GBASE®

GBase 8s SQL 指南：参考

www.gbase.cn

GBase 8s SQL 指南：参考，南大通用数据技术股份有限公司

通讯方式

南大通用数据技术股份有限公司

天津华苑产业区海泰发展六道 6 号海泰绿色产业基地 J 座(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

目　　录

1 SQL 指南：参考 ..... - 1 -
2 系统目录表 ..... - 2 -
2.1 系统目录表跟踪的对象 ..... - 6 -
2.2 使用系统目录 ..... - 6 -
2.2.1 添加到 systables 系统目录表中的行 ..... - 8 -
2.2.2 添加到 syscolumns 或 syscolumnsext 系统目录表中的行 ..... - 8 -
2.2.3 添加到 sysviews 系统目录表中的行 ..... - 9 -
2.2.4 添加到 systabauth 系统目录表中的行 ..... - 9 -
2.2.5 添加到 syscolauth 系统目录表中的行 ..... - 10 -
2.2.6 添加到 sysindexes 或 sysindices 表中的行 ..... - 10 -
2.3 系统目录的结构 ..... - 12 -
2.4 SYSAGGREGATES ..... - 15 -
2.5 SYSAMS ..... - 16 -
2.6 SYSATTRTYPES ..... - 19 -
2.7 SYSAUTOLOCATE ..... - 20 -
2.8 SYSBLOBS ..... - 20 -
2.9 SYSCASTS ..... - 21 -
2.10 SYSCHECKS ..... - 21 -
2.11 SYSCHECKUDRDEP ..... - 22 -
2.12 SYSCOLATTRIBS ..... - 22 -
2.13 SYSCOLAUTH ..... - 23 -
2.14 SYSCOLDEPEND ..... - 24 -
2.15 SYSCOLUMNS ..... - 24 -
2.16 SYSCOLUMNSEXT ..... - 29 -
2.17 SYSCOLCOMMS ..... - 31 -
2.18 SYSCOLCOMMENTS ..... - 31 -
2.19 SYSCOMMAUTH ..... - 32 -
2.20 SYSCOMMS ..... - 32 -
2.21 SYSCOMMENTS ..... - 32 -
2.22 SYSCONSTRAINTS ..... - 32 -
2.23 SYSDEFAULTS ..... - 33 -
2.24 SYSDEFAULTSEXPR ..... - 34 -
2.25 SYSDEPEND ..... - 35 -

2. 26 SYSDIRECTIVES ..... - 35 -
2. 27 SYSDISTRIB ..... - 36 -
2. 28 SYSDOMAINS ..... - 37 -
2. 29 SYSERRORS ..... - 38 -
2. 30 SYSEXTCOLS ..... - 38 -
2. 31 SYSEXTDFILES ..... - 39 -
2. 32 SYSEXTERNAL ..... - 39 -
2. 33 SYSFRAGAUTH ..... - 40 -
2. 34 SYSFRAGDIST ..... - 41 -
2. 35 SYSFRAGMENTS ..... - 42 -
2. 36 SYSINDEXES ..... - 44 -
2. 37 SYSINDICES ..... - 46 -
2. 38 SYSINHERITS ..... - 48 -
2. 39 SYSLANGAUTH ..... - 48 -
2. 40 SYSLOGMAP ..... - 49 -
2. 41 SYSOBJSTATE ..... - 49 -
2. 42 SYSOPCLASSES ..... - 50 -
2. 43 SYSOPCLSTR ..... - 51 -
2. 44 SYSPROCAUTH ..... - 52 -
2. 45 SYSPROCBODY ..... - 52 -
2. 46 SYSPROCCOLUMNS ..... - 53 -
2. 47 SYSPROCEDURES ..... - 54 -
2. 48 SYSPROCPLAN ..... - 56 -
2. 49 SYSREFERENCES ..... - 57 -
2. 50 SYSROLEAUTH ..... - 57 -
2. 51 SYSROUTINELANGS ..... - 57 -
2. 52 SYSSECLABELAUTH ..... - 58 -
2. 53 SYSSECLABELCOMPONENTS ..... - 58 -
2. 54 SYSSECLABELCOMPONENTELEMENTS ..... - 58 -
2. 55 2.48 SYSSECLABELNAMES ..... - 59 -
2. 56 SYSSECLABELS ..... - 59 -
2. 57 SYSSECPOLICIES ..... - 59 -
2. 58 SYSSECPOLICYCOMPONENTS ..... - 60 -
2. 59 SYSSECPOLICYEXEMPTIONS ..... - 60 -

2.60 SYSSEQUENCES ..... - 60 -
2.61 SYSSURROGATEAUTH ..... - 61 -
2.62 SYSSYNONYMS ..... - 61 -
2.63 SYSSYNTABLE ..... - 62 -
2.64 SYSTABAMDATA ..... - 62 -
2.65 SYSTABAUTH ..... - 63 -
2.66 SYSTABLES ..... - 63 -
2.67 SYSTRACECLASSES ..... - 66 -
2.68 SYSTRACEMSGS ..... - 67 -
2.69 SYSTRIGBODY ..... - 67 -
2.70 SYSTRIGGERS ..... - 68 -
2.71 SYSUSERS ..... - 69 -
2.72 SYSVIEWS ..... - 69 -
2.73 SYSVIOLATIONS ..... - 70 -
2.74 SYSXADATASOURCES ..... - 70 -
2.75 SYSXASOURCETYPES ..... - 70 -
2.76 SYSXTDDESC ..... - 71 -
2.77 SYSXTDTYPEAUTH ..... - 71 -
2.78 SYSXTDTYPES ..... - 72 -
2.79 DUAL ..... - 73 -
2.80 信息模式 ..... - 73 -
2.80.1 生成信息模式视图 ..... - 74 -
2.80.2 访问信息模式视图 ..... - 74 -
2.80.3 信息模式视图的结构 ..... - 74 -
3 数据类型 ..... - 78 -
3.1 数据类型的摘要 ..... - 78 -
3.2 ANSI 到 GBase 8s 的数据类型映射 ..... - 82 -
3.3 数据类型的描述 ..... - 83 -
3.3.1 BIGINT 数据类型 ..... - 83 -
3.3.2 BIGSERIAL 数据类型 ..... - 83 -
3.3.3 BLOB 数据类型 ..... - 84 -
3.3.4 BOOLEAN 数据类型 ..... - 85 -
3.3.5 BYTE 数据类型 ..... - 85 -
3.3.6 CHAR(n) 数据类型 ..... - 86 -

3. 3. 7 CHARACTER(n) 数据类型 ..... - 87 -
3. 3. 8 CHARACTER VARYING(m,r) 数据类型 ..... - 87 -
3. 3. 9 CLOB 数据类型 ..... - 88 -
3. 3. 10 DATE 数据类型 ..... - 89 -
3. 3. 11 DATETIME 数据类型 ..... - 89 -
3. 3. 12 DEC 数据类型 ..... - 92 -
3. 3. 13 DECIMAL 数据类型 ..... - 92 -
3. 3. 14 DISTINCT 数据类型 ..... - 94 -
3. 3. 15 DOUBLE PRECISION 数据类型 ..... - 94 -
3. 3. 16 FLOAT(n) 数据类型 ..... - 95 -
3. 3. 17 IDSSECURITYLABEL 数据类型 ..... - 95 -
3. 3. 18 INT 数据类型 ..... - 95 -
3. 3. 19 INT8 数据类型 ..... - 95 -
3. 3. 20 INTEGER 数据类型 ..... - 96 -
3. 3. 21 INTERVAL 数据类型 ..... - 96 -
3. 3. 22 LIST(e) 数据类型 ..... - 98 -
3. 3. 23 LVARCHAR(m) 数据类型 ..... - 99 -
3. 3. 24 MONEY(p,s) 数据类型 ..... - 100 -
3. 3. 25 MULTISET(e) 数据类型 ..... - 101 -
3. 3. 26 NCHAR(n) 数据类型 ..... - 102 -
3. 3. 27 NUMERIC(p,s) 数据类型 ..... - 102 -
3. 3. 28 NVARCHAR(m,r) 数据类型 ..... - 102 -
3. 3. 29 OPAQUE 数据类型 ..... - 103 -
3. 3. 30 REAL 数据类型 ..... - 103 -
3. 3. 31 ROW 数据类型，已命名 ..... - 103 -
3. 3. 32 ROW 数据类型，未命名 ..... - 104 -
3. 3. 33 SERIAL(n) 数据类型 ..... - 106 -
3. 3. 34 SERIAL8(n) 数据类型 ..... - 107 -
3. 3. 35 SET(e) 数据类型 ..... - 108 -
3. 3. 36 SMALLFLOAT 数据类型 ..... - 109 -
3. 3. 37 SMALLINT 数据类型 ..... - 109 -
3. 3. 38 TEXT 数据类型 ..... - 110 -
3. 3. 39 TIMESTAMP 数据类型 ..... - 111 -
3. 3. 40 VARCHAR(m,r) 数据类型 ..... - 111 -

