Vastbase自身特点融合了现网DBA慢SQL诊断经验，该工具可以支持慢SQL根因15+，能同时按照可能性大小输出多个根因并提供针对性的建议。

**语法格式**

用户可以通过 --help 选项获得该模式的帮助信息。

gs\_dbmind component slow\_query\_diagnosis --help

usage: [-h] -c DIRECTORY [--query SLOW\_QUERY]  
 [--start-time TIMESTAMP\_IN\_MICROSECONDS]  
 [--end-time TIMESTAMP\_IN\_MICROSECONDS] [--retention-days DAYS]  
 {show,clean}  
  
Slow Query Diagnosis: Analyse the root cause of slow query  
  
positional arguments:  
 {show,clean} choose a functionality to perform  
  
optional arguments:  
 -h, --help show this help message and exit  
 -c DIRECTORY, --conf DIRECTORY  
 set the directory of configuration files  
 --query SLOW\_QUERY set a slow query you want to retrieve  
 --start-time TIMESTAMP\_IN\_MICROSECONDS  
 set the start time of a slow SQL diagnosis result to  
 be retrieved  
 --end-time TIMESTAMP\_IN\_MICROSECONDS  
 set the end time of a slow SQL diagnosis result to be  
 retrieved  
 --retention-days DAYS  
 clear historical diagnosis results and set the maximum  
 number of days to retain data

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| -h, --help | 帮助命令 | - |
| action | 动作参数 | show：结果展示clean：清理结果 |
| -c，--conf | 配置目录 | - |
| --query | 慢SQL文本 | \* |
| --start-time | 显示开始时间的时间戳，单位毫秒 | 非负整数 |
| --end-time | 显示结束时间的时间戳，单位毫秒 | 非负整数 |
| --retention-days | 清理天数级结果 | 非负实数 |

**注意事项**

* 数据库运行正常。
* 指标采集系统运行正常。
* 已完成confpath目录初始化配置，可参考SERVICE子命令章节中配置初始化目录内容。

**示例**

* 仅启动慢SQL诊断功能（输出Top3根因），启动命令如下（更多用法参考对service子命令的说明）：

gs\_dbmind service start -c confpath --only-run slow\_query\_diagnosis

* 交互式慢SQL诊断，命令如下：

gs\_dbmind component slow\_query\_diagnosis show -c confpath --query SQL --start-time timestamps0 --end-time timestamps1

* 手动清理历史预测结果，命令如下：

gs\_dbmind component slow\_query\_diagnosis clean -c confpath --retention-days DAYS

* 停止已启动的服务，命令如下：

gs\_dbmind service stop -c confpath

**常见问题**

* 如果用户对没有执行过的慢SQL执行交互式诊断命令，则无法给出诊断结果。
* exporter指标采集功能没有启动时运行慢SQL诊断功能，此时功能无法正常运行。
* 配置文件中的参数重新设置后，需要重新启动服务进程才能生效。

#### 趋势预测

**功能描述**

趋势预测功能模块主要实现基于历史时序数据预测未来时序变化趋势。该模块框架解耦，可以实现不同预测算法的灵活替换，并且该模块功能可以实现不同特征时序的算法自动选择，支持线性特征时序预测LR回归算法和非线性特征预测ARIMA算法。目前该模块可以覆盖线性时序、非线性时序和周期时序的准确预测。

**语法格式**

gs\_dbmind component forecast --help

usage: [-h] -c DIRECTORY [--metric-name METRIC\_NAME] [--host HOST] [--start-time TIMESTAMP\_IN\_MICROSECONDS] [--end-time TIMESTAMP\_IN\_MICROSECONDS] [--retention-days DAYS]  
 {show,clean}  
  
Workload Forecasting: Forecast monitoring metrics  
  
positional arguments:  
 {show,clean} choose a functionality to perform  
  
optional arguments:  
 -h, --help show this help message and exit  
 -c DIRECTORY, --conf DIRECTORY  
 set the directory of configuration files  
 --metric-name METRIC\_NAME  
 set a metric name you want to retrieve  
 --host HOST set a host you want to retrieve  
 --start-time TIMESTAMP\_IN\_MICROSECONDS  
 set a start time of for retrieving  
 --end-time TIMESTAMP\_IN\_MICROSECONDS  
 set a end time of for retrieving  
 --retention-days DAYS  
 clear historical diagnosis results and set the maximum number of days to retain data

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| -h, --help | 帮助命令 | - |
| action | 动作参数 | show：结果展示clean：清理结果 |
| -c，--conf | 配置目录 | - |
| --metric-name | 指定显示指标名，用于过滤 | - |
| --host | 指定服务IP和端口号，用于过滤 | - |
| --start-time | 显示开始时间的时间戳，单位毫秒 | 非负实数 |
| --end-time | 显示结束时间的时间戳，单位毫秒 | 非负实数 |
| --retention-days | 保留结果天数 | 非负实数 |

**注意事项**

* 数据库运行正常。
* 指标采集系统运行正常。
* 已完成confpath目录初始化配置，可参考SERVICE子命令章节中配置初始化目录内容。

**示例**

* 仅启动趋势预测功能，启动命令如下（更多用法参考对service子命令的说明）：

gs\_dbmind service start -c confpath --only-run forecast

* 用户交互式趋势预测，查看timestamps0到timestamps1时间段内的预测结果，命令如下：

gs\_dbmind component forecast show -c confpath --start-time timestamps0 --end-time timestamps1

* 用户手动清理手动清理历史预测结果，命令如下：

gs\_dbmind component forecast clean -c confpath --retention-days DAYS

* 停止已启动的服务，命令如下：

gs\_dbmind service stop -c confpath

**常见问题**

* 综合实际业务与模型预测效果考虑，趋势预测时长建议不要太短，建议大于3600秒（如果指标采集周期为15秒，则数据量为240个），否则预测效果会变差，并且数据量极小时，服务会异常，因此默认配置为3600秒
* 配置文件中的参数重新设置后，需要重新启动服务进程才能生效。

#### SQL DIAG：慢SQL发现

**功能描述**

SQLdiag是一个SQL语句执行时间预测工具，通过模板化方法或者深度学习方法，实现在不获取SQL语句执行计划的前提下，依据语句逻辑相似度与历史执行记录，预测SQL语句的执行时间并以此发现异常SQL。

**语法格式**

gs\_dbmind component sqldiag --help

usage: [-h] [-f CSV\_FILE] [--predicted-file PREDICTED\_FILE]  
 [--model {template,dnn}] --model-path MODEL\_PATH  
 [--config-file CONFIG\_FILE]  
 {train,predict,finetune}  
  
SQLdiag integrated by vastbase.  
  
positional arguments:  
 {train,predict,finetune}  
 The training mode is to perform feature extraction and  
 model training based on historical SQL statements. The  
 prediction mode is to predict the execution time of a  
 new SQL statement through the trained model.  
  
optional arguments:  
 -h, --help show this help message and exit  
 -f CSV\_FILE, --csv-file CSV\_FILE  
 The data set for training or prediction. The file  
 format is CSV. If it is two columns, the format is  
 (SQL statement, duration time). If it is three  
 columns, the format is (timestamp of SQL statement  
 execution time, SQL statement, duration time).  
 --predicted-file PREDICTED\_FILE  
 The file path to save the predicted result.  
 --model {template,dnn}  
 Choose the model model to use.  
 --model-path MODEL\_PATH  
 The storage path of the model file, used to read or  
 save the model file.  
 --config-file CONFIG\_FILE

**参数说明**

| **参数** | **参数说明** | **取值范围** |
| --- | --- | --- |
| -f | 训练或预测文件位置 | - |
| --predicted-file | 预测结果存储位置 | - |
| --model | 模型选择 | template, dnn |
| --model-path | 训练模型存储位置 | - |

**注意事项**

* 需要保证用户提供训练数据。
* 如果用户通过提供的工具收集训练数据，则需要启用WDR功能，涉及到的参数为track\_stmt\_stat\_level和log\_min\_duration\_statement，具体情况见下面小结。
* 为保证预测准确率，用户提供的历史语句日志应尽可能全面并具有代表性。

**SQL流水采集方法**

本工具需要用户提前准备数据，训练数据格式如下，每个样本通过换行符分隔：

SQL,EXECUTION\_TIME

预测数据格式如下：

SQL

其中SQL表示SQL语句的文本，EXECUTION\_TIME表示SQL语句的执行时间，样例数据见sample\_data中的train.csv和predict.csv。

用户可以按照要求格式自己收集训练数据，工具也提供了脚本自动采集（load\_sql\_from\_rd），该脚本基于WDR报告获取SQL信息，涉及到的参数有log\_min\_duration\_statement和track\_stmt\_stat\_level：

* 其中log\_min\_duration\_statement表示慢SQL阈值，如果为0则全量收集，时间单位为毫秒；
* track\_stmt\_stat\_level表示信息捕获的级别，建议设置为track\_stmt\_stat\_level='L0,L0'

参数开启后，可能占用一定的系统资源，但一般不大。持续的高并发场景可能产生5%以内的损耗，数据库并发较低的场景，性能损耗可忽略。下述脚本存在于sqldiag根目录（$GAUSSHOME/bin/components/sqldiag）中。

使用脚本获取训练集方式：

load\_sql\_from\_wdr.py [-h] --port PORT --start\_time START\_TIME  
 --finish\_time FINISH\_TIME [--save\_path SAVE\_PATH]

**操作步骤**

1、提供历史日志以供模型训练

2、进行训练与预测操作。

基于模板法的训练与预测：

gs\_dbmind component sqldiag [train, predict] -f FILE --model template --model-path template\_model\_path

基于DNN的训练与预测：

gs\_dbmind component sqldiag [train, predict] -f FILE --model dnn --model-path dnn\_model\_path

**示例**

* 使用提供的测试数据进行模板化训练：

gs\_dbmind component sqldiag train -f ./sample\_data/train.csv --model template --model-path ./template

* 使用提供的测试数据进行模板化预测：

gs\_dbmind component sqldiag predict -f ./sample\_data/predict.csv --model template --model-path ./template --predicted-file ./result/t\_result

* 使用提供的测试数据进行模板化模型更新：

gs\_dbmind component sqldiag finetune -f ./sample\_data/train.csv --model template --model-path ./template

* 使用提供的测试数据进行DNN训练：

gs\_dbmind component sqldiag train -f ./sample\_data/train.csv --model dnn --model-path ./dnn\_model

* 使用提供的测试数据进行DNN预测：

gs\_dbmind component sqldiag predict -f ./sample\_data/predict.csv --model dnn --model-path ./dnn\_model --predicted-file

* 使用提供的测试数据进行DNN模型更新：

gs\_dbmind component sqldiag finetune -f ./sample\_data/train.csv --model dnn --model-path ./dnn\_model

**常见问题**

* 训练场景失败：请检查历史日志文件路径是否正确，且文件格式符合上文规定。
* 预测场景失败：请检查模型路径是否正确。确保待预测负载文件格式正确。

## DB4AI: 数据库驱动AI

DB4AI是指利用数据库的能力驱动AI任务，实现数据存储、技术栈的同构。通过在数据库内集成AI算法，令Vastbase具备数据库原生AI计算引擎、模型管理、AI算子、AI原生执行计划的能力，为用户提供普惠AI技术。不同于传统的AI建模流程，DB4AI“一站式”建模可以解决数据在各平台的反复流转问题，同时简化开发流程，并可通过数据库规划出最优执行路径，让开发者更专注于具体业务和模型的调优上，具备同类产品不具备的易用性与性能优势。

### 原生DB4AI引擎

Vastbase当前版本支持了原生DB4AI能力，通过引入原生AI算子，简化操作流程，充分利用数据库优化器、执行器的优化与执行能力，获得高性能的数据库内模型训练能力。更简化的模型训练与预测流程、更高的性能表现，让开发者在更短时间内能更专注于模型的调优与数据分析上，而避免了碎片化的技术栈与冗余的代码实现。

**关键字解析**

**表 1** DB4AI语法及关键字

| **名称** | | **描述** |
| --- | --- | --- |
| 语法 | CREATE MODEL | 创建模型并进行训练，同时保存模型。 |
| PREDICT BY | 利用已有模型进行推断。 |
| DROP MODEL | 删除模型。 |
| 关键字 | TARGET | 训练/推断任务的目标列名。 |
| FEATURES | 训练/推断任务的数据特征列名。 |
| MODEL | 训练任务的模型名称。 |

**使用指导**

1、 本版本支持的算法概述。

当前版本的DB4AI新增支持算法如下：

**表 2** 支持算法

| **优化算法** | **算法** |
| --- | --- |
| GD | logistic\_regression |
| linear\_regression |
| svm\_classification |
| PCA |
| multiclass |
| Kmeans | kmeans |
| xgboost | xgboost\_regression\_logistic |
| xgboost\_binary\_logistic |
| xgboost\_regression\_squarederror |
| xgboost\_regression\_gamma |

2、 模型训练语法说明。

* CREATE MODEL
* 使用“CREATE MODEL”语句可以进行模型的创建和训练。模型训练SQL语句，选用公开数据集鸢尾花数据集iris。
* 以multiclass为例，训练一个模型。从tb.*iris训练集中指定sepal.*length, sepal.*width,petal.*length,petal.*widt为特征列，使用multiclass算法，创建并保存模型iris.*classification.\_model。

vastbase=# CREATE MODEL iris\_classification\_model USING xgboost\_regression\_logistic FEATURES sepal\_length, sepal\_width,petal\_length,petal\_width TARGET target\_type < 2 FROM tb\_iris\_1 WITH nthread=4, max\_depth=8;  
MODEL CREATED. PROCESSED 1

上述命令中：

* “CREATE MODEL”语句用于模型的训练和保存。
* USING关键字指定算法名称。
* FEATURES用于指定训练模模型的特征，需根据训练数据表的列名添加。
* TARGET指定模型的训练目标，它可以是训练所需数据表的列名，也可以是一个表达式，例如: price .> 10000。
* WITH用于指定训练模型时的超参数。当超参未被用户进行设置的时候，框架会使用默认数值。
* 针对不同的算子，框架支持不同的超参组合：
* **表 3** 算子支持的超参

| **算子** | **超参** |
| --- | --- |
| GD  (logistic\_regression、linear\_regression、svm\_classification) | optimizer(char); verbose(bool); max\_iterations(int); max\_seconds(double); batch\_size(int); learning\_rate(double); decay(double); tolerance(double)  其中，SVM限定超参lambda(double) |
| Kmeans | max\_iterations(int); num\_centroids(int); tolerance(double); batch\_size(int); num\_features(int); distance\_function(char); seeding\_function(char); verbose(int);seed(int) |
| GD(pca) | batch\_size(int);max\_iterations(int);max\_seconds(int);tolerance(float8);verbose(bool);number\_components(int);seed(int) |
| GD(multiclass) | classifier(char)  注意：multiclass的其他超参种类取决于选择的分类器中类 |
| xgboost\_regression\_logistic、xgboost\_binary\_logistic、xgboost\_regression\_squarederror、xgboost\_regression\_gamma | batch\_size(int);booster(char);tree\_method(char);eval\_metric(char\*);seed(int);nthread(int);max\_depth(int);gamma(float8);eta(float8);min\_child\_weight(int);verbosity(int) |

当前各个超参数设置的默认值和取值范围如下：

* **表 4** 超参的默认值以及取值范围

| **算子** | **超参(默认值)** | **取值范围** | **超参描述** |
| --- | --- | --- | --- |
| GD:  logistic\_regression、linear\_regression、svm\_classification、pca | optimizer = gd（梯度下降法） | gd/ngd（自然梯度下降） | 优化器 |
| verbose = false | T/F | 日志显示 |
| max\_iterations = 100 | (0, 10000] | 最大迭代次数 |
| max\_seconds = 0 (不对运行时长设限制) | [0,INT\_MAX\_VALUE] | 运行时长 |
| batch\_size = 1000 | (0, 1048575] | 一次训练所选取的样本数 |
| learning\_rate = 0.8 | (0, DOUBLE\_MAX\_VALUE] | 学习率 |
| decay = 0.95 | (0, DOUBLE\_MAX\_VALUE] | 权值衰减率 |
| tolerance = 0.0005 | (0, DOUBLE\_MAX\_VALUE] | 公差 |
| seed = 0（对seed取随机值） | [0, INT\_MAX\_VALUE] | 种子 |
| just for linear、SVM：kernel = “linear” | linear/gaussian/polynomial | 核函数 |
| just for linear、SVM：components = MAX(2\*features, 128) | [0, INT\_MAX\_VALUE] | 高维空间维数 |
| just for linear、SVM：gamma = 0.5 | (0, DOUBLE\_MAX\_VALUE] | gaussian核函数参数 |
| just for linear、SVM：degree = 2 | [2, 9] | polynomial核函数参数 |
| just for linear、SVM：coef0 = 1.0 | [0, DOUBLE\_MAX\_VALUE] | polynomial核函数的参数 |
| just for SVM：lambda = 0.01 | (0, DOUBLE\_MAX\_VALUE) | 正则化参数 |
| just for pca： number\_components | （0，INT\_MAX\_VALUE] | 降维的目标维度 |
| GD:  multiclass | classifier=“svm\_classification” | svm\_classification\logistic\_regression | 多分类任务的分类器 |
| Kmeans | max\_iterations = 10 | [1, 10000] | 最大迭代次数 |
| num\_centroids = 10 | [1, 1000000] | 簇的数目 |
| tolerance = 0.00001 | (0,1] | 中心点误差 |
| batch\_size = 10 | [1,1048575] | 一次训练所选取的样本数 |
| num\_features = 2 | [1, INT\_MAX\_VALUE] | 输入样本特征数 |
| distance\_function = “L2\_Squared” | L1\L2\L2\_Squared\Linf | 正则化方法 |
| seeding\_function = “Random++” | “Random++”\“KMeans||” | 初始化种子点方法 |
| verbose = 0U | { 0, 1, 2 } | 冗长模式 |
| seed = 0U | [0, INT\_MAX\_VALUE] | 种子 |
| xgboost:  xgboost\_regression\_logistic、  xgboost\_binary\_logistic、  xgboost\_regression\_gamma、xgboost\_regression\_squarederror | n\_iter=10 | (0, 10000] | 迭代次数 |
| batch\_size=10000 | (0, 1048575] | 一次训练所选取的样本数 |
| booster=“gbtree” | gbtree\gblinear\dart | booster种类 |
| tree\_method=“auto” | auto\exact\approx\hist\gpu\_hist  注意：gpu\_hist参数需要相应的库GPU版本，否则DB4AI平台不支持该值。 | 树构建算法 |
| eval\_metric=“rmse” | rmse\rmsle\map\mae\auc\aucpr | 验证数据的评估指标 |
| seed=0 | [0, 100] | 种子 |
| nthread=1 | (0, MAX\_MEMORY\_LIMIT] | 并发量 |
| max\_depth=5 | (0, MAX\_MEMORY\_LIMIT] | 树的最大深度，该超参仅对树型booster生效。 |
| gamma=0.0 | [0, 1] | 叶节点上进行进一步分区所需的最小损失减少 |
| eta=0.3 | [0, 1] | 更新中使用的步长收缩，以防止过拟合 |
| min\_child\_weight=1 | [0, INT\_MAX\_VALUE] | 孩子节点中所需的实例权重的最小总和 |
| verbosity=1 | 0 (silent)\1 (warning)\2 (info)\3 (debug) | 打印信息的详细程度 |
| MAX\_MEMORY\_LIMIT = 最大内存加载的元组数量 | | | |
| GS\_MAX\_COLS = 数据库单表最大属性数量 | | | |

* 模型保存成功，则返回创建成功信息：

MODEL CREATED. PROCESSED x

3、 查看模型信息。

当训练完成后模型会被存储到系统表gs.*model.*warehouse中。系统表gs.*model.*warehouse可以查看到关于模型本身和训练过程的相关信息。

关于模型的详细描述信息以二进制的形式存储在系统表中，用户可用过使用函数gs.*explain.*model完成对模型的查看，语句如下：

vastbase=# select \* from gs\_explain\_model("iris\_classification\_model");  
 DB4AI MODEL  
-------------------------------------------------------------  
 Name: iris\_classification\_model  
 Algorithm: xgboost\_regression\_logistic  
 Query: CREATE MODEL iris\_classification\_model  
 USING xgboost\_regression\_logistic  
 FEATURES sepal\_length, sepal\_width,petal\_length,petal\_width  
 TARGET target\_type < 2  
 FROM tb\_iris\_1  
 WITH nthread=4, max\_depth=8;  
 Return type: Float64  
 Pre-processing time: 0.000000  
 Execution time: 0.001443  
 Processed tuples: 78  
 Discarded tuples: 0  
 n\_iter: 10  
 batch\_size: 10000  
 max\_depth: 8  
 min\_child\_weight: 1  
 gamma: 0.0000000000  
 eta: 0.3000000000  
 nthread: 4  
 verbosity: 1  
 seed: 0  
 booster: gbtree  
 tree\_method: auto  
 eval\_metric: rmse  
 rmse: 0.2648450136  
 model size: 4613

4、 利用已存在的模型做推断任务。

使用“SELECT”和“PREDICT BY”关键字利用已有模型完成推断任务。

查询语法：SELECT…PREDICT BY….(FEATURES….)…FROM…;

vastbase=# SELECT id, PREDICT BY iris\_classification (FEATURES sepal\_length,sepal\_width,petal\_length,petal\_width) as "PREDICT" FROM tb\_iris limit 3;  
   
id | PREDICT  
-----+---------  
 84 | 2  
 85 | 0  
 86 | 0  
(3 rows)

针对相同的推断任务，同一个模型的结果是大致稳定的。且基于相同的超参数和训练集训练的模型也具有稳定性，同时AI模型训练存在随机成分（每个batch的数据分布、随机梯度下降），所以不同的模型间的计算表现、结果允许存在小的差别。

5、 查看执行计划。

使用explain语句可对“CREATE MODEL”和“PREDICT BY”的模型训练或预测过程中的执行计划进行分析。Explain关键字后可直接拼接CREATE MODEL/ PREDICT BY语句（子句），也可接可选的参数，支持的参数如下：

**表 5** EXPLAIN支持的参数

| **参数名** | **描述** |
| --- | --- |
| ANALYZE | 布尔型变量，追加运行时间、循环次数等描述信息 |
| VERBOSE | 布尔型变量，控制训练的运行信息是否输出到客户端 |
| COSTS | 布尔型变量 |
| CPU | 布尔型变量 |
| DETAIL | 布尔型变量，不可用。 |
| NODES | 布尔型变量，不可用 |
| NUM\_NODES | 布尔型变量，不可用 |
| BUFFERS | 布尔型变量 |
| TIMING | 布尔型变量 |
| PLAN | 布尔型变量 |
| FORMAT | 可选格式类型：TEXT / XML / JSON / YAML |

示例：

vastbase=# Explain CREATE MODEL patient\_logisitic\_regression USING logistic\_regression FEATURES second\_attack, treatment TARGET trait\_anxiety > 50 FROM patients WITH batch\_size=10, learning\_rate = 0.05;  
 QUERY PLAN  
-------------------------------------------------------------------------  
 Train Model - logistic\_regression (cost=0.00..0.00 rows=0 width=0)  
 -> Materialize (cost=0.00..41.08 rows=1776 width=12)  
 -> Seq Scan on patients (cost=0.00..32.20 rows=1776 width=12)  
(3 rows)

6、 异常场景。

**训练阶段**

* 场景一：当超参数的设置超出取值范围，模型训练失败，返回ERROR，并提示错误，例如：

vastbase=# CREATE MODEL patient\_linear\_regression USING linear\_regression FEATURES second\_attack,treatment TARGET trait\_anxiety FROM patients WITH optimizer='aa';  
ERROR: Invalid hyperparameter value for optimizer. Valid values are: gd, ngd.

* 场景二：当模型名称已存在，模型保存失败，返回ERROR，并提示错误原因，例如：

vastbase=# CREATE MODEL patient\_linear\_regression USING linear\_regression FEATURES second\_attack,treatment TARGET trait\_anxiety FROM patients;  
ERROR: The model name "patient\_linear\_regression" already exists in gs\_model\_warehouse.

* 场景三：FEATURE或者TARGETS列是.\*，返回ERROR，并提示错误原因，例如：

vastbase=# CREATE MODEL patient\_linear\_regression USING linear\_regression FEATURES \* TARGET trait\_anxiety FROM patients;  
ERROR: FEATURES clause cannot be \*  
-----------------------------------------------------------------------------------------------------------------------  
vastbase=# CREATE MODEL patient\_linear\_regression USING linear\_regression FEATURES second\_attack,treatment TARGET \* FROM patients;  
ERROR: TARGET clause cannot be \*

* 场景四：对于无监督学习方法使用TARGET关键字，或者在监督学习方法中不适用TARGET关键字，均会返回ERROR，并提示错误原因，例如：

vastbase=# CREATE MODEL patient\_linear\_regression USING linear\_regression FEATURES second\_attack,treatment FROM patients;  
ERROR: Supervised ML algorithms require TARGET clause  
-----------------------------------------------------------------------------------------------------------------------------  
CREATE MODEL patient\_linear\_regression USING linear\_regression TARGET trait\_anxiety FROM patients;   
ERROR: Supervised ML algorithms require FEATURES clause

* 场景五：当进行分类任务时TARGET列的分类只有1种情况，会返回ERROR，并提示错误原因，例如：

vastbase=# CREATE MODEL ecoli\_svmc USING multiclass FEATURES f1, f2, f3, f4, f5, f6, f7 TARGET cat FROM (SELECT \* FROM db4ai\_ecoli WHERE cat='cp');  
ERROR: At least two categories are needed

* 场景六：DB4AI在训练过程中会过滤掉含有空值的数据，当参与训练的模型数据为空的时候，会返回ERROR，并提示错误原因，例如：

vastbase=# create model iris\_classification\_model using xgboost\_regression\_logistic features message\_regular target error\_level from error\_code;  
ERROR: Training data is empty, please check the input data.

* 场景七：DB4AI的算法对于支持的数据类型是有限制的。当数据类型不在支持白名单中，会返回ERROR，并提示非法的oid，可通过pg.\_type查看OID确定非法的数据类型，例如：

vastbase=# CREATE MODEL ecoli\_svmc USING multiclass FEATURES f1, f2, f3, f4, f5, f6, f7, cat TARGET cat FROM db4ai\_ecoli ;  
ERROR: Oid type 1043 not yet supported

* 场景八：当GUC参数statement.*timeout设置了时长，训练超时执行的语句将被终止：执行CREATE MODEL语句。训练集的大小、训练轮数.(iteration.)、提前终止条件.(tolerance、max.*seconds.)、并行线程数.(nthread.)等参数都会影响训练时长。当时长超过数据库限制，语句被终止模型训练失败。

**模型解析**

* 场景九：当模型名在系统表中查找不到，数据库会报ERROR，例如：

vastbase=# select gs\_explain\_model("ecoli\_svmc");  
ERROR: column "ecoli\_svmc" does not exist

**推断阶段**

* 场景十：当模型名在系统表中查找不到，数据库会报ERROR，例如：

vastbase=# select id, PREDICT BY patient\_logistic\_regression (FEATURES second\_attack,treatment) FROM patients;  
ERROR: There is no model called "patient\_logistic\_regression".

* 场景十一：当做推断任务FEATURES的数据维度和数据类型与训练集存在不一致，将报ERROR，并提示错误原因，例如：

vastbase=# select id, PREDICT BY patient\_linear\_regression (FEATURES second\_attack) FROM patients;  
ERROR: Invalid number of features for prediction, provided 1, expected 2  
CONTEXT: referenced column: patient\_linear\_regression\_pred  
-------------------------------------------------------------------------------------------------------------------------------------  
vastbase=# select id, PREDICT BY patient\_linear\_regression (FEATURES 1,second\_attack,treatment) FROM patients;  
ERROR: Invalid number of features for prediction, provided 3, expected 2  
CONTEXT: referenced column: patient\_linear\_regression\_pre

fig: **说明：**   
DB4AI特性需要读取数据参与计算，不适用于密态数据库等情况。

### 全流程AI

传统的AI任务往往具有多个流程，如数据的收集过程包括数据的采集、数据清洗、数据存储等，在算法的训练过程中又包括数据的预处理、训练、模型的保存与管理等。其中，对于模型的训练过程，又包括超参数的调优过程。诸如此类机器学习模型生命周期的全过程，可大量集成于数据库内部。在距离数据存储侧最近处进行模型的训练、管理、优化等流程，在数据库端提供SQL语句式的开箱即用的AI全声明周期管理的功能，称之为全流程AI.

Vastbase实现了部分全流程AI的功能，将在本章节中详细展开。

#### PLPython Fenced模式

在fenced模式中添加plpython非安全语言。在数据库编译时需要将python集成进数据库中，在configure阶段加入--with-python选项。同时也可指定安装plpython的python路径，添加选项--with-includes='/python-dir=path'。

在启动数据库之前配置GUC参数unix\_socket\_directory ，指定unix\_socket进程间通讯的文件地址。用户需要提前在user-set-dir-path下创建文件夹，并将文件夹权限修改为可读可写可执行状态。

unix\_socket\_directory = '/user-set-dir-path'

配置完成，启动数据库。

将plpython加入数据库编译，并设置好GUC参数unix\_socket\_directory后，在启动数据库的过程中，自动创建fenced-Master进程。在数据库不进行python编译的情况下，fenced模式需要手动拉起master进程，在GUC参数设置完成后，输入创建master进程命令。

启动fenced-Master进程，命令为：

gaussdb --fenced -k /user-set-dir-path -D /user-set-dir-path &

完成fence模式配置，针对plpython-fenced UDF数据库将在fenced-worker进程中执行UDF计算。

**使用指导**

**创建extension**

* 当编译的plpython为python2时：

vastbase=### create extension plpythonu;  
CREATE EXTENSION

* 当编译的plpython为python3时：

vastbase=### create extension plpython3u;  
CREATE EXTENSION

下面示例是以python2为例。

* 创建plpython-fenced UDF

create or replace function pymax(a int, b int)  
returns INT  
language plpythonu fenced  
as $$  
import numpy  
if a > b:  
return a;  
else:  
return b;  
$$;  
CREATE FUNCTION

* 查看UDF信息

select \* from pg\_proc where proname='pymax';  
  
-[ RECORD 1 ]----+--------------  
proname | pymax  
pronamespace | 2200  
proowner | 10  
prolang | 16388  
procost | 100  
prorows | 0  
provariadic | 0  
protransform | -  
proisagg | f  
proiswindow | f  
prosecdef | f  
proleakproof | f  
proisstrict | f  
proretset | f  
provolatile | v  
pronargs | 2  
pronargdefaults | 0  
prorettype | 23  
proargtypes | 23 23  
proallargtypes |  
proargmodes |  
proargnames | {a,b}  
proargdefaults |  
prosrc |  
 | import numpy  
 | if a > b:  
 | return a;  
 | else:  
 | return b;  
 |  
probin |  
proconfig |  
proacl |  
prodefaultargpos |  
fencedmode | t  
proshippable | f  
propackage | f  
prokind | f  
proargsrc |

**运行UDF**

* 创建一个数据表：

vastbase=# create table temp (a int ,b int) ;  
CREATE TABLE  
vastbase=# insert into temp values (1,2),(2,3),(3,4),(4,5),(5,6);  
INSERT 0 5

* 运行UDF：

vastbase=# select pymax(a,b) from temp;  
 pymax  
-------  
 2  
 3  
 4  
 5  
 6  
(5 rows)

#### DB4AI-Snapshots数据版本管理

DB4AI-Snapshots是DB4AI模块用于管理数据集版本的功能。通过DB4ai-Snapshots组件，开发者可以简单、快速地进行特征筛选、类型转换等数据预处理操作，同时还可以像git一样对训练数据集进行版本控制。数据表快照创建成功后可以像视图一样进行使用，但是一经发布后，数据表快照便固化为不可变的静态数据，如需修改该数据表快照的内容，需要创建一个版本号不同的新数据表快照。

**DB4AI-Snapshots的生命周期**

DB4AI-Snapshots的状态包括published、archived以及purged。其中，published可以用于标记该DB4AI-Snapshots 已经发布，可以进行使用。archived表示当前 DB4AI-Snapshots 处于“存档期”，一般不进行新模型的训练，而是利用旧数据对新的模型进行验证。purged则是该DB4AI-Snapshots 已经被删除的状态，在数据库系统中无法再检索到。

需要注意的是快照管理功能是为了给用户提供统一的训练数据，不同团队成员可以使用给定的训练数据来重新训练机器学习模型，方便用户间协同。为此私有用户和三权分立状态(enableSeparationOfDuty=ON)等涉及不支持用户数据转写等情况将不支持Snapshot特性。

用户可以通过“CREATE SNAPSHOT”语句创建数据表快照，创建好的快照默认即为published状态。可以采用两种模式创建数据表快照，即为MSS以及CSS模式，它们可以通过GUC参数db4ai\_snapshot\_mode进行配置。对于MSS模式，它是采用物化算法进行实现的，存储了原始数据集的数据实体；CSS则是基于相对计算算法实现的，存储的是数据的增量信息。数据表快照的元信息存储在DB4AI的系统目录中。可以通过db4ai.snapshot 系统表查看到。

可以通过“ARCHIVE SNAPSHOT”语句将某一个数据表快照标记为archived状态，可以通过“PUBLISH SNAPSHOT”语句将其再度标记为published状态。标记数据表快照的状态，是为了帮助数据科学家进行团队合作使用的。

当一个数据表快照已经丧失存在价值时，可以通过“PURGE SNAPSHOT”语句删除它，以便永久删除其数据并恢复存储空间。

**DB4AI-Snapshots使用指导**

1、 创建表以及插入表数据。

数据库内存在已有的数据表，可根据该已有的数据表创建对应的数据表快照。为了后续演示，在此处新建一个名为 t1 的数据表，并向其中插入测试数据。

create table t1 (id int, name varchar);  
insert into t1 values (1, 'zhangsan');  
insert into t1 values (2, 'lisi');  
insert into t1 values (3, 'wangwu');  
insert into t1 values (4, 'lisa');  
insert into t1 values (5, 'jack');

通过SQL语句，查询搭配数据表内容。

SELECT \* FROM t1;  
id | name  
----+----------  
 1 | zhangsan  
 2 | lisi  
 3 | wangwu  
 4 | lisa  
 5 | jack  
(5 rows)

2、 使用DB4AI-Snapshots。

* 创建DB4AI-Snapshots

示例1：CREATE SNAPSHOT…AS

示例如下，其中，默认版本分隔符为 “@”, 默认子版本分割符为 “.”，该分割符可以分别通过GUC参数db4ai\_snapshot\_version\_delimiter以及db4ai\_snapshot\_version\_separator进行设置。

create snapshot s1@1.0 comment is 'first version' as select \* from t1;  
schema | name  
--------+--------  
 public | s1@1.0  
(1 row)

上述结果提示已经创建了数据表 s1的快照，版本号为 1.0。创建好后的数据表快照可以像使用一般视图一样进行查询，但不支持通过“INSERT INTO”语句进行更新。例如下面几种语句都可以查询到数据表快照s1的对应版本 1.0的内容：

SELECT \* FROM s1@1.0;  
SELECT \* FROM public.s1@1.0;  
SELECT \* FROM public . s1 @ 1.0;  
id | name  
----+----------  
 1 | zhangsan  
 2 | lisi  
 3 | wangwu  
 4 | lisa  
 5 | jack  
(5 rows)

可以通过下列SQL语句修改数据表t1的内容：

UPDATE t1 SET name = 'tom' where id = 4;  
insert into t1 values (6, 'john');  
insert into t1 values (7, 'tim');

再检索数据表t1的内容时，发现虽然数据表t1的内容已经发生变化，但是数据表快照 s1@1.0 版本的查询结果并未发生变化。由于数据表t1的数据已经发生了改变，如果将当前数据表的内容作为版本2.0，则可创建快照**s1@2.0**,创建的SQL语句如下：

create snapshot s1@2.0 as select \* from t1;

通过上述例子，我们可以发现，数据表快照可以固化数据表的内容，避免中途对数据的改动造成机器学习模型训练时的不稳定，同时可以避免多用户同时访问、修改同一个表时造成的锁冲突。

示例2：CREATE SNAPSHOT…FROM

SQL语句可以对一个已经创建好的数据表快照进行继承，利用在此基础上进行的数据修改产生一个新的数据表快照。例如：

create snapshot s1@3.0 from @1.0 comment is 'inherits from @1.0' using (INSERT VALUES(6, 'john'), (7, 'tim'); DELETE WHERE id = 1);  
schema | name  
--------+--------  
 public | s1@3.0  
(1 row)

其中，“@”为数据表快照的版本分隔符，from子句后加上已存在的数据表快照，用法为“@”+版本号，USING关键字后加入可选的几个操作关键字（INSERT …/UPDATE …/DELETE …/ALTER …）,其中 “INSERT INTO”以及“DELETE FROM”语句中的“INTO”、“FROM”等与数据表快照名字相关联的子句可以省略。

示例中，基于前述s1@1.0快照，插入2条数据，删除1条新的数据，新生成的快照s1@3.0，检索该s1@3.0：

SELECT \* FROM s1@3.0;  
id | name  
----+----------  
 2 | lisi  
 3 | wangwu  
 4 | lisa  
 5 | jack  
 6 | john  
 7 | tim  
(7 rows)

* 删除数据表快照SNAPSHOT

purge snapshot s1@3.0;  
schema | name  
--------+--------  
 public | s1@3.0  
(1 row)

此时，已经无法再从s1@3.0 中检索到数据了，同时该数据表快照在db4ai.snapshot视图中的记录也会被清除。删除该版本的数据表快照不会影响其他版本的数据表快照。

* 从数据表快照中采样

示例：从snapshot s1中抽取数据，使用0.5抽样率。

sample snapshot s1@2.0 stratify by name as nick at ratio .5;  
schema | name  
--------+------------  
 public | s1nick@2.0  
(1 row)

可以利用该功能创建训练集与测试集，例如：

SAMPLE SNAPSHOT s1@2.0 STRATIFY BY name AS \_test AT RATIO .2, AS \_train AT RATIO .8 COMMENT IS 'training';  
schema | name  
--------+----------------  
 public | s1\_test@2.0  
 public | s1\_train@2.0  
(2 rows)

* 发布数据表快照

采用下述SQL语句将数据表快照 s1@2.0 标记为published 状态：

publish snapshot s1@2.0;  
schema | name  
--------+--------  
 public | s1@2.0  
(1 row)

* 存档数据表快照

采用下述语句可以将数据表快照标记为 archived 状态：

archive snapshot s1@2.0;  
schema | name  
--------+--------  
 public | s1@2.0  
(1 row)

可以通过db4ai-snapshots提供的视图查看当前数据表快照的状态以及其他信息：

select \* from db4ai.snapshot;  
id | parent\_id | matrix\_id | root\_id | schema | name | owner | commands | comment | published | archived | created | row\_count  
----+-----------+-----------+---------+--------+------------+--------+------------------------------------------+---------+-----------+----------+----------------------------+-----------  
 1 | | | 1 | public | s1@2.0 | omm | {"select \*","from t1 where id > 3",NULL} | | t | f | 2021-04-17 09:24:11.139868 | 2  
 2 | 1 | | 1 | public | s1nick@2.0 | omm | {"SAMPLE nick .5 {name}"} | | f | f | 2021-04-17 10:02:31.73923 | 0

3、 异常场景

* 数据表或db4ai-snapshots不存在时。

purge snapshot s1nick@2.0;  
publish snapshot s1nick@2.0;  
---------  
ERROR: snapshot public."s1nick@2.0" does not exist  
CONTEXT: PL/pgSQL function db4ai.publish\_snapshot(name,name) line 11 at assignment  
   
archive snapshot s1nick@2.0;  
----------  
ERROR: snapshot public."s1nick@2.0" does not exist  
CONTEXT: PL/pgSQL function db4ai.archive\_snapshot(name,name) line 11 at assignment

* 删除snapshot时，有依赖该快照的其他snapshot，需先确保删除对本快照所依赖的其他快照。

purge snapshot s1@1.0;  
ERROR: cannot purge root snapshot 'public."s1@1.0"' having dependent snapshots  
HINT: purge all dependent snapshots first  
CONTEXT: referenced column: purge\_snapshot\_internal  
SQL statement "SELECT db4ai.purge\_snapshot\_internal(i\_schema, i\_name)"  
PL/pgSQL function db4ai.purge\_snapshot(name,name) line 71 at PERFORM

4、 相关GUC参数

* db4ai\_snapshot\_mode：
* Snapshot有2种模式：MSS（物化模式，存储数据实体）和CSS（计算模式，存储增量信息）。Snapshot可在MSS和CSS之间切换快照模式，默认是MSS模式。
* db4ai\_snapshot\_version\_delimiter：
* 该参数为数据表快照版本分隔符。“@”为数据表快照的默认版本分隔符。
* db4ai\_snapshot\_version\_separator
* 该参数为数据表快照子版本分隔符。“.”为数据表快照的默认版本分隔符。

5、 DB4AI Schema下的数据表快照详情db4ai.snapshot。

vastbase=# d db4ai.snapshot  
 Table "db4ai.snapshot"  
 Column | Type | Modifiers  
-----------+-----------------------------+---------------------------  
 id | bigint |  
 parent\_id | bigint |  
 matrix\_id | bigint |  
 root\_id | bigint |  
 schema | name | not null  
 name | name | not null  
 owner | name | not null  
 commands | text[] | not null  
 comment | text |  
 published | boolean | not null default false  
 archived | boolean | not null default false  
 created | timestamp without time zone | default pg\_systimestamp()  
 row\_count | bigint | not null  
Indexes:  
 "snapshot\_pkey" PRIMARY KEY, btree (schema, name) TABLESPACE pg\_default  
 "snapshot\_id\_key" UNIQUE CONSTRAINT, btree (id) TABLESPACE pg\_default

fig: **说明：**   
命名空间DB4AI是本功能的私有域，不支持在DB4AI的命令空间下创建函数索引（functional index）。

## AI in DB: 数据库内AI功能

### 智能Explain: SQL语句查询时间预测

#### 概述

本功能名为Predictor，是基于机器学习且具有在线学习能力的查询时间预测工具。通过不断学习数据库内收集的历史执行信息，实现计划的执行时间预测功能。

本特性需要拉起python进程AIEngine，用于模型的训练和推理。

该功能所在目录为 $GAUSSHOME/bin/dbmind/components/predictor. 由于该模块中某些功能涉及到相对复杂的搭建，因此，需要用户切换到该目录中寻找对应文件，并按照本章说明进行部署。

#### 环境部署

**前提条件**

需要保证Vastbase处于正常状态，用户通过身份验证成功登录Vastbase；用户执行的SQL语法正确无报错，且不会导致数据库异常等；历史性能数据窗口内Vastbase并发量稳定，表结构、表数量不变，数据量无突变，涉及查询性能的guc参数不变；进行预测时，需要保证模型已训练并收敛；AiEngine运行环境稳定。

**请求样例**

AiEngine进程与内核进程使用https发送请求进行通信，请求样例如下：

curl -X POST -d '{"modelName":"modelname"}' -H 'Content-Type: application/json' 'https://IP-address:port/request-API'

**表 1** ： AI-Engine对外接口

| **Request-API** | **功能** |
| --- | --- |
| /check | 检查模型是否被正常拉起 |
| /configure | 设置模型参数 |
| /train | 模型训练 |
| /track\_process | 查看模型训练日志 |
| /setup | 加载历史模型 |
| /predict | 模型预测 |

**证书生成**

请参考:《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->身份认证与通信加密》章节。

**环境准备**

1、 将工具代码文件夹拷贝至目标环境。

* 假设安装路径为$INSTALL\_FOLDER
* 假设目标环境路径为/home/ai\_user

scp -r $INSTALL\_FOLDER/bin/dbmind/predictor ai\_user@127.0.0.1:path\_to\_Predictor

2、 拷贝CA证书文件夹至aiEngine环境中某路径下。

cp -r $GAUSSHOME/CA ai\_user@127.0.0.1:path\_to\_CA

3、 安装predictor/install/requirements(-gpu).txt工具（**该功能比较特殊，与其他AI功能不同，需要独立安装依赖**）。

有GPU：pip install -r requirements-gpu.txt  
无GPU：pip install -r requirements.txt

**拉起AiEngine**

1、 切换至aiEngine环境（即拷贝predictor的目标环境 ai\_user）。

设置predictor/python/settings.py 中的相关参数。

DEFAULT\_FLASK\_SERVER\_HOST = '127.0.0.1' （aiEngine运行IP地址）  
DEFAULT\_FLASK\_SERVER\_PORT = '5000' （aiEngine运行端口号）

2、 运行aiEngine启动脚本。

python path\_to\_Predictor/python/run.py

此时，aiEngine即在相应端口保持拉起状态，等待内核侧时间预测功能的请求指令。

至此，aiEngine工具部署完成。

#### 使用指导

**数据收集**

1、 打开数据收集。

a、 设置ActiveSQL operator信息相关参数:

enable\_resource\_track=on  
resource\_track\_level=operator  
enable\_resource\_record=on  
resource\_track\_cost=10（默认值为100000）

fig: **说明：**

* resource\_track\_cost需设置为小于需要收集的查询总代价，满足条件的信息才能被收集。
* Cgroups功能正常加载。

b、 信息收集：

执行业务查询语句。

查看实时收集数据：

select \* from gs\_wlm\_plan\_operator\_history;

预期：满足resource\_track\_duration和resource\_track\_cost的作业被全量收集。

2、 关闭数据收集。  
a、 设置ActiveSQL operator信息相关参数：

enable\_resource\_track=off 或  
resource\_track\_level=none 或  
resource\_track\_level=query

b、 执行业务查询语句。

等待三分钟之后查看当前节点上的数据：

select \* from gs\_wlm\_plan\_operator\_info;

预期：所查表和视图无新增数据。

3、 数据持久化保存。  
a、 设置ActiveSQL operator信息相关参数：

enable\_resource\_track=on  
resource\_track\_level=operator  
enable\_resource\_record=on  
resource\_track\_duration=0（默认值为60s）  
resource\_track\_cost=10（默认值为100000）

fig: **说明：**

* resource\_track\_cost需设置为小于需要收集的查询总代价，满足条件的信息才能被收集。
* Cgroups功能正常加载。

b、 执行业务查询语句。

等待三分钟之后查看当前节点上的数据：

select \* from gs\_wlm\_plan\_operator\_info;

预期：满足resource\_track\_duration和resource\_track\_cost的作业被全量收集。

**模型管理（系统管理员用户）**

fig: **说明：**

模型管理操作需要在数据库正常的状态下进行。

1、 新增模型：

INSERT INTO gs\_opt\_model values('......')；

示例：

INSERT INTO gs\_opt\_model values('rlstm', 'model\_name', 'datname', '127.0.0.1', 5000, 2000, 1, -1, 64, 512, 0 , false, false, '{S, T}', '{0,0}', '{0,0}', 'Text');

fig: **说明：**

* 具体模型参数设置请参考GS\_OPT\_MODEL。
* 目前 "template\_name" 列只支持 "rlstm";
* "datname" 列请和用于模型使用和训练的数据库保持一致，否则无法使用。
* "model\_name" 一列需要满足unique约束。

2、 修改模型参数：

UPDATE gs\_opt\_model SET <attribute> = <value> WHERE model\_name = <target\_model\_name>;

3、 删除模型：

DELETE FROM gs\_opt\_model WHERE model\_name = <target\_model\_name>;

4、 查询现有模型及其状态：

SELECT \* FROM gs\_opt\_model;

**模型训练（系统管理员用户）**

1、 配置/添加模型训练参数：参考模型管理（系统管理员用户）

例：

模型添加：

INSERT INTO gs\_opt\_model values('rlstm', 'default', 'postgres', '127.0.0.1', 5000, 2000, 1, -1, 64, 512, 0 , false, false, '{S, T}', '{0,0}', '{0,0}', 'Text');

训练参数更新：

UPDATE gs\_opt\_model SET <attribute> = <value> WHERE model\_name = <target\_model\_name>;

2、 前提条件为数据库状态正常且历史数据正常收集：

删除原有encoding数据：

DELETE FROM gs\_wlm\_plan\_encoding\_table;

进行数据编码，需要指定数据库名：

SELECT gather\_encoding\_info('postgres');

开始训练：

SELECT model\_train\_opt('rlstm', 'default');

3、 获取AI Engine侧模型训练日志相对路径：

SELECT \* FROM track\_model\_train\_opt('rlstm', 'default');

**模型预测**

fig: **说明：**

* 模型预测功能需在数据库状态正常、指定模型已被训练且收敛的条件下进行。
* 目前，模型训练参数的标签设置中需要包含“S”标签，explain中才可显示“p-time”预测值。  
  例：INSERT INTO gs\_opt\_model values('rlstm', 'default', 'postgres', '127.0.0.1', 5000, 1000, 1, -1, 50, 500, 0 , false, false, '{**S**, T}', '{0,0}', '{0,0}', 'Text');

1、 调用explain接口：

explain (analyze on, predictor <model\_name>)  
SELECT ...