3.4 内置数据类型 - 113 -
3.4.1 字符数据类型 - 113 -
3.4.2 大对象数据类型 - 116 -
3.4.3 时间数据类型 - 117 -
3.4.4 SYS_REFCURSOR 游标类型 - 123 -
3.5 扩展数据类型 - 123 -
3.6 复杂数据类型 - 124 -
3.6.1 单值数据类型 - 127 -
3.6.2 不透明数据类型 - 127 -
3.7 运算符优先级 - 133 -
4 环境变量 - 134 -
4.1 环境变量的类型 - 134 -
4.2 环境变量的限制 - 134 -
4.3 在 UNIX 上使用环境变量 - 135 -
4.3.1 在配置文件中设置环境变量 - 135 -
4.3.2 设置登录时的环境变量 - 136 -
4.3.3 设置环境变量的语法 - 136 -
4.3.4 取消设置环境变量 - 137 -
4.3.5 修改环境变量设置 - 137 -
4.3.6 查看环境变量设置 - 137 -
4.3.7 使用 chkenv 实用程序检查环境变量 - 138 -
4.3.8 环境变量的优先顺序规则 - 138 -
4.4 在 Windows 上使用环境变量 - 139 -
4.4.1 在 Windows 上设置环境变量的位置 - 139 -
4.4.2 环境设置 - 140 -
4.4.3 Windows 环境变量的优先顺序规则 - 141 -
4.5 GBase 8s 产品中的环境变量 - 142 -
4.5.1 ANSIOWNER 环境变量 - 142 -
4.5.2 CPFIRST 环境变量 - 143 -
4.5.3 CMCONFIG 环境变量 - 143 -
4.5.4 DBACCNOIGN 环境变量 - 144 -
4.5.5 DBANSIWARN 环境变量 - 145 -
4.5.6 DBBLOBBUF 环境变量 - 145 -
4.5.7 DBCENTURY 环境变量 - 145 -

4.5.8 DBDATE 环境变量 - 148 -
4.5.9 DBDELIMITER 环境变量 - 150 -
4.5.10 DBEDIT 环境变量 - 151 -
4.5.11 DBFLTMASK 环境变量 - 151 -
4.5.12 DBLANG 环境变量 - 152 -
4.5.13 DBMONEY 环境变量 - 153 -
4.5.14 DBONPLOAD 环境变量 - 154 -
4.5.15 DBPATH 环境变量 - 154 -
4.5.16 DBPRINT 环境变量 - 156 -
4.5.17 DBREMOTECMD 环境变量 (UNIX) - 156 -
4.5.18 DBSPACETEMP 环境变量 - 156 -
4.5.19 DBTEMP 环境变量 - 158 -
4.5.20 DBTIME 环境变量 - 158 -
4.5.21 DBUPSPACE 环境变量 - 161 -
4.5.22 DEFAULT_ATTACH 环境变量 - 161 -
4.5.23 DELIMIDENT 环境变量 - 162 -
4.5.24 ENVIGNORE 环境变量 (UNIX™) - 163 -
4.5.25 FET_BUF_SIZE 环境变量 - 164 -
4.5.26 IFX_DEF_TABLE_LOCKMODE 环境变量 - 164 -
4.5.27 IFX_DIRECTIVES 环境变量 - 165 -
4.5.28 IFX_EXTDIRECTIVES 环境变量 - 166 -
4.5.29 IFX_LARGE_PAGES 环境变量 - 167 -
4.5.30 IFX_LOB_XFERSIZE 环境变量 - 168 -
4.5.31 IFX_LONGID 环境变量 - 168 -
4.5.32 IFX_NETBUF_PVTPOOL_SIZE 环境变量 (UNIX™) - 169 -
4.5.33 IFX_NETBUF_SIZE 环境变量 - 169 -
4.5.34 IFX_NO_SECURITY_CHECK 环境变量 (UNIX™) - 169 -
4.5.35 IFX_NO_TIMELIMIT_WARNING 环境变量 - 170 -
4.5.36 IFX_NODBPROC 环境变量 - 170 -
4.5.37 IFX_NOT_STRICT_THOUS_SEP 环境变量 - 170 -
4.5.38 IFX_ONTAPE_FILE_PREFIX 环境变量 - 171 -
4.5.39 IFX_PAD_VARCHAR 环境变量 - 171 -
4.5.40 IFX_SMX_TIMEOUT 环境变量 - 172 -
4.5.41 IFX_SMX_TIMEOUT_RETRY 环境变量 - 172 -

4.5.42 IFX_UNLOAD_EILSEQ_MODE 环境变量 ..... - 173 -
4.5.43 IFX_UPDDESC 环境变量 ..... - 173 -
4.5.44 IFX_XASTDCOMPLIANCE_XAEND 环境变量 ..... - 173 -
4.5.45 IFX_XFER_SHMBASE 环境变量 ..... - 173 -
4.5.46 IFXRESFILE 环境变量 (Linux) ..... - 174 -
4.5.47 IMCADMIN 环境变量 ..... - 174 -
4.5.48 IMCCONFIG 环境变量 ..... - 174 -
4.5.49 IMCSERVER 环境变量 ..... - 175 -
4.5.50 GBASEDBTC 环境变量 (UNIX™) ..... - 175 -
4.5.51 GBASEDBTCMNAME 环境变量 ..... - 176 -
4.5.52 GBASEDBTCMCONUNITNAME 环境变量 ..... - 176 -
4.5.53 GBASEDBTCONCSMCFG 环境变量 ..... - 176 -
4.5.54 GBASEDBTCONRETRY 环境变量 ..... - 176 -
4.5.55 GBASEDBTCONTIME 环境变量 ..... - 177 -
4.5.56 GBASEDBTCPPMAP 环境变量 ..... - 178 -
4.5.57 GBASEDBTDIR 环境变量 ..... - 178 -
4.5.58 GBASEDBTOPCACHE 环境变量 ..... - 178 -
4.5.59 GBASEDBTSERVER 环境变量 ..... - 179 -
4.5.60 GBASEDBTSHMBASE 环境变量 (UNIX™) ..... - 179 -
4.5.61 GBASEDBTSQLHOSTS 环境变量 ..... - 180 -
4.5.62 GBASEDBTSTACKSIZE 环境变量 ..... - 180 -
4.5.63 GBASEDBTTERM 环境变量 (UNIX) ..... - 180 -
4.5.64 INF_ROLE_SEP 环境变量 ..... - 181 -
4.5.65 ISM_COMPRESSION 环境变量 ..... - 181 -
4.5.66 ISM_DEBUG_FILE 环境变量 ..... - 182 -
4.5.67 ISM_DEBUG_LEVEL 环境变量 ..... - 182 -
4.5.68 ISM_ENCRYPTION 环境变量 ..... - 182 -
4.5.69 ISM_MAXLOGSIZE 环境变量 ..... - 183 -
4.5.70 ISM_MAXLOGVERS 环境变量 ..... - 183 -
4.5.71 JAR_TEMP_PATH 环境变量 ..... - 183 -
4.5.72 JAVA_COMPILER 环境变量 ..... - 183 -
4.5.73 JVM_MAX_HEAP_SIZE 环境变量 ..... - 183 -
4.5.74 LD_LIBRARY_PATH 环境变量 (UNIX™) ..... - 184 -
4.5.75 LIBPATH 环境变量 (UNIX™) ..... - 184 -

4.5.76 NODEFDAC 环境变量 ..... - 184 -
4.5.77 ONCONFIG 环境变量 ..... - 185 -
4.5.78 OPTCOMPIND 环境变量 ..... - 185 -
4.5.79 OPTMSG 环境变量 ..... - 186 -
4.5.80 OPTOFC 环境变量 ..... - 186 -
4.5.81 OPT_GOAL 环境变量 (UNIX™) ..... - 187 -
4.5.82 PATH 环境变量 ..... - 187 -
4.5.83 PDQPRIORITY 环境变量 ..... - 187 -
4.5.84 PLCONFIG 环境变量 ..... - 188 -
4.5.85 PLOAD_LO_PATH 环境变量 ..... - 188 -
4.5.86 PLOAD_SHMBASE 环境变量 ..... - 189 -
4.5.87 PSM_ACT_LOG 环境变量 ..... - 189 -
4.5.88 PSM_CATALOG_PATH 环境变量 ..... - 189 -
4.5.89 PSM_DBS_POOL 环境变量 ..... - 190 -
4.5.90 PSM_DEBUG 环境变量 ..... - 190 -
4.5.91 PSM_DEBUG_LOG 环境变量 ..... - 190 -
4.5.92 PSM_LOG_POOL 环境变量 ..... - 190 -
4.5.93 PSORT_DBTEMP 环境变量 ..... - 191 -
4.5.94 PSORT_NPROCS 环境变量 ..... - 191 -
4.5.95 RTREE_COST_ADJUST_VALUE 环境变量 ..... - 192 -
4.5.96 SHLIB_PATH 环境变量 (UNIX™) ..... - 192 -
4.5.97 SRV_FET_BUF_SIZE 环境变量 ..... - 193 -
4.5.98 STMT_CACHE 环境变量 ..... - 193 -
4.5.99 TERM 环境变量 (UNIX™) ..... - 194 -
4.5.100 TERMCAP 环境变量 (UNIX™) ..... - 194 -
4.5.101 TERMINFO 环境变量 (UNIX) ..... - 195 -
4.5.102 THREADLIB 环境变量 (UNIX™) ..... - 195 -
4.5.103 TZ 环境变量 ..... - 195 -
4.5.104 USETABLENAME 环境变量 ..... - 196 -
5 附录 ..... - 197 -
5.1 stores_demo 数据库 ..... - 197 -
5.2 superstores_demo 数据库 ..... - 198 -

1 SQL 指南：参考

这些主题包含 GBase 8s 中实现的 SQL 语言，GBase 8s 方言的系统目录表、数据类型和环境变量的参考信息。这些主题还包含有关 GBase 8s 随附的 stores_demo、sales_demo 和 superstore_demo 数据库的信息。

本资料主要针对以下用户：

数据库用户
数据库管理员
数据库安全性管理员
数据库应用程序员。

本资料假设您具有以下背景：

具有计算机、操作系统和操作系统提供的实用程序的工作知识。
使用过关系数据库或接触过数据库概念。
具有一些计算机编程经验

本文档具有以下主题：

系统目录表

系统目录由描述数据库结构的表和视图组成。这些表对象有时称为数据字典，它们包含数据库本身的所有信息。每个系统目录表都包含有关数据库中特定元素的信息。每个数据库都有它自己的系统目录。

数据类型

为数据库中的表的每一列指定数据类型。数据类型精确地定义可在该列中存储的值的种类。

环境变量

各种环境变量都会影响 GBase 8s 产品的功能。可设置环境变量来识别终端、指定软件的位置以及定义其他参数。

附录

2 系统目录表

系统目录由描述数据库结构的表和视图组成。这些表对象有时称为数据字典，它们包含数据库本身的所有信息。每个系统目录表都包含有关数据库中特定元素的信息。每个数据库都有它自己的系统目录。

这些主题提供了有关系统目录表的结构、内容和使用的信息。还包含了有关“信息模式”的信息，它提供有关用户会话当前所连接 GBase 8s 实例的所有数据库中表、视图和列的信息。

系统目录表跟踪的对象

- 使用系统目录
- 系统目录的结构
- SYSAGGREGATES
- SYSAMS
  sysams 系统目录表中包含使用内置访问方法和那些由 SQL CREATE ACCESS METHOD 语句创建的访问方法所需的信息。
- SYSATTRTYPES

  SYSAUTOLOCATE
  sysautolocate 系统目录表保留供将来使用。
- SYSBLOBS
- SYSCASTS
- SYSCHECKS
- SYSCHECKUDRDEP
- SYSCOLATTRIBS
  syscolattribs 系统目录表描述智能大对象(即 CLOB 和 BLOB 数据类型)的特征。
- SYSCOLAUTH
- SYSCOLDEPEND
- SYSCOLUMNS
  syscolumns 系统目录表描述数据库中的每个列。
- SYSCOLUMNSEXT

- SYSCOLCOMMS
- SYSCOLCOMMENTS
- SYSCOMMAUTH
- SYSCOMMS
- SYSCOMMENTS
- SYSCONSTRAINTS
- SYSDEFAULTS
  sysdefaults 系统目录表列出了对数据库中的每一列设置的用户定义的缺省值。对于每个用户定义的缺省值，都存在一行。
- SYSDEPEND
- SYSDEFAULTSEXPR
- SYSDIRECTIVES
- SYSDISTRIB
  sysdistrib 系统目录表存储数据分发信息以供查询优化器使用。数据分发为优化器提供了详细的表和列信息以改进 SELECT 语句的执行路径的选择。
- SYSDOMAINS
- SYSERRORS
- SYSEXTCOLS
  sysextcols 系统目录表包含一行，该行描述格式类型 (fmttype) 为 FIXED 的外部表 tabid 中的每个内部列。
- SYSEXTDFILES
  sysextdfiles 系统目录表包含外部表的标识代码和路径。
- SYSEXTERNAL
  对于每个外部表，sysexternal 系统目录表中都存在单个行。
- SYSFRAGAUTH
- SYSFRAGDIST
  sysfragdist 系统目录表会存储分段表和索引的分段级别列统计信息。对于每个表分段或索引分段，都存在一行。

- SYSFRAGMENTS
  sysfragments 系统目录表存储表和索引的个别分段的分段存储信息和 LOW 方式统计信息分发。对于每个表分段或索引分段，都存在一行。
- SYSINDEXES
  sysindexes 表是基于 sysindices 表的视图。它对数据库中的每个索引包含一行。
- SYSINDICES
  sysindices 系统目录表描述数据库中的索引。它存储所有索引的 LOW 方式统计信息，且对在数据库中定义的每个索引包含一行。
- SYSINHERITS
- SYSLANGAUTH
- SYSLOGMAP
- SYSOBJSTATE
- SYSOPCLASSES
- SYSOPCLSTR
  sysopclstr 系统目录表定义数据库中的每个光学集群。在该表中，每个光学集群对应一行。
- SYSPROCAUTH
- SYSPROCBODY
- SYSPROCCOLUMNS
- SYSPROCEDURES
  sysprocedures 系统目录表列出了数据库中注册的每个函数和过程的特征。它对每个例程包含一行。
- SYSPROCPLAN
- SYSREFERENCES
- SYSROLEAUTH
- SYSROUTINELANGS
- SYSSECLABELAUTH
- SYSSECLABELCOMPONENTS
- SYSSECLABELCOMPONENTELEMENTS

- SYSSECLABELNAMES
- SYSSECLABELS
- SYSSECPOLICIES
- SYSSECPOLICYCOMPONENTS
- SYSSECPOLICYEXEMPTIONS
- SYSSEQUENCES
- SYSSURROGATEAUTH
  syssurrogateauth 系统目录表会存储可信用户和代理用户信息。
- SYSSYNONYMS
- SYSSYNTABLE
- SYSTABAMDATA
- SYSTABAUTH
- SYSTABLES
  systables 系统目录表对在数据库（包括系统目录的表和视图）中定义的每个表对象（表、视图、同义词或 GBase 8s 中的序列）包含一行。
- SYSTRACECLASSES
- SYSTRACEMSGS
  systracemsgs 系统目录表存储可在调试用户定义的例程时使用的国际化跟踪消息。
- SYSTRIGBODY
- SYSTRIGGERS
- SYSUSERS
  sysusers 系统目录表列出每单个用户的权限标识，或列出拥有数据库级别访问特权的 PUBLIC 组的公共权限标识。此表还会列出拥有数据库中任何对象访问特权的每个角色的名称。
- SYSVIEWS
- SYSVIOLATIONS
  sysviolations 系统目录表存储有关基本表的约束违例的信息。
- SYSXADATASOURCES
  sysxadatasources 系统目录表存储 XA 数据源。

- SYSXASOURCETYPES
- SYSXTDDESC
- SYSXTDTYPEAUTH
  sysxtdtypeauth 系统目录表标识每个 UDT（用户定义的数据类型）上的特权。
- SYSXTDTYPES
  在 sysxtdtypes 系统目录表中，在数据库中定义的每个 UDT（用户定义的数据类型）对应一个条目，这些类型包括不透明和单值数据类型以及复杂数据类型（命名 ROW 类型、未命名 ROW 类型和 COLLECTION 类型）。
- DUAL
- 信息模式

2.1 系统目录表跟踪的对象

系统目录表维护有关数据库的信息，包括数据库对象的以下类别：

表、视图、同义词和表分段

列、约束、索引和索引分段

表、索引和分段的分发统计信息

表上的触发器和视图上的 INSTEAD OF 触发器

过程、函数、例程和关联的消息

存取数据库对象的授权用户、角色和特权

LBAC 安全策略、组件、标签和豁免权

数据类型和强制转型

用户定义的聚集函数

访问方法和运算符类

序列对象

BLOB 和 CLOB 对象的存储空间

外部优化器伪指令

继承关系

XA 数据源和 XA 数据源类型

可信用户和代理用户信息

2.2 使用系统目录

GBase 8s 在您创建数据库时自动生成系统目录表。可以如同查询数据库中的任何其他

表那样查询系统目录表。新创建的数据库的系统目录表位于称为*数据库空间*的公共磁盘区域中。每个数据库都有它自己的系统目录表。系统目录中的所有表和视图都有前缀 sys（例如：系统目录表 systables）。

并非所有具有前缀 sys 的表都是系统目录表。例如：数据库 syscdr 支持 Enterprise Replication 功能部件。但是，非目录表具有大于等于 100 的 tabid。系统目录表都具有小于 100 的 tabid。有关数据库服务器指定给表、视图、同义词和（GBase 8s 中）序列对象的 tabid 号码的更多信息，请参阅本节后面部分和 SYSTABLES。

提示： 不要混淆数据库的系统目录表与 **sysmaster**、**sysutils**、**syscdr** 或（用于 GBase 8s）**sysadmin** 和 **sysuser** 数据库中的表。这些数据库中的表的名称也具有 **sys** 前缀，但这些表包含有关整个数据库服务器的信息，数据库服务器可管理多个数据库。sysadmin、sysmaster、sysutils、syscdr 和 sysuser 表中的信息主要对数据库服务器管理员 (DBSA) 有帮助。另请参阅《*GBase 8s 管理员指南*》和《*GBase 8s 管理员参考*》。

数据库服务器经常访问系统目录。每次处理 SQL 语句时，数据库服务器都会访问系统目录来确定系统特权、添加或验证表或列名等等。

例如：以下 CREATE SCHEMA 块将 customer 表及其索引和特权添加至 stores_demo 数据库中。此块还添加了一个视图 california，它将 customer 表的数据限制为仅住在 California 的所有客户的客户姓名、公司名称和电话号码。

```
CREATE SCHEMA AUTHORIZATION maryl
CREATE TABLE customer (customer_num SERIAL(101), fname CHAR(15),
    lname CHAR(15), company CHAR(20), address1 CHAR(20), address2
CHAR(20),
    city CHAR(15), state CHAR(2), zipcode CHAR(5), phone CHAR(18))
GRANT ALTER, ALL ON customer TO cathl WITH GRANT OPTION AS maryl
GRANT SELECT ON customer TO public
GRANT UPDATE (fname, lname, phone) ON customer TO nhowe
CREATE VIEW california AS
    SELECT fname, lname, company, phone FROM customer WHERE state =
'CA'
CREATE UNIQUE INDEX c_num_ix ON customer (customer_num)
CREATE INDEX state_ix ON customer (state)
```