预期结果：

例：Row Adapter (cost=110481.35..110481.35 rows=100 p-time=99..182 width=100) (actual time=375.158..375.160 rows=2 loops=1)  
其中，“p-time”列为标签预测值。

**其他功能**

1、 检查AiEngine是否可连接:

vastbase=# select check\_engine\_status('aiEngine-ip-address',running-port);

2、 查看模型对应日志在AiEngine侧的保存路径：

vastbase=# select track\_model\_train\_opt('template\_name', 'model\_name');

#### 最佳实践

相关参数解释参考表**表 1** GS\_OPT\_MODEL。

**表 1** GS\_OPT\_MODEL

| **模型参数** | **参数建议** |
| --- | --- |
| template\_name | ‘rlstm’ |
| model\_name | 自定义，如‘open\_ai’，需满足unique约束。 |
| datname | 所服务database名称，如‘postgres’。 |
| ip | aiEngine-ip地址，如‘127.0.0.1’。 |
| port | aiEngine侦听端口，如‘5000’。 |
| max\_epoch | 迭代次数，推荐较大数值，保证收敛效果，如‘2000’。 |
| learning\_rate | (0, 1]浮点数，推荐较大的学习率，助于加快收敛速度。 |
| dim\_red | 特征值降维系数：‘-1’：不采用PCA降维，全量特征；‘（0，1] ’区间浮点数：越小，训练维度越小，收敛速度越快，但影响训练准确率。 |
| hidden\_units | 特征值维度较高时，建议适度增大此参数，提高模型复杂度，如 ‘64，128……’ |
| batch\_size | 根据编码数据量，较大数据量推荐适度增大此参数，加快模型收敛，如‘256，512……’ |
| 其他参数 | 参考表GS\_OPT\_MODEL |

推荐参数配置：

INSERT INTO gs\_opt\_model values('rlstm', 'open\_ai', 'postgres', '127.0.0.1', 5000, 2000，1, -1, 64, 512, 0 , false, false, '{S, T}', '{0,0}', '{0,0}', 'Text');

#### 常见问题处理

**AI Engine配置问题**

* **AiEngine启动失败**：请检查ip地址，端口是否可用；CA证书路径是否存在。
* **发起请求AiEngine无响应**：请检查通信双方CA证书是否一致。
* **训练，测试场景失败**：请检查模型文件保存路径是否存在；训练预测文件是否在正确下载。

**数据库内部报错问题**

问题：AiEngine链接失败。

ERROR: AI engine connection failed.  
CONTEXT: referenced column: model\_train\_opt

处理方法：检查AIEngine是否正常拉起或重启AIEngine；检查通信双方CA证书是否一致；检查模型配置信息中的ip和端口是否匹配；

问题：模型不存在。

ERROR: OPT\_Model not found for model name XXX  
CONTEXT: referenced column: track\_model\_train\_opt

处理方法：检查GS\_OPT\_MODEL表中是否存在执行语句中“model\_name”对应的模型；使用预测功能报错时，检查模型是否已被训练。

# GUC参数说明

## GUC使用说明

**功能描述**

数据库提供了许多运行参数，配置这些参数可以影响数据库系统的行为。在修改这些参数时请确保用户理解了这些参数对数据库的影响，否则可能会导致无法预料的结果。

**注意事项**

* 参数中如果取值范围为字符串，此字符串应遵循操作系统的路径和文件名命名规则。
* 取值范围最大值为INT\_MAX的参数，此选项最大值跟所在的操作系统有关。
* 取值范围最大值为DBL\_MAX的参数，此选项最大值跟所在的操作系统有关。

## 文件位置

数据库安装后会自动生成三个配置文件（postgresql.conf、pg\_hba.conf和pg\_ident.conf），并统一存放在数据目录（data）下。用户可以使用本节介绍的方法修改配置文件的名称和存放路径。

修改任意一个配置文件的存放目录时，postgresql.conf里的data\_directory参数必须设置为实际数据目录（data）。

fig: **须知：**   
考虑到配置文件修改一旦出错对数据库的影响很大，不建议安装后再修改本节的配置文件。

**data\_directory**

**参数说明：**设置Vastbase的数据目录（data目录）。此参数可以通过如下方式指定。

* 在安装Vastbase时指定。
* 该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，长度大于0

**默认值：**安装时指定，如果在安装时不指定，则默认不初始化数据库。

**config\_file**

**参数说明：**设置主服务器配置文件名称（postgresql.conf）。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置，不支持使用：配置运行参数->重设参数章节中表2的方式四进行修改。

**取值范围：**字符串，长度大于0

**默认值：**postgresql.conf（实际安装可能带有绝对目录）

**hba\_file**

**参数说明：**设置基于主机认证（HBA）的配置文件（pg\_hba.conf）。此参数只能在配置文件postgresql.conf中指定。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**pg\_hba.conf（实际安装可能带有绝对目录）

**ident\_file**

**参数说明：**设置用于客户端认证的配置文件的名称（pg\_ident.conf）。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**pg\_ident.conf（实际安装可能带有绝对目录）

**external\_pid\_file**

**参数说明：**声明可被服务器管理程序使用的额外PID文件。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
这个参数只能在数据库服务重新启动后生效。

**取值范围：**字符串

**默认值：**空

## 连接和认证

### 连接设置

介绍设置客户端和服务器连接方式相关的参数。

**listen\_addresses**

**参数说明：**声明服务器侦听客户端的TCP/IP地址。

该参数指定Vastbase服务器使用哪些IP地址进行侦听，如IPV4或IPV6（若支持）。服务器主机上可能存在多个网卡，每个网卡可以绑定多个IP地址，该参数就是控制Vastbase到底绑定在哪个或者哪几个IP地址上。而客户端则可以通过该参数中指定的IP地址来连接Vastbase或者给Vastbase发送请求。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**

* 主机名或IP地址，多个值之间用英文逗号分隔。
* 星号“*”或“0.0.0.0”表示侦听所有IP地址。配置侦听所有IP地址存在安全风险，不推荐用户使用。必须与有效地址结合使用（比如本地IP等），否则，可能造成Build失败的问题。同时，主备环境下配置为“*”或“0.0.0.0”时，主节点数据库路径下postgresql.conf文件中的localport端口号不能为数据库dataPortBase+1，否则会导致数据库无法启动。
* 置空则服务器不会侦听任何IP地址，这种情况下，只有Unix域套接字可以用于连接数据库。

**默认值：**数据库实例安装好后，根据XML配置文件中不同实例的IP地址配置不同默认值。DN的默认参数值为：listen\_addresses = 'x.x.x.x'。

**local\_bind\_address**

**参数说明：**声明当前节点连接Vastbase其他节点绑定的本地IP地址。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**默认值：**数据库实例安装好后，根据XML配置文件中不同实例的IP地址配置不同默认值。DN的默认参数值为：local\_bind\_address = 'x.x.x.x'。

**port**

**参数说明：**Vastbase服务侦听的TCP端口号。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **说明：**   
该参数由安装时的配置文件指定，请勿轻易修改，否则修改后会影响数据库正常通信。

**取值范围：**整型，1～65535

fig: **说明：**

* 设置端口号时，请设置一个未被占用的端口号。设置多个实例的端口号，不可冲突。
* 1~1023为操作系统保留端口号，请不要使用。
* 通过配置文件安装数据库实例时，配置文件中的端口号需要注意通信矩阵预留端口。如：DN还需保留dataPortBase+1作为内部工具使用端口，保留dataPortBase+6作为流引擎消息队列通信端口等。故数据库实例安装阶段，port最大值为：DN可设置65529，同时需要保证端口号不冲突。
* 通过gs\_guc set的方式修改了port值之后，需要手动修改下静态配置文件static\_config\_files里面的端口信息后，才可以生效。

**默认值：**5432（实际值由安装时的配置文件指定）static\_config\_files

**max\_connections**

**参数说明：**允许和数据库连接的最大并发连接数。此参数会影响Vastbase的并发能力。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型。最小值为10，最大值为262143。

**默认值：**

数据库节点：5000。如果该默认值超过内核支持的最大值（在执行vb\_initdb的时候判断），系统会提示错误。

**设置建议：**

* 数据库主节点中此参数建议保持默认值。数据库节点中此参数建议设置为数据库主节点的个数乘以数据库主节点中此参数的值。
* 增大这个参数可能导致Vastbase要求更多的SystemV共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。

**max\_inner\_tool\_connections**

**参数说明：**允许和数据库连接的工具的最大并发连接数。此参数会影响Vastbase的工具连接并发能力。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为1，最大值为262143。

**默认值：**数据库节点为50。如果该默认值超过内核支持的最大值（在执行vb\_initdb的时候判断），系统会提示错误。

**设置建议：**

数据库主节点中此参数建议保持默认值。

增大此参数可能导致Vastbase要求更多的SystemV共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。

**sysadmin\_reserved\_connections**

**参数说明：**为管理员用户预留的最少连接数，不建议设置过大。该参数和max\_connections参数配合使用，管理员用户的最大连接数等于max\_connections + sysadmin\_reserved\_connections。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为MIN(262143, max\_connections)，max\_connections的计算方法见上文。

**默认值：**3

注意：启用线程池功能时，若线程池占满将形成处理瓶颈，导致管理员预留连接无法正常建立，作为逃生手段，此时可使用gsql通过主端口+1端口号连入，清理无用会话，即可正常连入。

**unix\_socket\_directory**

**参数说明：**设置Vastbase服务器侦听客户端连接的Unix域套接字目录。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

该参数的长度限制于操作系统的长度，超过该限制将会导致Unix-domain socket path "xxx" is too long的问题。

**取值范围：**字符串

**默认值：**空字符串（实际值由安装时配置文件指定）

**unix\_socket\_group**

**参数说明：**设置Unix域套接字的所属组（套接字的所属用户总是启动服务器的用户）。可以与选项[unix\_socket\_permissions](#Xed8a066c3b8df0f598f0d6c65c09aabce2433e6)一起用于对套接字进行访问控制。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，其中空字符串表示当前用户的缺省组。

**默认值：**空字符串

**unix\_socket\_permissions**

**参数说明：**设置Unix域套接字的访问权限。

Unix域套接字使用普通的Unix文件系统权限集。这个参数的值应该是数值的格式（chmod和umask命令可接受的格式）。如果使用自定义的八进制格式，数字必须以0开头。

建议设置为0770（只有当前连接数据库的用户和同组的人可以访问）或者0700（只有当前连接数据库的用户自己可以访问，同组或者其他人都没有权限）。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0000-0777

**默认值：**0777

fig: **说明：**   
在Linux中，文档具有十个属性，其中第一个属性为文档类型，后面九个为权限属性，分别为Owner、Group及Others这三个组别的read、write、execute属性。  
文档的权限属性分别简写为r、w、x，这九个属性三个为一组，也可以使用数字来表示文档的权限，对照表如下：

* r：4
* w：2
* x：1
* -：0

同一组（owner/group/others）的三个属性是累加的。  
例如，-rwxrwx---表示这个文档的权限为：  
owner = rwx = 4+2+1 = 7  
group = rwx = 4+2+1 = 7  
others = --- = 0+0+0 = 0  
所以其权限为0770。

**application\_name**

**参数说明：**当前连接请求当中，所使用的客户端名称。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

在备机请求主机进行日志复制时，如果该参数非空串，那么会被用来作为备机在主机上的流复制槽名字。此时，如果该参数长度超过61个字节，那么流复制槽名字只会截取使用前61个字节的字符。

**取值范围：**字符串。

**默认值：**空字符串（连接到后端的应用名，以实际安装为准）

**connection\_info**

**参数说明：**连接数据库的驱动类型、驱动版本号、当前驱动的部署路径和进程属主用户。

该参数属于USERSET类型参数，属于运维类参数，不建议用户设置。

**取值范围：**字符串。

**默认值：**空字符串

fig: **说明：**

* 空字符串，表示当前连接数据库的驱动不支持自动设置connection\_info参数或应用程序未设置。
* 驱动连接数据库的时候自行拼接的connection\_info参数格式如下：
* {"driver\_name":"ODBC","driver\_version": "(Vastbase X.X.X build 13b34b53) compiled at 2020-05-08 02:59:43 commit 2143 last mr 131 debug","driver\_path":"/usr/local/lib/psqlodbcw.so","os\_user":"omm"}
* 默认显示driver\_name和driver\_version，driver\_path和os\_user的显示由用户控制。

### 安全和认证（postgresql.conf）

介绍设置客户端和服务器的安全认证方式的相关参数。

**authentication\_timeout**

**参数说明：**完成客户端认证的最长时间。如果一个客户端没有在这段时间里完成与服务器端的认证，则服务器自动中断与客户端的连接，这样就避免了出问题的客户端无限制地占用连接数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为1，最大值为600，最小单位为s。

**默认值：**1min

**auth\_iteration\_count**

**参数说明：**认证加密信息生成过程中使用的迭代次数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，2048-134217728。

**默认值：**10000

fig: **须知：**   
迭代次数设置过小会降低口令存储的安全性，设置过大会导致认证、用户创建等涉及口令加密的场景性能劣化，请根据实际硬件条件合理设置迭代次数，推荐采用默认迭代次数。

**session\_authorization**

**参数说明：**当前会话的用户标识。

该参数属于USERSET类型参数，此参数为内部参数，只能通过SET SESSION AUTHORIZATION语法设置，该语法请参考：SQL语法参考->SQL语法->SET SESSION AUTHORIZATION章节，不支持直接设置。

**取值范围：**字符串。

**默认值：**NULL

**session\_timeout**

**参数说明：**表明与服务器建立链接后，不进行任何操作的最长时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0-86400，最小单位为s，0表示关闭超时设置。

**默认值：**10min

fig: **须知：**   
Vastbase gsql客户端中有自动重连机制，所以针对初始化用户本地连接，超时后gsql表现的现象为断开后重连。

**ssl**

**参数说明：**启用SSL连接。请在使用这个选项之前阅读：《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->使用vsql连接》章节。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示启用SSL连接。
* off表示不启用SSL连接。

fig: **须知：**   
开启此参数需要同时配置ssl\_cert\_file、ssl\_key\_file和ssl\_ca\_file等参数及对应文件，不正确的配置可能会导致Vastbase无法正常启动。

**默认值：**on

**require\_ssl**

**参数说明：**设置服务器端是否强制要求SSL连接，该参数只有当参数ssl为on时才有效。请在使用这个选项之前阅读：《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->使用vsql连接》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示服务器端强制要求SSL连接。
* off表示服务器端对是否通过SSL连接不作强制要求。

fig: **须知：**   
Vastbase目前支持SSL的场景为客户端连接数据库主节点场景，该参数目前建议只在数据库主节点中开启。

**默认值：**off

**ssl\_ciphers**

**参数说明：**指定SSL支持的加密算法列表。ssl\_ciphers设置错误会导致数据库不能正常启动。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，如果指定多个加密算法，加密算法之间需要以分号分割。详细请参考：《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->身份认证与通信加密》章节，获取支持的加密算法。

**默认值：**ALL

**ssl\_renegotiation\_limit**

**参数说明：**指定在会话密钥重新协商之前，通过SSL加密通道可以传输的流量。这个重新协商流量限制机制可以减少攻击者针对大量数据使用密码分析法破解密钥的几率，但是也带来较大的性能损失。流量是指发送和接受的流量总和。使用SSL重协商机制可能引入其他风险，因此已禁用SSL重协商机制，为保持版本兼容保留此参数，修改参数配置不再起作用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为2147483647。单位为KB。其中0表示禁用重新协商机制。

**默认值：**0

**ssl\_cert\_file**

**参数说明：**指定包含SSL服务器证书的文件的名称，其相对路径是相对于数据目录的。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**server.crt

**ssl\_key\_file**

**参数说明：**指定包含SSL私钥的文件名称，其相对路径是相对于数据目录的。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**server.key

**ssl\_ca\_file**

**参数说明：**指定包含CA信息的文件的名称，其相对路径是相对于数据目录的。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，其中空字符串表示没有CA文件被加载，不进行客户端证书验证。

**默认值：**空

**ssl\_crl\_file**

**参数说明**：证书吊销列表，如果客户端证书在该列表中，则当前客户端证书被视为无效证书。必须使用相对路径，相对路径是相对于数据目录的。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串，空字符串表示没有吊销列表。

**默认值**：空

**krb\_server\_keyfile**

**参数说明：**指定Kerberos服务主配置文件的位置，详细请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->配置客户端接入认证》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**空

**krb\_srvname**

**参数说明：**设置Kerberos服务名，详细请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->数据库使用->连接数据库->配置客户端接入认证》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**postgres

**krb\_caseins\_users**

**参数说明：**设置Kerberos用户名是否大小写敏感。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示大小写不敏感。
* off表示大小写敏感。

**默认值：**off

**modify\_initial\_password**

**参数说明：**当Vastbase安装成功后，数据库中仅存在一个初始用户（UID为10的用户）。客户通过该帐户初次登录数据库进行操作时，该参数决定是否要对该初始帐户的密码进行修改。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

如果安装过程中未指定初始用户密码，则安装后初始用户密码默认为空，执行其他操作前需要先通过gsql客户端设置初始用户的密码。此参数功能不再生效，保留此参数仅为兼容升级场景。

**取值范围：**布尔型

* on表示Vastbase安装成功后初始用户首次登录操作前需要修改初始密码。
* off表示Vastbase安装成功后初始用户无需修改初始密码即可进行操作。

**默认值：**off

**password\_policy**

**参数说明：**在使用CREATE ROLE/USER或者ALTER ROLE/USER命令创建或者修改Vastbase帐户时，该参数决定是否进行密码复杂度检查。关于密码复杂度检查策略请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->安全性->管理用户及权限->设置安全策略->设置密码安全策略》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
从安全性考虑，请勿关闭密码复杂度策略。

**取值范围：**0、1

* 0表示不采用密码复杂度校验策略。
* 1表示采用默认密码复杂度校验策略。

**默认值：**1

**password\_reuse\_time**

**参数说明：**在使用ALTER USER或者ALTER ROLE修改用户密码时，该参数指定是否对新密码进行可重用天数检查。关于密码可重用策略请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->安全性->管理用户及权限->设置安全策略->设置密码安全策略》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
修改密码时会检查配置参数password\_reuse\_time和password\_reuse\_max。

* 当password\_reuse\_time和password\_reuse\_max都为正数时，只要满足其中任一个，即可认为密码可以重用。
* 当password\_reuse\_time为0时，表示不限制密码重用天数，仅限制密码重用次数。
* 当password\_reuse\_max为0时，表示不限制密码重用次数，仅限制密码重用天数。
* 当password\_reuse\_time和password\_reuse\_max都为0时，表示不对密码重用进行限制。

**取值范围：**浮点型（天），最小值为30，最大值为90。

正数表示新密码不能为该值指定的天数内使用过的密码。

**默认值：**90

**password\_reuse\_max**

**参数说明：**在使用ALTER USER或者ALTER ROLE修改用户密码时，该参数指定是否对新密码进行可重用次数检查。关于密码可重用策略请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->安全性->管理用户及权限->设置安全策略->设置密码安全策略》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
修改密码时会检查配置参数password\_reuse\_time和password\_reuse\_max。

* 当password\_reuse\_time和password\_reuse\_max都为正数时，只要满足其中任一个，即可认为密码可以重用。
* 当password\_reuse\_time为0时，表示不限制密码重用天数，仅限制密码重用次数。
* 当password\_reuse\_max为0时，表示不限制密码重用次数，仅限制密码重用天数。
* 当password\_reuse\_time和password\_reuse\_max都为0时，表示不对密码重用进行限制。

**取值范围：**整型，最小值为1，最大值为1000。

正整数表示新密码不能为该值指定的次数内使用过的密码。

**默认值：**3

**password\_lock\_time**

**参数说明：**该参数指定帐户被锁定后自动解锁的时间。关于帐户自动锁定策略请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->安全性->管理用户及权限->设置安全策略->设置密码安全策略》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
password\_lock\_time和failed\_login\_attempts必须都为正数时锁定和解锁功能才能生效。

**取值范围：**整型，最小值为1，最大值为525600，单位为分钟。

* 正数表示帐户被锁定后，当锁定时间超过password\_lock\_time设定的值时，帐户将会被自行解锁。
* 系统管理员可以通过ALTER USER joe ACCOUNT UNLOCK;手动解锁。

**默认值：**1440（即1天）

**failed\_login\_attempts**

**参数说明：**在任意时候，如果输入密码错误的次数达到failed\_login\_attempts参数设定的值，则当前帐户会被锁定。password\_lock\_time参数设定的天数过后，帐户自动解锁。例如，登录时输入密码失败，ALTER USER时修改密码失败等。关于帐户自动锁定策略请参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->安全性->管理用户及权限->设置安全策略->设置密码安全策略》章节。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
failed\_login\_attempts和password\_lock\_time必须都为正数时锁定和解锁功能才能生效。

**取值范围：**整型，最小值为1，最大值为1000。

* 正整数表示当错误密码次数达到failed\_login\_attempts设定的值时，当前帐户将被锁定。
* 系统管理员可以通过ALTER USER joe ACCOUNT UNLOCK;手动解锁。

**默认值：**5

**password\_encryption\_type**

**参数说明：**该字段决定采用何种加密方式对用户密码进行加密存储。修改此参数的配置不会自动触发已有用户密码加密方式的修改，只会影响新创建用户或修改用户密码操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0、1、2、3

* 0表示采用md5方式对密码加密。
* 1表示采用sha256和md5两种方式分别对密码加密。
* 2表示采用sha256方式对密码加密。
* 3表示采用sm3方式对密码加密。

fig: **须知：**   
md5加密算法安全性低，存在安全风险，不建议用户使用。

**默认值：**1

**password\_min\_length**

**参数说明：**该字段决定帐户密码的最小长度。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，6~999个字符。

**默认值：**8

**password\_max\_length**

**参数说明：**该字段决定帐户密码的最大长度。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，6~999个字符。

**默认值：**32

**password\_min\_uppercase**

**参数说明：**该字段决定帐户密码中至少需要包含大写字母个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~999

* 0表示没有限制。
* 1~999表示创建账户所指定的密码中至少需要包含大写字母个数。

**默认值：**0

**password\_min\_lowercase**

**参数说明：**该字段决定帐户密码中至少需要包含小写字母的个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~999

* 0表示没有限制。
* 1~999表示创建帐户所指定的密码中至少需要包含小写字母个数。

**默认值：**0

**password\_min\_digital**

**参数说明：**该字段决定帐户密码中至少需要包含数字的个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~999

* 0表示没有限制。
* 1~999表示创建帐户所指定的密码中至少需要包含数字个数。

**默认值：**0

**password\_min\_special**

**参数说明：**该字段决定帐户密码中至少需要包含特殊字符个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~999

* 0表示没有限制。
* 1~999表示创建帐户所指定的密码中至少需要包含特殊字符个数。

**默认值：**0

**password\_effect\_time**

**参数说明：**该字段决定帐户密码的有效时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**浮点型，最小值为30，最大值为100\*365，单位为天。

30~100\*365表示创建帐户所指定的密码有效期，临近或超过有效期系统会提示用户修改密码。

**默认值：**90

**password\_notify\_time**

**参数说明：**该字段决定帐户密码到期前提醒的天数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为1，最大值为90，单位为天。

1~90表示帐户密码到期前提醒的天数。

**默认值：**7

### 通信库参数

本节介绍通信库相关的参数设置及取值范围等内容。

**tcp\_keepalives\_idle**

**参数说明：**在支持TCP\_KEEPIDLE套接字选项的系统上，设置发送活跃信号的间隔秒数。不设置发送保持活跃信号，连接就会处于闲置状态。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

fig: **须知：**

* 如果操作系统不支持TCP\_KEEPIDLE选项，这个参数的值必须为0。
* 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。
* 将该值设置为0时，将使用系统的值。
* 该参数在不同的会话之间不共享，也就是说不同的会话连接可能有不同的值。
* 查看该参数时查出来的是当前会话连接内的参数值，而不是guc副本的值。

**取值范围：**0-3600，单位为s。

**默认值：**0

**tcp\_keepalives\_interval**

**参数说明：**在支持TCP\_KEEPINTVL套接字选项的操作系统上，以秒数声明在重新传输之间等待响应的时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0-180，单位为s。

**默认值：**0

fig: **须知：**

* 如果操作系统不支持TCP\_KEEPINTVL选项，这个参数的值必须为0。
* 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。
* 将该值设置为0时，将使用系统的值。
* 该参数在不同的会话之间不共享，也就是说不同的会话连接可能有不同的值。
* 查看该参数时查出来的是当前会话连接内的参数值，而不是guc副本的值。

**tcp\_keepalives\_count**

**参数说明：**在支持TCP\_KEEPCNT套接字选项的操作系统上，设置Vastbase服务端在断开与客户端连接之前可以等待的保持活跃信号个数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 如果操作系统不支持TCP\_KEEPCNT选项，这个参数的值必须为0。
* 在通过Unix域套接字进行连接的操作系统上，这个参数将被忽略。
* 将该值设置为0时，将使用系统的值。
* 该参数在不同的会话之间不共享，也就是说不同的会话连接可能有不同的值。
* 查看该参数时查出来的是当前会话连接内的参数值，而不是guc副本的值。

**取值范围：**0-100，其中0表示Vastbase未收到客户端反馈的保持活跃信号则立即断开连接。

**默认值：**0

**comm\_proxy\_attr**

**参数说明：**通信代理库相关参数配置。

fig: **说明：**

* 该参数仅支持欧拉2.9系统下的集中式ARM单机。
* 本功能在线程池开启状态下生效，即enable\_thread\_pool为on。
* 配置该参数时需同步配置GUC参数local\_bind\_address为libos\_kni的网卡IP。
* 参数模板：comm\_proxy\_attr = '{enable\_libnet:true, enable\_dfx:false, numa\_num:4, numa\_bind:[[30,31],[62,63],[94,95],[126,127]]}'
* 可配置参数说明。
* enable\_libnet：是否开启用户态协议，取值范围：true、false。
* enable\_dfx：是否开启通信代理库视图，取值范围：true、false。
* numa\_num：机器环境中numa的数量，支持2P、4P服务器，取值范围：4、8。
* numa\_bind：代理线程绑核参数，每个numa两个CPU绑核，共numa\_num组，取值范围：[0，cpu数-1]。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串，长度大于0。

**默认值：**'none'

## 资源消耗

### 内存

介绍与内存相关的参数设置。

fig: **须知：**这些参数只能在数据库服务重新启动后生效。

**memorypool\_enable**

**参数说明**：设置是否允许使用内存池。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示允许使用内存池。
* off表示不允许使用内存池。

**默认值**：off

**memorypool\_size**

**参数说明**：设置内存池大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，128\*1024～INT\_MAX/2，单位为KB。

**默认值**：512MB

**enable\_memory\_limit**

**参数说明：**启用逻辑内存管理模块。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示启用逻辑内存管理模块。
* off表示不启用逻辑内存管理模块。

**默认值：**on

fig: **注意：**

* 若max\_process\_memory-shared\_buffers-cstore\_buffers-元数据少于2G，Vastbase强制把enable\_memory\_limit设置为off。其中元数据是Vastbase内部使用的内存和部分并发参数，如max\_connections、thread\_pool\_attr、max\_prepared\_transactions等参数相关。
* 当该值为off时，不对数据库使用的内存做限制，在大并发或者复杂查询时，使用内存过多，可能导致操作系统OOM问题。

**max\_process\_memory**

**参数说明：**设置一个数据库节点可用的最大物理内存。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：整型，2*1024*1024～INT\_MAX，单位为KB。

**默认值：**12GB

**设置建议：**

数据库节点上该数值需要根据系统物理内存及单节点部署主数据库节点个数决定。建议计算公式如下：(物理内存大小 - vm.min\_free\_kbytes) \* 0.7 / (1 + 主节点个数)。该系数的目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min\_free\_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即，max\_process\_memory = 物理内存 \* 0.665 / (1 + 主节点个数)。

fig: **注意：**

当该值设置不合理，即大于服务器物理内存，可能导致操作系统OOM问题。

**enable\_memory\_context\_control**

**参数说明：**启用检查内存上下文是否超过给定限制的功能。仅适用于DEBUG版本。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示启用最大内存上下文限制检查功能。
* off表示关闭最大内存上下文限制检查功能。

**默认值：**off

**uncontrolled\_memory\_context**

**参数说明：**启用检查内存上下文是否超过给定限制的功能时，设置不受此功能约束。仅适用于DEBUG版本。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

查询时会在参数值的最前面添加标题含义字符串“MemoryContext white list:”。

**取值范围**：字符串

**默认值：**空

**shared\_buffers**

**参数说明：**设置Vastbase使用的共享内存大小。增加此参数的值会使Vastbase比系统默认设置需要更多的System V共享内存。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，16 ~ 1073741823，单位为8KB。

shared\_buffers需要设置为BLCKSZ的整数倍，BLCKSZ目前设置为8KB，即shared\_buffers需要设置为8KB整数倍。改变BLCKSZ的值会改变最小值。

**默认值：**8MB

**设置建议：**

建议设置shared\_buffers值为内存的40%以内。行存列存分开对待。行存设大，列存设小。列存：(单服务器内存/单服务器数据库节点个数)*0.4*0.25。

如果设置较大的shared\_buffers需要同时增加checkpoint\_segments的值，因为写入大量新增、修改数据需要消耗更多的时间周期。

**segment\_buffers**

**参数说明：**设置Vastbase段页式元数据页的内存大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值说明：**整型，16 ~ 1073741823，单位为8KB。

segment\_buffers需要设置为BLCKSZ的整数倍，BLCKSZ目前设置为8KB，即segment\_buffers需要设置为8KB整数倍。改变BLCKSZ的值会改变最小值。

**默认值：**8MB

**设置建议：**

segment\_buffers 用来缓存段页式段头的内容，属于关键元数据信息，为了提高性能建议常用的表的段头都能缓存在buffer中，不被置换出去。建议按照表的个数（包括索引和toast表）\* 分区数 \* 3 + 128 来设置。乘以3是因为每个表（分区）会有一些额外的元数据段，一般一个表有3个段。最后+128因为段页式表空间管理需要一定数量的buffer。

**bulk\_write\_ring\_size**

**参数说明：**大批量数据写入触发时（例如copy动作），该操作使用的环形缓冲区大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，16384 ~ 2147483647，单位为KB。

**默认值：**2GB

**设置建议：**建议导入压力大的场景中增加数据库节点中此参数配置。

**standby\_shared\_buffers\_fraction**

**参数说明：**备实例所在服务器使用shared\_buffers内存缓冲区大小的比例。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**双精度浮点型，0.1~1.0

**默认值：**0.3

**temp\_buffers**

**参数说明：**设置每个数据库会话使用的LOCAL临时缓冲区的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

在每个会话的第一次使用临时表之前可以改变temp\_buffers的值，之后的设置将是无效的。

一个会话将按照temp\_buffers给出的限制，根据需要分配临时缓冲区。如果在一个并不需要大量临时缓冲区的会话里设置一个大的数值，其开销只是一个缓冲区描述符的大小。当缓冲区被使用，就会额外消耗8192字节。

**取值范围：**整型，100~1073741823，单位为8KB。

**默认值：**1MB

**max\_prepared\_transactions**

**参数说明：**设置可以同时处于“预备”状态的事务的最大数目。增加此参数的值会使Vastbase比系统默认设置需要更多的System V共享内存。

当Vastbase部署为主备双机时，在备机上此参数的设置必须要高于或等于主机上的，否则无法在备机上进行查询操作。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~262143。

**默认值：**10

fig: **说明：**   
一般不需要对事务显式进行PREPARE操作，如果业务对事务进行显示PREPARE操作，为避免在准备步骤失败，需要调大该值，大于需要进行PREPARE业务的并发数。

**work\_mem**

**参数说明：**设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。ORDER BY、DISTINCT和merge joins都要用到排序操作。Hash表在散列连接、散列为基础的聚集、散列为基础的IN子查询处理中都要用到。

对于复杂的查询，可能会同时并发运行好几个排序或者散列操作，每个都可以使用此参数所声明的内存量，不足时会使用临时文件。同样，好几个正在运行的会话可能会同时进行排序操作。因此使用的总内存可能是work\_mem的好几倍。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，64~2147483647，单位为KB。

**默认值：**4096

**设置建议：**

依据查询特点和并发来确定，一旦work\_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。

* 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work\_mem=50%内存/10。
* 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work\_mem=50%内存/5。
* 对于并发场景，建议work\_mem=串行下的work\_mem/物理并发数。
* 对于BitmapScan的哈希表也会受到work\_mem的限制，但不会被严格管控下盘。完全Lossify的情况下，哈希表每占用1MB的内存，对应一次BitmapHeapScan的16GB的页面（Ustore为32GB），达到work\_mem上限后，会按此比例随数据访问量线性增长。

**query\_mem**

**参数说明：**设置执行作业所使用的内存。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0，或大于32M的整型，默认单位为KB。

**默认值：**0

fig: **须知：**

* 如果设置的query\_mem值大于0，在生成执行计划时，优化器会将作业的估算内存调整为该值。
* 如果设置值为负数或小于32MB，将设置为默认值0，此时优化器不会根据该值调整作业的估算内存。

**query\_max\_mem**

**参数说明：**设置执行作业所能够使用的最大内存。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0，或大于32M的整型，默认单位为KB。

**默认值：**0

fig: **须知：**

* 如果设置的query\_max\_mem值大于0，当作业执行时所使用内存超过该值时，将报错退出。
* 如果设置值为负数或小于32M，将设置为默认值0，此时不会根据该值限制作业的内存使用。

**maintenance\_work\_mem**

**参数说明：**设置在维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。该参数的设置会影响VACUUM、VACUUM FULL、CLUSTER、CREATE INDEX的执行效率。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1024~INT\_MAX，单位为KB。

**默认值：**16MB

**设置建议：**

* 建议设置此参数的值大于work\_mem可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。
* 当自动清理线程(参考：GUC参数说明->自动清理章节)运行时，autovacuum\_max\_workers倍数的内存将会被分配，所以此时设置maintenance\_work\_mem的值应该不小于work\_mem。
* 如果进行大数据量的cluster等，可以在session中调大该值。

**psort\_work\_mem**

**参数说明：**设置列存表在进行局部排序中，在开始写入临时磁盘文件之前使用的内存大小。带partial cluster key的表、带索引的表插入、创建表索引、删除表和更新表都会用到。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
同样，多个正在运行的会话可能会同时进行表的局部排序操作。因此，使用的总内存可能是psort\_work\_mem的几倍。

**取值范围：**整型64~2147483647，单位为KB。

**默认值：**512MB

**max\_loaded\_cudesc**

**参数说明：**设置列存表在做扫描时，每列缓存cudesc信息的个数。增大设置会提高查询性能，但也会增加内存占用，特别是当列存表的列非常多时。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
max\_loaded\_cudesc设置过高时，有可能引起内存分配不足。

**取值范围：**100~1073741823。

**默认值：**1024

**max\_stack\_depth**

**参数说明：**设置Vastbase执行堆栈的最大安全深度。需要这个安全界限是因为在服务器里，并非所有程序都检查了堆栈深度，只是在可能递规的过程，比如表达式计算这样的过程里面才进行检查。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，100~INT\_MAX，单位为KB。

**默认值：**2MB

**设置原则：**

* 数据库需要预留640KB堆栈深度，因此，此参数的最佳设置是=操作系统内核允许的最大值（就是ulimit -s的设置）- 640KB。
* 如果设置此参数的值大于实际的内核限制，则一个正在运行的递归函数可能会导致一个独立的服务器线程崩溃。在Vastbase能够检测内核限制的操作系统上，将自动限制设置为一个不安全的值。
* 因为并非所有的操作都能够检测，所以建议用户在此设置一个明确的值。
* 默认值2MB，这个值相对比较小，不容易导致系统崩溃。

**cstore\_buffers**

**参数说明：**设置列存所使用的共享缓冲区的大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，16384～1073741823，单位为KB。

**默认值：**128MB

**设置建议**：

列存表使用cstore\_buffers设置的共享缓冲区，几乎不用shared\_buffers。因此在列存表为主的场景中，应减少shared\_buffers，增加cstore\_buffers。

**bulk\_read\_ring\_size**

**参数说明：**大批量数据查询时（例如大表扫描），该操作使用的环形缓冲区大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，256~2147483647，单位为KB。

**默认值：**16MB

**enable\_early\_free**

**参数说明**：控制是否可以实现算子内存的提前释放。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示支持算子内存提前释放。
* off表示不支持算子内存提前释放。

**默认值**：on

**local\_syscache\_threshold**

**参数说明**：系统表cache在单个session缓存的大小。

该参数属于PG\_SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

如果enable\_global\_plancache已打开，为保证GPC生效，local\_syscache\_threshold设置值小于16MB时不会生效，最小为16MB。

如果enable\_global\_plancache和enable\_thread\_pool打开，该参数描述的是当前线程和绑定到当前线程上的session缓存的总大小。

**取值范围**：整型，1*1024 ~ 512*1024，单位KB。

**默认值**：256MB

### 磁盘空间

介绍与磁盘空间相关的参数，用于限制临时文件所占用的磁盘空间。

**sql\_use\_spacelimit**

**参数说明：**限制单个SQL在单个数据库节点上、触发落盘操作时、落盘文件的空间大小、管控的空间包括普通表、临时表以及中间结果集落盘占用的空间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，-1~2147483647，单位为KB。其中-1表示没有限制。

**默认值：**-1

**temp\_file\_limit**

**参数说明：**限制一个会话中，触发下盘操作时，单个下盘文件的空间大小。例如一次会话中，排序和哈希表使用的临时文件，或者游标占用的临时文件。

此设置为会话级别的下盘文件控制。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
SQL查询执行时使用的临时表空间不在此限制。

**取值范围：**整型，-1~2147483647，单位为KB。其中-1表示没有限制。

**默认值：**-1

### 内核资源使用

介绍与操作系统内核相关的参数，这些参数是否生效依赖于操作系统的设置。

**max\_files\_per\_process**

**参数说明：**设置每个服务器进程允许同时打开的最大文件数目。如果操作系统内核强制一个合理的数目，则不需要设置。

但是在一些平台上（特别是大多数BSD系统），内核允许独立进程打开比系统真正可以支持的数目大得多的文件数。如果用户发现有的“Too many open files”这样的失败现象，请尝试缩小这个设置。通常情况下需要满足，系统FD（file descriptor）数量>=最大并发数*数据库节点个数*max\_files\_per\_process\*3。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，25~2147483647。

**默认值：**1000

**shared\_preload\_libraries**

**参数说明：**此参数用于声明一个或者多个在服务器启动的时候预先装载的共享库，多个库名称之间用逗号分隔。比如 '$libdir/mylib' 会在加载标准库目录中的库文件之前预先加载mylib.so（某些平台上可能是mylib.sl）库文件。

可以用这个方法预先装载Vastbase的存储过程库，通常是使用 '$libdir/plXXX' 语法。XXX只能是pgsql、perl、tcl、python之一。

通过预先装载一个共享库并在需要的时候初始化它，可以避免第一次使用这个库的加载时间。但是启动每个服务器进程的时间可能会增加，即使进程从来没有使用过这些库。因此建议对那些将被大多数会话使用的库才使用这个选项。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 如果被声明的库不存在，Vastbase服务将会启动失败。
* 每一个支持Vastbase的库都有一个特殊的标记用于保证兼容性。因此，不支持Vastbase的库不能用这种方法加载。

**取值范围：**字符串

**默认值：**空

### 基于开销的清理延迟

这个特性的目的是允许管理员减少VACUUM和ANALYZE语句在并发活动的数据库上的I/O影响。比如，像VACUUM和ANALYZE这样的维护语句并不需要迅速完成，并且不希望他们严重干扰系统执行其他的数据库操作。基于开销的清理延迟为管理员提供了一个实现这个目的手段。

fig: **须知：**   
有些清理操作会持有关键的锁，这些操作应该尽快结束并释放锁。所以Vastbase的机制是，在这类操作过程中，基于开销的清理延迟不会发生作用。为了避免在这种情况下的长延时，实际的开销限制取下面两者之间的较大值：

* vacuum\_cost\_delay\*accumulated\_balance/vacuum\_cost\_limit
* vacuum\_cost\_delay\*4

**背景信息**

在：SQL语法参考->SQL语法->ANALYZE | ANALYSE章节和SQL语法参考->SQL语法->VACUUM章节，语句执行过程中，系统维护一个内部的记数器，跟踪所执行的各种I/O操作的近似开销。如果积累的开销达到了vacuum\_cost\_limit声明的限制，则执行这个操作的进程将睡眠vacuum\_cost\_delay指定的时间。然后它会重置记数器然后继续执行。

这个特性是缺省关闭的。如需开启，需要把vacuum\_cost\_delay变量设置为一个非零值。

**vacuum\_cost\_delay**

**参数说明：**指定开销超过vacuum\_cost\_limit的值时，进程睡眠的时间。

要注意在许多系统上，睡眠的有效分辨率是10毫秒。因此把vacuum\_cost\_delay设置为一个不是10的整数倍的数值与将它设置为下一个10的整数倍作用相同。

此参数一般设置较小，常见的设置是10或20毫秒。调整此特性资源占用率时，最好是调整其他参数，而不是此参数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~100，正数值表示打开基于开销的清理延迟特性；0表示关闭基于开销的清理延迟特性。

**默认值：**0

**vacuum\_cost\_page\_hit**

**参数说明：**清理一个在共享缓存里找到的缓冲区的预计开销。它代表锁住缓冲池、查找共享的Hash表、扫描页面内容的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~10000。

**默认值：**1

**vacuum\_cost\_page\_miss**

**参数说明：**清理一个要从磁盘上读取的缓冲区的预计开销。它代表锁住缓冲池、查找共享Hash表、从磁盘读取需要的数据块、扫描它的内容的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~10000。