要处理此 CREATE SCHEMA 块，数据库服务器首先访问系统目录来验证以下信息：

新表和视图名在数据库中尚未存在。（如果数据库符合 ANSI 标准，那么数据库服务器将验证指定所有者的新名称是否尚不存在。）

用户具有创建表和授予用户特权的许可权。

CREATE VIEW 和 CREATE INDEX 语句中的列名在 customer 表中存在。

除了验证此信息和创建两个新表之外，数据库服务器还将新行添加至下列系统目录表：

systables

syscolumns

syscolumnsext

sysviews

systabauth

syscolauth

sysindexes

sysindices

2.2.1 添加到 systables 系统目录表中的行

运行 CREATE SCHEMA 块之后，会将以下两行新信息添加到 systables 系统目录表中。

列名	第一行	第二行
tabname	customer	california
owner	maryl	maryl
partnum	16778361	0
tabid	101	102
rowsize	134	134
ncols	10	4
nindexes	2	0
nrows	0	0
created	01/26/2007	01/26/2007
version	1	0
tabtype	T	V
locklevel	P	B
npused	0	0
fextsize	16	0
nextsize	16	0
flags	0	0
site		
dbname		

对 **systables** 系统目录表中记录的每个表都指定一个 **tabid**（一个系统指定的顺序号，它唯一地标识数据中的每个表）。系统目录表接收 2 位的 **tabid** 号，而用户创建的表接收以 100 开头的顺序 **tabid** 号。

2.2.2 添加到 syscolumns 或 syscolumnsext 系统目录表中的行

CREATE SCHEMA 块将 14 行添加至 **syscolumns** 系统目录表(GBase 8s 的 **syscolumnsext** 表)。这些行对应于表 **customer** 和视图 **california** 中的列，如以下示例所示。

colname	tabid	colno	coltype	coltypname	collength	colmin	colmax
customer_num	101	1	262	SERIAL	4		
fname	101	2	0	CHAR	15		
lname	101	3	0	CHAR	15		
company	101	4	0	CHAR	20		
address1	101	5	0	CHAR	20		
address2	101	6	0	CHAR	20		
city	101	7	0	CHAR	15		
state	101	8	0	CHAR	2		
zipcode	101	9	0	CHAR	5		
phone	101	10	0	CHAR	18		
fname	102	1	0	CHAR	15		
lname	102	2	0	CHAR	15		
company	102	3	0	CHAR	20		
phone	102	4	0	CHAR	18		

在 **syscolumns** 表中，对表中的每个列都指定一个顺序列号 **colno**，它在列所在的表中唯一地标识该列。在 **colno** 列中，对 **customer** 表的 **fname** 列指定值 2，并对视图 **california** 的 **fname** 列指定值 1。

colmin 和 **colmax** 列是空的。当某一列是索引中的第一个键（或唯一的键）且没有 NULL 值或重复值，并且已运行 UPDATE STATISTICS 语句时，这些列就会包含值。

在 **syscolumnsext** 表中，对表中的每个列类型 **coltype** 指定 **coltypename**，来以字符形式显示列类型，这样用户查询表结构时，就可以直接获得列类型名称。

2.2.3 添加到 sysviews 系统目录表中的行

数据库服务器还将行添加至 **sysviews** 系统目录表中，该表的 **viewtext** 列包含定义视图的 CREATE VIEW 语句的每一行。该列中，在语句中列名前面的 **x0**（例如：**x0.fname**）起别名的作用，用来区分在自连接中使用的相同列。

2.2.4 添加到 systabauth 系统目录表中的行

CREATE SCHEMA 块还将行添加至 **systabauth** 系统目录表。这些行对应于对 **customer** 和 **california** 表授予的用户特权，如以下示例所示。

grantor	grantee	tabid	tabauth
maryl	public	101	su-idx--
maryl	cathl	101	SU-IDXAR
maryl	nhowe	101	--*-----
	maryl	102	SU-ID---

tabauth 列指定授予用户的对 **customer** 和 **california** 表的表级别特权。此列使用 8 字节模式（如 s（选择）、u（更新）、*（列级别特权）、i（插入）、d（删除）、x（索引）、a(改变)和 r(引用))来标识特权的类型。在此示例中，用户 **nhowe** 具有对 **customer** 表的列级别特权。连字符 (-) 表示未向用户授予 **tabauth** 值中由连字符占据其位置的特权。

如果 **tabauth** 特权代码是大写的(例如，表示 Select 的 S)，那么用户具有此特权，并可将该特权授予他人；但是，如果特权代码是小写的（例如：表示 Select 的 s），那么用户不能将该特权授予他人。

2.2.5 添加到 syscolauth 系统目录表中的行

另外，有三行被添加至 **syscolauth** 系统目录表。这些行对应于对 customer 表中的特定列授予的用户特权，如以下示例所示。

grantor	grantee	tabid	colno	colauth
maryl	nhowe	101	2	-u-
maryl	nhowe	101	3	-u-
maryl	nhowe	101	10	-u-

colauth 列指定对 **customer** 表授予的列级别特权。此列使用 3 字节模式（如 s (Select)、u (Update) 和 r (References)）来标识特权类型。例如：用户 **nhowe** 具有对 **customer** 表(由 **tabid** 值 101 指示)的第二个列(因为 **colno** 值是 2)的 Update 特权。

2.2.6 添加到 sysindexes 或 sysindices 表中的行

CREATE SCHEMA 块将两行添加至 **sysindexes** 系统目录表(GBase 8s 的 **sysindices** 表)。这些行对应于对 **customer** 表创建的索引，如在以下示例中所示。

idxname	c_num_ix	state_ix
owner	maryl	maryl
tabid	101	101
idxtype	U	D

idxname	c_num_ix	state_ix
集群		
part1	1	8
part2	0	0
part3	0	0
part4	0	0
part5	0	0
part6	0	0
part7	0	0
part8	0	0
part9	0	0
part10	0	0
part11	0	0
part12	0	0
part13	0	0
part14	0	0
part15	0	0
part16	0	0
levels		
leaves		
nunique		
clust		
idxflags		

在此表中，**idxtype** 列标识创建的索引是需要唯一值 (U) 还是接受重复的值 (D)。例如：**customer.customer_num** 列的 **c_num_ix** 索引是唯一的。

如果在执行查询或其他数据操作语言 (DML) 语句之前使用 UPDATE STATISTICS 语句来更新系统目录，那么可以确保查询执行优化器可用的信息是最新的。

访问系统目录

普通用户对系统目录的访问是只读的。具有 Connect 或 Resource 特权的用户不能变更目录，但他们可以使用标准 SELECT 语句以只读方式来访问系统目录表中的数据。

例如，下面的 SELECT 语句显示数据库中用户创建表的所有表名及对应的 tabid 代码：

```
SELECT tabname, tabid FROM systables WHERE tabid > 99
```

在使用 DB-Access 时，只显示您创建的表。要显示系统目录表，输入以下语句：

```
SELECT tabname, tabid FROM systables WHERE tabid < 100
```

可以使用 **SUBSTR** 或 **SUBSTRING** 函数来仅选择源字符串的一部分。要按列显示表的列表，输入以下语句：

```
SELECT SUBSTR(tabname, 1, 18), tabid FROM systables
```

虽然用户 **gbasedbt** 可以修改大部分系统目录表，但不应该在这些表中更新、删除或插入任何行。修改系统目录表的内容可能会影响数据库的完整性。然而，您可以安全地使用 ALTER TABLE 语句来修改系统目录表的下一个扩展数据块的大小。更改下一个扩展数据块大小不会影响已存在的扩展数据块。

但是，对于 GBase 8s 的某些目录表，将条目添加至系统目录表是有效的。例如，对于 **syserrors** 系统目录表和 **systracemsgs** 系统目录表，DataBlade® 模块开发者可以直接插入位于这些系统目录表中的条目。

更新系统目录数据

如果在执行查询或其他数据操作语言 (DML) 语句之前使用 UPDATE STATISTICS 语句来更新系统目录，那么可以确保查询执行优化器可用的信息是最新的。

在 GBase 8s 中，优化器会为执行 SQL 查询和其他 DML 操作确定最有效的策略。优化器允许您查询数据库而不必全面考虑要先在连接中搜索哪些表或要使用哪些索引。优化器使用来自系统目录中的信息以确定最佳查询策略。

当删除或修改表时，数据库服务器不会自动更新系统目录中的相关统计数据。例如：如果使用 DELETE 语句删除表中的一行或多行，那么 **systables** 系统目录表中用于保存该表行数的 **nrows** 列不会自动更新。

UPDATE STATISTICS 语句会让数据库服务器重新计算 **systables**、**sysdistrib**、**syscolumns** 和 **sysindices** 系统目录表中以及 **sysindexes** 视图中的数据。（对于在 STATLEVEL 属性设置为 FRAGMENT 的分段表上的操作，它还会更新 **sysfragdist** 和 **sysfragments** 系统目录表。）在运行 UPDATE STATISTICS 之后，**systables** 系统目录表就在 **nrows** 列中保存正确的值。如果在运行 UPDATE STATISTICS 时指定 MEDIUM 或 HIGH 方式，那么 **sysdistrib** 表会保存更新的列分布数据。如果在运行 UPDATE STATISTICS 时指定 MEDIUM 或 HIGH 方式，那么 **sysdistrib** 系统目录表会保存更新的列分布数据。对于分段级别的统计信息，**sysfragdist** 系统目录表会保存更新的列分布数据。

每当对数据表进行大量修改时，使用 UPDATE STATISTICS 语句来更新系统目录中的数据。有关 UPDATE STATISTICS 语句的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

2.3 系统目录的结构

以下系统目录表描述了 GBase 8s 数据库中的数据库对象。

系统目录表
SYSAGGREGATES
SYSAMS
SYSATTRTYPES
SYSAUTOLOCATE
SYSBLOBS
SYSCASTS
SYSCHECKS
SYSCHECKUDRDEP
SYSCOLATTRIBS
SYSCOLAUTH
SYSCOLDEPEND
SYSCOLUMNS
SYSCOLUMNSEXT
SYSCOLCOMMS
SYSCOLCOMENTS
SYSCOMMAUTH
SYSCOMMS
SYSCOMMENTS
SYSCONSTRAINTS
SYSDEFAULTS
SYSDEFAULTSEXPR
SYSDEPEND
SYSDIRECTIVES
SYSDISTRIB
SYSDOMAINS
SYSERRORS
SYSEXTCOLS
SYSEXTDFILES
SYSEXTERNAL
SYSFRAGAUTH
SYSFRAGDIST
SYSFRAGMENTS
SYSINDEXES
SYSINDICES
SYSINHERITS
SYSLANGAUTH
SYSLOGMAP
SYSOBJSTATE
SYSOPCLASSES

系统目录表
SYSOPCLSTR
SYSPROCAUTH
SYSPROCBODY
SYSPROCCOLUMNS
SYSPROCEDURES
SYSPROCPLAN
SYSREFERENCES
SYSROLEAUTH
SYSROUTINELANGS
SYSSECLABELAUTH
SYSSECLABELCOMPONENTS
SYSSECLABELCOMPONENTELEMENTS
SYSSECLABELNAMES
SYSSECLABELS
SYSSECPOLICIES
SYSSECPOLICYCOMPONENTS
SYSSECPOLICYEXEMPTIONS
SYSSEQUENCES
SYSSURROGATEAUTH
SYSSYNONYMS
SYSSYNTABLE
SYSTABAMDATA
SYSTABAUTH
SYSTABLES
SYSTRACECLASSES
SYSTRACEMSGS
SYSTRIGBODY
SYSTRIGGERS
SYSUSERS
SYSVIEWS
SYSVIOLATIONS
SYSXADATASOURCES
SYSXASOURCETYPES
SYSXTDDESC
SYSXTDTYPEAUTH
SYSXTDTYPES
DUAL

在使用缺省数据库语言环境的区分大小写数据库（美国英语 ISO 8859-1 代码集）中，

这些表中的字符列为 CHAR 和 VARCHAR 数据类型。对于所有其他语言环境，字符列为 NLS 数据类型(NCHAR 和 NVARCHAR)。有关字符数据类型整理顺序中差别的信息，请参阅《*GBase 8s GLS 用户指南*》。另请参阅本出版物的数据类型章节。

不区分大小写的数据库中的字符列

在使用 NLSCASE INSENSITIVE 关键字创建且使用缺省数据库语言环境（美国英语 ISO 8859-1 代码集）的数据库中，系统目录表中的字符列为 CHAR 和 VARCHAR 数据类型，这些类型支持区分大小写的查询。对于所有其他数据库语言环境，系统目录表中的字符列数据类型为 NLS 数据类型（NCHAR 和 NVARCHAR），但存在以下特定例外情况：

Table_name.Column_name	数据类型
sysams.am_sptype	CHAR(3)
syscolauth.colauth	CHAR(3)
sysdefaults.class	CHAR(1)
sysfragauth.fragauth	CHAR(6)
sysinherits.class	CHAR(1)
syslangauth.langauth	CHAR(1)
sysprocauth.procauth	CHAR(1)
sysprocedures.mode	CHAR(1)
systabauth.tabauth	CHAR(9)
systriggers.event	CHAR(1)
sysxtdtypeauth.auth	CHAR(2)

在上述每个列中，区分大小写的编码可记录数据库服务器实用程序在对这些系统目录表的查询中所需的信息。在不区分大小写的数据库中，如果数据库对象的不同属性编码为相同字母的不同大小写，查询可能从 NCHAR 或 NVARCHAR 列中存储的数据返回不正确的结果。要避免丢失信息，CHAR 数据类型用于上面列出的系统目录列。

2.4 SYSAGGREGATES

sysaggregates 系统目录表记录用户定义的聚集 (UDA)。**sysaggregates** 表具有以下列。

表 1. SYSAGGREGATES 表列描述

列	类型	解释
name	VARCHAR(128)	聚集的名称
owner	CHAR(32)	聚集所有者的名称

列	类型	解释
aggid	SERIAL	标识聚集的唯一代码
init_func	VARCHAR(128)	初始化 UDR 的名称
iter_func	VARCHAR(128)	迭代器 UDR 的名称
combine_func	VARCHAR(128)	组合 UDR 的名称
final_func	VARCHAR(128)	结束 UDR 的名称
handlesnulls	BOOLEAN	NULL 处理指示符： t = 处理 NULL f = 不处理 NULL

每个用户定义的聚集在 **sysaggregates** 中都有一个条目，该条目由其标识代码（**aggid** 值）唯一标识。只有用户定义的聚集（不是内置的聚集）才会在 **sysaggregates** 中具有条目。

对 **aggid** 列的简单索引以及对 **name** 和 **owner** 列的组合索引都需要唯一值。

2.5 SYSAMS

sysams 系统目录表中包含使用内置访问方法和那些由 SQL CREATE ACCESS METHOD 语句创建的访问方法所需的信息。

sysams 表具有以下列。

表 2. SYSAMS 表列描述

列	类型	解释
am_name	VARCHAR(128, 0)	访问方法的名称
am_owner	CHAR(32)	访问方法所有者的名称
am_id	INTEGER	访问方法的唯一标识代码 这对应于 systables、sysindices 和 sysopclasses 表中的 am_id 列。
am_type	CHAR(1)	访问方法的类型：P = 主要；S = 辅助

列	类型	解释
am_sptype	CHAR(3)	访问方法可存在的空间类型： A 意味着访问方法支持外部空间和智能大对象空间。如果访问方法是内置的（例如 B 型树），那么还支持数据库空间。 D 或 d 表示访问方法仅支持数据库空间。 DS 意味着访问方法支持数据库空间和智能大对象空间。 S 或 s 表示访问方法仅支持智能大对象空间。 X 或 x 表示访问方法仅支持外部空间。 sx 意味着访问方法支持智能大对象空间和外部空间。
am_defopclass	INTEGER	缺省运算符类的唯一标识代码 值为 sysopclasses 表中此运算符类的条目中的 opclassid。
am_keyscan	INTEGER	辅助访问方法是否支持键扫描 （如果键扫描能够从对 am_getnext 函数的调用中返回键和行标识，那么该访问方法支持键扫描。）（0 = FALSE；非零 = TRUE）
am_unique	INTEGER	辅助访问方法是否支持唯一键（0 = FALSE；非零 = TRUE）
am_cluster	INTEGER	主访问方法是否支持集群（0 = FALSE；非零 = TRUE）
am_rowids	INTEGER	主访问方法是否支持行标识（0 = FALSE；非零 = TRUE）
am_readwrite	INTEGER	主访问方法是否可读写（0 = 访问方法只可读；非零 = 访问方法可读/写）
am_parallel	INTEGER	访问方法是否支持并行执行（0 = FALSE；非零 = TRUE）

列	类型	解释
am_costfactor	SMALLFLOAT	乘以扫描成本以规范化为针对内置访问方法执行的成本计算的值 扫描成本是 am_scancost 函数的输出。
am_create	INTEGER	为此访问方法的 AM_CREATE 用途函数指定的例程 对于 sysprocedures 表中的例程，值 = procid。
am_drop	INTEGER	为此访问方法的 AM_DROP 用途函数指定的例程
am_open	INTEGER	为此访问方法的 AM_OPEN 用途函数指定的例程
am_close	INTEGER	为此访问方法的 AM_CLOSE 用途函数指定的例程
am_insert	INTEGER	为此访问方法的 AM_INSERT 用途函数指定的例程
am_delete	INTEGER	为此访问方法的 AM_DELETE 用途函数指定的例程
am_update	INTEGER	为此访问方法的 AM_UPDATE 用途函数指定的例程
am_stats	INTEGER	为此访问方法的 AM_STATS 用途函数指定的例程
am_scancost	INTEGER	为此访问方法的 AM_SCANCOST 用途函数指定的例程
am_check	INTEGER	为此访问方法的 AM_CHECK 用途函数指定的例程
am_beginscan	INTEGER	为此访问方法的 AM_BEGINSCAN 用途函数指定的例程
am_endscan	INTEGER	为此访问方法的 AM_ENDSCAN 用途函数指定的例程
am_rescan	INTEGER	为此访问方法的 AM_RESCAN 用途函数指定的例程

列	类型	解释
am_getnext	INTEGER	为此访问方法的 AM_GETNEXT 用途函数指定的例程
am_getbyid	INTEGER	为此访问方法的 AM_GETBYID 用途函数指定的例程
am_build	INTEGER	为此访问方法的 AM_BUILD 用途函数指定的例程
am_init	INTEGER	为此访问方法的 AM_INIT 用途函数指定的例程
am_truncate	INTEGER	为此访问方法的 AM_TRUNCATE 用途函数指定的例程
am_expr_pushdown	INTEGER	保留供将来使用是否支持参数描述符（0 = FALSE；非零 = TRUE）

对于包含用途函数例程的每个列，值为相应例程的 **sysprocedures.procid** 值。

此表中 **am_name** 和 **am_owner** 列的组合索引只允许唯一值。**am_id** 列具有唯一索引。有关访问方法函数的信息，请参阅访问方法的文档。