**默认值：**10

**vacuum\_cost\_page\_dirty**

**参数说明：**清理修改一个原先是干净的块的预计开销。它代表把一个脏的磁盘块再次刷新到磁盘上的额外开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~10000

**默认值：**20

**vacuum\_cost\_limit**

**参数说明：**设置清理进程休眠的开销限制。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整型，1~10000。

**默认值：**10000

### 后端写进程

介绍后端写（background writer）进程的参数配置。后端写进程的功能就是把共享缓冲区中的脏数据（指共享缓冲区中新增或者修改的内容）写入到磁盘。目的是让数据库进程在进行用户查询时可以很少或者几乎不等待写动作的发生（写动作由后端写进程完成）。

此机制同样也减少了检查点造成的性能下降。后端写进程将持续的把脏页面刷新到磁盘上，所以在检查点到来的时候，只有几个页面需要刷新到磁盘上。但是这样还是增加了I/O的总净负荷，因为以前的检查点间隔里，一个重复弄脏的页面可能只会冲刷一次，而同一个间隔里，后端写进程可能会写好几次。在大多数情况下，连续的低负荷要比周期性的尖峰负荷好，但是在本节讨论的参数可以用于按实际需要调节其行为。

**bgwriter\_delay**

**参数说明**：设置后端写进程写“脏”共享缓冲区之间的时间间隔。每一次，后端写进程都会为一些脏的缓冲区发出写操作（用bgwriter\_lru\_maxpages参数控制每次写的量），然后休眠bgwriter\_delay毫秒后才再次启动。

在许多系统上，休眠延时的有效分辨率是10毫秒。因此，设置一个不是10的倍数的数值与把它设置为下一个10的倍数是一样的效果。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，10~10000，单位为毫秒。

**默认值：**2s

**设置建议：**在数据写压力比较大的场景中可以尝试减小该值以降低checkpoint的压力。

**candidate\_buf\_percent\_target**

**参数说明**：设置用于增量检查点打开时，候选buffer链中可用buffer数目占据shared\_buffer内存缓冲区百分比的期望值，当前候选链中的数目少于目标值时，bgwriter线程会启动将满足条件的脏页刷盘。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：双精度浮点类型，0.1～0.85

**默认值**：0.3

**bgwriter\_lru\_maxpages**

**参数说明：**设置后端写进程每次可写入磁盘的“脏”缓存区的个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0～1000

fig: **说明：**   
此参数设置为0表示禁用后端写功能，禁用后端写功能不会对checkpoints产生影响。

**默认值：**100

**bgwriter\_lru\_multiplier**

**参数说明：**通过与已使用缓存区数目的乘积评估下次服务器需要的缓存区数目。

写“脏”缓存区到磁盘的数目取决于服务器最近几次使用的缓存区数目。最近的buffers数目的平均值乘以bgwriter\_lru\_multiplier是为了评估下次服务器进程需要的buffers数目。在有足够多的干净的、可用的缓存区之前，后端写进程会一直写“脏”缓存区的（每次写的缓存区数目不会超过bgwriter\_lru\_maxpages的值）。

设置bgwriter\_lru\_multiplier的值为1.0表示一种“实时”策略，其作用是精准预测下次写“脏”缓冲区的数目。设置为较大的值可以应对突然的需求高峰，而较小的值则可以让服务器进程执行更多的写操作。

设置较小的bgwriter\_lru\_maxpages和bgwriter\_lru\_multiplier会减小后端写进程导致的额外I/O开销，但是服务器进程必须自己发出写操作，增加了对查询的响应时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**浮点型，0~10。

**默认值：**2

**pagewriter\_thread\_num**

**参数说明**：设置用于增量检查点打开后后台刷页的线程数，主要是按照脏页置脏的顺序刷盘，用于推进recovery点。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置对应设置方法进行设置。

**取值范围**：整型，1～16

**默认值**：4

**dirty\_page\_percent\_max**

**参数说明**：设置用于增量检查点打开后脏页数量占shared\_buffers的百分比。达到这个设定值时，后台刷页线程将以设置的max\_io\_capacity计算出的最大值刷脏页。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0.1～1

**默认值**：0.9

**pagewriter\_sleep**

**参数说明**：设置用于增量检查点打开后，pagewrite线程每隔pagewriter\_sleep的时间刷一批脏页下盘。当脏页占据shared\_buffers的比例达到dirty\_page\_percent\_max时，每批页面数量以设定的max\_io\_capacity计算出的值刷页，其余情况每批页面数量按比例相对减少。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～3600000（毫秒）

**默认值**：5000ms

**max\_io\_capacity**

**参数说明：**设置后端写进程批量刷页每秒的IO上限，需要根据具体业务场景和机器磁盘IO能力进行设置。要求RTO很短时间或者数据量比共享内存大多倍的情况，业务访问数据量又是随机访问时，该值不宜过小。该参数设置较小会减小后端写进程刷页个数，如果业务触发页面淘汰多时，该值设置小会影响业务。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，30720~10485760。单位是KB。

**默认值：**512000KB（500MB）

**enable\_consider\_usecount**

**参数说明：**设置backend线程在页面置换时是否考虑页面热度，建议大容量场景下开启此参数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示考虑页面热度。
* off/false表示不考虑页面热度。

**默认值：**off

**dw\_file\_num**

**参数说明**：设置批量双写文件的数量，该值与pagewriter\_thread\_num有关，不会大于pagwriter\_thread\_num，如果设置过大，内部会纠正为pagewriter\_thread\_num。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1～16

**默认值**：1

**dw\_file\_size**

**参数说明**：设置每个批量双写文件的大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，32～256

**默认值**：256

### 异步IO

**enable\_adio\_debug**

**参数说明：**允许维护人员输出一些与ADIO相关的日志，便于定位ADIO相关问题。开发人员专用，不建议普通用户使用。

该参数属于SUSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on/true表示开启此日志开关。
* off/false表示关闭此日志开关。

**默认值：**off

fig: **说明：**   
当前版本暂不支持打开该开关，即使用户手动设置为打开，系统内部也会自动设置为关闭状态。

**enable\_adio\_function**

**参数说明：**是否开起ADIO功能。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

fig: **说明：**   
当前版本暂不支持开启异步IO功能，默认该功能关闭。

**取值范围：**布尔型

* on/true表示开启此功能。
* off/false表示关闭此功能。

**默认值：**off

**enable\_fast\_allocate**

**参数说明：**磁盘空间快速分配开关。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。只有在XFS文件系统上才能开启该开关。

**取值范围：**布尔型

* on/true表示开启此功能。
* off/false表示关闭此功能。

**默认值：**off

**prefetch\_quantity**

**参数说明：**描述行存储使用ADIO预读取IO量的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，128～131072，单位为8KB。

**默认值：**32MB (4096 \* 8KB)

**backwrite\_quantity**

**参数说明：**描述行存储使用ADIO写入IO量的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，128～131072，单位为8KB。

**默认值：**8MB (1024 \* 8KB)

**cstore\_prefetch\_quantity**

**参数说明：**描述列存储使用ADIO预取IO量的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1024～1048576，单位为KB。

**默认值：**32MB

**cstore\_backwrite\_quantity**

**参数说明：**描述列存储使用ADIO写入IO量的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1024～1048576，单位为KB。

**默认值：**8MB

**cstore\_backwrite\_max\_threshold**

**参数说明：**描述列存储使用ADIO写入数据库可缓存最大的IO量。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，4096～INT\_MAX/2，单位为KB。

**默认值：**2GB

**fast\_extend\_file\_size**

**参数说明：**描述列存储使用ADIO预扩展磁盘的大小。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1024～1048576，单位为KB。

**默认值：**8MB

**effective\_io\_concurrency**

**参数说明：**磁盘子系统可以同时有效处理的请求数。对于RAID阵列，此参数应该是阵列中驱动器主轴的数量。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~1000

**默认值：**1

**checkpoint\_flush\_after**

**参数说明**：设置checkpointer线程刷页个数超过设定的阈值时，告知操作系统开始将操作系统缓存中的页面异步刷盘。Vastbase中，磁盘页大小为8KB。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~256（0表示关闭异步刷盘功能）。例如，取值32，表示checkpointer线程连续写32个磁盘页，即32\*8=256KB磁盘空间后会进行异步刷盘。

**默认值**：32

**bgwriter\_flush\_after**

**参数说明**：设置background writer线程刷页个数超过设定的阈值时，告知操作系统开始将操作系统缓存中的页面异步刷盘。Vastbase中，磁盘页大小为8KB。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~256（0表示关闭异步刷盘功能），单位页面（8KB）。例如，取值64，表示background writer线程连续写64个磁盘页，即64\*8=512KB磁盘空间后会进行异步刷盘。

**默认值**：512KB（即64个页面）

**backend\_flush\_after**

**参数说明**：设置backend线程刷页个数超过设定的阈值时，告知操作系统开始将操作系统缓存中的页面异步刷盘。Vastbase中，磁盘页大小为8KB。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~256（0表示关闭异步刷盘功能），单位页面（8KB）。例如，取值64，表示backend线程连续写64个磁盘页，即64\*8=512KB磁盘空间后会进行异步刷盘。

**默认值**：0

## 预写式日志

### 设置

**wal\_level**

**参数说明：**设置写入WAL信息量的级别。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

须知：

* 如果需要启用WAL日志归档和主备机的数据流复制，必须将此参数设置为archive或者hot\_standby。
* 如果此参数设置为minimal, archive\_mode必须设置为off, hot\_standby必须设置为off，max\_wal\_senders参数设置为0, 且需为单机环境, 否则将导致数据库无法启动。
* 如果此参数设置为archive，hot\_standby必须设置为off，否则将导致数据库无法启动。但是，hot\_standby在双机环境中不能设置为off，具体参见hot\_standby参数说明。

**取值范围：**枚举类型

* minimal
* 优点：一些重要操作（包括创建表、创建索引、簇操作和表的复制）都能安全的跳过，这样就可以使操作变得更快。
* 缺点：WAL仅提供从数据库服务器崩溃或者紧急关闭状态恢复时所需要的基本信息，无法用WAL归档日志恢复数据。
* archive
* 这个参数增加了WAL归档需要的日志信息，从而可以支持数据库的归档恢复。
* hot\_standby
* 这个参数进一步增加了在备机上运行的SQL查询的信息，这个参数只能在数据库服务重新启动后生效。
* 为了在备机上开启只读查询，wal\_level必须在主机上设置成hot\_standby ，并且备机必须打开hot\_standby参数。hot\_standby和archive级别之间的性能只有微小的差异，如果它们的设置对产品的性能影响有明显差异，欢迎反馈。
* logical

这个参数表示WAL日志支持逻辑复制。

**默认值：**hot\_standby

**fsync**

**参数说明：**设置Vastbase服务器是否使用fsync()系统函数（请参见wal\_sync\_method）确保数据的更新及时写入物理磁盘中。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**

* 使用fsync()系统函数可以保证在操作系统或者硬件崩溃的情况下将数据恢复到一个已知的状态。
* 如果将此参数关闭，可能会在系统崩溃时无法恢复原来的数据，导致数据库不可用。

**取值范围：**布尔型

* on表示使用fsync()系统函数。
* off表示不使用fsync()系统函数。

**默认值：**on

**synchronous\_commit**

**参数说明：**设置当前事务的同步方式。开启后支持从库在主库commit后可直接读取已提交的数据。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

通常情况下，一个事务产生的日志的同步顺序如下：

1、主机将日志内容写入本地内存。

2、主机将本地内存中的日志写入本地文件系统。

3、主机将本地文件系统中的日志内容刷盘。

4、主机将日志内容发送给备机。

5、备机接受到日志内容，存入备机内存。

6、备机将备机内存中的日志写入备机文件系统。

7、备机将备机文件系统中的日志内容刷盘。

8、备机回放日志，完成对数据文件的增量更新。

**取值范围：**枚举类型

* on：表示主机事务提交需要等待备机将对应日志刷新到磁盘。
* off：表示主机事务提交无需等待主机自身将对应日志刷新到磁盘，通常也称为异步提交。
* local：表示主机事务提交需要等待主机自身将对应日志刷新到磁盘，通常也称为本地提交。
* remote\_write：表示主机事务提交需要等待备机将对应日志写到文件系统（无需刷新到磁盘）。
* remote\_receive：表示主机事务提交需要等待备机接收到对应日志数据（无需写入文件系统）。
* remote\_apply：表示主机事务提交需要等待备机完成对应日志的回放操作。
* true：同on。
* false：同off。
* yes：同on。
* no：同off。
* 1：同on。
* 0：同off。
* 2：同remote\_apply。

**默认值：**on

**wal\_sync\_method**

**参数说明：**设置向磁盘强制更新WAL数据的方法。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**如果将 fsync 关闭，这个参数的设置就没有意义，因为所有数据更新都不会强制写入磁盘。

**取值范围：**枚举类型

* open\_datasync表示用带O\_DSYNC选项的open()打开“WAL”文件。
* fdatasync表示每次提交的时候都调用fdatasync()。（支持suse10和suse11）。
* fsync\_writethrough表示每次提交的时候调用fsync()强制把缓冲区任何数据写入磁盘。
* 由于历史原因，我们允许在Windows平台上将wal\_sync\_method设置为fsync\_writethrough，尽管它和fsync等效。
* fsync表示每次提交的时候调用fsync()。（支持suse10和suse11）
* open\_sync表示用带O\_SYNC选项的open()写“WAL”文件。（支持suse10和suse11）

**默认值：**fdatasync

**full\_page\_writes**

**参数说明：**设置Vastbase服务器在检查点之后对页面的第一次修改时，是否将每个磁盘页面的全部内容写到WAL日志中。当增量检查点开关和enable\_double\_write同时打开时，则不使用full\_page\_writes。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**

* 设置这个参数是因为在操作系统崩溃过程中可能磁盘页面只写入了一部分内容，从而导致在同一个页面中包含新旧数据的混合。在崩溃后的恢复期间，由于在WAL日志中存储的行变化信息不够完整，因此无法完全恢复该页。把完整的页面影像保存下来就可以保证页面被正确还原，代价是增加了写入WAL日志的数据量。
* 关闭此参数，在系统崩溃的时候，可能无法恢复原来的数据。如果服务器硬件的特质（比如电池供电的磁盘控制器）可以减小部分页面的写入风险，或者文件系统特性支持（比如ReiserFS 4），并且清楚知道写入风险在一个可以接受的范畴，可以关闭这个参数。

**取值范围：**布尔型

* on表示启用此特性。
* off表示关闭此特性。

**默认值：**on

**wal\_log\_hints**

**参数说明：**设置在检查点之后对页面的第一次修改为页面上元组hint bits的修改时，是否将整个页面的全部内容写到WAL日志中。不推荐用户修改此设置。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示整个页面全部内容写到WAL日志中。
* off表示整个页面内容不会写到WAL日志中。

**默认值：**on

**wal\_buffers**

**参数说明：**设置用于存放WAL数据的共享内存空间的XLOG\_BLCKSZ数，XLOG\_BLCKSZ的大小默认为8KB。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**-1~218

* 如果设置为-1，表示wal\_buffers的大小随着参数shared\_buffers自动调整，为shared\_buffers的1/32，最小值为8个XLOG\_BLCKSZ，最大值为2048个XLOG\_BLCKSZ。
* 如果设置为其他值，当小于4时，会被默认设置为4。

**默认值：**16MB

**设置建议：**每次事务提交时，WAL缓冲区的内容都写入到磁盘中，因此设置为很大的值不会带来明显的性能提升。如果将它设置成几百兆，就可以在有很多即时事务提交的服务器上提高写入磁盘的性能。根据经验来说，默认值可以满足大多数的情况。

**wal\_writer\_delay**

**参数说明：**WalWriter进程的写间隔时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**如果时间过长可能造成WAL缓冲区的内存不足，时间过短会引起WAL不断写入，增加磁盘I/O负担。

**取值范围：**整型， 1～10000（毫秒）

**默认值：**200ms

**commit\_delay**

**参数说明：**表示一个已经提交的数据在WAL缓冲区中存放的时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**

* 设置为非 0 值时事务执行commit后不会立即写入WAL中，而仍存放在WAL缓冲区中，等待WalWriter进程周期性写入磁盘。
* 如果系统负载很高，在延迟时间内，其他事务可能已经准备好提交。但如果没有事务准备提交，这个延迟就是在浪费时间。

**取值范围：**整型， 0～100000（微秒），其中0表示无延迟。

**默认值：**0

**commit\_siblings**

**参数说明：**当一个事务发出提交请求时，如果数据库中正在执行的事务数量大于此参数的值，则该事务将等待一段时间（commit\_delay的值），否则该事务则直接写入WAL。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型， 0～1000

**默认值：**5

**wal\_block\_size**

**参数说明：**说明WAL日志段文件中日志页面的大小。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整型，单位为Byte。

**默认值：**8192

**wal\_segment\_size**

**参数说明：**说明WAL日志段文件的大小。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整型，单位为8KB。

**默认值：**16MB (2048 \* 8KB)

**walwriter\_cpu\_bind**

**参数说明：**绑定到WAL写入线程的CPU核，与thread pool配合使用。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，-1~核数减1。

**默认值：**-1

**walwriter\_sleep\_threshold**

**参数说明：**xlogflusher进入sleep之前空闲xlog刷新的次数，达到阈值会休眠。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~50000。

**默认值：**500

**wal\_file\_init\_num**

**参数说明：**WAL编写器将创建的xlog段文件的数量。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~1000000。

**默认值：**10

**force\_promote**

**参考说明：**备机强切功能开关。

备机强切在集群故障状态下，以丢失部分数据为代价换取集群尽可能快的恢复服务；是集群状态为不可用时的一个逃生方法，不建议频繁触发。如果操作者不清楚备机强切后丢失数据对业务的影响，请勿使用本功能。

**取值范围：**整型，0或1

0表示关闭，1表示开启

**默认值：**0

**wal\_flush\_timeout**

**参数说明：**Xlog刷盘自适应控制的刷盘IO遍历WalInsertStatusTable等待的最大时间，即遍历WalInsertStatusTable的超时时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**

* 如果时间过长可能造成Xlog刷盘频率降低，降低Xlog处理性能。
* 遍历WalInsertStatusTable是数据库内部的实现，所以通过合理设置wal\_flush\_timeout的值来控制IO频率比较困难。

**取值范围：**整型， 0 ~ 90000000（微秒）

**默认值：**2（微秒）

**wal\_flush\_delay**

**参数说明：**遍历WalInsertStatusTable时，遇到WAL\_NOT\_COPIED状态entry时等待的时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**须知：**参数wal\_flush\_delay值建议不要配置过大，原因如下：

* wal\_flush\_delay和wal\_flush\_timeout两个参数综合作用影响事务的提交。
* 一旦触发WALWriter休眠逻辑后，因为事务提交的等待时长为该参数配置的值，则当wal\_flush\_delay值设置过大时等待时长将过长。

**取值范围：**整型， 0 ~ 90000000（微秒）

**默认值：**1（微秒）

**vb\_wal\_directory**

**参数说明：**xlog日志的存放路径。

xlog的默认存放路径在$PGDATA/pg\_xlog下，如有更改路径的需求，可以修改postgresql.conf中的参数vb\_wal\_directory。新的存放路径应是存在的且可访问的。

**默认值：**$PGDATA/pg\_xlog

### 检查点

**checkpoint\_segments**

**参数说明：**设置checkpoint\_timeout周期内所保留的最少WAL日志段文件数量。每个日志文件大小为16MB。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~2147483646

提升此参数可加快大数据的导入速度，但需要结合checkpoint\_timeout、shared\_buffers这两个参数统一考虑。这个参数同时影响WAL日志段文件复用数量，通常情况下pg\_xlog文件夹下最大的复用文件个数为2倍的checkpoint\_segments个，复用的文件被改名为后续即将使用的WAL日志段文件，不会被真正删除。

**默认值：**64

**checkpoint\_timeout**

**参数说明：**设置自动WAL检查点之间的最长时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，30～3600（秒）

在提升checkpoint\_segments以加快大数据导入的场景也需将此参数调大，同时这两个参数提升会加大shared\_buffers的负担，需要综合考虑。

**默认值：**15min

**checkpoint\_completion\_target**

**参数说明：**指定检查点完成的目标。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：双精度浮点类型，0.0～1.0

**默认值：**0.5

fig: **说明：**   
默认值0.5表示每个checkpoint需要在checkpoints间隔时间的50%内完成。

**checkpoint\_warning**

**参数说明：**如果由于填充检查点段文件导致检查点发生的时间间隔接近这个参数表示的秒数，就向服务器日志发送一个建议增加checkpoint\_segments值的消息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~INT\_MAX，单位秒。

**默认值：**5min

**推荐值：**5min

**checkpoint\_wait\_timeout**

**参数说明：**设置请求检查点等待checkpointer线程启动的最长时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，2～3600（秒）

**默认值：**1min

**enable\_incremental\_checkpoint**

**参数说明：**增量检查点开关。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：布尔型

**默认值：**on

**enable\_double\_write**

**参数说明：**双写开关。当增量检查点开关打开时，同时enable\_double\_write打开，则使用enable\_double\_write双写特性保护，不再使用full\_page\_writes防止半页写问题。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

**默认值：**on

**incremental\_checkpoint\_timeout**

**参数说明：**增量检查点开关打开之后，设置自动WAL检查点之间的最长时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~3600（秒）

**默认值：**1min

\*\*enable\_xlog\_prune

**参数说明：**设置在任一备机断联时，主机是否根据xlog日志的大小超过参数max\_size\_for\_xlog\_prune的值而回收日志。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* 设置为on时，如果任一备机断联时，主机回收日志。
* 设置为off时，如果任一备机断联时，主机不回收日志。

**默认值：**on

**max\_redo\_log\_size**

**参数说明：**备DN表示当前回放的最新检查点位置和当前日志回放位置之间日志量的期望值，主DN表示恢复点到当前最新日志之间日志量的期望值，关注RTO的情况下，这个值建议不宜过大。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，163840～2147483647，单位为KB。

**默认值：**1GB

**max\_size\_for\_xlog\_prune**

**参数说明：**在enable\_xlog\_prune、synchronous\_commit都打开时生效，如果有备机断连且xlog日志大小大于此阈值，则回收日志。所有备机断联且无逻辑复制槽时，不回收日志。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～2147483647，单位为KB

**默认值：**2147483647，单位KB

### 日志回放

**recovery\_time\_target**

**参数说明：**设置recovery\_time\_target秒能够让备机完成日志写入和回放。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~3600（秒）

0是指不开启日志流控，1~3600是指备机能够在recovery\_time\_target时间内完成日志的写入和回放，可以保证主机与备机切换时能够在recovery\_time\_target秒完成日志写入和回放，保证备机能够快速升主机。recovery\_time\_target设置时间过小会影响主机的性能，设置过大会失去流控效果。另外，由于极致RTO自带流控，所以同时开启极致RTO与流控时会以极致RTO优先，在运行期间使流控不生效。

**默认值：**0

**recovery\_max\_workers**

**参数说明：**设置最大并行回放线程个数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~20

**默认值：**1（安装工具默认设置为4，以获得更好的性能）

**recovery\_parse\_workers**

**参数说明：**是极致RTO特性中ParseRedoRecord线程的数量。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：整型，1~16

仅在开启极致RTO情况下可以设置recovery\_parse\_workers为>1。需要配合recovery\_redo\_workers使用。若同时开启recovery\_parse\_workers和recovery\_max\_workers，以开启极致RTO的recovery\_parse\_workers为准，并行回放特性失效。因极致RTO不支持hot standby模式和主备从模式，仅在参数hot\_standby设置成off，replication\_type设置成1时可以设置recovery\_parse\_workers为>1。另外，极致RTO也不支持列存，在已经使用列存表或者即将使用列存表的系统中，请关闭极致RTO。由于极致RTO自带流控，所以同时开启极致RTO与流控时会以极致RTO优先，在运行期间使流控不生效。

**默认值：**1

**recovery\_redo\_workers**

**参数说明：**是极致RTO特性中每个ParseRedoRecord线程对应的PageRedoWorker数量。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：整型，1~8

需要配合recovery\_parse\_workers使用。在配合recovery\_parse\_workers使用时，只有recovery\_parse\_workers大于0，recovery\_redo\_workers参数才生效。

**默认值：**1

**recovery\_parallelism**

**参数说明：**查询实际回放线程个数，该参数为只读参数，无法修改。

该参数属于POSTMASTER类型参数，受recovery\_max\_workers以及recovery\_parse\_workers参数影响，任意一值大于0时，recover\_parallelism将被重新计算。

**取值范围**：整型，1~2147483647

**默认值：**1

**enable\_page\_lsn\_check**

**参数说明：**数据页lsn检查开关。回放时，检查数据页当前的lsn是否是期望的lsn。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

**默认值：**on

**recovery\_min\_apply\_delay**

**参数说明：**设置备节点回放的延迟时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

* 此参数主节点设置无效，必须设置在需要延迟的备节点上，推荐设置在异步备上，设置了延时的异步备如果升主RTO时间会比较长。
* 延迟时间是根据主服务器上事务提交的时间戳与备机上的当前时间来计算，因此需要保证主备系统时钟一致。
* 延迟时间设置过长时，可能会导致该备机XLOG文件所在的磁盘满，需要平衡考虑磁盘大小来设置延迟时间。
* 没有事务的操作不会被延迟。
* 主备切换之后，原主机若需延迟，需要再手动配置此参数。
* 当synchronous\_commit被设置为remote\_apply时，同步复制会受到这个延时的影响，每一个COMMIT都需要等待备机回放结束后才会返回。
* 使用这个特性也会让hot\_standby\_feedback被延迟，这可能导致主服务器的膨胀，两者一起使用时要小心。
* 主机执行了持有AccessExclusive锁的DDL操作，比如DROP和TRUNCATE操作，在备机延迟回放该条记录期间，在备机上对该操作对象执行查询操作会等待锁释放之后才会返回。
* 不支持MOT表。

**取值范围：**整型，0~INT\_MAX，单位为毫秒。

**默认值：**0（不增加延迟）

**redo\_bind\_cpu\_attr**

**参数说明**：用于控制回放线程的绑核操作，仅sysadmin用户可以访问。该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串，长度大于0，该参数不区分大小写。

* 'nobind'：线程不做绑核。
* 'nodebind: 1, 2'：利用NUMA组1,2中的CPU core进行绑核。
* 'cpubind: 0-30'：利用0-30号CPU core进行绑核。

**默认值**：'nobind'

### 归档

**archive\_mode**

**参数说明：**表示是否进行归档操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**  
当wal\_level设置成minimal时，archive\_mode参数无法使用。

无论是同步备机还是异步备机都能够开启归档，归档开启的方式与单机开启归档一致，将archive\_mode置为on，并设置正确的archive\_dest或者archive\_command即可。

注：若未开启最大可用模式以及有同步备机与主机断开连接时，主机会因为业务阻塞的原因无法给备机发送归档的位置，从而导致归档失败。

**取值范围**：布尔型

* on表示进行归档。
* off表示不进行归档。

**默认值：**off

**archive\_command**

**参数说明：**由管理员设置的用于归档WAL日志的命令，建议归档路径为绝对路径。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 当archive\_dest和archive\_command同时配置时，WAL日志优先保存到archive\_dest所设置的目录中，archive\_command配置的命令不生效。
* 字符串中任何%p都被要归档的文件的绝对路径代替，而任何%f都只被该文件名代替（相对路径都相对于数据目录的）。如果需要在命令里嵌入%字符就必须双写%。
* 这个命令当且仅当成功的时候才返回零。示例如下：
* archive\_command = 'cp --remove-destination %p /mnt/server/archivedir/%f'
* --remove-destination选项作用为：拷贝前如果目标文件已存在，会先删除已存在的目标文件，然后执行拷贝操作。
* 如果归档命令有多条，则需将其写入SHELL脚本文件中，然后将archive\_command配置为执行该脚本的命令。示例如下：
* --假设多条命令如下。  
  test ! -f dir/%f && cp %p dir/%f  
  --则test.sh脚本内容如下。  
  test ! -f dir/$2 && cp $1 dir/$2   
  --归档命令如下。  
  archive\_command='sh dir/test.sh %p %f'

**取值范围**：字符串

**默认值：**(disabled)

**archive\_dest**

**参数说明：**由管理员设置的用于归档WAL日志的目录，建议归档路径为绝对路径。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 当archive\_dest和archive\_command同时配置时，WAL日志优先保存到archive\_dest所设置的目录中，archive\_command配置的命令不生效。
* 字符串中如果是相对路径为相对于数据目录的。示例如下。
* archive\_dest = '/mnt/server/archivedir/'

**取值范围**：字符串

**默认值：**空字符串

**archive\_timeout**

**参数说明**：表示归档周期。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 超过该参数设定的时间时强制切换WAL段。
* 由于强制切换而提早关闭的归档文件仍然与完整的归档文件长度相同。因此，将archive\_timeout设为很小的值将导致占用巨大的归档存储空间，建议将archive\_timeout设置为60秒。

**取值范围**：整型，0 ~ INT\_MAX，单位为秒。其中0表示禁用该功能。

**默认值**：0

## 双机复制

### 发送端服务器

**max\_wal\_senders**

**参数说明：**指定事务日志发送进程的并发连接最大数量。不可大于等于max\_connections，参考：GUC参数说明->连接和认证->连接设置章节。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
wal\_level必须设置为archive或者hot\_standby以允许备机的连接。

**取值范围**：整型，0～1024

* 建议取值范围为8 ~ 100。
* 只有当使用单DN实例无主备场景下才可以设置为0。

**默认值：**单机环境默认值为4；主备环境默认值为8。

**wal\_keep\_segments**

**参数说明：**Xlog日志文件段数量。设置“pg\_xlog”目录下保留事务日志文件的最小数目，备机通过获取主机的日志进行流复制。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，2 ~ INT\_MAX

**默认值：**16

**设置建议：**

* 当服务器开启日志归档或者从检查点恢复时，保留的日志文件数量可能大于wal\_keep\_segments设定的值。
* 如果此参数设置过小，则在备机请求事务日志时，此事务日志可能已经被产生的新事务日志覆盖，导致请求失败，主备关系断开。
* 当双机为异步传输时，以COPY方式连续导入4G以上数据需要增大wal\_keep\_segments配置。以T6000单板为例，如果导入数据量为50G，建议调整参数为1000。您可以在导入完成并且日志同步正常后，动态恢复此参数设置。
* 若 synchronous\_commit 级别小于LOCAL\_FLUSH，重建备机时，建议调大改参数为1000，避免重建过程中，主机日志回收导致重建失败。

**wal\_sender\_timeout**

**参数说明：**设置本端等待事务日志接收端接收日志的最大等待时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 如果主机数据较大，重建备机数据库时需要增大此参数的值，主机数据在 500G时，此参数的参考值为600s。
* 此值不能大于wal\_receiver\_timeout或数据库重建时的超时参数。

**取值范围：**整型，0 ~ INT\_MAX，单位为毫秒（ms）。

**默认值：**6s

**max\_replication\_slots**

**参数说明：**设置主机端的日志复制slot个数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~1024（建议取值范围：8~100）

**默认值：**8

**设置建议**：

当使用双机复制、逻辑解码时，该参数值建议设为：当前物理流复制槽数+所需的逻辑复制槽数。如果实际设置值比上述建议值要小，那么可能造成这些功能不可用或异常。

* 物理流复制槽提供了一种自动化的方法来确保主节点在所有备节点或从备节点收到xlog之前，xlog不会被移除。也就是说物理流复制槽用于支撑主备HA。数据库所需要的物理流复制槽数为备节点加从备的和与主节点之间的比例。例如，假设数据库高可用方案为1主、1备、1从备，则所需物理流复制槽数为2。假设数据库的高可用方案为1主3备，则所需物理流复制槽数为3。
* 目前默认不支持主备从部署方式。
* 关于逻辑复制槽数，请按如下规则考虑：
* 一个逻辑复制槽只能解码一个数据库的修改，如果需要解码多个数据库，则需要创建多个逻辑复制槽。
* 如果需要多路逻辑复制同步给多个目标数据库，在源端数据库需要创建多个逻辑复制槽，每个逻辑复制槽对应一条逻辑复制链路。

**enable\_slot\_log**

**参数说明**：是否开启逻辑复制槽主备同步特性。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启逻辑复制槽主备同步特性。
* off表示不开启逻辑复制槽主备同步特性。

**默认值**：off

**max\_changes\_in\_memory**

**参数说明**：逻辑解码时单条事务在内存中缓存的DML语句数量上限。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~2147483647‬

**默认值**：4096

**max\_cached\_tuplebufs**

**参数说明**：逻辑解码时总元组信息在内存中缓存的数量上限。建议设置为max\_changes\_in\_memory的两倍以上。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~2147483647‬

**默认值**：8192

**enable\_wal\_shipping\_compression**

**参数说明：**在流式容灾模式下设置启动跨数据库实例日志压缩功能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

该参数仅作用于流式容灾中跨数据库实例传输的一对walsender与walreceiver中，在主数据库实例上配置。

**取值范围**：布尔型

* true表示打开流式容灾跨数据库实例日志压缩
* false表示关闭流式容灾跨数据库实例日志压缩

**默认值：**false

**replconninfo1**

**参数说明：**设置本端侦听和鉴权的第一个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第一个节点信息。

**默认值：**空字符串

**replconninfo2**

**参数说明：**设置本端侦听和鉴权的第二个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第二个节点信息。

**默认值：**空字符串

**replconninfo3**

**参数说明：**设置本端侦听和鉴权的第三个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第三个节点信息。

**默认值：**空字符串

**replconninfo4**

**参数说明：**设置本端侦听和鉴权的第四个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第四个节点信息。

**默认值：**空字符串

**replconninfo5**

**参数说明：**设置本端侦听和鉴权的第五个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第五个节点信息。

**默认值：**空字符串

**replconninfo6**

**参数说明：**设置本端侦听和鉴权的第六个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第六个节点信息。

**默认值：**空字符串

**replconninfo7**

**参数说明：**设置本端侦听和鉴权的第七个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第七个节点信息。

**默认值：**空字符串

**replconninfo8**

**参数说明：**设置本端侦听和鉴权的第八个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第八个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo1**

**参数说明：**设置跨集群的本端侦听和鉴权的第一个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第一个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo2**

**参数说明：**设置跨集群的本端侦听和鉴权的第二个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第二个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo3**

**参数说明：**设置跨集群的本端侦听和鉴权的第三个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第三个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo4**

**参数说明：**设置跨集群的本端侦听和鉴权的第四个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第四个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo5**

**参数说明：**设置跨集群的本端侦听和鉴权的第五个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第五个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo6**

**参数说明：**设置跨集群的本端侦听和鉴权的第六个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第六个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo7**

**参数说明：**设置跨集群的本端侦听和鉴权的第七个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第七个节点信息。

**默认值：**空字符串

**cross\_cluster\_replconninfo8**

**参数说明：**设置跨集群的本端侦听和鉴权的第八个节点信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置第八个节点信息。

**默认值：**空字符串

**available\_zone**

**参数说明：**设置本端节点所在区域信息。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。其中空字符串表示没有配置节点信息。

**默认值：**空字符串

### 主服务器

**synchronous\_standby\_names**

**参数说明：**潜在同步复制的备机名称列表，每个名称用逗号分隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 当前连接的同步备机是列表中的第一个名称。如果当前同步备机失去连接，则它会立即更换下一个优先级更高的备机，并将此备机的名称放入列表中。
* 备机名称可以通过设置环境变量PGAPPNAME指定。

**取值范围：**字符串。当取值为\*，表示匹配任意提供同步复制的备机名称。支持按如下格式配置：

* ANY num\_sync (standby\_name [, ...]) [, ANY num\_sync (standby\_name [, ...])]
* [FIRST] num\_sync (standby\_name [, ...])
* standby\_name [, ...]

fig: **说明：**

* 其中 num\_sync 是事务需要等待其回复的同步复制的备机的数量，standby\_name 是备机的名称，FIRST以及ANY指定从所列服务器中选取同步复制的备机的策略。
* ANY N (node1,node2,...) 表示在括号内任选N个主机名称作为同步复制的备机名称列表。例如，ANY 1 (node1,node2) 表示在node1和node2中任选一个作为同步复制的备机名称。
* ANY N1 (node1,node2,...), ANY N2 (node3,node4,...) 表示分组潜在同步复制的备机名称列表，在第一组括号内任选N1个主机名称作为第一组同步复制的备机名称列表，在第二组括号内任选N2个主机名称作为第二组同步复制的备机名称列表。此时两个分组之间为且关系，必须两个分组均达到各自需求的同步备机数，本地事务才可以被提交。
* FIRST N (node1,node2,...)表示在括号内按出现顺序的先后作为优先级选择前N个主机名称作为同步复制的备机名称列表。例如，FIRST 1 (node1,node2)表示选择node1作为同步复制的备机名称。
* node1,node2,...和FIRST 1 (node1,node2,...) 具有的含义相同。
* 若使用vb\_guc工具设置该参数，需要如下设置：

vb\_guc reload -Z datanode -D @DN\_PATH@ -c "synchronous\_standby\_names='ANY NODE 1(dn\_instanceId1, dn\_instanceId2)'";

或者：

vb\_guc reload -Z datanode -D @DN\_PATH@ -c "synchronous\_standby\_names='ANY 1(AZ1, AZ2)'";

**默认值：\***

fig: **说明：**

* 备机名称列表中不可出现重复的名称，配置中num\_sync 不可大于备机列表数量。
* 多分组同步备机配置 如ANY N1 (node1,node2,...), ANY N2 (node3,node4,...) 的时候，多个分组之间为且关系，当前仅支持多ANY分组。不允许使用 \* 来作为模糊匹配，不允许出现配置重复的备机。

**most\_available\_sync**

**参数说明：**在有同步备机故障时，主机事务不因同步备机故障而被阻塞。比如有两个同步备机，一个故障，另一个正常，这个时候主机事务只会等好的这个同步备，而不被故障的同步备所阻塞；  
再比如走quorum协议时，一主三同步备，配置ANY 2(node1,node2,node3)，当node1、node3故障，node2正常时，主机业务同样不被阻塞。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在有同步备机故障时，不阻塞主机。
* off表示在有同步备机故障时，阻塞主机。

**默认值**：off

**keep\_sync\_window**

**参数说明：**延迟进入最大可用模式的时间。

* 当最大可用模式most\_available\_sync配置为on，在主备场景下，当存在同步备发生故障，导致不满足当前所配置的同步备数量(详细可参考synchonous\_standby\_name的含义)时，如果配置了keep\_sync\_window参数, 则在keep\_sync\_window设置的时间窗口内，继续保持最大保护模式，即阻塞主机的事务提交, 延缓进入最大可用模式的时间。
* 若在keep\_sync\_window超时窗口内，同步备机故障恢复，且满足当前所配置的同步备数量, 则不阻塞事务, 恢复到正常状态.
* 如果设置keep\_sync\_window，推荐最小配置为5s，以避免监控系统监控到网络不稳定的误报。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整形，范围0~INT\_MAX，单位为秒。

* 0表示不设置keep\_sync\_window超时时间窗口, 即直接进入最大可用模式。
* 其余表示keep\_sync\_window超时时间窗口的大小。

**默认值**：0

fig: **须知：**

配置该参数可能会对RPO造成影响, 若主机在所配置的超时时间窗口内发生故障，则从开始阻塞到主机故障这段时间窗口内的数据可能丢失。

**enable\_stream\_replication**

**参数说明：**控制主备、主从是否进行数据和日志同步。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 此参数属于性能测试参数，用于测试带有备机和不带备机的性能参数。关闭参数后，不能进行切换、故障等异常场景测试，否则会出现主备从不一致的情况。
* 此参数属于受控参数，不建议正常业务场景下关闭此参数。
* 当前版本默认不支持主备从部署模式。

**取值范围：**布尔型

* on表示打开主备、主从同步。
* off表示关闭主备、主从同步。

**默认值**：on

**enable\_mix\_replication**

**参数说明：**控制主备、主从之间WAL日志及数据复制的方式。

该参数属于INTERNAL类型参数，默认值为off，不允许外部修改。

fig: **须知：**

* 此参数目前不允许正常业务场景下改变其值，即关闭WAL日志、数据页混合复制模式。
* 当前版本默认不支持主备从部署模式。

**取值范围：**布尔型

* on表示打开WAL日志、数据页混合复制模式。
* off表示关闭WAL日志、数据页混合复制模式。

**默认值**：off

**vacuum\_defer\_cleanup\_age**

**参数说明：**指定VACUUM使用的事务数，VACUUM会延迟清除无效的行存表记录，延迟的事务个数通过vacuum\_defer\_cleanup\_age进行设置。即VACUUM和VACUUM FULL操作不会立即清理刚刚被删除元组。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0～1000000，值为0表示不延迟。

**默认值**：0

**data\_replicate\_buffer\_size**

**参数说明：**发送端与接收端传递数据页时，队列占用内存的大小。此参数会影响主备之间复制的缓冲大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，4096~1072693248，单位为KB。

**默认值：**16MB（即16384KB）

**walsender\_max\_send\_size**

**参数说明：**设置主机端日志或数据发送缓冲区的大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，8~INT\_MAX，单位为KB。

**默认值：**8M（即8192KB）

**enable\_data\_replicate**

**参数说明：**当数据库在数据导入行存表时，主机与备机的数据同步方式可以进行选择。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示导入数据行存表时主备数据采用数据页的方式进行同步。当replication\_type参数为1时，不允许设置为on，如果此时用guc工具设置成on，会强制改为off。
* off表示导入数据行存表时主备数据采用日志（Xlog）方式进行同步。

**默认值：**off

**ha\_module\_debug**

**参数说明：**用于查看数据复制时具体数据块的复制状态日志。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示日志中将打印数据复制时每个数据块的状态。
* off表示日志中不打印数据复制时每个数据块的状态。

**默认值：**off

**enable\_incremental\_catchup**

**参数说明：**控制主备之间数据追赶（catchup）的方式，目前默认不支持主备从部署模式。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示备机catchup时用增量catchup方式，即从从备本地数据文件扫描获得主备差异数据文件列表，进行主备之间的catchup。
* off表示备机catchup时用全量catchup方式，即从主机本地所有数据文件扫描获得主备差异数据文件列表，进行主备之间的catchup。

**默认值**：on

**wait\_dummy\_time**

**参数说明：**同时控制增量数据追赶（catchup）时，Vastbase主备从按顺序启动时等待从备启动的最长时间以及等待从备发回扫描列表的最长时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，范围1~INT\_MAX，单位为秒

**默认值**：300

fig: **说明：**

* 单位只能设置为秒。
* 当前版本默认不支持主备从部署模式。

**catchup2normal\_wait\_time**

**参数说明：**打开最大可用模式most\_available\_sync，主备场景下，控制备机数据追赶（catchup）阻塞主机的最长时间。该时间为估算值，实际结果可能与参数值有偏差。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，范围-1~10000，单位为毫秒。

* -1表示主机阻塞直到备机数据追赶完成。
* 0表示备机数据追赶时始终不阻塞主机。
* 其余值表示备机数据追赶时阻塞主机的最长时间。例如，取值5000，表示当备机数据追赶完成时间还剩5s时，阻塞主机等待其完成。

**默认值**：-1

**sync\_config\_strategy**

**参数说明：**主机和备机、备机和级联备之间配置文件的同步策略。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

* all\_node: 主机配置为all\_node时，表示允许主机向所有备机主动同步配置文件；备机配置为all\_node时，表示允许当前备机向其主机发送同步请求，允许当前备机向其所有级联备主动同步配置文件；级联备配置为all\_node时，表示允许当前级联备向上一级备机发送同步请求。
* only\_sync\_node: 主机配置为only\_sync\_node时，表示仅允许主机向所有同步备机主动同步配置文件；备机配置为only\_sync\_node时，表示允许当前备机向其主机发送同步请求，不允许当前备机向其所有级联备机主动同步配置文件；级联备配置为only\_sync\_node时，表示允许当前级联备向其备机发送同步请求。
* none\_node: 主机配置为none\_node时，表示不允许主机向任何备机主动同步配置文件；备机配置为none\_node时，表示不允许当前备机向其主机发送同步请求，不允许当前备机向其所有级联备主动同步配置文件；级联备配置为none\_node时，表示不允许当前级联备向其备机发送同步请求。

**默认值：**all\_node

fig: **须知：**

* 在一个包含了主机、备机和级联备的Vastbase集群中，主机相对于备机是发送端，备机相对于主机是接收端，备机相对于级联备是发送端，级联备相对于备机是接收端。
* 发送端主动向接收端同步配置文件、接收端请求发送端同步配置文件是两个独立的事件，均会使得配置文件同步。若不希望配置文件同步，则需要将集群中所有节点的sync\_config\_strategy参数配置为none\_node；若仅希望主机与同步备机同步配置文件，则需要将主机的sync\_config\_strategy参数配置为only\_sync\_node，其余节点配置为none\_node；若希望所有节点同步配置文件，则需要将所有节点的sync\_config\_strategy参数配置为all\_node。目前暂不支持自定义指定任意节点间的同步策略。
* 配置参数同步的具体表现为，发送端发送配置文件，对接收端配置文件中的对应参数直接覆盖。若设置了配置文件需要同步的策略，则修改接收端配置参数后，发送端会立刻覆盖接收端的配置参数，使得接收端修改不生效。
* 即使设置了配置文件需要同步的策略，仍有部分配置参数不会被同步。它们是："application\_name"、 "archive\_command"、 "audit\_directory"、 "available\_zone"、 "comm\_control\_port"、 "comm\_sctp\_port"、 "listen\_addresses"、 "log\_directory"、 "port"、 "replconninfo1"、 "replconninfo2"、 "replconninfo3"、 "replconninfo4"、 "replconninfo5"、 "replconninfo6"、 "replconninfo7"、 "replconninfo8"、 "ssl"、 "ssl\_ca\_file"、 "ssl\_cert\_file"、 "ssl\_ciphers"、 "ssl\_crl\_file"、 "ssl\_key\_file"、 "ssl\_renegotiation\_limit"、 "ssl\_cert\_notify\_time"、 "synchronous\_standby\_names"、 "local\_bind\_address"、 "perf\_directory"、 "query\_log\_directory"、 "asp\_log\_directory"、 "streaming\_router\_port"、 "enable\_upsert\_to\_merge"、 "archive\_dest"、 "recovery\_min\_apply\_delay"、 "sync\_config\_strategy"。

### 备服务器

**hot\_standby**

**参数说明：**设置是否允许备机在恢复过程中连接和查询。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 如果此参数设置为on，wal\_level必须设置为hot\_standby，否则将导致数据库无法启动。
* 在双机环境中，因为会对双机其他一些功能产生影响，hot\_standby参数不能设置成off。
* 如果hot\_standby参数曾经被关闭，且wal\_level参数曾被设置低于hot\_standby等级，那么，再次打开hot\_standby参数之前，为了确保主备环境下备机上待回放的日志都可以支持备机查询功能，需要进行如下操作：
* 1、将主、备的wal\_level参数调整到hot\_standby等级或以上，并重启实例生效。
* 2、在主机上执行checkpoint操作，并通过查询pg\_stat\_get\_wal\_senders()系统函数，确认各个备机的receiver\_replay\_location追上主机当前的sender\_flush\_location，保证wal\_level的调整同步到备机并生效，且备机不需要再回放之前低等级的日志。
* 3、备的hot\_standby参数打开（设为on），并重启实例生效。

**取值范围：**布尔型

* on表示允许备机在恢复过程中连接和查询。
* off表示不允许备机在恢复过程中连接和查询。

**默认值：**on

**max\_standby\_archive\_delay**

**参数说明：**当开启双机热备模式时，如果备机正处理归档WAL日志数据，这时进行查询就会产生冲突，此参数就是设置备机取消查询之前所等待的时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
-1表示允许备机一直等待冲突的查询完成。

**取值范围：**整型，范围：-1~INT\_MAX，单位为毫秒。

**默认值：**3s（即3000ms）

**max\_standby\_streaming\_delay**

**参数说明：**当开启双机热备模式时，如果备机正通过流复制接收WAL日志数据，这时进行查询就会产生冲突，这个参数就是设置备机取消查询之前所等待的时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：** -1表示允许备机一直等待冲突的查询完成。

**取值范围：**整型（毫秒），范围：-1~INT\_MAX。

**默认值：**3s（即3000ms）

**wal\_receiver\_status\_interval**

**参数说明：**设置WAL日志接收进程的状态通知给主机的最大时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，范围：0 ~ INT\_MAX，单位为秒。

**默认值：**5s（即5000ms）

fig: **须知：**   
当该参数设置为0时，表示关闭备机向主机反馈日志接收位置等信息，可能会导致主机事务提交阻塞、switchover操作失败等异常现象。正常业务场景，不建议将该参数设置为0。

**hot\_standby\_feedback**

**参数说明：**设置是否允许将备机上执行查询的结果反馈给主机，这可以避免查询冲突。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许将备机上执行查询的最小事务号反馈给主机。
* off表示不允许将备机上执行查询的最小事务号反馈给主机。

**默认值：**off

fig: **须知：**   
当该参数为on时，主机的旧版本数据的清理会受限于备机正在读的事务，即主机只允许清理小于备机反馈回来的事务所作的更改。  
所以，若该参数开启时，会影响主机的性能。

**wal\_receiver\_timeout**

**参数说明：**设置从主机接收数据的最大等待时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ INT\_MAX，单位为毫秒。

**默认值：**6s（即6000ms）

**wal\_receiver\_connect\_timeout**

**参数说明：**设置连接主机的最大等待超时时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ INT\_MAX / 1000，单位为秒。

**默认值：**2s

**wal\_receiver\_connect\_retries**

**参数说明：**设置连接主机的最大尝试次数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~ INT\_MAX，单位为毫秒。

**默认值：**1

**wal\_receiver\_buffer\_size**

**参数说明：**备机与从备接收Xlog存放到内存缓冲区的大小，目前默认不支持主备从部署模式。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，4096~1047552，单位为KB。

**默认值：**64MB（即65536KB）

**primary\_slotname**

**参数说明：**设置备机对应主机的slot name，用于主备校验，与wal日志删除机制。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符型

**默认值：**空字符串

**max\_logical\_replication\_workers**

**参数说明**：订阅端apply worker线程的最大数量。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~262143

**默认值**：4

**max\_sync\_workers\_per\_subscription**

**参数说明**：订阅端每个订阅的tablesync worker线程的最大数量。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~262143

**默认值**：2

## 内存表

介绍内存表的配置参数。

**enable\_codegen\_mot**

**参数说明**：设置是否启用原生LLVM Lite执行简单查询。如果当前平台上不支持原生LLVM，那么将使用伪LLVM。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔值

**默认值**：true

**force\_pseudo\_codegen\_mot**

**参数说明**：设置是否强制伪LLVM Lite执行简单查询，即使当前平台上支持原生LLVM。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔值

**默认值**：true

fig: **说明：**   
即使将force\_pseudo\_codegen\_mot设置为true，而当前平台不支持原生LLVM，那么仍然会使用伪LLVM。

**enable\_codegen\_mot\_print**

**参数说明**：设置是否打印生成函数的IR字节码（如果使用伪 LLVM，则打印伪IR字节码）。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔值

**默认值**：true

**codegen\_mot\_limit**

**参数说明**：设置全局缓存计划源的数量限制以及每个会话的克隆计划。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：uint32

**默认值**：100

**mot\_allow\_index\_on\_nullable\_column**

**参数说明**：设置是否允许在内存表nullable列上创建索引。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔值

**默认值**：true

**mot\_config\_file**

**参数说明**：指定MOT的主配置文件。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

**默认值**：NULL

## 查询规划

介绍查询优化器方法配置、开销常量、规划算法以及一些配置参数。

fig: **说明：**   
优化器中涉及的两个参数：

* INT\_MAX数据类型INT的最大值，其值为2147483647。
* DBL\_MAX数据类型FLOAT的最大值。

### 优化器方法配置

这些配置参数提供了影响查询优化器选择查询规划的原始方法。如果优化器为特定的查询选择的缺省规划并不是最优的，可以通过使用这些配置参数强制优化器选择一个不同的规划来临时解决这个问题。更好的方法包括调节优化器开销常量、手动运行ANALYZE、增加配置参数default\_statistics\_target的值、增加使用ALTER TABLE SET STATISTICS为指定列增加收集的统计信息。

**enable\_bitmapscan**

**参数说明**：控制优化器对位图扫描规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_bitmapscan**

**参数说明**：控制优化器对位图扫描规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**force\_bitmapand**

**参数说明**：控制优化器强制使用bitmapand规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**enable\_hashagg**

**参数说明**：控制优化器对Hash聚集规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_hashjoin**

**参数说明**：控制优化器对Hash连接规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_indexscan**

**参数说明**：控制优化器对索引扫描规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_indexonlyscan**

**参数说明**：控制优化器对仅索引扫描规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_material**

**参数说明**：控制优化器对实体化的使用。消除整个实体化是不可能的，但是可以关闭这个变量以防止优化器插入实体节点。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_mergejoin**

**参数说明**：控制优化器对融合连接规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_nestloop**

**参数说明**：控制优化器对内表全表扫描嵌套循环连接规划类型的使用。完全消除嵌套循环连接是不可能的，但是关闭这个变量就会让优化器在存在其他方法的时候优先选择其他方法。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_index\_nestloop**

**参数说明**：控制优化器对内表参数化索引扫描嵌套循环连接规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_seqscan**

**参数说明**：控制优化器对顺序扫描规划类型的使用。完全消除顺序扫描是不可能的，但是关闭这个变量会让优化器在存在其他方法的时候优先选择其他方法。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_sort**

**参数说明**：控制优化器使用的排序步骤。完全消除明确的排序是不可能的，但是关闭这个变量可以让优化器在存在其他方法的时候优先选择其他方法。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_tidscan**

**参数说明**：控制优化器对TID扫描规划类型的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_kill\_query**

**参数说明**：CASCADE模式删除用户时，会删除此用户拥有的所有对象。此参数标识是否允许在删除用户的时候，取消锁定此用户所属对象的query。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示允许取消锁定。
* off表示不允许取消锁定。

**默认值**：off

**enforce\_a\_behavior**

**参数说明：**控制正则表达式的规则匹配模式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示正则表达式采用A格式的匹配规则。
* off表示正则表达式采用POSIX格式的匹配规则。

**默认值：**on

**max\_recursive\_times**

**参数说明**：控制with recursive的最大迭代次数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX。

**默认值**：1000

**enable\_vector\_engine**

**参数说明**：控制优化器对向量化执行引擎的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_change\_hjcost**

**参数说明**：控制优化器在Hash Join代价估算路径选择时，是否使用将内表运行时代价排除在Hash Join节点运行时代价外的估算方式。如果使用，则有利于选择条数少，但运行代价大的表做内表。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**enable\_absolute\_tablespace**

**参数说明：**控制表空间是否可以使用绝对路径。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示可以使用绝对路径。
* off表示不可以使用绝对路径。

**默认值：**on

**enable\_valuepartition\_pruning**

**参数说明：**控制是否对DFS分区表进行静态/动态优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示对DFS分区表进行静态/动态优化。
* off表示不对DFS分区表进行静态/动态优化。

**默认值：**on

**expected\_computing\_nodegroup**

**参数说明**：标识选定的计算Node Group模式或目标计算Node Group。Node Group目前为内部用机制，用户无需设置。

共4种计算Node Group模式，用于关联操作和聚集操作时选定计算Node Group。在每一种模式中，优化器有针对性地选定几个候选计算Node Group，然后根据代价，从中为当前算子挑选最佳计算Node Group。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：字符串

* optimal：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前用户具有COMPUTE权限的所有Node Group包含的数据库节点构成的Node Group
* query：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前查询涉及的所有基表所在Node Group包含的数据库节点构成的Node Group
* Node Group名（enable\_nodegroup\_debug被设置为off）：候选计算Node Group列表包含算子操作对象所在的Node Group和该指定的Node Group
* Node Group名（enable\_nodegroup\_debug被设置为on）：候选计算Node Group为指定的Node Group

**默认值**：空

**enable\_nodegroup\_debug**

**参数说明**：控制优化器在多Node Group环境下，是否使用强制弹性计算。Node Group目前为内部用机制，用户无需设置。

该参数只在expected\_computing\_nodegroup被设置为具体Node Group时生效。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：布尔型

**默认值**：空

**qrw\_inlist2join\_optmode**

**参数说明**：控制是否使用inlist-to-join查询重写。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

* disable：关闭inlist2join查询重写。
* cost\_base：基于代价的inlist2join查询重写。
* rule\_base：基于规则的inlist2join查询重写，即强制使用inlist2join查询重写。
* 任意正整数：inlist2join查询重写阈值，即list内元素个数大于该阈值，进行inlist2join查询重写。

**默认值**：cost\_base

**skew\_option**

**参数说明**：控制是否使用优化策略。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

* off：关闭策略。
* normal：采用激进策略。对于不确定是否出现倾斜的场景，认为存在倾斜，并进行相应优化。
* lazy：采用保守策略。对于不确定是否出现倾斜场景，认为不存在倾斜，不进行优化。

**默认值**：normal

**default\_limit\_rows**

**参数说明**：设置生成genericplan的缺省limit估算行数。此参数设置为正数时表示直接将设置的值作为估算limit的行数，设置为负数时表示使用百分比的形式设置默认的估算值，负数转换为默认百分比，即-5代表5%。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，-100～DBL\_MAX。

**默认值**：-10

**check\_implicit\_conversions**

**参数说明**：控制是否对查询中有隐式类型转换的索引列是否会生成候选索引路径进行检查。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示对查询中有隐式类型转换的索引列是否会生成候选索引路径进行检查。
* off表示不进行相关检查。

**默认值**：off

**cost\_weight\_index**

**参数说明**：设置index\_scan的代价权重。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，1e-10~1e+10。

**默认值**：1

**try\_vector\_engine\_strategy**

**参数说明**：设置行存表走向量化执行引擎的策略。通过设置该参数，可以使包含行存表的查询可以转换为向量化的执行计划执行计算，从而提升类AP场景的复杂查询的执行性能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：枚举型

* off，为默认取值，表示关闭本功能，即行存表不会转换为向量的执行计划执行。
* force，表示只要查询中不包含向量化引擎不支持的类型或者表达式，则不论查询的基表为行存表、列存表，还是行列混合存储的，强制将查询转换为向量化的执行计划执行计算。在这种情况下，针对不同的查询场景可能出现性能下降。
* optimal，表示在force的基础上，由优化器根据查询的复杂度进行选择是否将查询语句转换为向量化的执行计划，尽可能避免转换为向量化的执行计划后出现性能下降。

**默认值**：off

### 优化器开销常量

介绍优化器开销常量。这里描述的开销可以按照任意标准度量。只关心其相对值，因此以相同的系数缩放它们将不会对优化器的选择产生任何影响。缺省时，它们以抓取顺序页的开销为基本单位。也就是说将seq\_page\_cost设为1.0，同时其他开销参数以它为基准设置。也可以使用其他基准，比如以毫秒计的实际执行时间。

**seq\_page\_cost**

**参数说明**：设置优化器计算一次顺序磁盘抓取页面的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：1

**random\_page\_cost**

**参数说明**：设置优化器计算一次非顺序抓取磁盘页面的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
虽然服务器允许将random\_page\_cost设置的比seq\_page\_cost小，但是物理上实际不受影响。如果所有数据库都位于随机访问内存中时，两者设置为相等很合理。因为在此种情况下，非顺序抓取页并没有副作用。同样，在缓冲率很高的数据库上，应该相对于CPU参数同时降低这两个值，因为获取内存中的页要比通常情况下开销小很多。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：4

fig: **说明：**

* 对于特别表空间中的表和索引，可以通过设置同名的表空间的参数来覆盖这个值。
* 相对于seq\_page\_cost，减少这个值将导致系统更倾向于使用索引扫描，而增加这个值使得索引扫描开销比较高。可以通过同时增加或减少这两个值来调整磁盘I/O相对于CPU的开销。

**cpu\_tuple\_cost**

**参数说明**：设置优化器计算在一次查询中处理每一行数据的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：0.01

**cpu\_index\_tuple\_cost**

**参数说明**：设置优化器计算在一次索引扫描中处理每条索引的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：0.005

**cpu\_operator\_cost**

**参数说明**：设置优化器计算一次查询中执行一个操作符或函数的开销。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：0.0025

**effective\_cache\_size**

**参数说明**：设置优化器在一次单一的查询中可用的磁盘缓冲区的有效大小。

设置这个参数，还要考虑Vastbase的共享缓冲区以及内核的磁盘缓冲区。另外，还要考虑预计的在不同表之间的并发查询数目，因为它们将共享可用的空间。

这个参数对Vastbase分配的共享内存大小没有影响，它也不会使用内核磁盘缓冲，它只用于估算。数值是用磁盘页来计算的，通常每个页面是8192字节。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1～INT\_MAX，单位为8KB。

比默认值高的数值可能会导致使用索引扫描，更低的数值可能会导致选择顺序扫描。

**默认值**：128MB

**allocate\_mem\_cost**

**参数说明**：设置优化器计算Hash Join创建Hash表开辟内存空间所需的开销，供Hash join估算不准时调优使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～DBL\_MAX。

**默认值**：0

### 基因查询优化器

介绍基因查询优化器相关的参数。基因查询优化器（GEQO）是一种启发式的查询规划算法。这个算法减少了对复杂查询规划的时间，而且生成规划的开销有时也小于正常的详尽的查询算法。

**geqo**

**参数说明**：控制基因查询优化的使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
通常情况下在执行过程中不要关闭，geqo\_threshold变量提供了更精细的控制GEQO的方法。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**geqo\_threshold**

**参数说明**：如果执行语句的数量超过设计的FROM的项数，则会使用基因查询优化来执行查询。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 对于简单的查询，通常用详尽搜索方法，当涉及多个表的查询的时候，用GEQO可以更好的管理查询。
* 一个FULL OUTER JOIN构造仅作为一个FROM项。

**取值范围**：整型，2～INT\_MAX。

**默认值**：12

**geqo\_effort**

**参数说明**：控制GEQO在规划时间和规划质量之间的平衡。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
geqo\_effort实际上并没有直接做任何事情，只是用于计算其他影响GEQO的变量的默认值。如果愿意，可以手工设置其他参数。

**取值范围**：整型，1～10。

fig: **须知：**   
比默认值大的数值增加了查询规划的时间，但是也增加了选中有效查询的几率。

**默认值**：5

**geqo\_pool\_size**

**参数说明**：控制GEQO使用池的大小，也就是基因全体中的个体数量。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX。

fig: **须知：**   
至少是2，且有用的值一般在100到1000之间。设置为0，表示使用系统自适应方式，Vastbase会基于geqo\_effort和表的个数选取合适的值。

**默认值**：0

**geqo\_generations**

**参数说明**：控制GEQO使用的算法的迭代次数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX。

fig: **须知：**   
必须至少是1，且有用的值介于100和1000之间。如果设置为0，则基于geqo\_pool\_size选取合适的值。

**默认值**：0

**geqo\_selection\_bias**

**参数说明**：控制GEQO的选择性偏好，即就是一个种群中的选择性压力。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，1.5～2.0。

**默认值**：2

**geqo\_seed**

**参数说明**：控制GEQO使用的随机数生产器的初始化值，用来从顺序连接在一起的查询空间中查找随机路径。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0.0～1.0。

fig: **须知：**   
不同的值会改变搜索的连接路径，从而影响了所找路径的优劣。

**默认值**：0

### 其他优化器选项

**explain\_dna\_file**

**参数说明：** 指定explain\_perf\_mode为run，导出的csv信息的目标文件。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
这个参数的取值必须是绝对路径加上.csv格式的文件名。

**取值范围：** 字符串

**默认值：** 空

**explain\_perf\_mode**

**参数说明**：此参数用来指定explain的显示格式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：normal、pretty、summary、run

* normal：代表使用默认的打印格式。
* pretty：代表使用Vastbase改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。
* summary：代表是在pretty的基础上增加了对打印信息的分析。
* run：代表在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

**默认值**：normal（当前版本参数取值仅normal生效，若设置为非normal，显示格式依然为normal）

**analysis\_options**

**参数说明：**通过开启对应选项中所对应的功能选项使用相应的定位功能，包括数据校验、性能统计等，参见取值范围中的选项说明。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。设置时，选择开启或者关闭的选项请使用'on()'或'off()'包括，未被显示指定的功能选项会维持原来的值，参考格式：'on(option1, option2, ...)'。

**取值范围：**字符串

* LLVM\_COMPILE表示在explain performance显示界面中显示每个线程的codegen编译时间。
* HASH\_CONFLICT表示在数据库节点进程的pg\_log目录中的log日志中显示hash表的统计信息，包括hash表大小、hash链长、hash冲突情况。
* STREAM\_DATA\_CHECK表示对网络传输前后的数据进行CRC校验。

fig: **说明：**   
pretty模式当前版本不支持。

**默认值：**ALL,on(),off(LLVM\_COMPILE、HASH\_CONFLICT、STREAM\_DATA\_CHECK)，不开启任何定位功能。

**cost\_param**

**参数说明**：该参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

当cost\_param & 1不为0，表示对于求不等值连接选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确。目前，已弃用cost\_param & 1 不为0时的路径，默认选择更优的估算公式；

当cost\_param & 2不为0，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确；

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX

**默认值**：0

**enable\_partitionwise**

**参数说明**：分区表连接操作是否选择智能算法。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示选择智能算法。
* off表示不选择智能算法。

**默认值**：off

**rewrite\_rule**

**参数说明**：标识开启的可选查询重写规则。有部分查询重写规则是可选的，开启它们并不能总是对查询效率有提升效果。在特定的客户场景中，通过此GUC参数对查询重写规则进行设置，使得查询效率最优。

此参数可以控制查询重写规则的组合，比如有多个重写规则：rule1、rule2、rule3、rule4。可以设置：

set rewrite\_rule=rule1; --启用查询重写规则rule1  
set rewrite\_rule=rule2,rule3; --启用查询重写规则rule2和rule3  
set rewrite\_rule=none; --关闭所有可选查询重写规则

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

* none：不使用任何可选查询重写规则。
* lazyagg：使用Lazy Agg查询重写规则（消除子查询中的聚集运算）。
* magicset : 使用Magic Set查询重写规则（从主查询中下推条件到子查询）。
* partialpush：使用Partial Push查询重写规则。
* uniquecheck：使用Unique Check查询重写规则（提升目标列中无agg的子查询语句，在执行时检查返回行数是否为1行）。
* disablerep：使用Disable Replicate查询重写规则。
* intargetlist：使用In Target List查询重写规则（提升目标列中的子查询）。
* predpushnormal：使用Predicate Push查询重写规则（下推谓词条件到子查询中）。
* predpushforce：使用Predicate Push查询重写规则（下推谓词条件到子查询中，尽可能的利用索引加速）。
* predpush：在predpushnormal和predpushforce中根据代价选择最优计划。

**默认值**：magicset

**enable\_pbe\_optimization**

**参数说明**：设置优化器是否对以PBE（Parse Bind Execute）形式执行的语句进行查询计划的优化。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型。

* on表示优化器将优化PBE语句的查询计划。
* off表示不使用优化。

**默认值**：on

**enable\_global\_plancache**

**参数说明**：设置是否对PBE查询的执行计划进行缓存共享，开启该功能可以节省高并发下数据库节点的内存使用。

在打开enable\_global\_plancache的情况下，为保证GPC生效，默认local\_syscache\_threshold不小于16MB。即如当前local\_syscache\_threshold小于16MB，则设置为16MB，如大于16MB，则不改变。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型。

* on表示对PBE查询的执行计划进行缓存共享。
* off表示不共享。

**默认值**：off

fig: **说明：**   
enable\_global\_plancache不支持与vbplsql\_check同时设置为on，如果同事设置为on则出现报错提示：此参数不能和vbplsql\_check同时设置为on。

**gpc\_clean\_timeout\***\*

**参数说明**：开启enable\_global\_plancache的情况下，如果共享计划列表里的计划超过gpc\_clean\_timeout的时间没有被使用，则会被清理掉。本参数用于控制没有使用的共享计划的保留时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，300～86400，单位为秒

**默认值**：1800，即30min

**enable\_global\_stats**

**参数说明：**标识当前统计信息模式，区别采用全局统计信息收集模式还是单节点统计信息收集模式，默认创建为采用全局统计信息模式。当关闭该参数时，则默认收集Vastbase第一个节点的统计信息，此时可能会影响生成查询计划的质量，但信息收集性能较优，建议客户谨慎考虑。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示全局统计信息。
* off/false表示数据库节点统计信息。

**默认值：**on

**enable\_opfusion**

**参数说明**：控制是否对简单增删改查进行优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

简单查询限制如下：

* 只支持indexscan和indexonlyscan，且全部WHERE语句的过滤条件都在索引上。
* 只支持单表增删改查，不支持join、using。
* 只支持行存表，不支持分区表，表不支持有触发器。
* 不支持active sql、QPS等信息统计特性。
* 不支持正在扩容和缩容的表。
* 不支持查询或者修改系统列。
* 只支持简单SELECT语句，例如：
* SELECT c3 FROM t1 WHERE c1 = ? and c2 =10;
* 仅可以查询目标表的列，c1和c2列为索引列，后边可以是常量或者参数，可以使用 for update。
* 只支持简单INSERT语句，例如：
* INSERT INTO t1 VALUES (?,10,?);
* 仅支持一个VALUES，VALUES里面的类型可以是常量和参数，不支持returning。
* 只支持简单DELETE语句，例如：
* DELETE FROM t1 WHERE c1 = ? and c2 = 10;
* c1和c2列为索引列，后边可以是常量或者参数。
* 只支持简单UPDATE语句，例如：
* UPDATE t1 SET c3 = c3+? WHERE c1 = ? and c2 = 10;
* c3列修改的值可以是常量和参数，也可以是一个简单的表达式，c1和c2列为索引列，后边可以是常量或者参数。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：on

**enable\_partition\_opfusion\***\*

**参数说明**：在enable\_opfusion参数打开的状态下，如果开启该参数，可以对分区表的简单查询进行查询优化，提升SQL执行性能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**sql\_beta\_feature**

**参数说明**：标识开启的可选SQL引擎Beta特性，其中包括对行数估算、查询等价估算等优化。

开启它们可以对特定的场景进行优化，但也可能会导致部分没有被测试覆盖的场景发生性能劣化。在特定的客户场景中，通过此GUC参数对查询重写规则进行设置，使得查询效率最优。

此参数可以控制SQL引擎Beta特性的组合，比如有多个Beta特性：feature1、feature2、feature3、feature4。可以设置：

--启用SQL引擎Beta特性feature1。  
set sql\_beta\_feature=feature1;  
  
--启用SQL引擎Beta特性feature2和feature3。  
set sql\_beta\_feature=feature2,feature3;  
  
--关闭所有可选SQL引擎Beta特性。  
set sql\_beta\_feature=none;

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

* none：不使用任何Beta优化器特性。
* sel\_semi\_poisson：使用泊松分布对等值的半连接和反连接选择率进行校准。
* sel\_expr\_instr: 使用字符串匹配的行数估算方法对instr(col, 'const') > 0、= 0、 = 1进行更准确的估算。
* param\_path\_gen：生成更多可能的参数化路径。
* rand\_cost\_opt：对小数据量表的随机读取代价进行优化。
* param\_path\_opt：利用表的膨胀系数优化索引analyze信息。
* page\_est\_opt：优化对非列存表索引analyze信息的relpages估算。
* no\_unique\_index\_first：关闭主键索引扫描路径优先的优化。
* join\_sel\_with\_cast\_func：估算join行数的时候支持类型转换函数。
* canonical\_pathkey：正则化pathkey生成置后（pathkey是指标记数据有序性键值的集合）。
* index\_cost\_with\_leaf\_pages\_only：估算索引代价时考虑索引叶子结点。
* partition\_opfusion：开启分区表优化。
* a\_style\_coerce：开启Decode类型转换规则兼容O，详见对于case，在ORA兼容模式下的处理，请参考：SQL语法参考->类型转换->UNION CASE和相关构造章节。
* partition\_fdw\_on：支持基于分区表创建postgres foreign table下的相关SQL。
* predpush\_same\_level：开启predpush hint控制同层参数化路径的功能。

**默认值**：none

**ngram\_gram\_size**

**参数说明：**ngram解析器分词的长度。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1～4

**默认值：**2

**ngram\_grapsymbol\_ignore**

**参数说明：**ngram解析器是否忽略图形化字符。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示忽略图形化字符。
* off表示不忽略图形化字符。

**默认值：**off

**ngram\_punctuation\_ignore**

**参数说明：**ngram解析器是否忽略标点符号。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示忽略标点符号。
* off表示不忽略标点符号。

**默认值：**on

**default\_statistics\_target**

**参数说明**：为没有用ALTER TABLE SET STATISTICS设置字段目标的表设置缺省统计目标。此参数设置为正数时，代表统计信息的样本数量；设置为负数时，代表使用百分比的形式设置统计目标，负数转换为对应的百分比，即-5代表5%。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，-100～10000。

fig: **须知：**

* 比默认值大的正数数值增加了ANALYZE所需的时间，但是可能会改善优化器的估计质量。
* 调整此参数可能存在性能劣化的风险，如果某个查询劣化，可以考虑：  
   1、恢复默认的统计信息。  
   2、使用plan hint来调整到之前的查询计划。（详细参见：《[VASTDATA]Vastbase G100 V2.2管理员指南->性能调优->SQL调优指南->使用Plan Hint进行调优》章节。
* 当此guc参数设置为负数时，如果计算的采样样本数大于等于总数据量的2%，且用户表的数据量小于1600000时，ANALYZE所需时间相比guc参数为默认值的时间会有所增加。
* 当此guc参数设置为负数时，则autoanalyze不生效。

**默认值**：100

**constraint\_exclusion**

**参数说明**：控制查询优化器使用表约束查询的优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：枚举类型

* on表示检查所有表的约束。
* off表示不检查约束。
* partition表示只检查继承的子表和UNION ALL子查询。

fig: **须知：**   
当constraint\_exclusion为on，优化器用查询条件和表的CHECK约束比较，并且在查询条件和约束冲突的时候忽略对表的扫描。

**默认值**：partition

fig: **说明：**   
目前， constraint\_exclusion缺省被打开，通常用来实现表分区。如果所有的表都打开它，对于简单的查询强加了额外的规划，并且对简单查询没有益处。如果不用分区表，可以关掉它。

**cursor\_tuple\_fraction**

**参数说明**：优化器估计游标获取行数在总行数中的占比。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0.0～1.0。

fig: **须知：**   
比默认值小的值与使用“fast start”为游标规划的值相偏离，从而使得前几行恢复的很快而抓取全部的行需要很长的时间。比默认值大的值加大了总的估计的时间。在最大的值1.0处，像正常的查询一样规划游标，只考虑总的估计时间和传送第一行的时间。

**默认值**：0.1

**from\_collapse\_limit**

**参数说明**：根据生成的FROM列表的项数来判断优化器是否将把子查询合并到上层查询，如果FROM列表项个数小于等于该参数值，优化器会将子查询合并到上层查询。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1～INT\_MAX。

fig: **须知：**   
比默认值小的数值将降低规划时间，但是可能生成差的执行计划。

**默认值**：8

**join\_collapse\_limit**

**参数说明**：根据得出的列表项数来判断优化器是否执行把除FULL JOINS之外的JOIN构造重写到FROM列表中。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1～INT\_MAX。

fig: **须知：**

* 设置为1会避免任何JOIN重排。这样就使得查询中指定的连接顺序就是实际的连接顺序。查询优化器并不是总能选取最优的连接顺序，高级用户可以选择暂时把这个变量设置为1，然后指定它们需要的连接顺序。
* 比默认值小的数值减少规划时间但也降低了执行计划的质量。

**默认值**：8

**plan\_mode\_seed**

**参数说明**：该参数为调测参数，目前仅支持OPTIMIZE\_PLAN和RANDOM\_PLAN两种。其中：  
OPTIMIZE\_PLAN表示通过动态规划算法进行代价估算的最优plan，参数值设置为0；RANDOM\_PLAN表示随机生成的plan；如果设置guc参数值为-1，表示用户不指定随机数的种子标识符seed值，由优化器随机生成[1, 2147483647]范围整型值的随机数，并根据随机数生成随机的执行计划。  
如果设置guc参数值为[1, 2147483647]范围的整型值，表示指定的生成随机数的种子标识符seed，优化器需要根据seed值生成随机的执行计划。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，-1~ 2147483647

**默认值**：0

fig: **须知：**

* 当该参数设置为随机执行计划模式时，优化器会生成不同的随机执行计划，该执行计划可能不是最优计划。因此在随机计划模式下，会对查询性能产生影响，所以建议在升级、扩容、缩容等正常业务操作或运维过程中将该参数保持为默认值0。
* 当该参数不为0时，查询指定的plan hint不会生效。

**hashagg\_table\_size**

**参数说明**：用于设置执行HASH JOIN操作时HASH表的大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX/2。

**默认值**：0

**enable\_codegen**

**参数说明：**标识是否允许开启代码生成优化，目前代码生成使用的是LLVM优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许开启代码生成优化。
* off表示不允许开启代码生成优化。

fig: **须知：**   
目前LLVM优化仅支持向量化执行引擎特性和SQL on Hadoop特性，在其他场景下建议关闭此参数。

**默认值：**on

**codegen\_strategy**

**参数说明：**标识在表达式codegen化过程中所使用的代码生成优化策略。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

* partial表示当所计算表达式中即使包含部分未被codegen化的函数时，仍可借助表达式全codegen框架调用LLVM动态编译优化策略。
* pure表示当所计算表达式整体可被codegen化时，才考虑调用LLVM动态编译优化策略。

fig: **须知：**   
在开启代码生成优化会导致查询性能下降的场景下可以设置此参数为pure，其他场景下建议不改变此参数的默认值partial。

**默认值：**partial

**enable\_codegen\_print**

**参数说明：**标识是否允许在log日志中打印所生成的LLVM IR函数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许在log日志中打印IR函数。
* off表示不允许在log日志中打印IR函数。

**默认值：**off

**codegen\_cost\_threshold**

**参数说明：**由于LLVM编译生成最终的可执行机器码需要一定时间，因此只有当实际执行的代价大于编译生成机器码所需要的代码和优化后的执行代价之和时，利用代码生成才有收益。codegen\_cost\_threshold标识代价的阈值，当执行估算代价大于该代价时，使用LLVM优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 2147483647。

**默认值：**10000

**enable\_bloom\_filter**

**参数说明：**标识是否允许使用BloomFilter优化。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许使用BloomFilter优化。
* off表示不允许使用BloomFilter优化。

**默认值：**on

**enable\_extrapolation\_stats**

**参数说明：**标识对于日期类型是否允许基于历史统计信息使用推理估算的逻辑。使用该逻辑对于未及时收集统计信息的表可以增大估算准确的可能性，但也存在错误推理导致估算过大的可能性，需要对于日期类型数据定期插入的场景开启此开关。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许基于历史统计信息使用推理估算的逻辑。
* off表示不允许基于历史统计信息使用推理估算的逻辑。

**默认值：**off

**autoanalyze**

**参数说明：**标识是否允许在生成计划的时候，对于没有统计信息的表进行统计信息自动收集。对于外表和临时表，不支持autoanalyze，如果需要收集统计信息，用户需手动执行analyze操作。如果在auto analyze某个表的过程中数据库发生异常，当数据库正常运行之后再执行语句有可能仍提示需要收集此表的统计信息。此时需要用户对该表手动执行一次analyze操作，以同步统计信息数据。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许自动进行统计信息收集。
* off表示不允许自动进行统计信息收集。

**默认值：**off

**enable\_analyze\_check**

**参数说明：**标识是否允许在生成计划的时候，对于在pg\_class中显示reltuples和relpages均为0的表，检查该表是否曾进行过统计信息收集。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许检查。
* off表示不允许检查。

**默认值：**off

**enable\_sonic\_hashagg**

**参数说明：**标识是否依据规则约束使用基于面向列的hash表设计的Hash Agg算子。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在满足约束条件时使用基于面向列的hash表设计的Hash Agg算子。
* off表示不使用面向列的hash表设计的Hash Agg算子。

fig: **说明：**

* 在开启enable\_sonic\_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，查询对应的Hash Agg算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景enable\_codegen打开后获得较大性能提升，对应的算子查询性能可能会出现劣化。
* 开启enable\_sonic\_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Aggregation”，而未达到该约束条件时，算子名称将显示为“Hash Aggregation”，Explain详解请参见：SQL语法参考->SQL语法->EXPLAIN章节。

**默认值：**on

**enable\_sonic\_hashjoin**

**参数说明：**标识是否依据规则约束使用基于面向列的hash表设计的Hash Join算子。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在满足约束条件时使用基于面向列的hash表设计的Hash Join算子。
* off表示不使用面向列的hash表设计的Hash Join算子。

fig: **说明：**

* 当前开关仅适用于Inner Join的场景。
* 在开启enable\_sonic\_hashjoin，查询对应的Hash Inner算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景，对应的算子查询性能可能会出现劣化。
* 开启enable\_sonic\_hashjoin，且查询达到约束条件使用基于面向列的hash表设计的Hash Join算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Join”，而未达到该约束条件时，算子名称将显示为“Hash Join”，Explain详解请参见：SQL语法参考->SQL语法->EXPLAIN章节。

**默认值：**on

**enable\_sonic\_optspill**

**参数说明：**标识是否对面向列的hash表设计的Hash Join算子进行下盘文件数优化。该参数打开时，在Hash Join算子下盘文件较多的时候，下盘文件数不会显著增加。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示优化面向列的hash表设计的Hash Join算子的下盘文件数。
* off表示不优化面向列的hash表设计的Hash Join算子的下盘文件数。

**默认值：**on

**log\_parser\_stats**

**参数说明**：控制优化器输出parser模块的性能日志。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**log\_planner\_stats**

**参数说明**：控制优化器输出planner模块的性能日志。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**log\_executor\_stats**

**参数说明**：控制优化器输出executor模块的性能日志。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**log\_statement\_stats**

**参数说明**：控制优化器输出该语句的性能日志。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**plan\_cache\_mode**

**参数说明**：标识在prepare语句中，选择生成执行计划的策略。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：枚举类型

* auto表示按照默认的方式选择custom plan或者generic plan。
* force\_generic\_plan表示强制走generic plan。
* force\_custom\_plan表示强制走custom plan。

fig: **说明：**

* 此参数只对prepare语句生效，一般用在prepare语句中参数化字段存在比较严重的数据倾斜的场景下。
* custom plan是指对于preapre语句，在执行execute的时候，把execute语句中的参数嵌套到语句之后生成的计划。custom plan会根据execute语句中具体的参数生成计划，这种方案的优点是每次都按照具体的参数生成优选计划，执行性能比较好；缺点是每次执行前都需要重新生成计划，存在大量的重复的优化器开销。
* generic plan是指对于preapre语句生成计划，该计划策略会在执行execute语句的时候把参数bind到plan中，然后执行计划。这种方案的优点是每次执行可以省去重复的优化器开销；缺点是当bind参数字段上数据存在倾斜时该计划可能不是最优的，部分bind参数场景下执行性能较差。

**默认值**：auto

**enable\_hypo\_index\***\*

**参数说明**：控制优化器执行EXPLAIN命令时是否考虑虚拟索引。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用。
* off表示不使用。

**默认值**：off

**enable\_force\_vector\_engine**

**参数说明**：对于支持向量化的执行器算子，如果其子节点是非向量化的算子，通过设置此参数为on，强制生成向量化的执行计划。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示可以向量化的算子强制生成向量化。
* off表示由向量化算子优化器决定是否向量化。

**默认值**：off

**enable\_auto\_explain\***\*

**参数说明**：控制是否开启自动打印执行计划。该参数是用来定位慢存储过程或慢查询，只对当前连接的数据库主节点有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* true表示开启。
* false表示关闭。

**默认值**：false

**auto\_explain\_level**

**参数说明**：控制自动打印执行计划的日志等级。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：枚举型，LOG或NOTICE。NOTICE表示以提示知的形式打印出计划。

* LOG表示在日志中打印执行计划。
* NOTICE表示以提示知的形式打印出计划。

**默认值**：LOG

**query\_dop**

**参数说明**：用户自定义的查询并行度。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~64。打开固定SMP功能，系统会使用固定并行度。

fig: **说明：**   
在开启并行查询后，请保证系统CPU、内存、网络等资源充足，以达到最佳效果。

**默认值：**1

**enable\_startwith\_debug\***\*

**参数说明**：该参数为start with/connect by用于debug的参数，打开参数可以显示start with特性所有涉及的尾列相关信息。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启。
* off表示关闭。

**默认值**：off

## 错误报告和日志

### 记录日志的位置

**log\_destination**

**参数说明：**Vastbase支持多种方法记录服务器日志，log\_destination的取值为一个逗号分隔开的列表（如log\_destination="stderr,csvlog"）。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

有效值为stderr、csvlog、syslog、eventlog。

* 取值为stderr，表示日志打印到屏幕。
* 取值为csvlog，表示日志的输出格式为“逗号分隔值”即CSV（Comma Separated Value）格式。使用csvlog记录日志的前提是将logging\_collector设置为on，请参见：GUC参数说明->错误报告和日志->使用CSV格式写日志章节。
* 取值为syslog，表示通过操作系统的syslog记录日志。Vastbase使用syslog的LOCAL0～LOCAL7记录日志，请参见syslog\_facility。使用syslog记录日志需在操作系统后台服务配置文件中添加代码：

local0.\* /var/log/omm

**默认值：**stderr

**logging\_collector**

**参数说明：**控制开启后端日志收集进程logger进行日志收集。该进程捕获发送到stderr或csvlog的日志消息并写入日志文件。

这种记录日志的方法比将日志记录到syslog更加有效，因为某些类型的消息在syslog的输出中无法显示。例如动态链接库加载失败消息和脚本（例如archive\_command）产生的错误消息。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

fig: **须知：**   
将服务器日志发送到stderr时可以不使用logging\_collector参数，此时日志消息会被发送到服务器的stderr指向的空间。这种方法的缺点是日志回滚困难，只适用于较小的日志容量。

**取值范围：**布尔型

* on表示开启日志收集功能。
* off表示关闭日志收集功能。

**默认值：**on

**log\_directory**

**参数说明：**logging\_collector设置为on时，log\_directory决定存放服务器日志文件的目录。它可以是绝对路径或者是相对路径（相对于数据目录的路径）。log\_directory支持动态修改，可以通过vb\_guc reload实现。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 当配置文件中log\_directory的值为非法路径时，会导致Vastbase无法重新启动。
* 通过vb\_guc reload动态修改log\_directory时，当指定路径为合法路径时，日志输出到新的路径下。当指定路径为非法路径时，日志输出到上一次合法的日志输出路径下而不影响数据库正常运行。此时即使指定的log\_directory的值非法，也会写入到配置文件中。
* 在沙箱环境，路径中不可以包含/var/chroot，例如log的绝对路径是/var/chroot/var/lib/log/Ruby/pg\_log/cn\_log，则只需要设置为/var/lib/log/Ruby/pg\_log/cn\_log。

fig: **说明：**

* 合法路径：用户对此路径有读写权限。
* 非法路径：用户对此路径无读写权限。

**取值范围：**字符串

**默认值：**安装时指定。

**log\_filename**

**参数说明：**logging\_collector设置为on时，log\_filename决定服务器运行日志文件的名称。通常日志文件名是按照strftime模式生成，因此可以用系统时间定义日志文件名，用%转义字符实现。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 建议使用%转义字符定义日志文件名称，否则难以对日志文件进行有效的管理。
* 当log\_destination设为csvlog时，系统会生成附加了时间戳的日志文件名，文件格式为csv格式，例如“server\_log.1093827753.csv”。

**取值范围：**字符串

**默认值：**postgresql-%Y-%m-%d\_%H%M%S.log

**log\_file\_mode**

**参数说明：**logging\_collector设置为on时，log\_file\_mode设置服务器日志文件的权限。通常log\_file\_mode的取值是能够被chmod和umask系统调用接受的数字。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* 使用此选项前请设置log\_directory，将日志存储到数据目录之外的地方。
* 因日志文件可能含有敏感数据，故不能将其设为对外可读。

**取值范围：**整型，0000～0777（8进制计数，转化为十进制 0 ~ 511）。

fig: **说明：**

* 0600表示只允许服务器管理员读写日志文件。
* 0640表示允许管理员所在用户组成员只能读日志文件。

**默认值：**0600

**log\_truncate\_on\_rotation**

**参数说明：**logging\_collector设置为on时，log\_truncate\_on\_rotation设置日志消息的写入方式。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

示例如下：

假设日志需要保留7天，每天生成一个日志文件，日志文件名设置为server\_log.Mon、server\_log.Tue等。第二周的周二生成的日志消息会覆盖写入到server\_log.Tue。设置方法：将log\_filename设置为server\_log.%a ，log\_truncate\_on\_rotation设置为on，log\_rotation\_age设置为1440，即日志有效时间为1天。

**取值范围：**布尔型

* on表示Vastbase以覆盖写入的方式写服务器日志消息。
* off表示Vastbase将日志消息附加到同名的现有日志文件上。

**默认值：**off

**log\_rotation\_age**

**参数说明：**logging\_collector设置为on时，log\_rotation\_age决定创建一个新日志文件的时间间隔。当现在的时间减去上次创建一个服务器日志的时间超过了log\_rotation\_age的值时，将生成一个新的日志文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 35791394，单位为min。其中0表示关闭基于时间的新日志文件的创建。

**默认值：**1440(min)

**log\_rotation\_size**

**参数说明：**logging\_collector设置为on时，log\_rotation\_size决定服务器日志文件的最大容量。当日志消息的总量超过日志文件容量时，服务器将生成一个新的日志文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ INT\_MAX / 1024，单位为KB。

0表示关闭基于容量的新日志文件的创建。

建议该值大小设置级别至少为MB级，利于日志文件的及时划分。

**默认值：**10MB

**syslog\_facility**

**参数说明：**log\_destination设置为syslog时，syslog\_facility配置使用syslog记录日志的“设备”。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型，有效值有local0、local1、local2、local3、local4、local5、local6、local7。

**默认值：**local0

**syslog\_ident**

**参数说明：**log\_destination设置为syslog时，syslog\_ident设置在syslog日志中Vastbase日志消息的标识。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**postgres