2.6 SYSATTRTYPES

sysattrtypes 系统目录表包含有关复杂数据类型的成员的信息。**sysattrtypes** 的每一行都包含有关集合数据类型的元素或行数据类型的字段的信息。

sysattrtypes 表具有以下列。

表 3. SYSATTRTYPES 表列描述

列	类型	解释
extended_id	INTEGER	扩展数据类型的标识代码 值与 **sysxtdtypes** 表中的相同（SYSXTDTYPES）。
seqno	SMALLINT	具有 **extended_id** 类型的条目的标识代码
levelno	SMALLINT	集合层次结构中成员的位置
parent_no	SMALLINT	包含此成员的复杂数据类型的 **seqno** 列中的值
fieldname	VARCHAR(128)	行类型中字段的名称 对其他复杂数据类型为 NULL
fieldno	SMALLINT	系统按顺序指定的字段号（在每个行类型中从左到右指定）

列	类型	解释
type	SMALLINT	数据类型的代码 请参阅 **syscolumns.coltype** 的描述（SYSCOLUMNS 页）。
length	SMALLINT	成员的长度（以字节计）
xtd_type_id	INTEGER	标识此数据类型的代码 请参阅 **sysxtdtypes.extended_id** 的描述 (SYSXTDTYPES)。

extended_id 列和 **xtd_type_id** 列的两个索引允许重复值。**extended_id** 和 **seqno** 列的组合索引只允许唯一值。

2.7 SYSAUTOLOCATE

sysautolocate 系统目录表保留供将来使用。

表 4. SYSAUTOLOCATE 表列描述

列	类型	解释
dbsnum	INTEGER	保留供将来使用。
dbsname	VARCHAR(128,0)	保留供将来使用。
pagesize	SMALLINT	保留供将来使用。
flags	INTEGER	保留供将来使用。

2.8 SYSBLOBS

sysblobs 系统目录表指定了 BYTE 和 TEXT 列值的存储位置。其名称基于 BYTE 和 TEXT 列的旧术语 Blob（也称为*简单大对象*），但不是指 GBase 8s 的 BLOB 数据类型。**sysblobs** 表包含了每个 BYTE 或 TEXT 列的一行，并具有以下列。

表 5. SYSBLOBS 表列描述

列	类型	解释
spacename	VARCHAR(128)	分区、数据库空间或系列的名称
type	CHAR(1)	标识存储介质类型的代码：M = 磁标识存储介质类型的代码：M = 磁 O = 光学
tabid	INTEGER	标识表的代码
colno	SMALLINT	列在其表中的列号

tabid 和 **colno** 列的组合索引只允许唯一值。

有关 TEXT、BYTE、BLOB 和 CLOB 列的 Blob 空间、数据库空间和智能大对象空间的块位置和大小的信息，请参阅《*GBase 8s 管理员指南*》和《*GBase 8s 管理员参考*》。

2.9 SYSCASTS

syscasts 系统目录表描述数据库中的强制转型。它对每个内置强制转型、每个隐式强制转型和用户定义的每个显式强制转型包含一行。**syscasts** 表具有以下列。

表 6. SYSCASTS 表列描述

列	类型	解释
owner	CHAR(32)	强制转型的所有者（用户 **gbasedbt** 表示内置强制转型，*用户*名表示隐式和显式强制转型）
argument_type	SMALLINT	对其进行强制转型的源数据类型
argument_xid	INTEGER	在 **argument_type** 列中指定的源数据类型的代码
result_type	SMALLINT	强制转型返回的数据类型的代码
result_xid	INTEGER	在 **result_type** 列中命名的数据类型的数据类型代码
routine_name	VARCHAR(128)	实现强制转型的函数或过程
routine_owner	CHAR(32)	在 **routine_name** 列中指定的函数或过程所有者的名称
class	CHAR(1)	强制转型的类型：E = 显式强制转型 I = 隐式强制转型 S = 内置强制转型

如果 **routine_name** 和 **routine_owner** 具有 NULL 值，那么这表示定义强制转型时未使用例程。如果在 **argument_type** 和 **result_type** 列中指定的数据类型都具有相同的长度和对齐方式，并且都通过引用传递或者都通过值传递，那么会发生这种情况。

列 **argument_type**、**argument_xid**、**result_type** 和 **result_xid** 的组合索引只允许唯一值。列 **result_type** 和 **result_xid** 的组合索引允许重复值。

2.10 SYSCHECKS

syschecks 系统目录表描述在数据库中定义的每个检查约束。由于 **syschecks** 表同时存储 ASCII 文本和二进制编码格式的检查约束，因此它对每个检查约束包含多个行。**syschecks** 表具有以下列。

表 7. SYSCHECKS 表列描述

列	类型	解释

列	类型	解释
constrid	INTEGER	标识约束的唯一代码
type	CHAR(1)	存储检查约束的格式：B = 二进制编码 s = 选择 T = 文本
seqno	SMALLINT	检查约束的行号
checktext	CHAR(32)	检查约束的文本

与类型列中的 B 类型相关联的 **checktext** 列中的文本采用的是计算机可读格式。要查看与特定检查约束相关联的文本，将以下查询与适当的 **constrid** 代码配合使用：

```
SELECT * FROM syschecks WHERE constrid=10 AND type='T'
```

2.11 SYSCHECKUDRDEP

udr_id 和 **constraint_id** 列的组合索引要求这些值的组合是唯一的。 **syscheckudrdep** 系统目录表描述数据库中用户定义的例程 (UDR) 引用的每个检查约束。**syscheckudrdep** 表具有以下列。

表 8. SYSCHECKUDRDEP 表列描述

列	类型	解释
udr_id	INTEGER	标识 UDR 的唯一代码
constraint_id	INTEGER	标识检查约束的唯一代码

在 **syscheckudrdep** 表中描述的每个检查约束在 **sysconstraints** 系统目录表中也有它自己的行，其中 **constrid** 列与 **syscheckudrdep** 的 **constraint_id** 列具有相同的值。

2.12 SYSCOLATTRIBS

syscolattribs 系统目录表描述智能大对象（即 CLOB 和 BLOB 数据类型）的特征。

在该表中，在 CREATE TABLE 语句或 ALTER TABLE 语句的 PUT 子句中引用的每个智能大对象空间对应一行。

表 9. SYSCOLATTRIBS 表列描述

列	类型	解释
tabid	INTEGER	唯一地标识表的代码
colno	SMALLINT	包含智能大对象的列的列号
extentsize	INTEGER	智能大对象扩展数据块中的页，以 KB 表示

列	类型	解释
flags	INTEGER	下列参数的十六进制值的组合（通过相加）的整数表示法： LO_NOLOG (0x00000001 = 1) = 不记录此智能大对象。 LO_LOG (0x00000010 = 2) = 智能大对象的日志记录遵循数据库的当前日志方式。 LO_KEEP_LASTACCESS_TIME (0x00000100 = 4) = 保存用户最近访问此列的时间记录。 LO_NOKEEP_LASTACCESS_TIME (0x00001000 = 8) = 不保存用户最近访问此列的时间记录。 HI_INTEG (0x00010000= 16) = 智能大对象空间数据页具有页眉和页脚，用于检测未完成的写操作和数据损坏。 MODERATE_INTEG (0x00100000= 32) = 数据页具有页眉，但没有页脚。
flags1	INTEGER	保留供将来使用
sbspace	VARCHAR(128)	智能大对象空间的名称

tabid、colno 和 **sbspace** 列的组合索引只允许这三个值的唯一组合。

2.13 SYSCOLAUTH

syscolauth 系统目录表描述对列授予的每组自主访问特权。当前授予用户、角色或数据库中某列上 PUBLIC 组的每组列级别特权各占一行。syscolauth 表具有以下列。

列	类型	解释
grantor	VARCHAR(32)	授权者的授权标识
grantee	VARCHAR(32)	被授权者的授权标识
tabid	INTEGER	唯一地标识表的代码
colno	SMALLINT	表中的列号
colauth	CHAR(3)	指定列特权的 3 字节模式：s *或* S = Select、u *或* U = Update、r *或* R = References

如果 colauth 特权代码是大写的（例如：S 表示 Select），那么具有此特权的用户还可以将此特权授予他人。如果 colauth 特权代码是小写的（例如：s 表示 Select），那么具有此特权的用户不能将该特权授予他人。连字符（ - ）指示在 colauth 模式内的该位置缺少对应的特权。

tabid、grantor、grantee 和 colno 列的组合索引只允许唯一值。tabid 和 grantee 列的组合索引允许重复值。

2.14 SYSCOLDEPEND

syscoldepend 系统目录表跟踪在检查约束和 NOT NULL 约束中指定的表列。由于检查约束可涉及表中的多列，所以 syscoldepend 表可以对每个检查约束包含多行；为约束所涉及的每列创建一行。syscoldepend 表具有以下列。

列	类型	解释
constrid	INTEGER	唯一地标识约束的代码
tabid	INTEGER	唯一地标识表的代码
colno	SMALLINT	表中的列号

constrid、tabid 和 colno 列的组合索引只允许唯一值。tabid 和 colno 列的组合索引允许重复值。

2.15 SYSCOLUMNS

syscolumns 系统目录表描述数据库中的每个列。

对于在表或视图中定义的每一列，都会有一行存在。

表 10. SYSCOLUMNS 表

列	类型	解释
colname	VARCHAR(128)	列名
tabid	INTEGER	包含列的表的标识代码
colno	SMALLINT	列号 系统按顺序指定此列号（在每个表中从左到右）。
coltype	SMALLINT	指示该列的数据类型的代码： 0 = CHAR 1 = SMALLINT 2 = INTEGER 3 = FLOAT 4 = SMALLFLOAT