**event\_source**

**参数说明：**该参数仅在windows环境下生效，Vastbase暂不支持。log\_destination设置为eventlog时，event\_source设置在日志中Vastbase日志消息的标识。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**PostgreSQL

### 记录日志的时间

**client\_min\_messages**

**参数说明：**控制发送到客户端的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，发送给客户端的消息就越少。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
当client\_min\_messages和log\_min\_messages取相同值时，其值所代表的级别不同。

**取值范围：**枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见：配置运行参数->重设参数章节中表1。在实际设置过程中，如果设置的级别大于error，为fatal或panic，系统会默认将级别转为error。

**默认值：**notice

**log\_min\_messages**

**参数说明：**控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
当client\_min\_messages和log\_min\_messages取相同值log时所代表的消息级别不同。  
部分日志信息的打印需要同时配置该参数与logging\_modules，即设置该参数打开后可能还需要设置logging\_modules打开对应模块的日志打印开关。

**取值范围：**枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见：配置运行参数->重设参数章节中表1。

**默认值：**warning

**log\_min\_error\_statement**

**参数说明：**控制在服务器日志中记录错误的SQL语句。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见：配置运行参数->重设参数章节中表1。

fig: **说明：**

* 设置为error，表示导致错误、日志消息、致命错误、panic的语句都将被记录。
* 设置为panic，表示关闭此特性。

**默认值：**error

**log\_min\_duration\_statement**

**参数说明：**当某条语句的持续时间大于或者等于特定的毫秒数时，log\_min\_duration\_statement参数用于控制记录每条完成语句的持续时间。

设置log\_min\_duration\_statement可以很方便地跟踪需要优化的查询语句。对于使用扩展查询协议的客户端，语法分析、绑定、执行每一步所花时间被独立记录。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
当此选项与log\_statement同时使用时，已经被log\_statement记录的语句文本不会被重复记录。在没有使用syslog情况下，推荐使用log\_line\_prefix记录PID或会话ID，方便将当前语句消息连接到最后的持续时间消息。

**取值范围：**整型，-1 ~ INT\_MAX，单位为毫秒。

* 设置为250，所有运行时间不短于250ms的SQL语句都会被记录。
* 设置为0，输出所有语句的持续时间。
* 设置为-1，关闭此功能。

**默认值：**5s

**backtrace\_min\_messages**

**参数说明：**控制当产生该设置参数级别相等或更高级别的信息时，会打印函数的堆栈信息到服务器日志文件中。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
该参数作为客户现场问题定位手段使用，且由于频繁的打印函数栈会对系统的开销及稳定性有一定的影响，因此如果需要进行问题定位时，建议避免将backtrace\_min\_messages的值设置为fatal及panic以外的级别。

**取值范围：**枚举类型

有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见下表表1：信息严重程度分类。

**默认值：**panic

[表1](#X9f3efdb30a4ee1f5a5d9c9f508c1cf1395cecd1)解释Vastbase中使用的消息安全级别。当日志输出到syslog或者eventlog（仅windows环境下，Vastbase版本不涉及该参数）时，Vastbase进行如表中的转换。

**表 1** 信息严重程度分类

| **信息严重程度类型** | **详细说明** | **系统日志** | **事件日志** |
| --- | --- | --- | --- |
| debug[1-5] | 报告详细调试信息。 | DEBUG | INFORMATION |
| log | 报告对数据库管理员有用的信息， 比如检查点操作统计信息。 | INFO | INFORMATION |
| info | 报告用户可能需求的信息， 比如在VACUUM VERBOSE过程中的信息。 | INFO | INFORMATION |
| notice | 报告可能对用户有帮助的信息， 比如， 长标识符的截断， 作为主键一部分创建的索引等。 | NOTICE | INFORMATION |
| warning | 报告警告信息， 比如在事务块范围之外的COMMIT。 | NOTICE | WARNING |
| error | 报告导致当前命令退出的错误。 | WARNING | ERROR |
| fatal | 报告导致当前会话终止的原因。 | ERR | ERROR |
| panic | 报告导致整个数据库被关闭的原因。 | CRIT | ERROR |

**plog\_merge\_age**

**参数说明：**该参数用于控制性能日志数据输出的周期。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
该参数以毫秒为单位，建议在使用过程中设置值为1000的整数倍，即设置值以秒为最小单位。该参数所控制的性能日志文件以prf为扩展名，文件放置在$GAUSSLOG/gs\_profile/<node\_name>目录下面，其中node\_name是由postgres.conf文件中的pgxc\_node\_name的值，不建议外部使用该参数。

**取值范围：**0~2147483647，单位为毫秒（ms）。

当设置为0时，当前会话不再输出性能日志数据。当设置为非0时，当前会话按照指定的时间周期进行输出性能日志数据。

该参数设置得越小，输出的日志数据越多，对性能的负面影响越大。

**默认值：**0

### 记录日志的内容

**debug\_print\_parse**

**参数说明：**用于控制打印解析树结果。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启打印结果的功能。
* off表示关闭打印结果的功能。

**默认值：**off

**debug\_print\_rewritten**

**参数说明：**用于控制打印查询重写结果。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启打印结果的功能。
* off表示关闭打印结果的功能。

**默认值：**off

**debug\_print\_plan**

**参数说明：**用于设置是否将查询的执行计划打印到日志中。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启打印结果的功能。
* off表示关闭打印结果的功能。

**默认值：**off

fig: **须知：**

* 只有当日志的级别为log及以上时，debug\_print\_parse、debug\_print\_rewritten和debug\_print\_plan的调试信息才会输出。当这些选项打开时，调试信息只会记录在服务器的日志中，而不会输出到客户端的日志中。通过设置GUC参数说明->错误报告和日志->记录日志的时间章节的client\_min\_messages和log\_min\_messages参数可以改变日志级别。
* 在打开debug\_print\_plan开关的情况下需尽量避免调用gs\_encrypt\_aes128及gs\_decrypt\_aes128函数，避免敏感参数信息在日志中泄露的风险。同时建议用户在打开debug\_print\_plan开关生成的日志中对gs\_encrypt\_aes128及gs\_decrypt\_aes128函数的参数信息进行过滤后再提供给外部维护人员定位，日志使用完成后请及时删除。

**debug\_pretty\_print**

**参数说明：**设置此选项对debug\_print\_parse、debug\_print\_rewritten和debug\_print\_plan产生的日志进行缩进，会生成易读但比设置为off时更长的输出格式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示进行缩进。
* off表示不进行缩进。

**默认值：**on

**log\_checkpoints**

**参数说明：**控制在服务器日志中记录检查点和重启点的信息。打开此参数时，服务器日志消息包含涉及检查点和重启点的统计量，其中包含需要写的缓存区的数量及写入所花费的时间等。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示打开此参数时，服务器日志消息包含涉及检查点和重启点的统计量。
* off表示关闭此参数时，服务器日志消息包含不涉及检查点和重启点的统计量。

**默认值：**off

**log\_connections**

**参数说明：**控制记录客户端的连接请求信息。

该参数属于BACKEND类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
有些客户端程序（例如vsql），在判断是否需要口令的时候会尝试连接两次，因此日志消息中重复的“connection receive”（收到连接请求）并不意味着一定是问题。

**取值范围：**布尔型

* on表示记录信息。
* off表示不记录信息。

**默认值：**off

**log\_disconnections**

**参数说明：**控制记录客户端结束连接信息。

该参数属于BACKEND类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示记录信息。
* off表示不记录信息。

**默认值：**off

**log\_duration**

**参数说明：**控制记录每个已完成SQL语句的执行时间。对使用扩展查询协议的客户端、会记录语法分析、绑定和执行每一步所花费的时间。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* 设置为off，该选项与GUC参数说明->错误报告和日志->记录日志的时间中log\_min\_duration\_statement参数的不同之处在于log\_min\_duration\_statement强制记录查询文本。
* 设置为on并且log\_min\_duration\_statement大于零，记录所有持续时间，但是仅记录超过阈值的语句。这可用于在高负载情况下搜集统计信息。

**默认值：**on

**log\_error\_verbosity**

**参数说明：**控制服务器日志中每条记录的消息写入的详细度。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

* terse代表输出不包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录。
* verbose代表输出包括SQLSTATE错误代码、源代码文件名、函数名及产生错误所在的行号。
* default代表输出包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录，不包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。

**默认值：**default

**log\_hostname**

**参数说明：**选项关闭状态下，连接消息日志只显示正在连接主机的IP地址。打开此选项同时可以记录主机名。由于解析主机名可能需要一定的时间，可能影响数据库的性能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示可以同时记录主机名。
* off表示不可以同时记录主机名。

**默认值：**on

**log\_line\_prefix**

**参数说明：**控制每条日志信息的前缀格式。日志前缀类似于printf风格的字符串，在日志的每行开头输出。用以%为开头的“转义字符”代替配置运行参数->重设参数章节中表1中的状态信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**表 1** 转义字符表

|  |  |
| --- | --- |
| **转义字符** | **效果** |
| %a | 应用程序名称。 |
| %u | 用户名。 |
| %d | 数据库名。 |
| %r | 远端主机名或者IP地址以及远端端口，在不启动log\_hostname时显示IP地址及远端端口。 |
| %h | 远端主机名或者IP地址，在不启动log\_hostname时只显示IP地址。 |
| %p | 线程ID。 |
| %t | 时间戳（没有毫秒）。 |
| %m | 带毫秒的时间戳。 |
| %n | 表示指定错误从哪个节点上报的。 |
| %i | 命令标签：会话当前执行的命令类型。 |
| %e | SQLSTATE错误码。 |
| %c | 会话ID，详见说明。 |
| %l | 每个会话或线程的日志编号，从1开始。 |
| %s | 进程启动时间。 |
| %v | 虚拟事务ID（backendID/ localXID）。 |
| %x | 事务ID（0表示没有分配事务ID）。 |
| %q | 不产生任何输出。如果当前线程是后端线程，忽略这个转义序列，继续处理后面的转义序列；如果当前线程不是后端线程，忽略这个转义序列和它后面的所有转义序列。 |
| %S | 会话ID。 |
| %% | 字符%。 |

fig: **说明：**   
 转义字符%c打印一个会话ID，由两个4字节的十六进制数组成，通过字符 “.”分开。这两个十六进制数分别表示进程的启动时间及进程编号，所以%c 也可以看作是保存打印这些名目的途径的空间。比如，从pg\_stat\_activity 中产生会话ID，可以用下面的查询：

SELECT to\_hex(EXTRACT(EPOCH FROM backend\_start)::integer) || '.' ||   
 to\_hex(pid)   
FROM pg\_stat\_activity;

* 当log\_line\_prefix设置为非空值时，请将其最后一个字符作为一个独立的段，以此来直观地与后续的日志进行区分，也可以使用一个标点符号。
* Syslog生成自己的时间戳及进程ID信息，所以当登录日志时，不需要包含这些转义字符。

**取值范围：**字符串

**默认值：**空

fig: **说明：**

%m %c %d %p %a %x %n %e表示在日志开头附加会话开始时间戳、会话ID、数据库名、线程ID、应用程序名、事务ID、报错节点、SQLSTATE错误码。

**log\_lock\_waits**

**参数说明：**当一个会话的等待获得一个锁的时间超过GUC参数说明->锁管理中deadlock\_timeout参数值时，此选项控制在数据库日志中记录此消息。这对于决定锁等待是否会产生一个坏的行为是非常有用的。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示记录此信息。
* off表示不记录此信息。

**默认值：**off

**log\_statement**

**参数说明：**控制记录SQL语句。对于使用扩展查询协议的客户端，记录接收到执行消息的事件和绑定参数的值（内置单引号要双写）。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

即使log\_statement设置为all，包含简单语法错误的语句也不会被记录，因为仅在完成基本的语法分析并确定了语句类型之后才记录日志。在使用扩展查询协议的情况下，在执行阶段之前（语法分析或规划阶段）同样不会记录。将log\_min\_error\_statement设为ERROR或更低才能记录这些语句。

**取值范围：**枚举类型

* none表示不记录语句。
* ddl表示记录所有的数据定义语句，比如CREATE、ALTER和DROP语句。
* mod表示记录所有DDL语句，还包括数据修改语句INSERT、UPDATE、DELETE、TRUNCATE和COPY FROM 。
* all表示记录所有语句，PREPARE、EXECUTE和EXPLAIN ANALYZE语句也同样被记录。

**默认值：**none

**log\_temp\_files**

**参数说明：**控制记录临时文件的删除信息。临时文件可以用来排序、哈希及临时查询结果。当一个临时文件被删除时，将会产生一条日志消息。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为-1，最大值为2147483647，单位KB。

* 正整数表示只记录比log\_temp\_files设定值大的临时文件的删除信息。
* 值0表示记录所有的临时文件的删除信息。
* 值-1表示不记录任何临时文件的删除信息。

**默认值：**-1

**log\_timezone**

**参数说明：**设置服务器写日志文件时使用的时区。与GUC参数说明->客户端连接缺省设置->区域和格式化中TimeZone参数不同，这个值是数据库范围的，针对所有连接到本数据库的会话生效。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，可查询视图PG\_TIMEZONE\_NAMES获得。

**默认值：**GMT

fig: **说明：**

vb\_initdb进行相应系统环境设置时会对默认值进行修改。

**logging\_module**

**参数说明：**用于设置或者显示模块日志在服务端的可输出性。该参数属于会话级参数，不建议通过gs\_guc工具来设置。

该参数属于USERSET类型参数，设置请参考：配置运行参数->重设参数章节中表1中对应设置的方法进行设置。

**取值范围：**字符串

**默认值：**所有模块日志在服务端是不输出的，可由SHOW logging\_module查看。为ALL,on(),off(DFS、GUC、ORC、SLRU、MEM\_CTL、AUTOVAC、CACHE、ADIO、SSL、TBLSPC、WLM、EXECUTOR、OPFUSION、VEC\_EXECUTOR、LLVM、OPT、OPT\_REWRITE、OPT\_JOIN、OPT\_AGG、OPT\_SUBPLAN、OPT\_SETOP、OPT\_SKEW、UDF、COOP\_ANALYZE、WLMCP、ACCELERATE、PLANHINT、SNAPSHOT、XACT、HANDLE、CLOG、EC、REMOTE、CN\_RETRY、PLSQL、TEXTSEARCH、SEQ、REDO、FUNCTION、PARSER、INSTR、INCRE\_CKPT、DBL\_WRT、RTO、HEARTBEAT)。

**设置方法**：首先，可以通过SHOW logging\_module来查看哪些模块是支持可控制的。例如，查询输出结果为：

Vastbase=# show logging\_module;  
logging\_module  
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
ALL,on(),off(DFS,GUC,ORC,SLRU,MEM\_CTL,AUTOVAC,CACHE,ADIO,SSL,TBLSPC,WLM,EXECUTOR,VEC\_EXECUTOR,LLVM,OPT,OPT\_REWRITE,OPT\_JOIN,OPT\_AGG,OPT\_SUBPLAN,OPT\_SETOP,OPT\_SKEW,UDF,COOP\_ANALYZE,WLMCP,ACCELERATE,T,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN\_RETRY,PLSQL,TEXTSEARCH,SEQ,REDO,FUNCTION,PARSER,INSTR,INCRE\_CKPT,DBL\_WRT,RTO,HEARTBEAT)  
(1 row)

支持可控制的模块使用大写来标识，特殊标识ALL用于对所有模块日志进行设置。可以使用on/off来控制模块日志的输出。设置SSL模块日志为可输出，使用如下命令：

Vastbase=# set logging\_module='on(SSL)';  
SET  
Vastbase=# show logging\_module; logging\_module   
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ALL,on(SSL),off(DFS,GUC,ORC,SLRU,MEM\_CTL,AUTOVAC,CACHE,ADIO,TBLSPC,WLM,EXECUTOR,VEC\_EXECUTOR,LLVM,OPT,OPT\_REWRITE,OPT\_JOIN,OPT\_AGG,OPT\_SUBPLAN,OPT\_SETOP,OPT\_SKEW,UDF,COOP\_ANALYZE,WLMCP,ACCELERATE,,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN\_RETRY,PLSQL,TEXTSEARCH,SEQ,REDO,FUNCTION,PARSER,INSTR,INCRE\_CKPT,DBL\_WRT,RTO,HEARTBEAT,COMM\_IPC,COMM\_PARAM)  
(1 row)

可以看到模块SSL的日志输出被打开。

ALL标识是相当于一个快捷操作，即对所有模块的日志可输出进行开启或关闭。

Vastbase=# set logging\_module='off(ALL)';  
SET  
Vastbase=# show logging\_module; logging\_module   
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ALL,on(),off(DFS,GUC,ORC,SLRU,MEM\_CTL,AUTOVAC,CACHE,ADIO,SSL,TBLSPC,WLM,EXECUTOR,VEC\_EXECUTOR,LLVM,OPT,OPT\_REWRITE,OPT\_JOIN,OPT\_AGG,OPT\_SUBPLAN,OPT\_SETOP,OPT\_SKEW,UDF,COOP\_ANALYZE,WLMCP,ACCELERATE,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN\_RETRY,PLSQL,TEXTSEARCH,SEQ,REDO,FUNCTION,PARSER,INSTR,INCRE\_CKPT,DBL\_WRT,RTO,HEARTBEAT)  
(1 row)  
  
Vastbase=# set logging\_module='on(ALL)';  
SET  
Vastbase=# show logging\_module; logging\_module   
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- ALL,on(DFS,GUC,ORC,SLRU,MEM\_CTL,AUTOVAC,CACHE,ADIO,SSL,TBLSPC,WLM,EXECUTOR,VEC\_EXECUTOR,LLVM,OPT,OPT\_REWRITE,OPT\_JOIN,OPT\_AGG,OPT\_SUBPLAN,OPT\_SETOP,OPT\_SKEW,UDF,COOP\_ANALYZE,WLMCP,ACCELERATE,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN\_RETRY,PLSQL,TEXTSEARCH,SEQ,REDO,FUNCTION,PARSER,INSTR,INCRE\_CKPT,DBL\_WRT,RTO,HEARTBEAT),off()  
(1 row)

**依赖关系**：该参数依赖于log\_min\_messages参数的设置。

**opfusion\_debug\_mode**

**参数说明：**用于调试简单查询是否进行查询优化。设置成log级别可以在数据库节点的执行计划中看到没有查询优化的具体原因。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

* off表示不打开该功能。
* log表示打开该功能，可以在数据库节点的执行计划中看到没有查询优化的具体原因。

fig: **须知：**

提供在log中显示语句没有查询优化的具体原因，需要将参数设置成log级别，log\_min\_messages设置成debug4级别，logging\_module设置'on(OPFUSION)'，注意log内容可能会比较多，尽可能在调优期间执行少量作业使用。

**默认值：**off

**enable\_debug\_vacuum**

**参数说明：**允许输出一些与VACUUM相关的日志，便于定位VACUUM相关问题。开发人员专用，不建议普通用户使用。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示开启此日志开关。
* off/false表示关闭此日志开关。

**默认值：**off

### 使用CSV格式写日志

**前提条件**

* GUC参数说明->错误报告和日志->记录日志的位置中log\_destination的值设置为csvlog。
* GUC参数说明->错误报告和日志->记录日志的位置中logging\_collector的值设置为on。

**csvlog定义**

以“逗号分隔值” 即CSV（Comma Separated Value）的形式发出日志。

以下是简单的用来存储CSV形式日志输出的表定义：

CREATE TABLE gaussdb\_log  
(  
log\_time timestamp(3) with time zone,  
node\_name text,  
user\_name text,  
database\_name text,  
process\_id bigint,  
connection\_from text,  
"session\_id" text,  
session\_line\_num bigint,  
command\_tag text,  
session\_start\_time timestamp with time zone,  
virtual\_transaction\_id text,  
transaction\_id bigint,  
query\_id bigint,  
module text,  
error\_severity text,  
sql\_state\_code text,  
message text,  
detail text,  
hint text,  
internal\_query text,  
internal\_query\_pos integer,  
context text,  
query text,  
query\_pos integer,  
location text,  
application\_name text  
);

详细说明请参见[表1](#X4b619bbd80c8fa32e3a10b2206ae767c1b0e2b1)。

**表 1** csvlog字段含义表

| **字段名** | **字段含义** | **字段名** | **字段含义** |
| --- | --- | --- | --- |
| log\_time | 毫秒级的时间戳 | module | 日志所属模块 |
| node\_name | 节点名称 | error\_severity | ERRORSTATE代码 |
| user\_name | 用户名 | sql\_state\_code | SQLSTATE代码 |
| database\_name | 数据库名 | message | 错误消息 |
| process\_id | 进程ID | detail | 详细错误消息 |
| connection\_from | 客户主机：端口号 | hint | 提示 |
| session\_id | 会话ID | internal\_query | 内部查询（查询那些导致错误的信息，如果有的话） |
| session\_line\_num | 每个会话的行数 | internal\_query\_pos | 内部查询指针 |
| command\_tag | 命令标签 | context | 环境 |
| session\_start\_time | 会话开始时间 | query | 错误发生位置的字符统计 |
| virtual\_transaction\_id | 常规事务 | query\_pos | 错误发生位置指针 |
| transaction\_id | 事务ID | location | 在openGauss源代码中错误的位置（如果GUC参数说明->错误报告和日志->记录日志的内容中log\_error\_verbosity参数的值设为verbose） |
| query\_id | 查询ID | application\_name | 应用名称 |

使用COPY FROM命令将日志文件导入这个表：

COPY gaussdb\_log FROM '/home/vastbase/data/vastbase/pg\_log/logfile.csv' WITH csv;

fig: **说明：**   
此处的日志名“logfile.csv”要换成实际生成的日志的名称。

**简化输入**

简化输入到CSV日志文件，可以通过如下操作：

* 设置GUC参数说明->错误报告和日志->记录日志的位置中log\_filename和log\_rotation\_age，为日志文件提供一个一致的、可预测的命名方案。通过日志文件名，预测一个独立的日志文件完成并进入准备导入状态的时间。
* 将GUC参数说明->错误报告和日志->记录日志的位置中log\_rotation\_size设为0来终止基于尺寸的日志回滚，因为基于尺寸的日志回滚让预测日志文件名变得非常的困难。
* 将GUC参数说明->错误报告和日志->记录日志的位置中log\_truncate\_on\_rotation设为on以便区分在同一日志文件中旧的日志数据和新的日志数据。

## 告警检测

在Vastbase运行的过程中，会对数据库中的错误场景进行检测，便于用户及早感知到Vastbase的错误。

**enable\_alarm**

**参数说明：**允许打开告警检测线程，检测数据库中可能的错误场景。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许打开告警检测线程。
* off表示不允许打开告警检测线程。

**默认值：**off

**connection\_alarm\_rate**

**参数说明：**允许和数据库连接的最大并发连接数的比率限制。数据库连接的最大并发连接数为max\_connections（参考：GUC参数说明->连接和认证->连接设置）\* connection\_alarm\_rate。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**浮点型，0.0~1.0

**默认值：**0.9

**alarm\_report\_interval**

**参数说明：**指定告警上报的时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位为秒。

**默认值：**10

**alarm\_component**

**参数说明：**在对告警做上报时，会进行告警抑制，即同一个实例的同一个告警项在alarm\_report\_interval（默认值为10s）内不做重复上报。在这种情况下设置用于处理告警内容的告警组件的位置。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**/opt/snas/bin/snas\_cm\_cmd

**table\_skewness\_warning\_threshold**

**参数说明**：设置用于表倾斜告警的阈值。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：浮点型，0～1

**默认值**：1

**table\_skewness\_warning\_rows**

**参数说明**：设置用于表倾斜告警的行数。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX

**默认值**：100000

## 运行时统计

### 查询和索引统计收集器

查询和索引统计收集器负责收集数据库系统运行中的统计数据，如在一个表和索引上进行了多少次插入与更新操作、磁盘块的数量和元组的数量、每个表上最近一次执行清理和分析操作的时间等。可以通过查询系统视图pg\_stats和pg\_statistic查看统计数据。下面的参数设置服务器范围内的统计收集特性。

**track\_activities**

**参数说明：**控制收集每个会话中当前正在执行命令的统计数据。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启收集功能。
* off表示关闭收集功能。

**默认值：**on

**track\_counts**

**参数说明：**控制收集数据库活动的统计数据。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启收集功能。
* off表示关闭收集功能。

**默认值：**on

fig: **说明：**   
在AutoVacuum自动清理进程中选择清理的数据库时，需要数据库的统计数据，故默认值设为on。

**track\_io\_timing**

**参数说明：**控制收集数据库I/O调用时序的统计数据。I/O时序统计数据可以在pg\_stat\_database中查询。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启收集功能，开启时，收集器会在重复地去查询当前时间的操作系统，这可能会引起某些平台的重大开销，故默认值设置为off。
* off表示关闭收集功能。

**默认值：**off

**track\_functions**

**参数说明：**控制收集函数的调用次数和调用耗时的统计数据。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
当SQL语言函数设置为调用查询的“内联”函数时，不管是否设置此选项，这些SQL语言函数无法被追踪到。

**取值范围：**枚举类型

* pl表示只追踪过程语言函数。
* all表示追踪SQL语言函数。
* none表示关闭函数追踪功能。

**默认值：**none

**track\_activity\_query\_size**

**参数说明：**设置用于跟踪每一个活动会话的当前正在执行命令的字节数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，100～102400

**默认值：**1024

**stats\_temp\_directory**

**参数说明：**设置存储临时统计数据的目录。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**   
将其设置为一个基于RAM的文件系统目录会减少实际的I/O开销并可以提升其性能。

**取值范围：**字符串

**默认值：**pg\_stat\_tmp

**track\_thread\_wait\_status\_interval**

**参数说明：**用来定期收集thread状态信息的时间间隔。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0～1天，单位为min。

**默认值：**30min

**enable\_save\_datachanged\_timestamp**

**参数说明：**确定是否收集insert/update/delete, exchange/truncate/drop partition操作对表数据改动的时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许收集相关操作对表数据改动的时间。
* off表示禁止收集相关操作对表数据改动的时间。

**默认值：**on

**track\_sql\_count**

**参数说明：**控制对每个会话中当前正在执行的SELECT、INSERT、UPDATE、DELETE、MERGE INTO语句进行计数的统计数据。

在x86平台集中式部署下，硬件配置规格为32核CPU/256GB内存，使用Benchmark SQL 5.0工具测试性能，开关此参数性能影响约0.8%。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启计数功能。
* off表示关闭计数功能。

**默认值：on**

fig: **说明：**

* track\_sql\_count参数受track\_activities约束：
* track\_activities开启而track\_sql\_count关闭时，如果查询了gs\_sql\_count视图，日志中将会有WARNING提示track\_sql\_count是关闭的；
* track\_activities和track\_sql\_count同时关闭，那么此时日志中将会有两条WARNING，分别提示track\_activities是关闭的和track\_sql\_count是关闭的；
* track\_activities关闭而track\_sql\_count开启，此时日志中将仅有WARNING提示track\_activities是关闭。
* 当参数关闭时，查询视图的结果为0行。

### 性能统计

在数据库运行过程中，会涉及到锁的访问、磁盘IO操作、无效消息的处理，这些操作都可能是数据库的性能瓶颈，通过Vastbase提供的性能统计方法，可以方便定位性能问题。

**输出性能统计日志**

**参数说明：**对每条查询，以下4个选项控制在服务器日志里记录相应模块的性能统计数据，具体含义如下：

* log\_parser\_stats控制在服务器日志里记录解析器的性能统计数据。
* log\_planner\_stats控制在服务器日志里记录查询优化器的性能统计数据。
* log\_executor\_stats控制在服务器日志里记录执行器的性能统计数据。
* log\_statement\_stats控制在服务器日志里记录整个语句的性能统计数据。

这些参数只能辅助管理员进行粗略分析，类似Linux中的操作系统工具getrusage() 。

这些参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig: **须知：**

* log\_statement\_stats记录总的语句统计数据，而其他的只记录针对每个模块的统计数据。
* log\_statement\_stats不能和其他任何针对每个模块统计的选项一起打开。

**取值范围：**布尔型

* on表示开启记录性能统计数据的功能。
* off表示关闭记录性能统计数据的功能。

**默认值：**off

## 负载管理

未对数据库资源做控制时，容易出现并发任务抢占资源导致操作系统过载甚至最终崩溃。操作系统过载时，其响应用户任务的速度会变慢甚至无响应；操作系统崩溃时，整个系统将无法对用户提供任何服务。Vastbase的负载管理功能能够基于可用资源的多少均衡数据库的负载，以避免数据库系统过载。

**use\_workload\_manager**

**参数说明**：是否开启资源管理功能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示打开资源管理。
* off表示关闭资源管理。

fig: **说明：**

* 当使用：配置运行参数->重设参数章节中表2中的方式二来修改参数值时，新参数值只能对更改操作执行后启动的线程生效。此外，对于后台线程以及线程复用执行的新作业，该参数值的改动不会生效。如果希望这类线程即时识别参数变化，可以使用kill session或重启节点的方式来实现。
* use\_workload\_manager参数由off变为on状态后，不会统计off时的存储资源。如果需要统计off时用户使用的存储资源，请在数据库中执行以下命令：
* select gs\_wlm\_readjust\_user\_space(0);

**默认值**：on

**cgroup\_name**

**参数说明**：设置当前使用的Cgroups的名称或者调整当前group下排队的优先级。

即如果先设置cgroup\_name，再设置session\_respool，那么session\_respool关联的控制组起作用，如果再切换cgroup\_name，那么新切换的cgroup\_name起作用。

切换cgroup\_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中方式三的方法进行设置。

建议尽量不要混合使用cgroup\_name和session\_respool。

**取值范围**：字符串

**默认值**：InvalidGroup

**cpu\_collect\_timer**

**参数说明**：设置语句执行时在数据库节点上收集CPU时间的周期。

数据库管理员需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持较合适的收集周期，太小会影响执行效率，太大会影响异常处理的精确度。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，1～INT\_MAX， 单位为秒。

**默认值**：30

**memory\_tracking\_mode**

**参数说明**：设置记录内存信息的模式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：

* none：表示不启动内存统计功能。
* peak：表示统计query级内存peak值，此数值计入数据库日志，也可以由explain analyze输出。
* normal：表示仅做内存实时统计，不生成文件。
* executor：表示生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。
* fullexec：表示生成文件包含执行层申请过的所有内存上下文信息。

**默认值**：none

**memory\_detail\_tracking**

**参数说明**：设置需要的线程内分配内存上下文的顺序号以及当前线程所在query的plannodeid。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：字符型

**默认值**：空

fig: **须知：**  
该参数不允许用户进行设置，建议保持默认值。

**enable\_resource\_track**

**参数说明**：是否开启资源实时监控功能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示打开资源监控。
* off表示关闭资源监控。

**默认值**：on

**enable\_resource\_record**

**参数说明**：是否开启资源监控记录归档功能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启资源监控记录归档功能。
* off表示关闭资源监控记录归档功能。

**默认值**：off

**enable\_logical\_io\_statistics**

**参数说明**：设置是否开启资源监控逻辑IO统计功能。开启时，对于PG\_TOTAL\_USER\_RESOURCE\_INFO视图中的read\_kbytes、write\_kbytes、read\_counts、write\_counts、read\_speed和write\_speed字段，会统计对应用户的逻辑读写字节数、次数以及速率。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启资源监控逻辑IO统计功能。
* off表示关闭资源监控逻辑IO统计功能。

**默认值**：on

**enable\_user\_metric\_persistent**

**参数说明**：设置是否开启用户历史资源监控转存功能。开启时，对于PG\_TOTAL\_USER\_RESOURCE\_INFO视图中数据，会定期采样保存到系统表和系统视图->系统表->GS\_WLM\_EC\_OPERATOR\_INFO表中。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启用户历史资源监控转存功能。
* off表示关闭用户历史资源监控转存功能。

**默认值**：on

**user\_metric\_retention\_time**

**参数说明**：设置用户历史资源监控数据的保存天数。该参数仅在 enable\_user\_metric\_persistent为on时有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中的方法一和方法二进行设置。

**取值范围**：整型，0～3650，单位为天。

* 值等于0时，用户历史资源监控数据将永久保存。
* 值大于0时，用户历史资源监控数据将保存对应天数。

**默认值**：7

**enable\_instance\_metric\_persistent**

**参数说明**：设置是否开启实例资源监控转存功能。开启时，对实例的监控数据会保存到系统表和系统视图->系统表->GS\_WLM\_INSTANCE\_HISTORY表中。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启实例资源监控转存功能。
* off表示关闭实例资源监控转存功能。

**默认值**：on

**instance\_metric\_retention\_time**

**参数说明**：设置实例历史资源监控数据的保存天数。该参数仅在enable\_instance\_metric\_persistent为on时有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中的方法一和方法二进行设置。

**取值范围**：整型，0～3650，单位为天。

* 值等于0时，实例历史资源监控数据将永久保存。
* 值大于0时，实例历史资源监控数据将保存对应设置天数。

**默认值**：7

**resource\_track\_level**

**参数说明**：设置当前会话的资源监控的等级。该参数只有当参数enable\_resource\_track为on时才有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：枚举型

* none：表示不开启资源监控功能。
* query：表示开启query级别资源监控功能。
* operator：表示开启query级别和算子级别资源监控功能。

**默认值**：query

**resource\_track\_cost**

**参数说明**：设置对当前会话的语句进行资源监控的最小执行代价。该参数只有当参数enable\_resource\_track为on时才有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，-1～INT\_MAX

* 值为-1时，不进行资源监控。
* 值大于或等于0且小于等于9时，对执行代价大于等于10的语句进行资源监控。
* 值大于或等于10时，对执行代价超过该参数值的语句进行资源监控。

**默认值**：100000

**resource\_track\_duration**

**参数说明**：设置资源监控实时视图中记录的语句执行结束后进行历史信息转存的最小执行时间。当执行完成的作业，其执行时间不小于此参数值时，作业信息会从实时视图（以statistics为后缀的视图）转存到相应的历史视图（以history为后缀的视图）中。该参数只有当enable\_resource\_track为on时才有效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，0～INT\_MAX，单位为秒。

* 值为0时，资源监控实时视图中记录的所有语句都进行历史信息归档。
* 值大于0时，资源监控实时视图中记录的语句的执行时间超过这个值就会进行历史信息归档。

**默认值**：1min

**disable\_memory\_protect**

**参数说明**：设置是否禁止内存保护功能。当系统内存不足时如果需要查询系统视图，可以先将此参数置为on，禁止内存保护功能，保证视图可以正常查询。该参数只适用于在系统内存不足时进行系统诊断和调试，正常运行时请保持该参数配置为off。

该参数属于USERSET类型参数，且只对当前会话有效。请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示禁止内存保护功能。
* off表示启动内存保护功能。

**默认值**：off

**query\_band**

**参数说明**：用于标示当前会话的作业类型，由用户自定义。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：字符型

**默认值**：空

**memory\_fault\_percent**

**参数说明**：内存故障测试时内存申请失败的比例，仅用在DEBUG版本。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，0～2147483647

**默认值**：0

**enable\_bbox\_dump**

**参数说明**：是否开启黑匣子功能，在系统不配置core机制的时候仍可产生core文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示打开黑匣子功能。
* off表示关闭黑匣子功能。

**默认值**：on

fig: **须知：**   
黑匣子功能生成core文件依赖操作系统开放ptrace接口。若发生权限不足(errno = 1)，请确保/proc/sys/kernel/yama/ptrace\_scope配置合理。

**bbox\_dump\_count**

**参数说明**：在[bbox\_dump\_path](#X052337f31d64c680f87e3b7a382a940c841a2d5)定义的路径下，允许存储的Vastbase所产生core文件最大数。超过此数量，旧的core文件会被删除。此参数只有当[enable\_bbox\_dump](#X368431c0f6d6d49b65dff24e5a99ae39802e9d4)为on时才生效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，1～20

**默认值**：8

fig: **说明：**   
在并发产生core文件时，core文件的产生个数可能大于bbox\_dump\_count。

**bbox\_dump\_path**

**参数说明**：黑匣子core文件的生成路径。此参数只有当[enable\_bbox\_dump](#X368431c0f6d6d49b65dff24e5a99ae39802e9d4)为on时才生效。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：字符型

**默认值**：空。默认生成黑匣子core文件的路径为读取/proc/sys/kernel/core\_pattern下的路径，如果这个路径不是一个目录，或者用户对此目录没有写权限，黑匣子core文件将生成在数据库的data目录下。或者以安装时指定的目录为准。

**enable\_ffic\_log**

**参数说明**：是否开启FFIC（First Failure Info Capture）功能。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：布尔型

* on表示打开FFIC功能。
* off表示关闭FFIC功能。

**默认值**：on

**io\_limits**

**参数说明**：每秒触发IO的上限。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，0～1073741823

**默认值**：0

**io\_priority**

**参数说明**：IO利用率高达50%时，重消耗IO作业进行IO资源管控时关联的优先级等级。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：枚举型

* None：表示不受控。
* Low：表示限制iops为该作业原始触发数值的10%。
* Medium：表示限制iops为该作业原始触发数值的20%。
* High：表示限制iops为该作业原始触发数值的50%。

**默认值**：None

**io\_control\_unit**

**参数说明**：行存场景下，io管控时用来对io次数进行计数的单位。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2中对应类型的设置方法进行设置。

记多少次io触发为一计数单位，通过此计数单位所记录的次数进行io管控。

**取值范围**：整型，1000～1000000

**默认值：**6000

**session\_respool**

**参数说明**：当前的session关联的resource pool。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中对应类型的设置方法进行设置。

即如果先设置cgroup\_name，再设置session\_respool，那么session\_respool关联的控制组起作用，如果再切换cgroup\_name，那么新切换的cgroup\_name起作用。

切换cgroup\_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

建议尽量不要混合使用cgroup\_name和session\_respool。

**取值范围**：string类型，通过create resource pool所设置的资源池。

**默认值**：invalid\_pool

**session\_statistics\_memory**

**参数说明**：设置实时查询视图的内存大小。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，5 \* 1024 ～ max\_process\_memory的50%，单位KB。

**默认值**：5MB

**topsql\_retention\_time**

**参数说明**：设置历史TopSQL中gs\_wlm\_operator\_info表中数据的保存时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，0～3650，单位为天。

* 值为0时，表示数据永久保存。
* 值大于0时，表示数据能够保存的对应天数。

**默认值**：0

**session\_history\_memory**

**参数说明**：设置历史查询视图的内存大小。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围**：整型，10 \* 1024 ～ max\_process\_memory的50%，单位KB。

**默认值**：10MB

**transaction\_pending\_time**

**参数说明**：事务块语句和存储过程语句排队的最大时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，-1～INT\_MAX/2，单位为秒。

* 值为-1或0时，事务块语句和存储过程语句无超时判断，排队至资源满足可执行条件。
* 值大于0时，事务块语句和存储过程语句排队超过所设数值的时间后，无视当前资源情况强制执行。

**默认值**：0

## 自动清理

系统自动清理线程（autovacuum）自动执行VACUUM和ANALYZE命令，回收被标识为删除状态的记录空间，并更新表的统计数据。

**autovacuum**

**参数说明：**控制数据库自动清理线程（autovacuum）的启动。自动清理线程运行的前提是将GUC参数说明->运行时统计->查询和索引统计收集器中track\_counts参数设置为on。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

fig: **说明：**

* 如果希望系统在故障恢复后，具备自动清理两阶段事务的功能，请将autovacuum设置为on;
* 当设置autovacuum为on，autovacuum\_max\_workers为0时，表示系统不会自动进行autovacuum，只会在故障恢复后，自动清理两阶段事务；
* 当设置autovacuum为on，autovacuum\_max\_workers大于0时，表示系统不仅在故障恢复后，自动清理两阶段事务，并且还可以自动清理线程。

fig: **须知：**   
即使此参数设置为off，当事务ID回绕即将发生时，数据库也会自动启动自动清理线程。对于create/drop database发生异常时，可能有的节点提交或回滚，有的节点未提交（prepared状态），此时系统不能自动修复，需要手动修复。

**取值范围：**布尔型

* on表示开启数据库自动清理线程。
* off表示关闭数据库自动清理线程。

**默认值：**on

**autovacuum\_mode**

**参数说明：**该参数仅在autovacuum设置为on的场景下生效，它控制autoanalyze或autovacuum的打开情况。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* analyze表示只做autoanalyze。
* vacuum表示只做autovacuum。
* mix表示autoanalyze和autovacuum都做。
* none表示二者都不做。

**默认值：**mix

**autoanalyze\_timeout**

**参数说明：**设置autoanalyze的超时时间。在对某张表做autoanalyze时，如果该表的analyze时长超过了autoanalyze\_timeout，则自动取消该表此次analyze。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0~2147483，单位是秒。

**默认值：**5min（即300s）

**autovacuum\_io\_limits**

**参数说明：**控制autovacuum线程每秒触发IO的上限。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0～1073741823和-1。其中-1表示不控制，而是使用系统默认控制组。

**默认值：**-1

**log\_autovacuum\_min\_duration**

**参数说明：**当自动清理的执行时间大于或者等于某个特定的值时，向服务器日志中记录自动清理执行的每一步操作。设置此选项有助于追踪自动清理的行为。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

举例如下：

将log\_autovacuum\_min\_duration设置为250ms，记录所有运行大于或者等于250ms的自动清理命令的相关信息。

**取值范围：**整型，最小值为-1，最大值为2147483647，单位为毫秒。

* 当参数设置为0时，表示所有的自动清理操作都记录到日志中。
* 当参数设置为-1时，表示所有的自动清理操作都不记录到日志中。
* 当参数设置为非-1、非0时，当由于锁冲突的存在导致一个自动清理操作被跳过，记录一条消息。

**默认值：**-1

**autovacuum\_max\_workers**

**参数说明：**设置能同时运行的自动清理线程的最大数量，该参数的取值上限与GUC参数max\_connections和job\_queue\_processes大小有关。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值为0（表示不会自动进行autovacuum），理论最大值为262143，实际最大值为动态值，计算公式为“262143 - max\_connections的值 - job\_queue\_processes的值 - 辅助线程数 – autovacuum的lancher线程数 - 1”，其中辅助线程数和autovacuum的lancher线程数由两个宏来指定，当前版本的默认值分别为20和2。

**默认值：**3

**autovacuum\_naptime**

**参数说明：**设置两次自动清理操作的时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，单位为s，最小值为1，最大值为2147483。

**默认值：**10min（即600s）

**autovacuum\_vacuum\_threshold**

**参数说明：**设置触发VACUUM的阈值。当表上被删除或更新的记录数超过设定的阈值时才会对这个表执行VACUUM操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为2147483647。

**默认值：**50

**autovacuum\_analyze\_threshold**

**参数说明：**设置触发ANALYZE操作的阈值。当表上被删除、插入或更新的记录数超过设定的阈值时才会对这个表执行ANALYZE操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为2147483647。

**默认值：**50

**autovacuum\_vacuum\_scale\_factor**

**参数说明：**设置触发一个VACUUM时增加到autovacuum\_vacuum\_threshold的表大小的缩放系数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**浮点型，0.0～100.0

**默认值：**0.2

**autovacuum\_analyze\_scale\_factor**

**参数说明：**设置触发一个ANALYZE时增加到autovacuum\_analyze\_threshold的表大小的缩放系数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**浮点型，0.0～100.0

**默认值：**0.1

**autovacuum\_freeze\_max\_age**

**参数说明：**设置事务内的最大时间，使得表的pg\_class.relfrozenxid字段在VACUUM操作执行之前被写入。

* VACUUM也可以删除pg\_clog/子目录中的旧文件。
* 即使自动清理线程被禁止，系统也会调用自动清理线程来防止循环重复。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**长整型，100 000～576 460 752 303 423 487

**默认值：**4000000000

**autovacuum\_vacuum\_cost\_delay**

**参数说明：**设置在自动VACUUM操作里使用的开销延迟数值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，-1～100，单位为毫秒（ms）。其中-1表示使用常规的vacuum\_cost\_delay。

**默认值：**20ms

**autovacuum\_vacuum\_cost\_limit**

**参数说明：**设置在自动VACUUM操作里使用的开销限制数值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，-1～10000。其中-1表示使用常规的vacuum\_cost\_limit。

**默认值：**-1

**defer\_csn\_cleanup\_time**

**参数说明：**用来指定本地回收时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0~INT\_MAX，单位为毫秒（ms）。

**默认值**：5s（即5000ms）

## 客户端连接缺省设置

### 语句行为

介绍SQL语句执行过程的相关默认参数。

**search\_path**

**参数说明：**当一个被引用对象没有指定模式时，此参数设置模式搜索顺序。它的值由一个或多个模式名构成，不同的模式名用逗号隔开。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

* 当前会话如果存放临时表的模式时，可以使用别名pg\_temp将它列在搜索路径中，如'pg\_temp，public'。存放临时表的模式始终会作为第一个被搜索的对象，排在pg\_catalog和search\_path中所有模式的前面，即具有第一搜索优先级。建议用户不要在search\_path中显示设置pg\_temp。如果在search\_path中指定了pg\_temp，但不是在最前面，系统会提示设置无效，pg\_temp仍被优先搜索。通过使用别名pg\_temp，系统只会在存放临时表的模式中搜索表、视图和数据类型这样的数据库对象，不会在里面搜索函数或运算符这样的数据库对象。
* 系统表所在的模式pg\_catalog，总是排在search\_path中指定的所有模式前面被搜索，即具有第二搜索优先级（pg\_temp具有第一搜索优先级）。建议用户不要在search\_path中显式设置pg\_catalog。如果在search\_path中指定了pg\_catalog，但不是在最前面，系统会提示设置无效，pg\_catalog仍被第二优先搜索。
* 当没有指定一个特定模式而创建一个对象时，它们被放置到以search\_path为命名的第一个有效模式中。当搜索路径为空时，会报错误。
* 通过SQL函数current\_schema可以检测当前搜索路径的有效值。这和检测search\_path的值不尽相同，因为current\_schema显示search\_path中首位有效的模式名称。

**取值范围：**字符串

fig: **说明：**

* 设置为“$user”，public时，支持共享数据库（没有用户具有私有模式和所有共享使用public）、用户私有模式和这些功能的组合使用。可以通过改变默认搜索路径来获得其他效果，无论是全局化的还是私有化的。
* 设置为空串（''）的时候，系统会自动转换成一对双引号。
* 设置的内容中包含双引号，系统会认为是不安全字符，会将每个双引号转换成一对双引号。

**默认值：**“$user”,public

fig: **说明：**   
$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

**current\_schema**

**参数说明：**此参数设置当前的模式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

**默认值：**"$user",public

fig: **说明：**   
$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

**default\_tablespace**

**参数说明：**当CREATE命令没有明确声明表空间时，所创建对象（表和索引等）的缺省表空间。

* 值是一个表空间的名称或者一个表示使用当前数据库缺省表空间的空字符串。若指定的是一个非默认表空间，用户必须具有它的CREATE权限，否则尝试创建会失败。
* 临时表不使用此参数，可以用[temp\_tablespaces](#X71ff3ed4d8779c921b39fe4b723f66f23c180e9)代替。
* 创建数据库时不使用此参数。默认情况下，一个新的数据库从模板数据库继承表空间配置。
* 该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串，其中空表示使用默认表空间。

**默认值：**空

**temp\_tablespaces**

**参数说明：**当一个CREATE命令没有明确指定一个表空间时，temp\_tablespaces指定了创建临时对象（临时表和临时表的索引）所在的表空间。在这些表空间中创建临时文件用来做大型数据的排序工作。

其值是一系列表空间名的列表。如果列表中有多个表空间时，每次临时对象的创建，Vastbase会在列表中随机选择一个表空间；如果在事务中，连续创建的临时对象被放置在列表里连续的表空间中。如果选择的列表中的元素是一个空串，Vastbase将自动将当前的数据库设为默认的表空间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串。空字符串表示所有的临时对象仅在当前数据库默认的表空间中创建，请参见[default\_tablespace](#X1cd62ea43b2f608445689551cdbee7f5e2a8689)。

**默认值：**空

**check\_function\_bodies**

**参数说明：**设置是否在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。为了避免产生问题（比如避免从转储中恢复函数定义时向前引用的问题），偶尔会禁用验证。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。
* off表示在CREATE FUNCTION执行过程中不进行函数体字符串的合法性验证。

**默认值：**on

**default\_transaction\_isolation**

**参数说明：**设置默认的事务隔离级别。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* read committed表示事务读已提交。
* repeatable read表示事务可重复读。
* serializable，Vastbase目前功能上不支持此隔离级别，等价于repeatable read。

**默认值：**read committed

**default\_transaction\_read\_only**

**参数说明：**设置每个新创建事务是否是只读状态。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示只读状态。
* off表示非只读状态。

**默认值：**off

**default\_transaction\_deferrable**

**参数说明：**控制每个新事务的默认延迟状态。只读事务或者那些比序列化更加低的隔离级别的事务除外。

Vastbase不支持可串行化的隔离级别，因此，该参数无实际意义。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示默认延迟。
* off表示默认不延迟。

**默认值：**off

**session\_replication\_role**

**参数说明：**控制当前会话与复制相关的触发器和规则的行为。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

fig: **须知：**   
设置此参数会丢弃之前任何缓存的执行计划。

**取值范围：**枚举类型

* origin表示从当前会话中复制插入、删除、更新等操作。
* replica表示从其他地方复制插入、删除、更新等操作到当前会话。
* local表示函数执行复制时会检测当前登录数据库的角色并采取相应的操作。

**默认值：**origin

**statement\_timeout**

**参数说明：**当语句执行时间超过该参数设置的时间（从服务器收到命令时开始计时）时，该语句将会报错并退出执行。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为2147483647，单位为毫秒。

**默认值：**0

**vacuum\_freeze\_min\_age**

**参数说明：**指定VACUUM在扫描一个表时用于判断是否用FrozenXID替换事务ID的中断寿命（在同一个事务中）。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0～576 460 752 303 423 487

fig: **说明：**   
尽管随时可以将此参数设为0到10亿之间的任意值，但是，VACUUM将默认其有效值范围限制在GUC参数说明->自动清理中autovacuum\_freeze\_max\_age参数的50%以内。

**默认值：**2000000000

**vacuum\_freeze\_table\_age**

**参数说明：**指定VACUUM对全表的扫描冻结元组的时间。如果表的pg\_class.relfrozenxid字段的值已经达到了参数指定的时间，VACUUM对全表进行扫描。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0～576 460 752 303 423 487

fig: **说明：**   
尽管随时可以将此参数设为零到20亿之间的值，但是，VACUUM将默认其有效值范围限制在GUC参数说明->自动清理中autovacuum\_freeze\_max\_age参数的95%以内。定期的手动VACUUM可以在对此表的反重叠自动清理启动之前运行。

**默认值：**15000000000

**bytea\_output**

**参数说明：**设置bytea类型值的输出格式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* hex：将二进制数据编码为每字节2位十六进制数字。
* escape：传统化的PostgreSQL格式。采用以ASCII字符序列表示二进制串的方法，同时将那些无法表示成ASCII字符的二进制串转换成特殊的转义序列。

**默认值：**hex

**xmlbinary**

**参数说明：**设置二进制值是如何在XML中进行编码的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* base64
* hex

**默认值：**base64

**xmloption**

**参数说明：**当XML和字符串值之间进行转换时，设置document或content是否是隐含的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* document：表示HTML格式的文档。
* content：普通的字符串。

**默认值：**content

**max\_compile\_functions**

**参数说明：**设置服务器存储的函数编译结果的最大数量。存储过多的函数和存储过程的编译结果可能占用很大内存。将此参数设置为一个合理的值，有助于减少内存占用，提升系统性能。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，1~ 2147483647。

**默认值：**1000

**gin\_pending\_list\_limit**

**参数说明：**设置当GIN索引启用fastupdate时，pending list容量的最大值。当pending list的容量大于设置值时，会把pending list中数据批量移动到GIN索引数据结构中以进行清理。单个GIN索引可通过更改索引存储参数覆盖此设置值。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值为64，最大值为INT\_MAX，单位为KB。

**默认值：**4MB

### 区域和格式化

介绍时间格式设置的相关参数。

**DateStyle**

**参数说明：**设置日期和时间值的显示格式，以及有歧义的输入值的解析规则。

这个变量包含两个独立的加载部分：输出格式声明（ISO、Postgres、SQL、German）和输入输出的年/月/日顺序（DMY、MDY、YMD）。这两个可以独立设置或者一起设置。关键字Euro和European等价于DMY；关键字US、NonEuro、NonEuropean等价于MDY。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

**默认值：**ISO,MDY

fig: **说明：**   
vb\_initdb会将这个参数初始化成与[lc\_time](#Xe2075b6e04779ac6058d80e367a1dfd2064fb67)一致的值。

**设置建议：**优先推荐使用ISO格式。Postgres、SQL和German均采用字母缩写的形式来表示时区，例如“EST、WST、CST”等。这些缩写可同时指代不同的时区，比如CST可同时代表美国中部时间（Central Standard Time (USA) UT-6:00）、澳大利亚中部时间（Central Standard Time (Australia) UT+9:30）、中国标准时间（China Standard Time UT+8:00）、古巴标准时间（Cuba Standard Time UT-4:00）。这种情况下在时区转化时可能会得不到正确的结果，从而引发其他问题。

**IntervalStyle**

**参数说明：**设置区间值的显示格式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**枚举类型

* sql\_standard表示产生与SQL标准规定匹配的输出。
* postgres表示产生与PostgreSQL 8.4版本相匹配的输出，当[DateStyle](#Xf1d5e51bc829da866085b953cdb4d458ec84031)参数被设为ISO时。
* postgres\_verbose表示产生与PostgreSQL 8.4版本相匹配的输出，当[DateStyle](#Xf1d5e51bc829da866085b953cdb4d458ec84031)参数被设为non\_ISO时。
* iso\_8601表示产生与在ISO 8601中定义的“格式与代号”相匹配的输出。
* a表示与numtodsinterval函数相匹配的输出结果，详细请参考：SQL语法参考->SQL基本要素->函数和操作符->时间和日期处理函数和操作符中numtodsinterval。

fig: **须知：**   
IntervalStyle参数也会影响不明确的间隔输入的说明。

**默认值：**postgres

**TimeZone**

**参数说明：**设置显示和解释时间类型数值时使用的时区。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串，可查询视图系统表和系统视图->系统视图->PG\_TIMEZONE\_NAMES获得。

**默认值：**PRC

fig: **说明：**   
vb\_initdb将设置一个与其系统环境一致的时区值。

**timezone\_abbreviations**

**参数说明：**设置服务器接受的时区缩写值。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串，可查询视图pg\_timezone\_names获得。

**默认值：**Default

fig: **说明：**   
Default表示通用时区的缩写，适合绝大部分情况。但也可设置其他诸如 'Australia' 和 'India' 等用来定义特定的安装。而设置除此之外的时区缩写, 需要在建数据库之前通过相应的配置文件进行设置。

**extra\_float\_digits**

**参数说明：**这个参数为浮点数值调整显示的数据位数，浮点类型包括float4、float8以及几何数据类型。参数值加在标准的数据位数上（FLT\_DIG或DBL\_DIG中合适的）。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，-15～3

fig: **说明：**

* 设置为3，表示包括部分关键的数据位。这个功能对转储那些需要精确恢复的浮点数据特别有用。
* 设置为负数，表示消除不需要的数据位。

**默认值：**0

**client\_encoding**

**参数说明：**设置客户端的字符编码类型。

请根据前端业务的情况确定。尽量客户端编码和服务器端编码一致，提高效率。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**兼容PostgreSQL所有的字符编码类型。其中UTF8表示使用数据库的字符编码类型。

fig: **说明：**

* 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
* 默认情况下，vb\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。
* 参数建议保持默认值，不建议通过vb\_guc工具或其他方式直接在postgresql.conf文件中设置client\_encoding参数，即使设置也不会生效，以保证Vastbase内部通信编码格式一致。

**默认值：**UTF8

**推荐值：**SQL\_ASCII/UTF8

**lc\_messages**

**参数说明：**设置信息显示的语言。

* 可接受的值是与系统相关的。
* 在一些系统上，这个区域范畴并不存在，不过仍然允许设置这个变量，只是不会有任何效果。同样，也有可能是所期望的语言的翻译信息不存在。在这种情况下，用户仍然能看到英文信息。
* 该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**

* 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
* 默认情况下，vb\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

**默认值：**en\_US.utf8

**lc\_monetary**

**参数说明：**设置货币值的显示格式，影响to\_char之类的函数的输出。可接受的值是系统相关的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**

* 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
* 默认情况下，vb\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

**默认值：**en\_US.utf8

**lc\_numeric**

**参数说明：**设置数值的显示格式，影响to\_char之类的函数的输出。可接受的值是系统相关的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**

* 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
* 默认情况下，vb\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

**默认值：**en\_US.utf8

**lc\_time**

**参数说明：**设置时间和区域的显示格式，影响to\_char之类的函数的输出。可接受的值是系统相关的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**

* 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
* 默认情况下，vb\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

**默认值：**en\_US.utf8

**default\_text\_search\_config**

**参数说明：**设置全文检索的配置信息。

如果设置为不存在的文本搜索配置时将会报错。如果default\_text\_search\_config对应的文本搜索配置被删除，需要重新设置default\_text\_search\_config，否则会报设置错误。

* 其被文本搜索函数使用，这些函数并没有一个明确指定的配置。
* 当与环境相匹配的配置文件确定时，vb\_initdb会选择一个与环境相对应的设置来初始化配置文件。
* 该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**   
Vastbase支持pg\_catalog.english、pg\_catalog.simple两种配置。

**默认值：**pg\_catalog.english

### 其他缺省

主要介绍数据库系统默认的库加载参数。

**dynamic\_library\_path**

**参数说明：**设置数据查找动态加载的共享库文件的路径。当需要打开一个可以动态装载的模块并且在CREATE FUNCTION或LOAD命令里面声明的名称没有目录部分时，系统将搜索这个目录以查找声明的文件。

用于dynamic\_library\_path的数值必须是一个冒号分隔的绝对路径列表。当一个路径名称以特殊变量$libdir为开头时，会替换为Vastbase发布提供的模块安装路径。例如：

dynamic\_library\_path = '/usr/local/lib/gaussdb:/opt/testgs/lib:$libdir'

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

fig: **说明：**   
设置为空字符串，表示关闭自动路径搜索。

**默认值：**$libdir

**gin\_fuzzy\_search\_limit**

**参数说明：**设置GIN索引返回的集合大小的上限。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0~2147483647

**默认值：**0

**local\_preload\_libraries**

**参数说明：**指定一个或多个共享库，它们在开始连接前预先加载。多个加载库之间用逗号分隔，除了双引号，所有的库名都转换为小写。

* 并非只有系统管理员才能更改此选项，因此只能加载安装的标准库目录下plugins子目录中的库文件，数据库管理员有责任确保该目录中的库都是安全的。local\_preload\_libraries中指定的项可以明确含有该目录，例如$libdir/plugins/mylib；也可以仅指定库的名称，例如mylib（等价于$libdir/plugins/mylib）。
* 与shared\_preload\_libraries不同，在会话开始之前加载模块与在会话中使用到该模块的时候临时加载相比并不具有性能优势。相反，这个特性的目的是为了调试或者测量在特定会话中不明确使用LOAD加载的库。例如针对某个用户将该参数设为ALTER USER SET来进行调试。
* 当指定的库未找到时，连接会失败。
* 每一个支持Vastbase的库都有一个“magic block”用于确保兼容性，因此不支持Vastbase的库不能通过这个方法加载。

该参数属于BACKEND类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

**默认值：**空

## 锁管理

在Vastbase中，并发执行的事务由于竞争资源会导致死锁。本节介绍的参数主要管理事务锁的机制。

**deadlock\_timeout**

**参数说明：**设置死锁超时检测时间，以毫秒为单位。当申请的锁超过设定值时，系统会检查是否产生了死锁。

* 死锁的检查代价是比较高的，服务器不会在每次等待锁的时候都运行这个过程。在系统运行过程中死锁是不经常出现的，因此在检查死锁前只需等待一个相对较短的时间。增加这个值就减少了无用的死锁检查浪费的时间，但是会减慢真正的死锁错误报告的速度。在一个负载过重的服务器上，用户可能需要增大它。这个值的设置应该超过事务持续时间，这样就可以减少在锁释放之前就开始死锁检查的问题。
* 当设置GUC参数说明->错误报告和日志->记录日志的内容中log\_lock\_waits参数为on时，deadlock\_timeout决定一个等待时间来将查询执行过程中的锁等待耗时信息写入日志。如果要研究锁延时情况，可以设置deadlock\_timeout的值比正常情况小。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，1~2147483647，单位为毫秒（ms）。

**默认值：**1s

**lockwait\_timeout**

**参数说明：**控制单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0 ~ INT\_MAX，单位为毫秒（ms）。

**默认值：**20min

**update\_lockwait\_timeout**

**参数说明：**允许并发更新参数开启情况下，该参数控制并发更新同一行时单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0 ~ INT\_MAX，单位为毫秒（ms）。

**默认值：**20min

**max\_locks\_per\_transaction**

**参数说明：**控制每个事务能够得到的平均的对象锁的数量。

* 共享的锁表的大小是以假设任意时刻最多只有max\_locks\_per\_transaction\*(max\_connections+max\_prepared\_transactions) 个独立的对象需要被锁住为基础进行计算的。不超过设定数量的多个对象可以在任一时刻同时被锁定。当在一个事务里面修改很多不同的表时，可能需要提高这个默认数值。只能在数据库启动的时候设置。
* 增大这个参数可能导致Vastbase请求更多的System V共享内存，有可能超过操作系统的缺省配置。
* 当运行备机时，请将此参数设置不小于主机上的值，否则，在备机上查询操作不会被允许。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，10 ~ INT\_MAX

**默认值：**256

**max\_pred\_locks\_per\_transaction**

**参数说明：**控制每个事务允许断定锁的最大数量，是一个平均值。

* 共享的断定锁表的大小是以假设任意时刻最多只有max\_pred\_locks\_per\_transaction\*(max\_connections+max\_prepared\_transactions) 个独立的对象需要被锁住为基础进行计算的。不超过设定数量的多个对象可以在任一时刻同时被锁定。当在一个事务里面修改很多不同的表时，可能需要提高这个默认数值。只能在服务器启动的时候设置。
* 增大这个参数可能导致Vastbase请求更多的System V共享内存，有可能超过操作系统的缺省配置。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，10 ~ INT\_MAX

**默认值：**64

**gs\_clean\_timeout**

**参数说明：**控制主节点周期性清理临时表的时间，是一个平均值。

* 数据库连接异常终止时，通常会有临时表残留，此时需要对数据库中的临时表进行清理。
* 增大这个参数可能导致Vastbase临时表清理时间延长。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0 ~ 2147483，单位为秒（s）。

**默认值：**1min

**partition\_lock\_upgrade\_timeout**

**参数说明：**在执行某些查询语句的过程中，会需要将分区表上的锁级别由允许读的ExclusiveLock级别升级到读写阻塞的AccessExclusiveLock级别。如果此时已经存在并发的读事务，那么该锁升级操作将阻塞等待。partition\_lock\_upgrade\_timeout为尝试锁升级的等待超时时间。

* 在分区表上进行MERGE PARTITION和CLUSTER PARTITION操作时，都利用了临时表进行数据重排和文件交换，为了最大程度提高分区上的操作并发度，在数据重排阶段给相关分区加锁ExclusiveLock，在文件交换阶段加锁AccessExclusiveLock。
* 常规加锁方式是等待加锁，直到加锁成功，或者等待时间超过[lockwait\_timeout](#Xd769d401893cdfabacd98361406af0c570de225)发生超时失败。
* 在分区表上进行MERGE PARTITION或CLUSTER PARTITION操作时，进入文件交换阶段需要申请加锁AccessExclusiveLock，加锁方式是尝试性加锁，加锁成功了则立即返回，不成功则等待50ms后继续下次尝试，加锁超时时间使用会话级设置参数partition\_lock\_upgrade\_timeout。
* 特殊值：若partition\_lock\_upgrade\_timeout取值-1，表示无限等待，即不停的尝试锁升级，直到加锁成功。
* 该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值-1，最大值3000，单位为秒（s）。

**默认值：**1800

**fault\_mon\_timeout**

**参数说明：**轻量级死锁检测周期。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，最小值0，最大值1440，单位为分钟（min）

**默认值：**5min

**enable\_online\_ddl\_waitlock**

**参数说明：**控制DDL是否会阻塞等待pg\_advisory\_lock/pgxc\_lock\_for\_backup等Vastbase锁。主要用于OM在线操作场景，不建议用户设置。

该参数属于SIGHUP类型参数，参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启。
* off表示关闭。

**默认值：**off

**xloginsert\_locks**

参数说明：控制用于并发写预写式日志锁的个数。主要用于提高写预写式日志的效率。

该参数属于POSTMASTER类型参数，参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

取值范围：整型，最小值1，最大值1000

默认值：8

**num\_internal\_lock\_partitions**

**参数说明：**控制内部轻量级锁分区的个数。主要用于各类场景的性能调优。内容以关键字和数字的KV方式组织，各个不同类型锁之间以逗号隔开。先后顺序对设置结果不影响，例如“CLOG\_PART=256,CSNLOG\_PART=512”等同于“CSNLOG\_PART=512,CLOG\_PART=256”。重复设置同一关键字时，以最后一次设置为准，例如“CLOG\_PART=256,CLOG\_PART=2”，设置的结果为CLOG\_PART=2。当没有设置关键字时，则为默认值，各类锁的使用描述和最大、最小、默认值如下。

* CLOG\_PART：CLOG文件控制器的个数。增大该值可以提高CLOG日志写入效率，提升事务提交性能，但是会增大内存使用；减小该值会减少相应内存使用，但可能使得CLOG日志写入冲突变大，影响性能。最小值为1，最大值为256。
* CSNLOG\_PART：CSNLOG文件控制器的个数。增大该值可以提高CSNLOG日志写入效率，提升事务提交性能，但是会增大内存使用；减小该值会减少相应内存使用，但可能使得CSNLOG日志写入冲突变大，影响性能。最小值为1，最大值为512。
* LOG2\_LOCKTABLE\_PART：常规锁表锁分区个数的2对数。增大该值可以提升正常流程常规锁获取锁的并行度，但是可能增加锁转移和锁消除时的耗时，对于等待事件在LockMgrLock时，可以调大该锁增加性能。最小值为4，即锁分区数为16；最大值为16，即锁分区数为65536。
* TWOPHASE\_PART：两阶段事务锁的分区数。调大该值可以提高两阶段事务提交的并发数。最小值为1，最大值为64。

该参数属于POSTMASTER类型参数，参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串

**默认值：**

* CLOG\_PART：256
* CSNLOG\_PART：512
* LOG2\_LOCKTABLE\_PART：4
* TWOPHASE\_PART：1

## 版本和平台兼容性

### 历史版本兼容性

Vastbase介绍数据库的向下兼容性和对外兼容性特性的参数控制。数据库系统的向后兼容性能够为旧版本的数据库应用提供支持。本节介绍的参数主要控制数据库的向后兼容性。

**array\_nulls**

**参数说明：**控制数组输入解析器是否将未用引用的NULL识别为数组的一个NULL元素。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许向数组中输入空元素。
* off表示向下兼容旧式模式。仍然能够创建包含NULL值的数组。

**默认值：**on

**backslash\_quote**

**参数说明：**控制字符串文本中的单引号是否能够用'表示。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

fig: **须知：**   
在字符串文本符合SQL标准的情况下，没有任何其他含义。这个参数影响的是如何处理不符合标准的字符串文本，包括明确的字符串转义语法是（E'...'）。

**取值范围：**枚举类型

* on表示一直允许使用'表示。
* off表示拒绝使用'表示。
* safe\_encoding表示仅在客户端字符集编码不会在多字节字符末尾包含的ASCII值时允许。

**默认值：**safe\_encoding

**escape\_string\_warning**

**参数说明：**警告在普通字符串中直接使用反斜杠转义。

* 如果需要使用反斜杠作为转义，可以调整为使用转义字符串语法（E'...'）来做转义，因为在每个SQL标准中，普通字符串的默认行为现在将反斜杠作为一个普通字符。
* 这个变量可以帮助定位需要改变的代码。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

**默认值：**on

**lo\_compat\_privileges**

**参数说明：**控制是否启动对大对象权限检查的向后兼容模式。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

on表示当读取或修改大对象时禁用权限检查，与PostgreSQL 9.0以前的版本兼容。

**默认值：**off

**quote\_all\_identifiers**

**参数说明：**当数据库生成SQL时，此选项强制引用所有的标识符（包括非关键字）。这将影响到EXPLAIN的输出及函数的结果，例如pg\_get\_viewdef。详细说明请参见gs\_dump的--quote-all-identifiers选项。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示打开强制引用。
* off表示关闭强制引用。

**默认值：**off

**sql\_inheritance**

**参数说明：**控制继承语义。用来控制继承表的访问策略，off表示各种命令不能访问子表，即默认使用ONLY关键字。这是为了兼容旧版本而设置的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示可以访问子表。
* off表示不访问子表。

**默认值：**on

**standard\_conforming\_strings**

**参数说明：**控制普通字符串文本（'...'）中是否按照SQL标准把反斜扛当普通文本。

* 应用程序通过检查这个参数可以判断字符串文本的处理方式。
* 建议明确使用转义字符串语法（E'...'）来转义字符。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示打开控制功能。
* off表示关闭控制功能。

**默认值：**on

**synchronize\_seqscans**

**参数说明：**控制启动同步的顺序扫描。在大约相同的时间内并行扫描读取相同的数据块，共享I/O负载。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示扫描可能从表的中间开始，然后选择“环绕”方式来覆盖所有的行，为了与已经在进行中的扫描活动同步。这可能会造成没有用ORDER BY子句的查询得到行排序造成不可预测的后果。
* off表示确保顺序扫描是从表头开始的。

**默认值：**on

**enable\_beta\_features**

**参数说明：**控制开启某些非正式发布的特性，仅用于POC验证。这些特性属于延伸特性，建议客户谨慎开启，在某些功能场景下可能存在问题。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启这些功能受限的特性，保持前向兼容。但某些场景可能存在功能上的问题。
* off表示禁止使用这些特性。

**默认值：**off

### 平台和客户端兼容性

很多平台都使用数据库系统，数据库系统的对外兼容性给平台提供了很大的方便。

**convert\_string\_to\_digit**

**参数说明**：设置隐式转换优先级，是否优先将字符串转为数字。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示优先将字符串转为数字。
* off表示不优先将字符串转为数字。

**默认值**：on

fig: **须知：**  
该参数调整会修改内部数据类型转换规则，导致不可预期的行为，请谨慎操作。

**nls\_timestamp\_format**

**参数说明**：设置时间戳默认格式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串

**默认值**：DD-Mon-YYYY HH:MI:SS.FF AM

**max\_function\_args**

**参数说明**：函数参数最大个数。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：整型

**默认值**：8192

**transform\_null\_equals**

**参数说明：**控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。如果expr得出NULL值则返回真，否则返回假。

* 正确的SQL标准兼容的expr = NULL总是返回NULL（未知）。
* Microsoft Access里的过滤表单生成的查询使用expr = NULL来测试空值。打开这个选项，可以使用该接口来访问数据库。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。
* off表示不控制，即expr = NULL总是返回NULL（未知）。

**默认值：**off

fig: **说明：**   
新用户经常在涉及NULL的表达式上语义混淆，故默认值设为off。

**support\_extended\_features**

**参数说明：**控制是否支持数据库的扩展特性。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示支持数据库的扩展特性。
* off表示不支持数据库的扩展特性。

**默认值：**off

**sql\_compatibility**

**参数说明：**控制数据库的SQL语法和语句行为同哪一个主流数据库兼容。

该参数属于INTERNAL类型参数，用户无法修改，只能查看。

**取值范围：**枚举型

* A表示同O数据库兼容。
* B表示同MY数据库兼容。
* C表示同TD数据库兼容。
* PG表示同POSTGRES数据库兼容。

**默认值：**A

fig: **须知：**

* 该参数只能在执行SQL语法参考->SQL语法->CREATE DATABASE命令创建数据库的时候设置。
* 在数据库中，该参数只能是确定的一个值，要么始终设置为A，要么始终设置为B，请勿任意改动，否则会导致数据库行为不一致。

**behavior\_compat\_options**

**参数说明：**数据库兼容性行为配置项，该参数的值由若干个配置项用逗号隔开构成。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**reduce\_tailing\_zero

fig: **说明：**

* 当前只支持“表1兼容性配置项”中的内容
* 配置多个兼容性配置项时，相邻配置项用逗号隔开，例如：set behavior\_compat\_options='end\_month\_calculate,display\_leading\_zero';

**表 1** 兼容性配置项

|  |  |
| --- | --- |
| **兼容性配置项** | **兼容性行为控制** |
| display\_leading\_zero | 浮点数显示配置项。 不设置此配置项时，对于-1~0和0~1之间的小数，不显示小数点前的0。比如，0.25显示为.25。 设置此配置项时，对于-1~0和0~1之间的小数，显示小数点前的0。比如，0.25显示为0.25。 |
| end\_month\_calculate | add\_months函数计算逻辑配置项。  假定函数add\_months的两个参数分别为param1和param2，param1的月份和param2的和为月份result。  不设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期小，计算结果中的日期字段（Day字段）和param1的日期字段保持一致。比如：   |  | | --- | | postgres=# select add\_months('2018-02-28',3) from sys\_dummy;  add\_months  ---------------------  2018-05-28 00:00:00  (1 row) |   设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期比小，计算结果中的日期字段（Day字段）和result的月末日期保持一致。比如：   |  | | --- | | postgres=# select add\_months('2018-02-28',3) from sys\_dummy;  add\_months  ---------------------  2018-05-28 00:00:00  (1 row) | |
| compat\_analyze\_sample | analyze采样行为配置项。  设置此配置项时，会优化analyze的采样行为，主要体现在analyze时全局采样会更精确的控制在3万条左右，更好的控制analyze时DBnode端的内存消耗，保证analyze性能的稳定性。 |
| bind\_schema\_tablespace | 绑定模式与同名表空间配置项。  如果存在与模式名sche\_name相同的表空间名，那么如果设置search\_path为sche\_name， default\_tablespace也会同步切换到sche\_name。 |
| bind\_procedure\_searchpath | 未指定模式名的数据库对象的搜索路径配置项。  在存储过程中如果不显示指定模式名，会优先在存储过程所属的模式下搜索。  如果找不到，则有两种情况：   * 若不设置此参数，报错退出。 * 若设置此参数，按照search\_path中指定的顺序继续搜索。如果还是找不到，报错退出。 |
| correct\_to\_number | 控制to\_number()结果兼容性的配置项。  若设置此配置项，则to\_number()函数结果与pg11保持一致，否则默认与O db保持一致。 |
| unbind\_dive\_bound | 控制对整数除法的结果进行范围校验。  若设置此配置项，则不需要对除法结果做范围校验，例如，INT\_MIN/(-1)可以得到输出结果为INT\_MAX+1，反之，则会因为超过结果大于INT\_MAX而报越界错误。 |
| merge\_update\_multi | 控制merge into匹配多行时是否进行update操作。  若设置此配置项，匹配多行时update不报错，否则默认与a db保持一致，报错。 |
| return\_null\_string | 控制函数lpad()和rpad()结果为空字符串”的显示配置项。  不设置此配置项时，空字符串显示为NULL。   |  | | --- | | postgres=# select length(lpad('123',0,'\*')) from sys\_dummy;  length  --------  (1 row) |   设置此配置项时，空字符串显示为”。   |  | | --- | | postgres=# select length(lpad('123',0,'\*')) from sys\_dummy;  length  --------  0  (1 row) | |
| compat\_concat\_variadic | 控制函数concat()和concat\_ws()对variadic类型结果兼容性的配置项。  若设置此配置项，当concat函数参数为variadic类型时，保留a db和Teradata兼容模式下不同的结果形式；否则默认a db和Teradata兼容模式下结果相同，且与a db保持一致。由于MY无variadic类型，所以该选项对MY无影响。 |
| merge\_update\_multi | 控制在使用MERGE INTO … WHEN MATCHED THEN UPDATE（参考[MERGE INTO](https://docs.vastdata.com.cn/zh/docs/VastbaseG100Ver2.2.5/doc/%E5%BC%80%E5%8F%91%E8%80%85%E6%8C%87%E5%8D%97/MERGE-INTO.html)）和INSERT … ON DUPLICATE KEY UPDATE（参考[INSERT](https://docs.vastdata.com.cn/zh/docs/VastbaseG100Ver2.2.5/doc/%E5%BC%80%E5%8F%91%E8%80%85%E6%8C%87%E5%8D%97/INSERT.html)）时，当目标表中一条目标数据与多条源数据冲突时UPDATE行为。  若设置此配置项，当存在上述场景时，该冲突行将会多次执行UPDATE；否则（默认）报错，即MERGE或INSERT操作失败。 |
| hide\_tailing\_zero | numeric显示配置项。不设置此项时，numeric按照指定精度显示。设置此项时，隐藏小数点后的末尾0。   |  | | --- | | set behavior\_compat\_options='hide\_tailing\_zero';  select cast(123.123 as numeric(15,10));  numeric  ---------  123.123 | |
| rownum\_type\_compat | 控制ROWNUM的类型，ROWNUM默认类型为INT8，设置此参数后，ROWNUM类型变更为NUMERIC类型。 |
| aformat\_null\_test | 控制rowtype类型判空逻辑,设置此项时，对于rowtype is not null判断，当一行数据有一列不为空的时候返回ture。  否则，对于rowtype is not null判断，当一行数据所有列不为空的时候返回ture. |
| aformat\_regexp\_match | 控制正则表达式函数的匹配行为。  设置此项，且sql\_compatibility参数的值为A或B时，正则表达式的 flags 参数支持的选项含义有变更：   * 默认不能匹配 'n' 字符。 * flags 中包含n选项时， 能够匹配 'n' 字符。 * regexp\_replace(source, pattern replacement) 函数替换所有匹配的子串。 * regexp\_replace(source, pattern, replacement, flags) 在 flags值为” 或者null时，返回值为null。   否则，正则表达式的 flags 参数支持的选项含义：   * 默认能匹配 'n' 字符。 * flags 中的 n 选项表示按照多行模式匹配。 * regexp\_replace(source, pattern replacement) 函数仅替换第一个匹配到的子串。 * regexp\_replace(source, pattern, replacement, flags) 在 flags值为” 或者null时，返回值为替换后的字符串。 |
| compat\_cursor | 控制隐式游标状态兼容行为。设置此项，且兼容O，隐式游标状态（SQL%FOUND、SQL%NOTFOUND、SQL%ISOPNE、SQL%ROWCOUNT）由原先的仅在当前执行的函数有效，拓展到包括本函数调用的子函数有效。 |
| proc\_outparam\_override | 控制存储过程出参的重载行为，打开该参数后，对于存储过程只有out出参部分不同的情况下，也可以正常调用。 |
| proc\_implicit\_for\_loop\_variable | 控制存储过程中FOR\_LOOP查询语句行为设置此项时，在FOR rec IN query LOOP语句中，若rec已经定义，不会复用已经定义的rec变量，而且重新建立一个新的变量。否则，会复用已经定义的rec变量，不会建立新的变量。 |
| allow\_procedure\_compile\_check | 控制存储过程中select语句和open cursor语句的编译检查设置此项时，在存储过程中执行select语句、open cursor for语句、cursor%rowtype语句、for rec in语句时，若查询的表不存在，则无法创建创建存储过程，不支持trigger函数的编译检查，若查询的表存在，则成功创建存储过程。 |
| char\_coerce\_compat | 控制char(n)类型向其它变长字符串类型转换时的行为。默认情况下char(n)类型转换其它变长字符串类型时会省略尾部的空格，开启该参数后，转换时不再省略尾部的空格，并且在转换时如果char(n)类型的长度超过其它变长字符串类型时将会报错。该参数仅在sql\_compatibility参数的值为A时生效。 |
| pgformat\_substr | 控制substr(str, from, for)在不同场景下的表现。默认情况下，当from小于0时，substr将从字符串尾部开始计数；当for小于1时，substr将返回NULL。开启该参数后，当from小于0时，将从字符串的第一位的前(-from + 1)位开始计数；当for小于0时，substr将报错。该参数仅在sql\_compatibility参数的值为PG时生效。 |

**plpgsql.variable\_conflict**

**参数说明**：设置同名的存储过程变量和表的列的使用优先级。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围**：字符串

* error表示遇到存储过程变量和表的列名同名则编译报错。
* use\_variable表示存储过程变量和表的列名同名则优先使用变量。
* use\_column表示存储过程变量和表的列名同名则优先使用列名。

**默认值**：空

**td\_compatible\_truncation**

**参数说明：**控制是否开启与Teradata数据库相应兼容的特征。该参数在用户连接上与TD兼容的数据库时，可以将参数设置成为on（即超长字符串自动截断功能启用），该功能启用后，在后续的insert语句中，对目标表中char和varchar类型的列插入超长字符串时，会按照目标表中相应列定义的最大长度对超长字符串进行自动截断。保证数据都能插入目标表中，而不是报错。

fig: **说明：**  
超长字符串自动截断功能不适用于insert语句包含外表的场景。如果向字符集为字节类型编码（SQL\_ASCII、LATIN1等）的数据库中插入多字节字符数据（如汉字等)，且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示启动超长字符串自动截断功能。
* off表示停止超长字符串自动截断功能。

**默认值：**off

**lastval\_supported**

**参数说明：**控制是否可以使用lastval函数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示支持lastval函数，同时nextval函数不支持下推。
* off表示不支持lastval函数，同时nextval函数可以下推。

**默认值：**off

## 容错性

当数据库系统发生错误时，以下参数控制服务器处理错误的方式。

**exit\_on\_error**

**参数说明：**打开该开关，ERROR级别报错会升级为PANIC报错，从而可以产生core堆栈。主要用于问题定位和业务测试。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示ERROR级别报错会升级为PANIC报错。
* off表示不会对ERROR级别报错进行升级。

**默认值：**off

**restart\_after\_crash**

**参数说明：**设置为on，后端进程崩溃时，Vastbase将自动重新初始化此后端进程。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示能够最大限度地提高数据库的可用性。
* 在某些情况（比如当采用管理工具（例如xCAT）管理Vastbase时），能够最大限度地提高数据库的可用性。
* off表示能够使得管理工具在后端进程崩溃时获取控制权并采取适当的措施进行处理。

**默认值：**on

**omit\_encoding\_error**

**参数说明：**设置为on，数据库的客户端字符集编码为UTF-8时，出现的字符编码转换错误将打印在日志中，有转换错误的被转换字符会被忽略，以“?”代替。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示有转换错误的字符将被忽略，以“?”代替，打印错误信息到日志中。
* off表示有转换错误的字符不能被转换，打印错误信息到终端。

**默认值：**off

**max\_query\_retry\_times**

**参数说明：**指定SQL语句出错自动重试功能的最大重跑次数（目前支持重跑的错误类型为“Connection reset by peer”、“Lock wait timeout”和“Connection timed out”等），设定为0时关闭重跑功能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，0~20。

**默认值：**0

**cn\_send\_buffer\_size**

**参数说明：**指定数据库主节点发送数据缓存区的大小。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**整型，8~128，单位为KB。

**默认值：**8KB

**max\_cn\_temp\_file\_size**

**参数说明：**指定SQL语句出错自动重试功能中数据库主节点端使用临时文件的最大值，设定为0表示不使用临时文件。

该参数属于SIGHUP类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整型，0~10485760， 单位为KB。

**默认值：**空

**retry\_ecode\_list**

**参数说明：**指定SQL语句出错自动重试功能支持的错误类型列表。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**字符串。

**默认值：**YY001 YY002 YY003 YY004 YY005 YY006 YY007 YY008 YY009 YY010 YY011 YY012 YY013 YY014 YY015 53200 08006 08000 57P01 XX003 XX009 YY016

**data\_sync\_retry**

**参数说明：**控制当fsync到磁盘失败后是否继续运行数据库。由于在某些操作系统的场景下，fsync失败后重试阶段即使再次fsync失败也不会报错，从而导致数据丢失。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示当fsync同步到磁盘失败后采取重试机制，数据库继续运行。
* off表示当fsync同步到磁盘失败后直接报panic，停止数据库。

**默认值：**off

**remote\_read\_mode**

**参数说明**：远程读功能开关。读取主机上的页面失败时可以从备机上读取对应的页面。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：枚举类型

* off表示关闭远程读功能。
* non\_authentication表示开启远程读功能，但不进行证书认证。
* authentication表示开启远程读功能，但要进行证书认证。

**默认值**：authentication

## 连接池参数

当使用连接池访问数据库时，在系统运行过程中，数据库连接是被当作对象存储在内存中的，当用户需要访问数据库时，并非建立一个新的连接，而是从连接池中取出一个已建立的空闲连接来使用。用户使用完毕后，数据库并非将连接关闭，而是将连接放回连接池中，以供下一个请求访问使用。

**pooler\_maximum\_idle\_time**

**参数说明：**Pooler链接自动清理功能使用，当链接池中链接空闲时间超过所设置值时，会触发自动清理机制，清理各节点的空闲链接数到minimum\_pool\_size。

fig: **说明：**此参数在该版本不生效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为INT\_MAX，最小单位为分钟

**默认值：**1h（即60min）

**minimum\_pool\_size**

**参数说明：**Pooler链接自动清理功能使用，自动清理后各pooler链接池对应节点的链接数最小剩余量，当参数设置为0时，可以关闭pooler链接自动清理功能。

fig: **说明：**此参数在该版本不生效。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表2中对应设置方法进行设置。

**取值范围：**整型，最小值为1，最大值为65535

**默认值：**200

**cache\_connection**

**参数说明：**是否回收连接池的连接。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表2对应设置方法进行设置。

**取值范围：**布尔型

* on表示回收连接池的连接。
* off表示不回收连接池的连接。

**默认值：**on

## 事务

介绍Vastbase事务隔离、事务只读、最大prepared事务数、维护模式目的参数设置及取值范围等内容。

**transaction\_isolation**

**参数说明：**设置当前事务的隔离级别。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**字符串，只识别以下字符串，大小写空格敏感：

* serializable：Vastbase中等价于REPEATABLE READ。
* read committed：只能读取已提交的事务的数据（缺省），不能读取到未提交的数据。
* repeatable read：仅能读取事务开始之前提交的数据，不能读取未提交的数据以及在事务执行期间由其它并发事务提交的修改。
* default：设置为default\_transaction\_isolation所设隔离级别。

**默认值：**read committed

**transaction\_read\_only**

**参数说明：**设置当前事务是只读事务。

该参数在数据库恢复过程中或者在备机里，固定为on；否则，固定为default\_transaction\_read\_only的值。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on表示设置当前事务为只读事务。
* off表示该事务可以是非只读事务。

**默认值：**off

**xc\_maintenance\_mode**

**参数说明：**设置系统进入维护模式。

该参数属于SUSET类型参数，仅支持：配置运行参数->重设参数章节中表1中的方式三进行设置。

**取值范围：**布尔型

* on表示该功能启用。
* off表示该功能被禁用。

fig: **须知：**

谨慎打开这个开关，避免引起Vastbase数据不一致。

**默认值：**off

**allow\_concurrent\_tuple\_update**

**参数说明：**设置是否允许并发更新。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示该功能启用。
* off表示该功能被禁用。

**默认值：**on

**transaction\_deferrable**

**参数说明：**指定是否允许一个只读串行事务延迟执行，使其不会执行失败。该参数设置为on时，当一个只读事务发现读取的元组正在被其他事务修改，则延迟该只读事务直到其他事务修改完成。该参数为预留参数，该版本不生效。与该参数类似的还有一个请参考：GUC参数说明->客户端连接缺省设置->语句行为中default\_transaction\_deferrable参数，设置它来指定一个事务是否允许延迟。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on表示允许执行。
* off表示不允许执行。

**默认值：**off

**enable\_show\_any\_tuples**

**参数说明：**该参数只有在只读事务中可用，用于分析。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示表中元组的所有版本都会可见。
* off/false表示表中元组的所有版本都不可见。

**默认值：**off

**replication\_type**

**参数说明：**标记当前HA模式是单主机模式、主备从模式还是一主多备模式。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

该参数用户不能自己去设置参数值。

**取值范围：**0~2

* 2表示单主机模式，此模式无法扩展备机。
* 1表示使用一主多备模式，全场景覆盖，推荐使用。
* 0表示主备从模式，目前此模式暂不支持。

**默认值：**1

**pgxc\_node\_name**

**参数说明：**指定节点名称。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

在备机请求主机进行日志复制时，如果application\_name参数没有设置，那么该参数会被用来作为备机在主机上的流复制槽名字。该流复制槽的命名方式为 "该参数值*备机ip*备机port"。其中，备机ip和备机port取自replconninfo参数中指定的备机ip和端口号。该流复制槽最大长度为61个字符，如果拼接后的字符串超过该长度，则会使用截断后的pgxc\_node\_name进行拼接，以保证流复制槽名字长度小于等于61个字符。

fig: **注意：**

此参数修改后会导致连接数据库实例失败，不建议进行修改。

**取值范围：**字符串

**默认值：**当前节点名称

**enable\_defer\_calculate\_snapshot**

**参数说明**：延迟计算快照的xmin和oldestxmin，执行1000个事务或者间隔1s才触发计算，设置为on时可以在高负载场景下减少计算快照的开销，但是会导致oldestxmin推进较慢，影响垃圾元组回收，设置为off时xmin和oldestxmin可以实时推进，但是会增加计算快照时的开销。

该参数属于SIGHUP类型参数，改请参考：配置运行参数->重设参数章节中表2进行设置

**取值范围：**布尔型。

* on表示延迟计算快照xmin和oldestxmin。
* off表示实时计算快照xmin和oldestxmin。

**默认值：**on

## 双数据库实例复制参数

**RepOriginId**

**参数说明：**该参数是一个会话级别的GUC参数，在双向逻辑复制的场景下，为避免数据循环复制，需要设置为一个非0的值。

该参数属于USERSET类型参数，仅支持：配置运行参数->重设参数章节中表1中的方式三进行设置。

**取值范围：**整型，0~2147483647

**默认值：**0

## 开发人员选项

**allow\_system\_table\_mods**

**参数说明：**设置是否允许普通用户修改系统表，无法限制超级用户对系统表的修改。因修改系统表可能导致数据库运行异常或无法启动等故障问题，请谨慎操作。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许修改系统表的结构。
* off表示不允许修改系统表的结构。