列	类型	解释
		5 = DECIMAL 6 = SERIAL[1] 7 = DATE 8 = MONEY 9 = NULL 10 = DATETIME 11 = BYTE 12 = TEXT 13 = VARCHAR 14 = INTERVAL 15 = NCHAR 16 = NVARCHAR 17 = INT8 18 = SERIAL8[1] 19 = SET 20 = MULTISET 21 = LIST 22 = ROW（未命名） 23 = COLLECTION 40 = 可变长度不透明类型 [2] 41 = 固定长度不透明类型 [2] 43 = LVARCHAR（仅适用于客户机端） 45 = BOOLEAN 52 = BIGINT 53 = BIGSERIAL [1] 2061 = IDSSECURITYLABEL [2] 4118 = ROW（命名）
collength	任何以下数据类型： 基于整数 可变长度字符 时间 定点 简单大对象 IDSSECURITYLABEL	值决定于列的数据类型。对于某些数据类型，值是列长（以字节为单位）。请参阅存储列长以获取更多信息。
colmin	INTEGER	最小列长（以字节计）
colmax	INTEGER	最大列长（以字节计）
extended_id	INTEGER	在 **coltype** 列中指定的数据类型的数据类型代码（来自 **sysxtdtypes** 表）
seclabelid	INTEGER	安全标号的标号标识与该列相关联（如果该列为受保护的列）。否则，其标号标识为 NULL。
colattr	SMALLINT	**HIDDEN** 1 - 隐藏的列 **ROWVER**

列	类型	解释
		2 - 行版本列 **ROW_CHKSUM** 4 - 行键列 **ER_CHECKVER** 8 - ER 行版本列 **UPGRD1_COL** 16 - ER 自动主键列 **UPGRD2_COL** 32 - ER 自动主键列 **UPGRD3_COL** 64 - ER 自动主键列 **PK_NOTNULL** 128 - NOT NULL by PRIMARY KEY

注：

[1] 在 DB-Access 中，总是将偏移值 256 添加到这些 **coltype** 代码上，因为 DB-Access 会将 SERIAL、SERIAL8 和 BIGSERIAL 列设置为 NOT NULL。

[2] 请参阅不透明数据类型以获取更多信息。

tabid 和 **colno** 列的组合索引只允许唯一值。

coltype 代码可通过显示列的以下特征的位图递增。

位值	设置位时的有效值
0x0100	不允许使用 NULL 值
0x0200	值来自主变量
0x0400	用于联网的数据库服务器的点分十进制
0x0800	DISTINCT 数据类型
0x1000	命名 ROW 类型
0x2000	LVARCHAR 基本类型中的 DISTINCT 类型
0x4000	BOOLEAN 基本类型中的 DISTINCT 类型
0x8000	在客户机系统上处理的集合

例如，命名行类型的 **coltype** 值 4118 是十六进制值 0x1016 的十进制表示法，该十六进制值与设置了命名行类型位的未命名行类型 0x016 的十六进制 **coltype** 值相同。文件 **$GBASEDBTDIR/incl/esql/sqltypes.h** 包含有关 **syscolumns.coltype** 代码的其他信息。

NOT NULL 约束

同样，如果列不允许使用 NULL 值，那么 **coltype** 值将增加 256。要确定这种列的数

据类型，根据可能的 coltype 值，从值中减去 256 并求余数。例如，如果 coltype 值为 262，减去 256 得到余数 6，那么这表示列具有 SERIAL 数据类型。

存储列数据类型

数据库服务器将 coltype 值作为位图存储，如 SYSCOLUMNS 中列示的那样。

不透明数据类型

特定数据类型是由数据库服务器作为内置不透明数据类型来实现的。内置不透明数据类型的类型定义由数据库服务器提供。

内置不透明类型没有唯一的 coltype 值。相反，coltype 值基于不透明类型的类别。以下表列出内置不透明数据类型的 coltype 值：

不透明数据类型的类别	预定义数据类型	coltype 列的值
固定长度不透明类型	BLOB、BOOLEAN 和 CLOB	41
长度可变的不透明类型	LVARCHAR	40
VARCHAR(128) 的 DISTINCT	IDSSECURITYLABEL	2061

使用 sysxtdtypes 系统目录表中的 extended_id 列来区分不同的固定长度不透明类型。

存储列长度

collength 列值取决于列的数据类型。

基于整数的数据类型

BIGINT、BIGSERIAL、DATE、INTEGER、INT8、SERIAL、SERIAL8 或 SMALLINT 列的 **collength** 值与机器无关。数据库服务器对 SQL 语言的这些基于整数的数据类型使用以下长度。

基于整数的数据类型	长度（以字节计）
SMALLINT	2
DATE、INTEGER 和 SERIAL	4
INT8 和 SERIAL8	10
BIGINT 和 BIGSERIAL	8

可变长度字符数据类型

对于 LVARCHAR 类型的 GBase 8s 列，collength 具有来自数据类型声明的 *max* 值或 2048（如果没有指定最大值）。

对于 VARCHAR 或 NVARCHAR 列，*max_size* 和 *min_space* 值是使用以下某个公式编码在 collength 列中：

如果 collength 值是正的：

collength = (*min_space* * 256) + *max_size*

如果 collength 值是负的：

collength + 65536 = (*min_space* * 256) + *max_size*

时间数据类型

如前面所述，DATE 列在 collength 列中的值为 4。

对于类型为 DATETIME 或 INTERVAL 的列，collength 是使用以下公式确定的：

(*length* * 256) + (*first_qualifier* * 16) + *last_qualifier*

长度是 DATETIME 或 INTERVAL 字段的实际长度，*first_qualifier* 和 *last_qualifier* 具有下表所显示的值。

字段限定符	值	字段限定符	值
YEAR	0	FRACTION(1)	11
MONTH	2	FRACTION(2)	12
DAY	4	FRACTION(3)	13
HOUR	6	FRACTION(4)	14
MINUTE	8	FRACTION(5)	15
SECOND	10		

例如：如果 DATETIME YEAR TO MINUTE 列长度为 12（例如：YYYY:DD:MO:HH:MI），*first_qualifier* 值为 0（用于 YEAR），并且 *last_qualifier* 值为 8（用于 MINUTE），那么 collength 值为 3080（由 (256 * 12) + (0 * 16) + 8 计算所得）。

定点数据类型

MONEY 或 DECIMAL(*p*,*s*) 列的 collength 值可使用以下公式来计算：

(*precision* * 256) + *scale*

简单大对象数据类型

如果列的数据类型为 BYTE 或 TEXT，那么 collength 保存描述符的长度。

存储最大值和最小值

colmin 和 **colmax** 值分别保存该列中第二小和第二大的数据值。例如：如果索引列中的各个值为 1、2、3、4 和 5，那么 **colmin** 值为 2 而 **colmax** 值为 4。存储第二小的数据值和第二大的数据值会让查询优化器对列中的值范围进行假设并进一步依次优化搜索策略。

仅当对列建立了索引并且 UPDATE STATISTICS 语句显式或隐式地计算了列分发时，**colmin** 和 **colmax** 列才会包含值。如果在表空间中存储 BYTE 或 TEXT 数据，那么 **colmin** 值会编码为 -1。

colmin 和 **colmax** 列仅对适合四个字节的数据类型有效：SMALLFLOAT、SMALLINT、INTEGER 以及 CHAR 的前四个字节。所有其他非整数列类型的值是最大值或最小值的头四个字节，它们被视为整数。

使用 UPDATE STATISTICS MEDIUM 比依赖于 **colmin** 和 **colmax** 值要好。UPDATE STATISTICS MEDIUM 提供了更好的信息并且对于所有数据类型都有效。

GBase 8s 不计算用户定义的数据类型的 **colmin** 和 **colmax** 值。但是，如果用户定义的辅助访问方法提供了用户定义的数据类型，那么这些列具有用户定义的数据类型的值。

2.16 SYSCOLUMNSEXT

syscolumnsext 表是基于 **syscolumns** 系统目录表的视图。它对于在表或视图中定义的每一列，都会有一行存在。

syscolumnsext 表具有以下列：

列	类型	解释
colname	VARCHAR(128)	列名
tabid	INTEGER	包含列的表的标识代码
colno	SMALLINT	列号 系统按顺序指定此列号（在每个表中从左到右）。
coltype	SMALLINT	指示该列的数据类型的代码。 有关代码的详细信息，请参阅 syscolunms 的 coltype 列。
coltypename	LVARCHAR（2048）	数据类型名称。Oracle 兼容格式。 CHAR、SMALLINT 、 INTEGER、FLOAT、 SMALLFLOAT 、DECIMAL、 SERIAL 、DATE 、 MONEY 、NULL 、 DATETIME、BYTE 、 TEXT 、 VARCHAR 、 INTERVAL、 NCHAR 、NVARCHAR、INT8 、 SERIAL8 、SET 、 MULTISET 、LIST 、 ROW、COLLECTION 、 BOOL 、ROWREF、 BIGINT 、BIGSERIAL。
coltypename2	LVARCHAR(2048)	数据类型名称。特定于 GBase 8s 格式。 指示该列的数据类型可为： CHAR、SMALLINT 、 INTEGER、FLOAT、 SMALLFLOAT 、DECIMAL、 SERIAL 、DATE 、