**默认值：**off

fig: **注意：** 不建议修改该参数默认值，若设置为on，可能导致系统表损坏，甚至数据库无法启动。

**debug\_assertions**

**参数说明：**控制打开各种断言检查。能够协助调试，当遇到奇怪的问题或者崩溃，请把此参数打开，因为它能暴露编程的错误。要使用这个参数，必须在编译Vastbase的时候定义宏USE\_ASSERT\_CHECKING（通过configure选项 --enable-cassert完成）。

该参数属于USERSET类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on表示打开断言检查。
* off表示不打开断言检查。

fig: **说明：**当启用断言选项编译Vastbase时，debug\_assertions缺省值为on 。

**默认值：**off

**ignore\_checksum\_failure**

**参数说明：**设置读取数据时是否忽略校验信息检查失败（但仍然会告警），继续执行可能导致崩溃，传播或隐藏损坏数据，无法从远程节点恢复数据及其他严重问题。不建议用户修改设置。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示忽略数据校验错误。
* off表示数据校验错误正常报错。

**默认值：**off

**ignore\_system\_indexes**

**参数说明：**读取系统表时忽略系统索引（但是修改系统表时依然同时修改索引）。

该参数属于BACKEND类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

fig:**须知：**这个参数在从系统索引被破坏的表中恢复数据的时候非常有用。

**取值范围：**布尔型

* on表示忽略系统索引。
* off表示不忽略系统索引。

**默认值：**off

**post\_auth\_delay**

**参数说明：**在认证成功后，延迟指定时间，启动服务器连接。允许调试器附加到启动进程上。

该参数属于BACKEND类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为0，最大值为2147，单位为秒。

**默认值：**0

fig: **说明：**   
此参数只用于调试和问题定位，为避免影响正常业务运行，生产环境下请确保参数值为默认值0。参数设置为非0时可能会因认证延迟时间过长导致数据库实例状态异常。

**pre\_auth\_delay**

**参数说明：**启动服务器连接后，延迟指定时间，进行认证。允许调试器附加到认证过程上。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，最小值为0～60，单位为秒。

**默认值：**0

fig: **说明：**   
此参数只用于调试和问题定位，为避免影响正常业务运行，生产环境下请确保参数值为默认值0。参数设置为非0时可能会因认证延迟时间过长导致数据库实例状态异常。

**trace\_notify**

**参数说明：**为LISTEN和NOTIFY命令生成大量调试输出。GUC参数说明->错误报告和日志->记录日志的时间中client\_min\_messages、log\_min\_messages参数级别必须是DEBUG1或者更低时，才能把这些输出分别发送到客户端或者服务器日志。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示打开输出功能。
* off表示关闭输出功能。

**默认值：**off

**trace\_recovery\_messages**

**参数说明：**启用恢复相关调试输出的日志录，否则将不会被记录。该参数允许覆盖正常设置GUC参数说明->错误报告和日志->记录日志的时间中的log\_min\_messages参数，但是仅限于特定的消息，这是为了在调试备机中使用。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、log，取值的详细信息请参见：GUC参数说明->错误报告和日志->记录日志的时间中log\_min\_messages参数。

**默认值：**log

fig: **说明：**

* 默认值log表示不影响记录决策。
* 除默认值外，其他值会导致优先级更高的恢复相关调试信息被记录，因为它们有log优先权。对于常见的log\_min\_messages设置，这会导致无条件地将它们记录到服务器日志上。

**trace\_sort**

**参数说明：**控制是否在日志中打印排序操作中的资源使用相关信息。这个选项只有在编译Vastbase的时候定义了TRACE\_SORT宏的时候才可用，不过目前TRACE\_SORT是由缺省定义的。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示打开控制功能。
* off表示关闭控制功能。

**默认值：**off

**zero\_damaged\_pages**

**参数说明：**控制检测导致Vastbase报告错误的损坏的页头，终止当前事务。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

设置为on时，会导致系统报告一个警告，把损坏的页面填充为零然后继续处理。这种行为会破坏数据，也就是所有在已经损坏页面上的行记录。但是它允许绕开坏页面然后从表中尚存的未损坏页面上继续检索数据行。因此它在因为硬件或者软件错误导致的崩溃中进行恢复是很有用的。通常不应该把它设置为on，除非不需要从崩溃的页面中恢复数据。

**默认值：**off

**remotetype**

**参数说明**：设置远程连接类型。

该参数不支持修改。

**取值范围**：枚举类型，有效值有application、datanode、internaltool。

**默认值**：application

**max\_user\_defined\_exception**

**参数说明**：异常最大个数，默认值不可更改。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，当前只能取固定值1000

**默认值**：1000

**enable\_fast\_numeric**

**参数说明：**标识是否开启Numeric类型数据运算优化。Numeric数据运算是较为耗时的操作之一，通过将Numeric转化为int64/int128类型，提高Numeric运算的性能。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示开启Numeric优化。
* off/false表示关闭Numeric优化。

**默认值：**on

**enable\_compress\_spill**

**参数说明：**标识是否开启下盘压缩功能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示开启下盘优化。
* off/false表示关闭下盘优化。

**默认值：**on

**resource\_track\_log**

**参数说明**：控制自诊断的日志级别。目前仅对多列统计信息进行控制。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

* summary：显示简略的诊断信息。
* detail：显示详细的诊断信息。

目前这两个参数值只在显示多列统计信息未收集的告警的情况下有差别，summary不显示未收集多列统计信息的告警，detail会显示这类告警。

**默认值：**summary

**show\_acce\_estimate\_detail**

**参数说明**：评估信息一般用于运维人员在维护工作中使用，因此该参数默认关闭，此外为了避免这些信息干扰正常的explain信息显示，只有在explain命令的verbose选项打开的情况下才显示评估信息

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示可以在explain命令的输出中显示评估信息。
* off表示不在explain命令的输出中显示评估信息。

**默认值**：off

fig: **说明：**当前版本不支持加速数据库实例，因此该参数设置后不生效。

**support\_batch\_bind**

**参数说明**：控制是否允许通过JDBC、ODBC、Libpq等接口批量绑定和执行PBE形式的语句。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示使用批量绑定和执行。
* off表示不使用批量绑定和执行。

**默认值**：on

**numa\_distribute\_mode**

**参数说明**：用于控制部分共享数据和线程在NUMA节点间分布的属性。用于大型多NUMA节点的ARM服务器性能调优，一般不用设置。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串，当前有效取值为'none'，'all'。

* 'none'：表示不启用本特性。
* 'all'：表示将部分共享数据和线程分布到不同的NUMA节点下，减少远端访存次数，提高性能。目前仅适用于拥有多个NUMA节点的ARM服务器，并且要求全部NUMA节点都可用于数据库进程，不支持仅选择一部分NUMA节点。

fig: **说明：**当前版本x86平台下不支持numa\_distribute\_mode设置为all。

**默认值**：'none'

**log\_pagewriter**

**参数说明**：设置用于增量检查点打开后，显示线程的刷页信息以及增量检查点的详细信息，信息比较多，不建议设置为true。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

**默认值**：off

**advance\_xlog\_file\_num**

**参数说明**：用于控制在后台周期性地提前初始化xlog文件的数目。该参数是为了避免事务提交时执行xlog文件初始化影响性能，但仅在超重负载时才可能出现，因此一般不用配置。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~1000000（0表示不提前初始化）。例如，取值10，表示后台线程会周期性地根据当前xlog写入位置提前初始化10个xlog文件。

**默认值**：0

**enable\_beta\_opfusion**

**参数说明**：在enable\_opfusion参数打开的状态下，如果开启该参数，可以支持TPCC中出现的聚集函数，排序两类SQL语句的加速执行，提升SQL执行性能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启。
* off表示不开启。

**默认值**：off

**string\_hash\_compatible**

**参数说明：**该参数用来说明char类型和varchar/text类型的hash值计算方式是否相同，以此来判断进行分布列从char类型到相同值的varchar/text类型转换，数据分布变化时，是否需要进行重分布。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示计算方式相同，不需要进行重分布。
* off表示计算方式不同，需要进行重分布。

fig: **说明：**   
计算方式的不同主要体现在字符串计算hash值时传入的字节长度上。（如果为char，则会忽略字符串后面空格的长度，如果为text或varchar，则会保留字符串后面空格的长度。）hash值的计算会影响到查询的计算结果，因此此参数一旦设置后，在整个数据库使用过程中不能再对其进行修改，以避免查询错误。

**默认值：**off

**pldebugger\_timeout**

**参数说明：**该参数用来控制pldebugger server端等待debug端响应的超时时间。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1 ~ 86400，单位为秒。

**默认值**：15min

**plsql\_show\_all\_error**

**参数说明：**该参数用来控制编译PLPGSQL对象时是否支持跳过报错继续编译，具体影响请参考：schema->Information-Schema->DBE\_PLDEVELOPER内的说明。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

**默认值**：off

## 审计

### 审计开关

**audit\_enabled**

**参数说明：**控制审计进程的开启和关闭。审计进程开启后，将从管道读取后台进程写入的审计信息，并写入审计文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示启动审计功能。
* off表示关闭审计功能。

**默认值：**off

**audit\_directory**

**参数说明：**审计文件的存储目录。一个相对于数据目录data的路径，可自行指定。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**pg\_audit。如果使用om工具部署Vastbase，则审计日志路径为“$GAUSSLOG/pg\_audit/实例名称”。

**audit\_data\_format**

**参数说明：**审计日志文件的格式。当前仅支持二进制格式。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**binary

**audit\_rotation\_interval**

**参数说明**：指定创建一个新审计日志文件的时间间隔。当现在的时间减去上次创建一个审计日志的时间超过了此参数值时，服务器将生成一个新的审计日志文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1~INT\_MAX/60，单位为min。

**默认值**：1d

fig: **须知：**   
请不要随意调整此参数，否则可能会导致audit\_resource\_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用[audit\_resource\_policy](#X6918d4c02531ccf74987af4f8f4cb8ec1dc1ddd)、[audit\_space\_limit](#X1296535e0d6b3f97de3d122e04f51813700acce)和[audit\_file\_remain\_time](#X0069dc4be076f2efc787d952d514f0aea1b6a95)参数进行控制。

**audit\_rotation\_size**

**参数说明**：指定审计日志文件的最大容量。当审计日志消息的总量超过此参数值时，服务器将生成一个新的审计日志文件。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1024~1048576，单位为KB。

**默认值**：10MB

fig: **须知：**   
请不要随意调整此参数，否则可能会导致audit\_resource\_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用audit\_resource\_policy、audit\_space\_limit和audit\_file\_remain\_time参数进行控制。

**audit\_resource\_policy**

**参数说明**：控制审计日志的保存策略，以空间还是时间限制为优先策略。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示采用空间优先策略，最多存储[audit\_space\_limit](#X1296535e0d6b3f97de3d122e04f51813700acce)大小的日志。
* off表示采用时间优先策略，最少存储[audit\_file\_remain\_time](#X0069dc4be076f2efc787d952d514f0aea1b6a95)长度时间的日志。

**默认值：**on

**audit\_file\_remain\_time**

**参数说明**：表示需记录审计日志的最短时间要求，该参数在[audit\_resource\_policy](#X6918d4c02531ccf74987af4f8f4cb8ec1dc1ddd)为off时生效。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~730，单位为day，0表示无时间限制。

**默认值**：90

**audit\_space\_limit**

**参数说明：**审计文件占用的磁盘空间总量。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1024KB~1024GB，单位为KB。

**默认值：**1GB

**audit\_file\_remain\_threshold**

**参数说明：**审计目录下审计文件个数的最大值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~1048576

**默认值：**1048576

fig: **须知：**   
请尽量保证此参数为1048576，并不要随意调整此参数，否则可能会导致audit\_resource\_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用audit\_resource\_policy、audit\_space\_limit和audit\_file\_remain\_time参数进行控制。

**audit\_thread\_num**

**参数说明：**审计线程的个数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~48

**默认值：**1

fig: **须知：**  
当audit\_dml\_state开关打开且在高性能场景下，建议增大此参数保证审计消息可以被及时处理和记录。

### 用户和权限审计

**audit\_login\_logout**

**参数说明：**这个参数决定是否审计Vastbase用户的登录（包括登录成功和登录失败）、注销。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~7。

* 0表示关闭用户登录、注销审计功能。
* 1表示只审计用户登录成功。
* 2表示只审计用户登录失败。
* 3表示只审计用户登录成功和失败。
* 4表示只审计用户注销。
* 5表示只审计用户注销和登录成功。
* 6表示只审计用户注销和登录失败。
* 7表示审计用户登录成功、失败和注销。

**默认值：**7

**audit\_database\_process**

**参数说明：**该参数决定是否对Vastbase的启动、停止、切换和恢复进行审计。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭Vastbase启动、停止、恢复和切换审计功能。
* 1表示开启Vastbase启动、停止、恢复和切换审计功能。

**默认值：**1

fig: **说明：**

Vastbase启动时DN执行备升主流程，因此DN启动时审计日志中类型为system\_switch。

**audit\_user\_locked**

**参数说明：**该参数决定是否审计Vastbase用户的锁定和解锁。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭用户锁定和解锁审计功能。
* 1表示开启审计用户锁定和解锁功能。

**默认值：**1

**audit\_user\_violation**

**参数说明：**该参数决定是否审计用户的越权访问操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭用户越权操作审计功能。
* 1表示开启用户越权操作审计功能。

**默认值：**0

**audit\_grant\_revoke**

**参数说明：**该参数决定是否审计Vastbase用户权限授予和回收的操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭审计用户权限授予和回收功能。
* 1表示开启审计用户权限授予和回收功能。

**默认值：**1

### 操作审计

**audit\_system\_object**

**参数说明：**该参数决定是否对Vastbase数据库对象的CREATE、DROP、ALTER操作进行审计。Vastbase数据库对象包括DATABASE、USER、schema、TABLE等。通过修改该配置参数的值，可以只审计需要的数据库对象的操作。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0～134217727

* 0代表关闭Vastbase数据库对象的CREATE、DROP、ALTER操作审计功能。
* 非0代表只审计Vastbase的某类或者某些数据库对象的CREATE、DROP、ALTER操作。

**取值说明**：

该参数的值由26个二进制位的组合求出，这26个二进制位分别代表Vastbase的26类数据库对象。如果对应的二进制位取值为0，表示不审计对应的数据库对象的CREATE、DROP、ALTER操作；取值为1，表示审计对应的数据库对象的CREATE、DROP、ALTER操作。这26个二进制位代表的具体审计内容请参见[表1](#X973352317830db4ab0555449a1bb69189bce140)。

**默认值：**12295

**表 1** audit\_system\_object取值含义说明

| **二进制位** | **含义** | **取值说明** |
| --- | --- | --- |
| 第0位 | 是否审计DATABASE对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第1位 | 是否审计SCHEMA对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第2位 | 是否审计USER对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第3位 | 是否审计TABLE对象的CREATE、DROP、ALTER、TRUNCATE操作。 | 0表示不审计该对象的CREATE、DROP、ALTER、TRUNCATE操作；1表示审计该对象的CREATE、DROP、ALTER、TRUNCATE操作。 |
| 第4位 | 是否审计INDEX对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第5位 | 是否审计VIEW/MATVIEW对象的CREATE、DROP操作。 | 0表示不审计该对象的CREATE、DROP操作；1表示审计该对象的CREATE、DROP操作。 |
| 第6位 | 是否审计TRIGGER对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第7位 | 是否审计PROCEDURE/FUNCTION对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第8位 | 是否审计TABLESPACE对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第9位 | 是否审计RESOURCE POOL对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作 |
| 第10位 | 是否审计WORKLOAD对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作 |
| 第11位 | 保留 |  |
| 第12位 | 是否审计DATA SOURCE对象的CRAETE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第13位 | 保留 | - |
| 第14位 | 是否审计ROW LEVEL SECURITY对象的CREATE、DROP、ALTER操作。 | 0表示不审计该对象的CREATE、DROP、ALTER操作；1表示审计该对象的CREATE、DROP、ALTER操作。 |
| 第15位 | 是否审计TYPE对象的CREATE、DROP、ALTER操作。 | 0表示不审计TYPE对象的CREATE、DROP、ALTER操作；1表示审计TYPE对象的CREATE、DROP、ALTER操作。 |
| 第16位 | 是否审计TEXT SEARCH对象（CONFIGURATION和DICTIONARY）的CREATE、DROP、ALTER操作。 | 0表示不审计TEXT SEARCH对象的CREATE、DROP、ALTER操作；1表示审计TEXT SEARCH对象的CREATE、DROP、ALTER操作。 |
| 第17位 | 是否审计DIRECTORY对象的CREATE、DROP、ALTER操作。 | 0表示不审计DIRECTORY对象的CREATE、DROP、ALTER操作；1表示审计DIRECTORY对象的CREATE、DROP、ALTER操作。 |
| 第18位 | 是否审计SYNONYM对象的CREATE、DROP、ALTER操作。 | 0表示不审计SYNONYM对象的CREATE、DROP、ALTER操作；1表示审计SYNONYM对象的CREATE、DROP、ALTER操作。 |
| 第19位 | 是否审计SEQUENCE对象的CREATE、DROP、ALTER操作。 | 0表示不审计SEQUENCE对象的CREATE、DROP、ALTER操作；1表示审计SEQUENCE对象的CREATE、DROP、ALTER操作。 |
| 第20位 | 是否审计CMK、CEK对象的CREATE、DROP操作。 | 0表示不审计CMK、CEK对象的CREATE、DROP操作；1表示审计CMK、CEK对象的CREATE、DROP操作。 |
| 第21位 | 是否审计PACKAGE对象的CREATE、DROP、ALTER操作。 | 0表示不审计PACKAGE对象的CREATE、DROP、ALTER操作；1表示审计PACKAGE对象的CREATE、DROP、ALTER操作。 |
| 第22位 | 是否审计MODEL对象的CREATE、DROP操作。 | 0表示不审计MODEL对象的CREATE、ALTER操作；1表示审计MODEL对象的CREATE、DROP操作。 |
| 第23位 | 是否审计PUBLICATION和SUBSCRIPTION对象的CREATE、DROP、ALTER操作。 | 0表示不审计PUBLICATION和SUBSCRIPTION对象的CREATE、DROP、ALTER操作；1表示审计PUBLICATION和SUBSCRIPTION对象的CREATE、DROP、ALTER操作。 |
| 第24位 | 是否审计对gs\_global\_config全局对象的ALTER、DROP操作。 | 0表示不审计对系统表gs\_global\_config全局对象的ALTER、DROP操作；1表示审计对系统表gs\_global\_config全局对象的ALTER、DROP操作。 |
| 第25位 | 是否审计FOREIGN DATA WRAPPER对象的CREATE、DROP、ALTER操作。 | 0表示不审计FOREIGN DATA WRAPPER对象的CREATE、DROP、ALTER操作；1表示审计FOREIGN DATA WRAPPER对象的CREATE、DROP、ALTER操作。 |

**audit\_dml\_state**

**参数说明：**这个参数决定是否对具体表的INSERT、UPDATE、DELETE操作进行审计。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭具体表的DML操作（SELECT除外）审计功能。
* 1表示开启具体表的DML操作（SELECT除外）审计功能。

**默认值：**0

**audit\_dml\_state\_select**

**参数说明：**这个参数决定是否对SELECT操作进行审计。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭SELECT操作审计功能。
* 1表示开启SELECT审计操作功能。

**默认值：**0

**audit\_function\_exec**

参数说明：这个参数决定在执行存储过程、匿名块或自定义函数（不包括系统自带函数）时是否记录审计信息。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭过程或函数执行的审计功能。
* 1表示开启过程或函数执行的审计功能。

**默认值：**0

**audit\_copy\_exec**

**参数说明：**这个参数决定是否对COPY操作进行审计。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭COPY审计功能。
* 1表示开启COPY审计功能。

**默认值：**1

**audit\_set\_parameter**

**参数说明：**这个参数决定是否对SET操作进行审计。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭SET审计功能。
* 1表示开启SET审计功能。

**默认值：**1

**audit\_xid\_info**

**参数说明：**这个参数决定是否在审计日志字段detail\_info中记录SQL语句的事务ID。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0、1。

* 0表示关闭审计日志记录事务ID功能。
* 1表示开启审计日志记录事务ID功能。

**默认值：**0

fig: **须知：** 如果开启此开关，审计日志中detail\_info信息则以xid开始，例如：

detail\_info: xid=14619 , create table t1(id int);

对于不存在事务ID的审计行为，记录xid=NA。

**enable\_Separation\_Of\_Duty**

**参数说明：**是否开启三权分立选项。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启三权分立。
* off表示不开启三权分立。

**默认值：**off

**enable\_nonsysadmin\_execute\_direct**

**参数说明：**是否允许非系统管理员和非监控管理员执行EXECUTE DIRECT ON语句。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示允许任意用户执行EXECUTE DIRECT ON语句。
* off表示只允许系统管理员和监控管理员执行EXECUTE DIRECT ON语句。

**默认值：**off

**enable\_access\_server\_directory**

**参数说明：**是否开启非初始用户创建、修改和删除DIRECTORY的权限。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示开启非初始用户创建、修改和删除DIRECTORY的权限。
* off表示不开启非初始用户创建、修改和删除DIRECTORY的权限。

**默认值：**off

fig: **须知：**

* 出于安全考虑，默认情况下，只有初始用户才能够创建、修改和删除DIRECTORY对象。
* 如果开启了enable\_access\_server\_directory，具有SYSADMIN权限的用户和继承了内置角色gs\_role\_directory\_create权限的用户可以创建directory对象；具有SYSADMIN权限的用户、directory对象的属主、被授予了该directory的DROP权限的用户或者继承了内置角色gs\_role\_directory\_drop权限的用户可以删除directory对象；具有SYSADMIN权限的用户和directory对象的属主可以修改directory对象的所有者，且要求该用户是新属主的成员。

## 升级参数

**IsInplaceUpgrade**

**参数说明：**标示是否在升级的过程中。该参数用户无法修改。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在升级过程中。
* off表示不在升级过程中。

**默认值：**off

**inplace\_upgrade\_next\_system\_object\_oids**

**参数说明：**标示就地升级过程中，新增系统对象的OID。该参数用户无法修改。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**空

**upgrade\_mode**

**参数说明**：标示升级模式。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整数，0~INT\_MAX

* 0表示不在升级过程中。
* 1表示在就地升级过程中。
* 2表示在灰度升级过程中。

**默认值**：0

fig: **说明：**   
特殊情况：在使用灰度升级的情况下，若选择策略为大版本升级，即需要执行升级脚本和替换二进制包，会将upgrade\_mode设置为2，选择策略为小版本升级，只替换二进制包，则不会设置upgrade\_mode设置为2。

## 其它选项

**enable\_default\_ustore\_table**

**参数说明：**指定是否开启默认支持Ustore存储引擎。该参数为on时，创建的表类型都为Ustore表。

该参数属于USERSET类型，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**[off,on]

**默认值：**off

**reserve\_space\_for\_nullable\_atts**

**参数说明：**指定是否为Ustore表的可空属性预留空间。该参数为on时默认为Ustore表的可空属性预留空间。

该参数属于USERSET类型，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**[off,on]

**默认值：**off

**ustore\_attr**

**参数说明：**Ustore测试参数。

该参数属于USERSET类型，可以设置包括enable\_ustore\_partial\_seqscan（仅在ustore表中顺序扫描时复制选择性列）、enable\_candidate\_buf\_usage\_count（是否脏页淘汰加入使用次数权重）、ustats\_tracker\_naptime（重新加载统计文件所用的时间）、umax\_search\_length\_for\_prune（扩展表前要修剪的块数）、ustore\_unit\_test（开启Ustore白盒测试）。设置方法为 ustore\_attr='需要设置的参数'，例如需要设置enable\_ustore\_partial\_seqscan时，ustore\_attr='enable\_ustore\_partial\_seqscan=on'。

**取值范围：**空

**server\_version**

**参数说明：**报告服务器版本号（字符串形式）。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。该参数继承自PostgreSQL内核，表示当前数据库内核兼容PostgreSQL对应的server\_version版本，无实际含义，为保持北向对外工具接口的生态兼容性（工具连接时查询），保留该参数。该参数不推荐使用，如想查询服务器版本号，可通过函数Vastbase\_version()获取。

**取值范围：**字符串

**默认值：**9.2.4

**server\_version\_num**

**参数说明：**报告服务器版本号（整数形式）。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。该参数继承自PostgreSQL内核，表示当前数据库内核兼容PostgreSQL对应的server\_version\_num版本，无实际含义，为保持北向对外工具接口的生态兼容性（工具连接时查询），保留该参数。

**取值范围：**整数

**默认值：**90204

**block\_size**

**参数说明：**报告当前数据库所使用的块大小。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**8192

**默认值：**8192

**segment\_size**

**参数说明：**报告当前数据库所使用的段文件大小。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**单位**: 8KB

**默认值：**131072，即1GB

**max\_index\_keys**

**参数说明：**报告当前数据库能够支持的索引键值的最大数目。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**默认值：**32

**integer\_datetimes**

**参数说明：**报告是否支持64位整数形式的日期和时间格式。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on表示支持。
* off表示不支持。

**默认值：**on

**lc\_collate**

**参数说明：**报告当前数据库的字符串排序区域设置。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**默认值：**依赖于Vastbase安装部署时的配置。

**lc\_ctype**

**参数说明：**报告当前数据库的字母类别区域设置。如：哪些字符属于字母，它对应的大写形式是什么。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**默认值：**依赖于Vastbase安装部署时的配置。

**max\_identifier\_length**

**参数说明：**报告当前系统允许的标识符最大长度。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**整型

**默认值：**127

**server\_encoding**

**参数说明：**报告当前数据库的服务端编码字符集。

默认情况下，gs\_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**默认值：**在创建数据库的时候由当前系统环境决定的。

**enable\_upgrade\_merge\_lock\_mode**

**参数说明：**当该参数设置为on时，通过提升deltamerge内部实现的锁级别，避免和update/delete并发操作时的报错。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：** 布尔型

* on：表示提升deltamerge内部实现的锁级别，并发执行deltamerge和update/delete操作时，一个操作先执行，另一个操作被阻塞，在前一个操作完成后，后一个操作再执行。
* off：表示在对表的delta table的同一行并发执行deltamerge和update/delete操作时，后一个对同一行数据更新的操作会报错退出。

**默认值：**off

**transparent\_encrypted\_string**

**参数说明：**它存储的是透明加密的一个样本串，使用数据库加密密钥加密固定串“TRANS\_ENCRYPT\_SAMPLE\_STRING”后的密文，用来校验二次启动时获取的DEK是否正确。如果校验失败，那么数据库节点将拒绝启动。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。该参数当前版本只适用于DWS场景。

**取值范围：**字符串，设置为空表示Vastbase非加密。

**默认值：**空

fig: **说明：**请勿手动设置该参数，设置不当将导致Vastbase不可用。

**transparent\_encrypt\_kms\_url**

**参数说明：**它存储的是透明加密的数据库密钥获取地址，内容要求不可出现RFC3986标准外的字符，最大长度2047字节。格式为“kms://协议@KMS主机名1;KMS主机名2:KMS端口号/kms”，例如kms://https@linux175:29800/。该参数当前版本只适用于DWS场景。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**空

**transparent\_encrypt\_kms\_region**

**参数说明：**它存储的是Vastbase的部署区域，内容要求不可出现RFC3986标准外的字符，最大长度2047字节。该参数当前版本只适用于DWS场景。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

**默认值：**空

**basebackup\_timeout**

**参数说明：**备份传输完成后连接无读写的超时时间。

通过gs\_basebackup工具作传输时，如果指定较高压缩率时，可能在传输表空间完成后超时（客户端需要压缩传输数据）。

**取值范围：**整型，0 ~ INT\_MAX，单位为秒。其中0表示禁用该功能。

**默认值：**600s

**datanode\_heartbeat\_interval**

**参数说明**：设置心跳线程间心跳消息发送时间间隔，建议值不超过wal\_receiver\_timeout / 2。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，1000～60000，单位为毫秒。

**默认值**：1s

**max\_concurrent\_autonomous\_transactions**

**参数说明**：自治事务最大链接数，同一时间自治事务执行的最大并发数。当设置为0时，将无法执行自治事务。

该参数属于INTERNAL类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：0-1024

**默认值**：10

**sql\_ignore\_strategy**

**参数说明：**在B兼容模式下，该参数可控制ignore\_error的hint在违反非空约束时的处理策略。

该参数属于USERSET类型，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

* ignore\_null: 忽略违反非空约束的行的处理。
* overwrite\_null: 将违反约束的null值覆写为目标类型的默认值。

**默认值：**ignore\_null

## 等待事件

**enable\_instr\_track\_wait**

**参数说明：**是否开启等待事件信息实时收集功能。

在x86平台集中式部署下，硬件配置规格为32核CPU/256GB内存，使用Benchmark SQL 5.0工具测试性能，开关此参数性能影响约1.4%。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on：表示打开等待事件信息收集功能。
* off：表示关闭等待事件信息收集功能。

**默认值**：on

## Query

**instr\_unique\_sql\_count**

**参数说明：**控制系统中unique sql信息实时收集功能。配置为0表示不启用unique sql信息收集功能。

该值由大变小将会清空系统中原有的数据重新统计（备机不支持此能力）；从小变大不受影响。

当系统中产生的unique sql条目数量大于instr\_unique\_sql\_count时，若开启了unique sql自动淘汰，则系统会按unique sql的更新时间由远到近自动淘汰一定比例的条目，使得新产生的unique sql信息可以继续被统计。若没有开启自动淘汰，则系统产生的新的unique sql信息将不再被统计。

在x86平台集中式部署下，硬件配置规格为32核CPU/256GB内存，使用Benchmark SQL 5.0工具测试性能，开关此参数性能影响约3%。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0~2147483647

**默认值**：100

fig: **注意：**

* 在开启自动淘汰的情况下，如果该值设置的较小，可能会导致系统频繁的进行自动淘汰，有可能会影响数据库系统性能，所以实际场景中建议不要将该值设置的过小，建议值为200000。
* 在开启自动淘汰的情况下，如果该值设置的较大（例如38347922），清理过程中可能会引发大内存问题而无法清理。

**instr\_unique\_sql\_track\_type**

**参数说明：**unique sql记录SQL方式。

该参数属于INTERNAL类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型

top：代表只记录顶层SQL。

**默认值**：top

**enable\_instr\_rt\_percentile**

**参数说明：**是否开启计算系统中80%和95%的SQL响应时间的功能。

该参数属于SIGHUP类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型

* on：表示打开SQL响应时间信息计算功能。
* off：表示关闭SQL响应时间信息计算功能。

**默认值**：on

**percentile**

**参数说明：**SQL响应时间百分比信息，后台计算线程根据设置的值计算相应的百分比信息。

该参数属于INTERNAL类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。

**默认值**：80,95

**instr\_rt\_percentile\_interval**

**参数说明：**SQL响应时间信息计算间隔，SQL响应时间信息计算功能打开后，后台计算线程每隔设置的时间进行一次计算。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0～3600，单位为秒。

**默认值**：10s

**enable\_instr\_cpu\_timer**

**参数说明：**是否捕获SQL执行的cpu时间消耗 。

在x86平台集中式部署下，硬件配置规格为32核CPU/256GB内存，使用Benchmark SQL 5.0工具测试性能，开关此参数性能影响约3.5%。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on：表示捕获SQL执行的CPU时间消耗。
* off：表示不捕获SQL执行的CPU时间消耗。

**默认值**：on

**enable\_stmt\_track**

**参数说明：**控制是否启用Full /Slow SQL特性。

在x86平台集中式部署下，硬件配置规格为32核CPU/256GB内存，使用Benchmark SQL 5.0工具测试性能，开关此参数性能影响约1.2%。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on：表示开启Full /Slow SQL捕获。
* off：表示关闭Full /Slow SQL捕获。

**默认值：**on

**track\_stmt\_session\_slot**

**参数说明：**设置一个session缓存的最大的全量/慢SQL的数量，超过这个数量，新的语句执行将不会被跟踪，直到落盘线程将缓存语句落盘，留出空闲的空间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 2147483647

**默认值：**1000

**track\_stmt\_details\_size**

**参数说明：**设置单语句可以收集的最大的执行事件的大小（byte）。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 100000000

**默认值：**4096

**track\_stmt\_retention\_time**

**参数说明：**组合参数，控制全量/慢SQL记录的保留时间。以60秒为周期读取该参数，并执行清理超过保留时间的记录，仅sysadmin用户可以访问。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符型

该参数分为两部分，形式为'full sql retention time, slow sql retention time'

full sql retention time为全量SQL的保留时间，取值范围为0 ~ 86400

slow sql retention time为慢SQL的保留时间，取值范围为0 ~ 604800

**默认值：**3600,604800

**track\_stmt\_stat\_level**

**参数说明：**控制语句执行跟踪的级别。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置，不区分英文字母大小写。

**取值范围：**字符型

该参数分为两部分，形式为'full sql stat level, slow sql stat level'

full sql stat level为全量SQL跟踪级别，取值范围为OFF、L0、L1、L2

slow sql stat level为慢SQL的跟踪级别，取值范围为OFF、L0、L1、L2

fig: **说明：**   
若全量SQL跟踪级别值为非OFF时，当前SQL跟踪级别值为全量SQL和慢SQL的较高级别（L2 > L1 > L0），级别说明请参见：系统表和系统视图->系统表->STATEMENT\_HISTORY中表1。

**默认值：**OFF,L0

**enable\_auto\_clean\_unique\_sql**

**参数说明：**当系统中产生的unique sql条目数量大于等于instr\_unique\_sql\_count时，是否启用unique sql自动淘汰功能。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

**默认值：**off

fig: **注意：**   
由于快照有部分信息是来源于unique sql，所以开启自动淘汰的情况下，在生成wdr报告时，如果选择的起始快照和终止快照跨过了淘汰发生的时间，会导致无法生成wdr报告。

## 系统性能快照

**enable\_wdr\_snapshot**

**参数说明：**是否开启数据库监控快照功能。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on：打开数据库监控快照功能。
* off：关闭数据库监控快照功能。

**默认值**：off

**wdr\_snapshot\_retention\_days**

**参数说明：**系统中数据库监控快照数据的保留天数。当数据库运行过程期间所生成的快照量数超过保留天数内允许生成的快照数量的最大值时，系统将每隔wdr\_snapshot\_interval时间间隔，清理snapshot\_id最小的快照数据。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1～8。

**默认值**：8

**wdr\_snapshot\_query\_timeout**

**参数说明：**系统执行数据库监控快照操作时，设置快照操作相关的sql语句的执行超时时间。如果语句超过设置的时间没有执行完并返回结果，则本次快照操作失败。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，100～INT\_MAX（秒）。

**默认值**：100s

**wdr\_snapshot\_interval**

**参数说明：**后台线程Snapshot自动对数据库监控数据执行快照操作的时间间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，10～60（分钟）。

**默认值**：1h

**asp\_flush\_mode**

**参数说明：**ASP刷新到磁盘上的方式分为写文件和写系统表，当为‘file’时，默认写文件，为‘table’时写系统表，为‘all’时，即写文件也写系统表，仅sysadmin用户可以访问。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，‘table’、‘file’、‘all’。

**默认值**：‘table’

**asp\_flush\_rate**

**参数说明：**当内存中样本个数达到asp\_sample\_num时，会按一定比例把内存中样本刷新到磁盘上，asp\_flush\_rate为刷新比例。该参数为10时表示按10：1进行刷新。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~10。

**默认值**：10

**asp\_log\_filename**

**参数说明：**当ASP写文件时，该参数设置文件名的格式，仅sysadmin用户可以访问。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。

**默认值**：“asp-%Y-%m-%d\_%H%M%S.log”

**asp\_retention\_days**

**参数说明：**当ASP样本写到系统表时，该参数表示保留的最大天数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~7。

**默认值**：2

**asp\_sample\_interval**

**参数说明：**每次采样的间隔。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~10，单位为秒。

**默认值**：1s

**asp\_sample\_num**

**参数说明：**LOCAL\_ACTIVE\_SESSION视图最大的样本个数，仅sysadmin用户可以访问。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，10～100000。

**默认值**：100000

**enable\_asp**

**参数说明：**是否开启活跃会话信息active session profile。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on：打开active session profile功能。
* off：关闭active session profile功能。

**默认值**：on

## 安全配置

**elastic\_search\_ip\_addr**

**参数说明：**Elastic Search系统IP地址。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。

**默认值**：'<https://127.0.0.1>'

**enable\_security\_policy**

**参数说明：**安全策略开关，控制统一审计和数据动态脱敏策略是否生效。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型。

on：安全策略开关打开。

off：安全策略开关关闭。

**默认值**：off

**use\_elastic\_search**

**参数说明：**使能统一审计发送日志至Elastic Search系统，enable\_security\_policy打开且本参数打开后，统一审计日志会通过http（https）传递至Elastic Search系统（默认使用https安全协议）。此参数打开后需要保证elastic\_search\_ip\_addr对应的es服务可正常连通，否则进程启动失败。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型。

on：使能统一审计日志发送至Elastic Search。

off：关闭统一审计日志发送至Elastic Search。

**默认值**：off

**is\_sysadmin**

**参数说明：**表示当前用户是否是初始用户。

该参数属于INTERNAL类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型。

on：是初始用户。

off：不是初始用户。

**默认值**：on

**enable\_tde**

**参数说明：**透明数据加密功能开关。创建加密表前需要将此参数置为on。当前参数值为off时，禁止创建新的加密表，对于已经创建的加密表只在读取数据时解密，写入数据时不再加密。

该参数属于POSTMASTER类型参数，为固定参数，用户无法修改此参数，只能查看。

**取值范围：**布尔型。

on：开启透明数据加密功能。

off：关闭透明数据加密功能。

**默认值**：off

**tde\_cmk\_id**

**参数说明：**透明数据加密功能使用的数据库实例主密钥CMK的ID编号，由使用的密钥管理服务KMS生成。数据库实例主密钥CMK用于对数据加密密钥DEK进行加密保护，当前需要对DEK进行解密时，需要给KMS发起请求报文，将DEK密文和对应CMK的ID编号一起发送给KMS。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串。

**默认值**：空

## 全局临时表

**max\_active\_global\_temporary\_table**

**参数说明：**全局临时表功能开关，控制是否可以创建全局临时表。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 1000000

* 0：全局临时表功能关闭。
* 大于0：全局临时表功能打开。

**默认值**：1000

**vacuum\_gtt\_defer\_check\_age**

**参数说明：**vacuum执行后检查全局临时表relfrozenxid与普通表的差异。如果全局临时表relfrozenxid落后超过指定参数值，就产生WARNING。一般不用修改。

该参数USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0 ~ 1000000

**默认值**：10000

## HyperLogLog

**hll\_default\_log2m**

**参数说明：**该参数可以指定hll数据结构桶的个数。桶的个数会影响hll计算distinct值的精度，桶的个数越多，误差越小。误差范围为：[-1.04/2log2m*1/2,+1.04/2log2m*1/2]。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，10~16。

**默认值：**14

**hll\_default\_log2explicit**

**参数说明：**该参数可以用来设置从Explicit模式到Sparse模式的默认阈值大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~12。0表示跳过Explicit模式，取1-12表示在基数到达2hll\_default\_log2explicit时切换模式。

**默认值：**10

**hll\_default\_log2sparse**

**参数说明：**该参数可以用来设置从Sparse模式到Full模式的默认阈值大小。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~14。0表示跳过Explicit模式，取1-14表示在基数到达2hll\_default\_log2sparse时切换模式。

**默认值：**12

**hll\_duplicate\_check**

**参数说明：**该参数可以用来指定是否默认开启duplicatecheck。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0，1。

* 0表示默认关闭。
* 1表示默认开启。

**默认值：**0

**hll\_default\_regwidth（废弃）**

**参数说明：**该参数可以指定hll数据结构每个桶的位数，该值越大，hll所占内存越高。hll\_default\_regwidth和hll\_default\_log2m可以决定当前hll能够计算的最大distinct value。当前regwidth设为固定值，该参数不再使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~5。

**默认值：**5

**hll\_default\_expthresh（废弃）**

**参数说明：**该参数可以用来设置从Explicit模式到Sparse模式的默认阈值大小。当前已经使用参数hll\_default\_log2explicit替代类似功能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，-1~7。-1表示自动模式，0表示跳过Explicit模式，取1-7表示在基数到达2hll\_default\_expthresh时切换模式。

**默认值：**-1

**hll\_default\_sparseon（废弃）**

**参数说明：**该参数可用来指定是否默认开启Sparse模式。当前已经使用参数hll\_default\_log2sparse替代类似功能，hll\_default\_log2sparse设置为0时关闭Sparse模式。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0，1。

* 0表示默认关闭。
* 1表示默认开启。

**默认值：**1

**hll\_max\_sparse（废弃）**

**参数说明：**该参数可以用来指定max\_sparse的大小。当前已经使用参数hll\_default\_log2sparse替代类似功能。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，-1~2147483647‬

**默认值：**-1

**enable\_compress\_hll（废弃）**

**参数说明：**该参数可以用来指定是否对hll开启内存优化模式。目前hll内存已经进行了优化设计，该参数不再使用。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on/true表示对hll开启内存优化模式。
* off/false表示不开启内存优化模式。

**默认值：**off

## 用户自定义函数

**udf\_memory\_limit**

**参数说明：**控制每个数据库节点执行UDF时可用的最大物理内存量。本参数当前版本不生效，请使用FencedUDFMemoryLimit和UDFWorkerMemHardLimit参数控制fenced udf worker虚存。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，200\*1024～max\_process\_memory，单位为KB。

**默认值：**200MB

**FencedUDFMemoryLimit**

**参数说明：**控制每个fenced udf worker进程使用的虚拟内存。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整数，0 ~ 2147483647，单位为KB，设置可带单位（KB，MB，GB）。其中0表示不做内存控制。

**默认值：**0

**UDFWorkerMemHardLimit**

**参数说明：**控制fencedUDFMemoryLimit的最大值。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整数，0 ~ 2147483647，单位为KB，设置时可带单位（KB，MB，GB）。

**默认值：**1GB

**pljava\_vmoptions**

**参数说明**：用户自定义设置PL/Java函数所使用的JVM虚拟机的启动参数，仅sysadmin用户可以访问。

该参数属于SUSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，支持：

* JDK8 JVM启动参数（可参见JDK[官方](https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html)说明）
* JDK8 JVM系统属性参数（以–D开头如–Djava.ext.dirs，可参见JDK[官方](https://docs.oracle.com/javase/tutorial/deployment/doingMoreWithRIA/properties.html)说明）
* 用户自定义参数（以–D开头，如–Duser.defined.option）

fig: **须知：**   
如果用户在pljava\_vmoptions中设置参数不满足上述取值范围，会在使用PL/Java语言函数时报错。

**默认值：**空

## 定时任务

**job\_queue\_processes**

**参数说明：**表示系统可以并发执行的job数目。该参数为postmaster级别，通过gs\_guc设置，需要重启gaussdb才能生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**0～1000

功能：

* 当job\_queue\_processes设置为0时，表示不启用定时任务功能，任何job都不会被执行（因为开启定时任务的功能会对系统的性能有影响，有些局点可能不需要定时任务的功能，可以通过设置为0不启用定时任务功能）。
* 当job\_queue\_processes设置为大于0时，表示启用定时任务功能且系统能够并发处理的最大任务数。

启用定时任务功能后，job\_scheduler线程会在定时时间间隔轮询pg\_job系统表，系统设置定时任务检查周期默认为1s。

由于并行运行的任务数太多会消耗更多的系统资源，因此需要设置系统并发处理的任务数，当前并发的任务数达到job\_queue\_processes时，且此时又有任务到期，那么这些任务本次得不到执行而延期到下一轮询周期。因此，建议用户需要根据每个任务的执行时长合理的设置任务的时间间隔（即submit接口中的interval参数），来避免由于任务执行时间太长而导致下个轮询周期无法正常执行。

注：如果同一时间内并行的job数很多，过小的参数值会导致job等待。而过大的参数值则消耗更多的系统资源，建议设置此参数为100，用户可以根据系统资源情况合理调整。

**默认值：**10

**enable\_prevent\_job\_task\_startup**

**参数说明**：控制是否启动job线程。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示不能启动job线程。
* off表示可以启动job线程。

**默认值**：off

## 线程池

**enable\_thread\_pool**

**参数说明**：控制是否使用线程池功能。该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启线程池功能。
* off表示不开启线程池功能。

**默认值**：off

**thread\_pool\_attr**

**参数说明**：用于控制线程池功能的详细属性，该参数仅在enable\_thread\_pool打开后生效。该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：字符串，长度大于0

该参数分为3个部分，'thread\_num、 group\_num、 cpubind\_info'，这3个部分的具体含义如下：

* thread\_num：线程池中的线程总数，取值范围是0~4096。其中0的含义是数据库根据系统CPU core的数量来自动配置线程池的线程数，如果参数值大于0，线程池中的线程数等于thread\_num。线程池大小推荐根据硬件配置设置，计算公式为thread\_num=CPU核数\*3~5，thread\_num最大值4096。
* group\_num：线程池中的线程分组个数，取值范围是0~64。其中0的含义是数据库根据系统NUMA组的个数来自动配置线程池的线程分组个数，如果参数值大于0，线程池中的线程组个数等于group\_num。
* cpubind\_info：线程池是否绑核的配置参数。可选择的配置方式有集中：
* '(nobind)' ，线程不做绑核；
* '(allbind)'，利用当前系统所有能查询到的CPU core做线程绑核；
* '(nodebind: 1, 2)'，利用NUMA组1，2中的CPU core进行绑核；
* '(cpubind: 0-30)'，利用0-30号CPU core进行绑核。
* '(numabind: 0-30)'，在NUMA组内利用0-30号CPU core进行绑核。利用0-30号CPU core进行绑核。该参数不区分大小写。

**默认值**：'16, 2, (nobind)'

## 备份恢复

**operation\_mode**

**参数说明：**标示系统进入备份恢复模式。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在备份恢复过程中。
* off表示不在备份恢复过程中。

**默认值：**off

**enable\_cbm\_tracking**

**参数说明：**当使用roach执行数据库实例的全量和增量备份时需要开启此参数，如果关闭会导致备份失败。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示追踪功能开启。
* off表示追踪功能关闭。

**默认值：**off

**hadr\_max\_size\_for\_xlog\_receiver**

**参数说明**：该参数为异地容灾参数，表示灾备数据库实例中实例获取obs端日志和本地回放日志的最大允许差距，若差距大于此值时停止获取obs端日志。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1中方式对应设置方法进行设置。

**修改建议：**该参数的取值应和本地磁盘大小相关，建议设置为磁盘大小的50%。

**取值范围**：整型，0~2147483647‬

**默认值**：256GB

## Undo

**undo\_space\_limit\_size**

**参数说明：**用于控制undo强制回收阈值，达到阈值的80%启动强制回收，用户需要根据自己的业务情况，设置该值，可以通过先设置一个较大值，然后观察实际业务运行占用undo空间，再将该值调整为合理值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，800MB~16TB

**默认值：**256GB

**undo\_limit\_size\_per\_transaction**

**参数说明：**用于控制单事务undo分配空间阈值，达到阈值时事务报错回滚。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，2MB~16TB

**默认值：**32GB

## DCF参数设置

**enable\_dcf**

**参数说明：**是否开启DCF模式，该参数不允许修改。（单机部署不支持该参数配置）

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型，on/off。on表示当前安装部署方式为DCF模式，off表示当前安装部署方式为非DCF模式。

**默认值：**off

**dcf\_ssl**

**参数说明：**是否开启SSL，重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型，on/off。on表示使用SSL，off表示不使用SSL。

**默认值：**on

**dcf\_config**

**参数说明：**用户安装时自定义配置信息，该参数不允许修改。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**默认值：**字符串，安装时用户自定义配置

**dcf\_data\_path**

**参数说明：**DCF数据路径，该参数不允许修改。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**默认值：**字符串，DN数据目录下的dcf\_data目录

**dcf\_log\_path**

**参数说明：**DCF日志路径，该参数不允许修改。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**默认值：**字符串，DN数据目录下的dcf\_log目录

**dcf\_node\_id**

**参数说明：**DCF所在DN节点ID，用户安装时自定义，该参数不允许修改。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**默认值：**整型，安装时用户自定义配置

**dcf\_max\_workers**

**参数说明：**DCF回调函数线程个数。如果节点数量超过7个，需要增加这个参数的数值（比如增加到40），否则可能会出现主节点一直处于promoting状态，主备节点日志不推进的状态。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，10~262143

**默认值：**10

**dcf\_truncate\_threshold**

**参数说明：**DN对DCF日志进行truncate的门限阈值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~2147483647

**默认值：**100000

**dcf\_election\_timeout**

**参数说明：**DCF leader和follower选举超时时间。选举超时时间数值依赖于当前DN之间的网络状况，在超时时间较小且网络极差的情形下，会有超时选举发生，待网络恢复选举恢复正常。建议根据当前网络状态合理设置超时时间。对DCF节点时钟的约束：DCF节点间最大时钟差异小于选举超时时间的一半。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位s，1~600

**默认值：**3

**dcf\_enable\_auto\_election\_priority**

**参数说明：**DCF优先级选主是否允许内部自动调整优先级值。0表示不允许，1表示允许内部自动调整。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~1

**默认值：**1

**dcf\_election\_switch\_threshold**

**参数说明：**DCF防频繁切主门限。推荐根据用户业务可接受的最大故障时间配置。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位s，0~2147483647

**默认值：**0

**dcf\_run\_mode**

**参数说明：**DCF选举模式，0表示自动选举模式，2表示去使能选举模式。目前去使能选举模式只限定少数派恢复场景使用，修改会导致数据库实例不可用。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举类型，0、2

**默认值：**0

**dcf\_log\_level**

**参数说明：**DCF日志级别。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串

* **关闭日志**：“NONE”，NONE表示关闭日志打印，不能与以下日志级别混合使用。
* **开启日志：“**RUN\_ERR|RUN\_WAR|RUN\_INF|DEBUG\_ERR|DEBUG\_WAR|DEBUG\_INF|TRACE|PROFILE|OPER”

日志级别可以从上述字符串中选取字符串并使用竖线组合使用，不能配置空串。

**默认值：**“RUN\_ERR|RUN\_WAR|DEBUG\_ERR|OPER|RUN\_INF|PROFILE”

**dcf\_log\_backup\_file\_count**

**参数说明：**DCF运行日志备份保留个数。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~100

**默认值：**10

**dcf\_max\_log\_file\_size**

**参数说明：**DCF运行日志单个文件最大大小。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，1~1000

**默认值：**10

**dcf\_socket\_timeout**

**参数说明：**DCF通信模块连接socket超时时间，参数重启生效。对于网络环境比较差的环境，若配置很小的超时时间，可能会导致建链不成功，此时需要适当增大此值。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位ms，10~600000

**默认值：**5000

**dcf\_connect\_timeout**

**参数说明：**DCF通信模块建立连接超时时间，参数重启生效。对于网络环境比较差的环境，若配置很小的超时时间，可能会导致建链不成功，此时需要适当增大此值。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位ms，10~600000

**默认值：**60000

**dcf\_mec\_fragment\_size**

**参数说明：**DCF通信模块fragment大小，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位KB，32~10240

**默认值：**64

**dcf\_stg\_pool\_max\_size**

**参数说明：**DCF存储模内存池最大值，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，32~2147483647

**默认值：**2048

**dcf\_stg\_pool\_init\_size**

**参数说明：**DCF存储模块内存池最小值，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，32~2147483647

**默认值：**32

**dcf\_mec\_pool\_max\_size**

**参数说明：**DCF通信模块内存池最大值，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，32~2147483647

**默认值：**200

**dcf\_flow\_control\_disk\_rawait\_threshold**

**参数说明：**DCF流控功能的磁盘等待阈值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位us，0~2147483647

**默认值：**100000

**dcf\_flow\_control\_net\_queue\_message\_num\_threshold**

**参数说明：**DCF流控功能的网络队列消息数阈值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~2147483647

**默认值：**1024

**dcf\_flow\_control\_cpu\_threshold**

**参数说明：**DCF CPU流控阈值。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位百分比，0~2147483647

**默认值：**100

**dcf\_mec\_batch\_size**

**参数说明：**DCF通信批量消息数，数值为0时，DCF会根据网络以及写入数据量自适应调整，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，0~1024

**默认值：**0

**dcf\_mem\_pool\_max\_size**

**参数说明：**DCF内存最大值，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，32~2147483647

**默认值：**2048

**dcf\_mem\_pool\_init\_size**

**参数说明：**DCF内存初始化大小，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，单位MB，32~2147483647

**默认值：**32

**dcf\_compress\_algorithm**

**参数说明：**DCF运行日志传输压缩算法，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型

* 0表示不压缩
* 1表示ZSTD压缩算法
* 2表示LZ4压缩算法

**默认值：**0

**dcf\_compress\_level**

**参数说明：**DCF日志传输压缩级别，参数重启生效，此参数生效前提必须配置有效的压缩算法，即设置合法的dcf\_compress\_algorithm参数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~22

若不开启压缩，配置的压缩级别将不生效。

**默认值：**1

**dcf\_mec\_channel\_num**

**参数说明：**DCF通信通道数量，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~64

**默认值：**1

**dcf\_rep\_append\_thread\_num**

**参数说明：**DCF日志复制线程数量，参数重启生效。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~1000

**默认值：**2

**dcf\_mec\_agent\_thread\_num**

**参数说明：**DCF通信工作线程数量，参数重启生效。dcf\_mec\_agent\_thread\_num值建议不少于2*节点数*dcf\_mec\_channel\_num。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~1000

**默认值：**10

**dcf\_mec\_reactor\_thread\_num**

**参数说明：**DCF使用reactor线程数量，参数重启生效。dcf\_mec\_reactor\_thread\_num与dcf\_mec\_agent\_thread\_num比例建议1：40。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~100

**默认值：**1

**dcf\_log\_file\_permission**

**参数说明：**DCF运行日志文件属性，参数重启生效，参数安装阶段配置，后续不支持修改。若用户需要支持同组的其他用户访问日志，首先需要所有的父目录都支持同组的其他用户也能访问。即若参数dcf\_log\_path\_permission配置为750，dcf\_log\_file\_permission只能为600或者640。若参数dcf\_log\_path\_permission配置为700，dcf\_log\_file\_permission只能为600。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举型，600、640

**默认值：**600

**dcf\_log\_path\_permission**

**参数说明：**DCF运行日志目录属性，参数重启生效，参数安装阶段配置，后续不支持修改。若用户需要支持同组的其他用户访问日志路径，需选择参数750，否则选择700。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**枚举型，700、750

**默认值：**700

## 闪回相关参数

本章节介绍闪回功能相关参数。

**enable\_recyclebin**

**参数说明**：用来控制回收站的实时打开和关闭。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

**默认值**：off

fig: **注意：**  
recyclebin不支持Astore，只支持Ustore。

**recyclebin\_retention\_time**

参数说明：设置回收站对象保留时间，超过该时间的回收站对象将被自动清理。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

取值范围：整型，单位为s，最小值为1，最大值为2147483647。

默认值：15min（即900s）

**version\_retention\_age**

**参数说明**：设置旧版本保留的事务数，超过该事务数的旧版本将被回收清理。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～576460752303423487，值为0表示不延迟。

**默认值**：0

fig: **注意：**   
该参数已弃用。

**vacuum\_defer\_cleanup\_age**

**参数说明**：指定VACUUM使用的事务数，VACUUM会延迟清除无效的行存表记录，延迟的事务个数通过vacuum\_defer\_cleanup\_age进行设置。即VACUUM和VACUUM FULL操作不会立即清理刚刚被删除元组。也可以通过设置该参数，配置闪回功能旧版本保留期限。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，0～1000000，值为0表示不延迟。**默认值**：0

fig: **注意：**   
在进行Ustore闪回时，无需关注该参数。其服务于之前版本的astore闪回功能，同时具有其他用途。本版本闪回功能已不使用。

**undo\_retention\_time**

**参数说明**：设置undo旧版本保留时间。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：整型，单位为s，最小值为0，最大值为2147483647。

**默认值**：0

## 回滚相关参数

**max\_undo\_workers**

**参数说明：**异步回滚调用的undoworker线程数量，参数重启生效。

该参数属于SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**整型，1~100

**默认值：**5

## 预留参数

fig: **说明：**   
下列参数为预留参数，该版本不生效。

* acce\_min\_datasize\_per\_thread
* cstore\_insert\_mode
* enable\_constraint\_optimization
* enable\_hadoop\_env
* enable\_hdfs\_predicate\_pushdown
* enable\_orc\_cache
* schedule\_splits\_threshold
* backend\_version
* undo\_zone\_count
* version\_retention\_age

## AI特性

**enable\_hypo\_index**

**参数说明：**该参数控制数据库的优化器进行EXPLAIN时是否考虑创建虚拟索引。通过对特定的查询语句执行explain，用户可根据优化器给出的执行计划评估该索引是否能够提升该查询语句的执行效率。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**布尔型

* on表示在进行EXPLAIN时创建虚拟索引。
* off表示在进行EXPLAIN时不创建虚拟索引。

**默认值：**off

**db4ai\_snapshot\_mode**

**参数说明：**snapshot有2种模式：MSS（物化模式，存储数据实体）和CSS（计算模式，存储增量信息）。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，MSS/CSS

* MSS表示物化模式，db4ai在创建快照的时候存储数据实体。
* CSS表示计算模式，db4ai在创建快照的时候存储增量信息。

**默认值：**MSS

**db4ai\_snapshot\_version\_delimiter**

**参数说明：**该参数为数据表快照版本分隔符。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，长度大于0

**默认值：**@

**db4ai\_snapshot\_version\_separator**

**参数说明：**该参数用于指定数据表快照子版本分隔符。

该参数属于USERSET类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，长度大于0

**默认值：**.

**unix\_socket\_directory**

**参数说明：**用于指定unix\_socket通信方式中，文件存放的路径。此参数只能在配置文件postgresql.conf中指定。再启动fenced模式前需要设定该GUC参数。

该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围：**字符串，长度大于等于0

**默认值： ‘ ‘**

## Global SysCache参数

**enable\_global\_syscache**

**参数说明**：控制是否使用全局系统缓存功能。该参数属于POSTMASTER类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

**取值范围**：布尔型

* on表示开启全局系统缓存功能。
* off表示不开启全局系统缓存功能。

**默认值**：on

推荐结合线程池参数使用。打开该参数后，如果需要访问备机，建议设置备机wal\_level级别为hot\_standby以上。

**global\_syscache\_threshold**

**参数说明：**全局系统缓存内存最大占用大小。

该参数属于PGC\_SIGHUP类型参数，请参考：配置运行参数->重设参数章节中表1对应设置方法进行设置。

需要打开enable\_global\_syscache参数。

**取值范围**：整型，16384~1073741824，单位为kB。

**默认值：**163840

推荐计算公式：热点DB个数和线程个数的最小值乘以每个DB分配的内存大小

即global\_syscache\_threshold = min(count(hot dbs)，count(threads)) \* memofdb

热点DB数即访问较为频繁的数据库，线程数在线程池模式下取线程池线程个数和后台线程个数之和，非线程池模式不需要计算这个值，直接使用热点DB数。

memofdb即平均每个db应该分配的内存，每个DB的底噪内存是2M，平均每增加一个表或者索引，增加11k内存。

如果设置的值过小，会导致内存频繁淘汰，内存存在大量碎片无法回收，导致内存控制失效。

## date类型控制参数

**功能描述**

date类型控制参数vb\_date\_type用来控制date类型的底层存储类型。

**语法格式**

* ORACLE兼容模式下，支持通过alter命令设置date类型控制参数vb\_date\_type，默认值为1。

alter system set vb\_date\_type = [1/2];

* ORACLE兼容模式下，通过修改postgresql.conf设置date类型控制参数

vi $PGDATA/postgresql.conf  
 //设置参数:  
vb\_date\_type = [1/2];

**参数说明**

* vb\_date\_type = 1时 date底层存储为oradate。
* vb\_date\_type = 2时 date底层存储为timestamp。

**注意事项**

* PG兼容模式下，不支持date类型控制参数。
* MYSQL兼容模式下，不支持date类型控制参数。
* 不指定兼容模式时，数据库默认为oracel兼容模式，支持date类型控制参数。

**示例**

**示例1**： oracle兼容下设置date类型控制参数。

1、创建数据库，设置兼容模式为oracle。

CREATE DATABASE date\_test\_oracle DBCOMPATIBILITY 'A';

2、配置date类型控制参数vb\_date\_type。

alter system set vb\_date\_type = 1;

3、重启数据库。

vb\_ctl restart

4、查看date兼容效果。

select '2022-08-19'::date;

查询结果显示为：

oradate  
---------------------  
 2022-08-19 00:00:00  
(1 row)

**示例2**：ORACLE兼容模式下，通过修改postgresql.conf设置date类型控制参数。

1、创建数据库。

CREATE DATABASE date\_test\_postgresqlconf DBCOMPATIBILITY 'A';

2、配置date类型控制参数。

vi $PGDATA/postgresql.conf  
//设置参数:  
vb\_date\_type = 2;

3、重启数据库。

vb\_ctl restart

4、查看date兼容效果。

\c date\_test\_postgresqlconf  
select '2022-08-19'::date;

查询结果显示为：

timestamp  
---------------------  
 2022-08-19 00:00:00  
(1 row)

# 配置运行参数

## 重设参数

**背景信息**

Vastbase提供了多种修改GUC参数的方法，用户可以方便的针对数据库、用户、会话进行设置。

* 参数名称不区分大小写。
* 参数取值有整型、浮点型、字符串、布尔型和枚举型五类。
* 布尔值可以是（on，off）、（true，false）、（yes，no）或者（1，0），且不区分大小写。
* 枚举类型的取值是在系统表pg\_settings的enumvals字段取值定义的。
* 对于有单位的参数，在设置时请指定单位，否则将使用默认的单位。
* 参数的默认单位在系统表pg\_settings的unit字段定义的。
* 内存单位有：KB（千字节）、MB（兆字节）和GB（吉字节）。
* 时间单位：ms（毫秒）、s（秒）、min（分钟）、h（小时）和d（天）。

具体参数说明请参见GUC参数说明章节。

**GUC参数设置**

Vastbase提供了六类GUC参数，具体分类和设置方式请参考[表1](#X81b6a11722c3a6b797a8adf9a85f9fe01812a67)：

**表 1** GUC参数分类

|  |  |  |
| --- | --- | --- |
| **参数类型** | **说明** | **设置方式** |
| INTERNAL | 固定参数，在创建数据库的时候确定，用户无法修改，只能通过show语法或者pg\_settings视图进行查看。 | 无 |
| POSTMASTER | 数据库服务端参数，在数据库启动时确定，可以通过配置文件指定。 | [支持表2中的方式一、方式四。](https://opengauss.org/zh/docs/3.0.0/docs/Developerguide/%E9%87%8D%E8%AE%BE%E5%8F%82%E6%95%B0.html" \l "zh-cn_topic_0283137176_zh-cn_topic_0237121562_zh-cn_topic_0059777490_t290c8f15953843db8d8e53d867cd893d) |
| SIGHUP | 数据库全局参数，可在数据库启动时设置或者在数据库启动后，发送指令重新加载。 | [支持表2中的方式一、方式二、方式四。](https://opengauss.org/zh/docs/3.0.0/docs/Developerguide/%E9%87%8D%E8%AE%BE%E5%8F%82%E6%95%B0.html" \l "zh-cn_topic_0283137176_zh-cn_topic_0237121562_zh-cn_topic_0059777490_t290c8f15953843db8d8e53d867cd893d) |
| BACKEND | 会话连接参数。在创建会话连接时指定，连接建立后无法修改。连接断掉后参数失效。内部使用参数，不推荐用户设置。 | [支持表2中的方式一、方式二、方式四。](https://opengauss.org/zh/docs/3.0.0/docs/Developerguide/%E9%87%8D%E8%AE%BE%E5%8F%82%E6%95%B0.html" \l "zh-cn_topic_0283137176_zh-cn_topic_0237121562_zh-cn_topic_0059777490_t290c8f15953843db8d8e53d867cd893d)  说明：  设置该参数后，下一次建立会话连接时生效 |
| SUSET | 数据库管理员参数。可在数据库启动时、数据库启动后或者数据库管理员通过SQL进行设置。 | [支持表2中的方式一、方式二或由数据库管理员通过方式三设置。](https://opengauss.org/zh/docs/3.0.0/docs/Developerguide/%E9%87%8D%E8%AE%BE%E5%8F%82%E6%95%B0.html" \l "zh-cn_topic_0283137176_zh-cn_topic_0237121562_zh-cn_topic_0059777490_t290c8f15953843db8d8e53d867cd893d) |
| USERSET | 普通用户参数。可被任何用户在任何时刻设置。 | [支持表2中的方式一、方式二或方式三设置。](https://opengauss.org/zh/docs/3.0.0/docs/Developerguide/%E9%87%8D%E8%AE%BE%E5%8F%82%E6%95%B0.html" \l "zh-cn_topic_0283137176_zh-cn_topic_0237121562_zh-cn_topic_0059777490_t290c8f15953843db8d8e53d867cd893d) |

**表 2** GUC参数设置方式

|  |  |
| --- | --- |
| **序号** | **设置方法** |
| 方式一 | 1、使用如下命令修改参数。  vb\_guc set -D datadir -c "paraname=value"  说明：  如果参数是一个字符串变量，则使用-c parameter="'value'"或者使用-c "parameter = 'value'"。  使用以下命令在数据库节点上设置cm\_agent某个参数。  vb\_guc set -Z cmagent -c "paraname=value"  使用以下命令在数据库节点上设置cm\_server某个参数。  vb\_guc set -Z cmserver -c "paraname=value"   1. 重启数据库使参数生效。   说明：  重启Vastbase操作会导致用户执行操作中断，请在操作之前规划好合适的执行窗口。  vb\_ctl restart |
| 方式二 | vb\_guc reload -D datadir -c "paraname=value" |
| 方式三 | 修改指定数据库、用户、会话级别的参数。  1、设置数据库级别的参数  ALTER DATABASE dbname SET paraname TO value;  在下次会话中生效。  2、设置用户级别的参数  ALTER USER username SET paraname TO value;  在下次会话中生效。  3、设置会话级别的参数  SET paraname TO value;  修改本次会话中的取值。退出会话后，设置将失效。  说明：  SET设置的会话级参数优先级最高，其次是ALTER设置的，其中ALTER DATABASE设置的参数值优先级高于ALTER USER设置，这三种设置方式设置的优先级都高于vb\_guc设置方式。 |
| 方式四 | 使用ALTER SYSTEM SET修改数据库参数。  设置POSTMASERT级别的参数  ALTER SYSTEM SET paraname TO value;  重启后生效。  设置SIGHUP级别的参数  ALTER SYSTEM SET paraname TO value;  立刻生效(实际等待线程重新加载参数略有延迟)。  设置BACKEND级别的参数  ALTER SYSTEM SET paraname TO value;  在下次会话中生效。 |

fig: **注意：**   
使用方式一和方式二设置参数时，若所设参数不属于当前环境，数据库会提示参数不在支持范围内的相关信息。

**示例**

使用方式一设置数据库参数，以在数据库主节点设置archive\_mode参数为例。

1、以操作系统用户vastbase登录数据库主节点。

2、查看archive\_mode参数，on表示日志要进行归档操作。

cat /home/vastbase/data/vastbase/postgresql.conf | grep archive\_mode

archive\_mode = on

3、设置archive\_mode参数为off，关闭日志的归档操作。

vb\_guc set -D $PGDATA -c "archive\_mode=off"

4、重启数据库使参数生效。

vb\_ctl restart

5、使用如下命令连接数据库，vastbase为需要连接的数据库名称，5432为数据库主节点的端口号。

vsql -d vastbase -p 5432

6、检查参数设置的正确性。

SHOW archive\_mode;  
 archive\_mode  
--------------  
 off  
(1 row)

使用方式二设置参数，以在数据库主节点设置authentication\_timeout参数为例。

1、以操作系统用户vastbase登录数据库主节点。

2、查看authentication\_timeout参数。

cat /home/vastbase/data/vastbase/postgresql.conf | grep authentication\_timeout

authentication\_timeout = 1min

3、设置authentication\_timeout参数为59s。

vb\_guc reload -D $PGDATA -c "authentication\_timeout = 59s"  
   
Total instances: 2. Failed instances: 0.  
Success to perform gs\_guc!

4、使用如下命令连接数据库，vastbase为需要连接的数据库名称，5432为数据库主节点的端口号。

vsql -d vastbase -p 5432

5、检查参数设置的正确性。

SHOW authentication\_timeout;  
 authentication\_timeout   
------------------------  
 59s  
(1 row)

使用方式三设置参数，以设置explain\_perf\_mode参数为例。

1、以操作系统用户vastbasq登录数据库主节点。

2、使用如下命令连接数据库，vastbase为需要连接的数据库名称，5432为数据库主节点的端口号。

vsql -d vastbase -p 5432

3、查看explain\_perf\_mode参数。

SHOW explain\_perf\_mode;  
 explain\_perf\_mode   
-------------------  
 normal  
(1 row)

4、设置explain\_perf\_mode参数。

使用以下任意方式进行设置：

* 设置数据库级别的参数

ALTER DATABASE postgres SET explain\_perf\_mode TO pretty;

当结果显示为如下信息，则表示设置成功。

ALTER DATABASE

在下次会话中生效。

* 设置用户级别的参数

ALTER USER vastbase SET explain\_perf\_mode TO pretty;

当结果显示为如下信息，则表示设置成功。

ALTER ROLE

在下次会话中生效。

* 设置会话级别的参数

SET explain\_perf\_mode TO pretty;

当结果显示为如下信息，则表示设置成功。

SET

5、检查参数设置的正确性。

SHOW explain\_perf\_mode;  
 explain\_perf\_mode  
--------------  
 pretty  
(1 row)

* 使用方式四设置参数，以设置autovacuum参数为例。

1、以操作系统用户vastbase登录数据库主节点。

2、使用如下命令连接数据库，vastbase为需要连接的数据库名称，5432为数据库主节点的端口号。

vsql -d vastbase -p 5432

3、查看autovacuum当前值。

show autovacuum;  
 autovacuum   
------------  
 off  
(1 row)

4、设置参数autovacuum为on。

ALTER SYSTEM SET autovacuum to on;

5、检查参数设置的正确性。

show autovacuum;  
 autovacuum   
------------  
 on  
(1 row)

## 查看参数当前取值

**功能描述**

Vastbase安装后，有一套默认的运行参数，为了使Vastbase与业务的配合度更高，用户需要根据业务场景和数据量的大小进行GUC参数调整。

**操作步骤**

1、以操作系统用户vastbase登录数据库主节点。

2、使用如下命令连接数据库。

vsql -d postgres -p 5432

postgres为需要连接的数据库名称，5432为数据库主节点的端口号。

连接成功后，系统显示类似如下信息：

vsql ((Vastbase x.x.x build 290d125f) compiled at 2020-05-08 02:59:43 commit 2143 last mr 131  
Non-SSL connection (SSL connection is recommended when requiring high-security)  
Type "help" for help.  
  
vastbase=#

3、查看数据库运行参数当前取值。

* 方法一：使用SHOW命令。

使用如下命令查看单个参数：

SHOW server\_version;

server\_version显示数据库版本信息的参数。

使用如下命令查看所有参数：

SHOW ALL;

* 方法二：使用pg\_settings视图。

使用如下命令查看单个参数：

SELECT \* FROM pg\_settings WHERE NAME='server\_version';

使用如下命令查看所有参数：

SELECT \* FROM pg\_settings;

**示例**

查看服务器的版本号。

vastbase=# SHOW server\_version;  
 server\_version   
----------------  
 9.2.4  
(1 row)

# 接口参考

## 应用程序接口

用户可以使用标准的数据库应用程序接口，开发基于Vastbase的应用程序，但Vastbase暂不支持JDBC和ODBC这两个接口。

**支持的应用程序接口**

每个应用程序是一个独立的Vastbase开发项目。应用程序通过API与数据库进行交互，在避免了应用程序直接操作数据库系统的同时，增强了应用程序的可移植性、扩展性和可维护性。

# 错误日志信息参考

## 内核错误信息

**ERRMSG:** "unsupported syntax: ENCRYPTED WITH in this operation"

SQLSTATE: 42601

CAUSE: "client encryption feature is not supported this operation."

ACTION: "Check client encryption feature whether supported this operation."

**ERRMSG:** "invalid grant operation"

SQLSTATE: 0LP01

CAUSE: "Grant options cannnot be granted to public."

ACTION: "Grant grant options to roles."

**ERRMSG:** "unrecognized object kind: %d"

SQLSTATE: XX004

CAUSE: "The object type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "unrecognized GrantStmt.targtype: %d"

SQLSTATE: XX004

CAUSE: "The target type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported target types."

**ERRMSG:** "invalid grant operation"

SQLSTATE: 0LP01

CAUSE: "Grant to public operation is forbidden in security mode."

ACTION: "Don't grant to public in security mode."

**ERRMSG:** "unrecognized object type"

SQLSTATE: XX004

CAUSE: "The object type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "invalid grant/revoke operation"

SQLSTATE: 0LP01

CAUSE: "Column privileges are only valid for relations in GRANT/REVOKE."

ACTION: "Use the column privileges only for relations."

**ERRMSG:** "invalid AccessPriv node"

SQLSTATE: 0LP01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "unrecognized GrantStmt.objtype: %d"

SQLSTATE: XX004

CAUSE: "The object type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "undefined client master key"

SQLSTATE: 42705

CAUSE: "The client master key does not exist."

ACTION: "Check whether the client master key exists."

**ERRMSG:** "undefined column encryption key"

SQLSTATE: 42705

CAUSE: "The column encryption key does not exist."

ACTION: "Check whether the column encryption key exists."

**ERRMSG:** "large object %u does not exist"

SQLSTATE: 42704

CAUSE: "The large object does not exist."

ACTION: "Check whether the large object exists."

**ERRMSG:** "redundant options"

SQLSTATE: 42601

CAUSE: "The syntax 'schemas' is redundant in ALTER DEFAULT PRIVILEGES statement."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax."

**ERRMSG:** "redundant options"

SQLSTATE: 42601

CAUSE: "The syntax 'roles' is redundant in ALTER DEFAULT PRIVILEGES statement."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax."

**ERRMSG:** "option '%s' not recognized"

SQLSTATE: 42601

CAUSE: "The option in ALTER DEFAULT PRIVILEGES statement is not supported."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax."

**ERRMSG:** "unrecognized GrantStmt.objtype: %d"

SQLSTATE: XX004

CAUSE: "The object type is not supported for ALTER DEFAULT PRIVILEGES."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax to obtain the supported object types."

**ERRMSG:** "invalid alter default privileges operation"

SQLSTATE: 0LP01

CAUSE: "Default privileges cannot be set for columns."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax."

**ERRMSG:** "unrecognized objtype: %d"

SQLSTATE: XX004

CAUSE: "The object type is not supported for default privileges."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax to obtain the supported object types."

**ERRMSG:** "could not find tuple for default ACL %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "unexpected default ACL type: %d"

SQLSTATE: 0LP01

CAUSE: "The object type is not supported for default privilege."

ACTION: "Check ALTER DEFAULT PRIVILEGES syntax to obtain the supported object types."

**ERRMSG:** "invalid object id"

SQLSTATE: 0LP01

CAUSE: "The object type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "undefined column"

SQLSTATE: 42703

CAUSE: "The column of the relation does not exist."

ACTION: "Check whether the column exists."

**ERRMSG:** "column number out of range"

SQLSTATE: 0LP01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for attribute %d of relation %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for relation %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "unsupported object type"

SQLSTATE: 42809

CAUSE: "Index type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "unsupported object type"

SQLSTATE: 42809

CAUSE: "Composite type is not supported for GRANT/REVOKE."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "wrong object type"

SQLSTATE: 42809

CAUSE: "GRANT/REVOKE SEQUENCE only support sequence objects."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "invalid privilege type USAGE for table"

SQLSTATE: 0LP01

CAUSE: "GRANT/REVOKE TABLE do not support USAGE privilege."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported privilege types for tables."

**ERRMSG:** "invalid privilege type %s for column"

SQLSTATE: 0LP01

CAUSE: "The privilege type is not supported for column object."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported privilege types for column object."

**ERRMSG:** "cache lookup failed for database %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for foreign-data wrapper %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for foreign server %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for function %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for language %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "Grant/revoke on untrusted languages if forbidden."

SQLSTATE: 0LP01

CAUSE: "Grant/revoke on untrusted languages if forbidden."

ACTION: "Support grant/revoke on trusted C languages"

**ERRMSG:** "Forbid grant language c to user with grant option."

SQLSTATE: 0A000

CAUSE: "Forbid grant language c to user with grant option."

ACTION: "Only support grant language c to user."

**ERRMSG:** "Forbid grant language c to public."

SQLSTATE: 0A000

CAUSE: "Forbid grant language c to public."

ACTION: "Grant language c to specified users."

**ERRMSG:** "cache lookup failed for large object %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for namespace %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for tablespace %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for type %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cannot set privileges of array types"

SQLSTATE: 0LP01

CAUSE: "Cannot set privileges of array types."

ACTION: "Set the privileges of the element type instead."

**ERRMSG:** "wrong object type"

SQLSTATE: 42809

CAUSE: "GRANT/REVOKE DOMAIN only support domain objects."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "cache lookup failed for data source %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for client master key %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for column encryption key %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "cache lookup failed for directory %u"

SQLSTATE: 29P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "unrecognized privilege type '%s'"

SQLSTATE: 42601

CAUSE: "The privilege type is not supported."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported privilege types."

**ERRMSG:** "unrecognized privilege: %d"

SQLSTATE: XX004

CAUSE: "The privilege type is not supported."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported privilege types."

**ERRMSG:** "unrecognized AclResult"

SQLSTATE: XX004

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "permission denied for column '%s' of relation '%s'"

SQLSTATE: 42501

CAUSE: "Insufficient privileges for the column."

ACTION: "Select the system tables to get the acl of the column."

**ERRMSG:** "role with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "unrecognized objkind: %d"

SQLSTATE: XX004

CAUSE: "The object type is not supported for privilege check."

ACTION: "Check GRANT/REVOKE syntax to obtain the supported object types."

**ERRMSG:** "attribute %d of relation with OID %u does not exist"

SQLSTATE: 42703

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "the column has been dropped"

SQLSTATE: 42703

CAUSE: "The column does not exist."

ACTION: "Check whether the column exists."

**ERRMSG:** "relation with OID %u does not exist"

SQLSTATE: 42P01

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "invalid group"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "database with OID %u does not exist"

SQLSTATE: 3D000

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "directory with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "function with OID %u does not exist"

SQLSTATE: 42883

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "client master key with OID %u does not exist"

SQLSTATE: 42705

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "language with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "large object %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "schema with OID %u does not exist"

SQLSTATE: 3F001

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "tablespace with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "foreign-data wrapper with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "foreign server with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "data source with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "type with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "operator with OID %u does not exist"

SQLSTATE: 42883

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "column encryption key with OID %u does not exist"

SQLSTATE: 42705

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "operator class with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "operator family with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "text search dictionary with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "text search configuration with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "collation with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "conversion with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "extension with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "synonym with OID %u does not exist"

SQLSTATE: 42704

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "package can not create the same name with schema."

SQLSTATE: 22023

CAUSE: "Package name conflict"

ACTION: "Please rename package name"

**ERRMSG:** "type is not exists %s."

SQLSTATE: 22023

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "This input type is not supported for tdigest\_in()"

SQLSTATE: 0A000

CAUSE: "input type is not supported"

ACTION: "Check tdigest\_in syntax to obtain the supported privilege types"

**ERRMSG:** "Failed to apply for memory"

SQLSTATE: 53200

CAUSE: "palloc failed"

ACTION: "Check memory"

**ERRMSG:** "Failed to get tde info from relation '%s'."

SQLSTATE: XX005

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "SPI\_connect failed: %s"

SQLSTATE: SP001

CAUSE: "System error."

ACTION: "Analyze the error message before the error"

**ERRMSG:** "permission denied for terminate snapshot thread"

SQLSTATE: 42501

CAUSE: "The user does not have system admin privilege"

ACTION: "Grant system admin to user"

**ERRMSG:** "terminate snapshot thread failed"

SQLSTATE: OP001

CAUSE: "Execution failed due to: %s"

ACTION: "check if snapshot thread exists"

**ERRMSG:** "terminate snapshot thread failed"

SQLSTATE: OP001

CAUSE: "restart wdr snapshot thread timeoutor The thread did not respond to the kill signal"

ACTION: "Check the wdr snapshot thread is restarted"

**ERRMSG:** "set lockwait\_timeout failed"

SQLSTATE: XX000

CAUSE: "System error."

ACTION: "Contact engineer to support."

**ERRMSG:** "permission denied for create WDR Snapshot"

SQLSTATE: 42501

CAUSE: "The user does not have system admin privilege"

ACTION: "Grant system admin to user"

**ERRMSG:** "WDR snapshot request can not be accepted, please retry later"

SQLSTATE: OP001

CAUSE: "wdr snapshot thread does not exist"

ACTION: "Check if wdr snapshot thread exists"

**ERRMSG:** "Cannot respond to WDR snapshot request"

SQLSTATE: OP001

CAUSE: "Execution failed due to: %s"

ACTION: "Check if wdr snapshot thread exists"

**ERRMSG:** "query(%s) can not get datum values"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "create sequence failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check if sequence can be created"

**ERRMSG:** "update snapshot end time stamp filled"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "query can not get datum values"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "SPI\_connect failed: %s"

SQLSTATE: XX000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "query(%s) execute failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "clean table of snap\_%s is failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "analyze table failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "insert into tables\_snap\_timestamp start time stamp is failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "insert data failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful and check whether the query can be executed"

**ERRMSG:** "update tables\_snap\_timestamp end time stamp is failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "clean snapshot id %lu is failed in snapshot table"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful and check whether the query can be executed"

**ERRMSG:** "clean snapshot failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "can not create snapshot stat table"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "create WDR snapshot data table failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "insert into tables\_snap\_timestamp start time stamp failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "insert into snap\_%s is failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "update tables\_snap\_timestamp end time stamp failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "create index failed"

SQLSTATE: 22000

CAUSE: "System error."

ACTION: "Check whether the query can be executed"

**ERRMSG:** "analyze table, connection failed: %s"

SQLSTATE: XX000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "snapshot thread SPI\_connect failed: %s"

SQLSTATE: XX000

CAUSE: "System error."

ACTION: "Check whether the snapshot retry is successful"

**ERRMSG:** "Distributed key column can't be transformed"

SQLSTATE: 42P10

CAUSE: "There is a risk of violating uniqueness when transforming distribution columns."

ACTION: "Change transform column."

**ERRMSG:** "cannot convert %s to %s"

SQLSTATE: 42804

CAUSE: "There is no conversion path in pg\_cast."

ACTION: "Rewrite or cast the expression."

**ERRMSG:** "create matview on TDE table failed"

SQLSTATE: 0A000

CAUSE: "create materialized views is not supported on TDE table"

ACTION: "check CREATE syntax about create the materialized views"

**ERRMSG:** "schema name can not same as package"

SQLSTATE: 22023

CAUSE: "schema name conflict"

ACTION: "rename schema name"

**ERRMSG:** "Unrecognized commandType when checking read-only attribute."

SQLSTATE: XX004

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Fail to generate subquery plan."

SQLSTATE: XX005

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Unrecognized node type when processing qual condition."

SQLSTATE: XX004

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Unrecognized node type when processing const parameters."

SQLSTATE: XX004

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "SELECT FOR UPDATE/SHARE is not allowed with UNION/INTERSECT/EXCEPT"

SQLSTATE: 0A000

CAUSE: "SQL uses unsupported feature."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "GROUP BY cannot be implemented."

SQLSTATE: 0A000

CAUSE: "GROUP BY uses unsupported datatypes."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "TSDB functions cannot be used if enable\_tsdb is off."

SQLSTATE: D0011

CAUSE: "Functions are not loaded."

ACTION: "Turn on enable\_tsdb according to manual."

**ERRMSG:** "Unrecognized node type when extracting index."

SQLSTATE: XX004

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Ordering operator cannot be identified."

SQLSTATE: 42883

CAUSE: "Grouping set columns must be able to sort their inputs."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "DISTINCT cannot be implemented."

SQLSTATE: 0A000

CAUSE: "DISTINCT uses unsupported datatypes."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "Failed to locate grouping columns."

SQLSTATE: 55000

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Resjunk output columns are not implemented."

SQLSTATE: 20000

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "PARTITION BY cannot be implemented."

SQLSTATE: 0A000

CAUSE: "PARTITION BY uses unsupported datatypes."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "ORDER BY cannot be implemented."

SQLSTATE: 0A000

CAUSE: "ORDER BY uses unsupported datatypes."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "Failed to deconstruct sort operators into partitioning/ordering operators."

SQLSTATE: D0011

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "OBS and HDFS foreign table can NOT be in the same plan."

SQLSTATE: XX008

CAUSE: "SQL uses unsupported feature."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "Pool size should not be zero"

SQLSTATE: 22012

CAUSE: "Compute pool configuration file contains error."

ACTION: "Please check the value of 'pl' in cp\_client.conf."

**ERRMSG:** "Failed to get the runtime info from the compute pool."

SQLSTATE: 22004

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "Version is not compatible between local cluster and the compute pool."

SQLSTATE: XX008

CAUSE: "Compute pool is not installed appropriately."

ACTION: "Configure compute pool according to manual."

**ERRMSG:** "No optional index path is found."

SQLSTATE: 01000

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "MERGE INTO on replicated table does not yet support using distributed tables."

SQLSTATE: 0A000

CAUSE: "SQL uses unsupported feature."

ACTION: "Modify SQL statement according to the manual."

**ERRMSG:** "Fail to find ForeignScan node!"

SQLSTATE: P0002

CAUSE: "System error."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "sql advisor don't support none table, temp table, system table."

SQLSTATE: 42601

CAUSE: "sql advisor don't support none table, temp table, system table."

ACTION: "check query component"

**ERRMSG:** "Invalid autonomous transaction return datatypes"

SQLSTATE: P0000

CAUSE: "PL/SQL uses unsupported feature."

ACTION: "Contact Huawei Engineer."

**ERRMSG:** "new row for relation '%s' violates check constraint '%s'"

SQLSTATE: 23514

CAUSE: "some rows copy failed"

ACTION: "check table defination"

**ERRMSG:** "new row for relation '%s' violates check constraint '%s'"

SQLSTATE: 23514

CAUSE: "some rows copy failed"

ACTION: "set client\_min\_messages = info for more details"

**ERRMSG:** "get gauss home path is NULL"

SQLSTATE: XX005

CAUSE: "gauss home path not set"

ACTION: "check if $GAUSSHOME is exist"

**ERRMSG:** "unable to open kms\_iam\_info.json file"

SQLSTATE: 58P03

CAUSE: "file not exist or broken"

ACTION: "check the kms\_iam\_info.json file"

**ERRMSG:** "can not get password plaintext"

SQLSTATE: XX005

CAUSE: "file not exist or broken"

ACTION: "check the password cipher rand file"

**ERRMSG:** "IAM info json key is NULL"

SQLSTATE: XX005

CAUSE: "IAM info value error"

ACTION: "check tde\_config kms\_iam\_info.json file"

**ERRMSG:** "get internal password is NULL"

SQLSTATE: XX005

CAUSE: "cipher rand file missing"

ACTION: "check password cipher rand file"

**ERRMSG:** "KMS info json key is NULL"

SQLSTATE: XX005

CAUSE: "KMS info value error"

ACTION: "check tde\_config kms\_iam\_info.json file"

**ERRMSG:** "unable to get json file"

SQLSTATE: 58P03

CAUSE: "parse json file failed"

ACTION: "check the kms\_iam\_info.json file format"

**ERRMSG:** "get JSON tree is NULL"

SQLSTATE: XX005

CAUSE: "get KMS JSON tree failed"

ACTION: "check input prarmeter or config.ini file"

**ERRMSG:** "failed to get json tree"

SQLSTATE: XX005

CAUSE: "config.ini json tree error"

ACTION: "check input prarmeter or config.ini file"

**ERRMSG:** "failed to set the value of json tree"

SQLSTATE: XX005

CAUSE: "config.ini json tree error"

ACTION: "check input prarmeter or config.ini file"

**ERRMSG:** "http request failed"

SQLSTATE: XX005

CAUSE: "http request error"

ACTION: "check KMS or IAM connect or config parameter"

**ERRMSG:** "get iam token or iam agency token is NULL"

SQLSTATE: XX005

CAUSE: "connect IAM failed"

ACTION: "check if your env can connect with IAM server"

**ERRMSG:** "KMS dek json key is NULL"

SQLSTATE: XX005

CAUSE: "KMS return value error"

ACTION: "check KMS config paramenter"

**ERRMSG:** "get kms dek is NULL"

SQLSTATE: XX005

CAUSE: "connect KMS failed"

ACTION: "check if your env can connect with KMS server"

**ERRMSG:** "get http header is NULL"

SQLSTATE: XX005

CAUSE: "http request failed"

ACTION: "check IAM config parameter"

**ERRMSG:** "create KMS dek failed"

SQLSTATE: XX005

CAUSE: "KMS error"

ACTION: "check KMS connect or config parameter"

**ERRMSG:** "get KMS dek failed"

SQLSTATE: XX005

CAUSE: "KMS error"

ACTION: "check KMS connect or config parameter"

**ERRMSG:** "get KMS DEK is NULL"

SQLSTATE: XX005

CAUSE: "get KMS dek\_plaintext failed"

ACTION: "check KMS network or cipher is right"

**ERRMSG:** "create matview with TDE failed"

SQLSTATE: 0A000

CAUSE: "TDE feature is not supported for Create materialized views"

ACTION: "check CREATE syntax about create the materialized views"

**ERRMSG:** "failed to add item to the index page"

SQLSTATE: XX002

CAUSE: "System error."

ACTION: "Check WARNINGS for the details."

**ERRMSG:** "index row size %lu exceeds maximum %lu for index '%s'"

SQLSTATE: 54000

CAUSE: "Values larger than 1/3 of a buffer page cannot be indexed."

ACTION: "Consider a function index of an MD5 hash of the value, or use full text indexing."



电话：010-82838118

地址：北京市海淀区学院路30号科大天工大厦B座6层

官网：www.vastdata.com.cn