列	类型	解释
		MONEY 、NULL 、 DATETIME [1] （特定于 G8ase 8s）、 BYTE 、 TEXT 、VARCHAR 、 INTERVAL [2]（特定于 G8ase 8s）、 NCHAR 、NVARCHAR、INT8 、 SERIAL8 、SET 、 MULTISET 、LIST 、 ROW、COLLECTION 、 BOOLEAN 、ROWREF、 BIGINT 、BIGSERIAL。
collength	任何以下数据类型： 基于整数 可变长度字符 时间 定点 简单大对象 IDSSECURITYLABEL	值决定于列的数据类型。对于某些数据类型，值是列长（以字节为单位）。请参阅存储列长以获取更多信息。
colmin	INTEGER	最小列长（以字节计）
colmax	INTEGER	最大列长（以字节计）
extended_id	INTEGER	在 **coltype** 列中指定的数据类型的数据类型代码（来自 **sysxtdtypes** 表）
seclabelid	INTEGER	安全标号的标号标识与该列相关联（如果该列为受保护的列）。否则，其标号标识为 NULL。
colattr	SMALLINT	**HIDDEN** 1 - 隐藏的列 **ROWVER** 2 - 行版本列 **ROW_CHKSUM** 4 - 行键列 **ER_CHECKVER** 8 - ER 行版本列 **UPGRD1_COL** 16 - ER 自动主键列 **UPGRD2_COL** 32 - ER 自动主键列 **UPGRD3_COL** 64 - ER 自动主键列

列	类型	解释
		PK_NOTNULL 128 - NOT NULL by PRIMARY KEY

注：

[1] 请参阅本手册第三章中 DETATIME 数据类型以获取更多信息。

[2] 请参阅本手册第三章中 INTERVAL 数据类型以获取更多信息。

此表是 GBase 8s 数据库基于 SYSCOLUMNS 的扩展。SYSCOLUMNS 仍可以作为系统目录表使用，您可对 SYSCOLUMNS 进行访问，请参阅 SYSCOLUMNS。它对于新增的视图和表的一列，都存在一行，并指示列的数据类型名称。

新增的 **coltypename** 列和 **coltypename2** 列提供对数据类型以字符形式显示 **coltype** 的支持。**coltypename** 兼容 **oracle** 数据类型格式。而 **coltypename2** 特定于 **GBase 8s** 格式。

2.17 SYSCOLCOMMS

syscolcomms 系统目录表用于存储列的注释信息。对于每个添加注释的列，都存在一行。如果从其所属表删除该列，则此行信息同步删除。

syscolcomms 表具有以下列：

列	数据类型	解释
tabint	CHAR(32)	表名
colno	SMALLINT	列 ID
comments	NVARCHAR(255)	用户为列添加注释的内容

2.18 SYSCOLCOMMENTS

syscolcomments 表是基于 **syscolcomms** 表的视图。它对于每个添加注释的列，都存在一行。如果从其所属表删除该列，则此行信息同步删除。

syscolcomments 表具有以下列：

列	数据类型	解释
tabname	CHAR(32)	表名
colname	CHAR(32)	列名
comments	NVARCHAR(255)	用户为列添加注释的内容

要查看已添加注释的列的信息，可以对该表执行 SELECT 语句。

2.19 SYSCOMMAUTH

syscommauth 系统目录表描述对表、视图或列授予的注释权限。对于授予注释权限的用户，都存在一行。 如果移除此用户的 COMMENT 权限，则从此表删除对应的一行。

syscommauth 表具有以下列：

字段	数据类型	解释
username	CHAR(32)	用户名

2.20 SYSCOMMS

syscomms 系统目录表用于存储表、视图注释的信息。对于每个要添加注释的表或视图，都存在一行。如果删除该表或视图，则此行信息同步删除。

syscomms 表具有以下列：

字段	数据类型	解释
tabid	INTEGER	唯一地标识表的代码
comments	NVARCHAR(255)	用户为表或视图添加的注释

2.21 SYSCOMMENTS

syscomments 表是基于 **syscomms** 系统目录表的视图。它对于每个要添加注释的表或视图，都存在一行。如果删除该表或视图，则此行信息同步删除。

syscomments 表具有以下列：

字段	数据类型	解释
tabname	CHAR(32)	表名、视图名
comments	NVARCHAR(255)	用户为表或视图添加的注释
tabtype	CHAR(1)	标识对象类型的代码： T = 表 V = 视图

要查看用户表的注释信息，可以对该表执行 SELECT 语句。

2.22 SYSCONSTRAINTS

sysconstraints 系统目录表列出了对每个数据库表中的各个列设置的约束。在 **sysindexes** 系统目录表(或 GBase 8s 的 **sysindices** 视图)中对在 **sysindexes** 或 **sysindices** 中尚未有对应条目的每个唯一的主键或引用约束也设置了一个条目。由于索引可以共享，所以多个约束可以与一个索引相关联。**sysconstraints** 表具有以下列。

表 11. SYSCONSTRAINTS 表列描述

列	类型	解释
constrid	SERIAL	唯一地标识约束的代码
constrname	VARCHAR(128)	约束的名称
owner	VARCHAR(32)	约束所有者的名称
tabid	INTEGER	唯一地标识表的代码
constrtype	CHAR(1)	标识约束类型的代码： C = 检查约束 N = Not NULL P = 主键 R = 引用 T = 表 U = 唯一
idxname	VARCHAR(128)	与约束相对应的索引的名称
collation	CHAR(32)	创建约束时的排列顺序。

constrname 和 **owner** 列的组合索引只允许唯一值。**tabid** 列的索引允许重复值，但 **constrid** 列的索引只允许唯一值。

对于检查约束（其中 **constrtype** = C），**idxname** 始终为 NULL。有关每个检查约束的其他信息包括在 **syschecks** 和 **syscoldepend** 系统目录表中。

2. 23 SYSDEFAULTS

sysdefaults 系统目录表列出了对数据库中的每一列设置的用户定义的缺省值。对于每个用户定义的缺省值，都存在一行。

sysdefaults 表具有以下列：

表 12. SYSDEFAULTS 表列描述

列	类型	解释
tabid	INTEGER	唯一地标识表的代码。当 class 列包含代码 P 时，tabid 列会引用过程标识，而不是表标识。
colno	SMALLINT	唯一地标识列的代码。

列	类型	解释
type	CHAR(1)	标识缺省值类型的代码：C = 当前 L = 文字值 N = NULLS = Dbservername *或* Sitename T = 今天 U = 用户
default	CHAR(256)	如果 **sysdefaults.type** = L，那么为文字缺省值。
class	CHAR(1)	标识列种类的代码：T = 表 t = ROW 类型 P = 过程

如果 CREATE TABLE 或 ALTER TABLE 语句中未明确指定缺省值，那么 **sysdefaults** 表中不存在该列的条目。

如果指定文字作为缺省值，那么它将以 ASCII 文本的形式存储在**缺省**列中。如果文字值不是下一段中列出的数据类型之一，那么 **default** 列由两部分组成。第一部分是缺省值结构的二进制值的 6 位表示法。第二部分是 ASCII 文本形式的缺省值。这两个部分由空格隔开。

如果列的数据类型不是 CHAR、NCHAR、NVARCHAR 或 VARCHAR 或者（用于 GBase 8s）BOOLEAN 或 LVARCHAR，那么在**缺省**列中对缺省值的二进制表示进行编码。

tabid、**colno** 和 **class** 列的组合索引只允许唯一值。

2.24 SYSDEFAULTSEXPR

sysdefaultsexpr 系统目录表存储 **default** 的表达式信息，它列出了对数据库中的每一列定义缺省值时，为其设置的函数的内容。

sysdefaultsexpr 表具有以下列：

表 12. SYSDEFAULTSEXPR 表列描述

列	类型	解释
tabid	INTEGER	唯一地标识表的代码。
colno	SMALLINT	唯一地标识列的代码。
type	CHAR(1)	标识存储类型的代码： 'T' 为原始文档的信息； 'B' 为表达式二进制内容。
seqno	SMALLINT	序号从 0 开始，表示同类信息分割后的顺序号。
default	CHAR(32)	存储相应的 **default** 内容。

如果 CREATE TABLE 或 ALTER TABLE 语句中未明确为缺省值设置函数，那么

sysdefaultsexpr 表中不存在该列的条目。

该表存储用户输入的表达式的两种形式：文本和二进制。文本内容为用户 SQL 的原始信息，用于在系统表中可视化查询相关信息。二进制为 default 表达式语法解析后生成的 statement 数据结构再经过一定编码处理后的信息，此信息用于使用 default 表达式进行求值。

2.25 SYSDEPEND

sysdepend 系统目录表描述每个视图或表与其他视图或表的依赖性。对于每个依赖性，此表中都存在一行，所以基于三个表的视图有三行。**sysdepend** 表具有以下列。

表 13. SYSDEPEND 表列描述

列	类型	解释
btabid	INTEGER	唯一地标识基本表或视图的代码
btype	CHAR(1)	基本对象类型：T = 表 V = 视图
dtabid	INTEGER	唯一地标识从属表或视图的代码
dtype	CHAR(1)	从属对象的类型的代码；当前，只实现了视图（V = 视图）

建立了 **btabid** 和 **dtabid** 列的索引，并且这些列允许重复值。

2.26 SYSDIRECTIVES

Sysdirectives 表存储可应用于查询的外部优化器伪指令。客户机应用程序中的查询是否能够使用这些优化器伪指令取决于客户机系统上环境变量 IFX_EXTDIRECTIVES 的设置（如第 3 章所述），并取决于数据库服务器的配置文件中的 EXT_DIRECTIVES 设置。

sysdirectives 表具有以下列：

表 14. SYSDIRECTIVES 表列描述

列	类型	解释
id	SERIAL	标识优化器伪指令的唯一代码
query	TEXT	查询的文本（如其在应用程序中所存在的一样）
directives	TEXT	优化器伪指令的文本（没有注释）
active	SMALLINT	整数代码，它标识此条目是有效 (= 1) 还是仅测试 (= 2)
hash_code	SMALLINT	仅供内部使用

NULL 值在 **query** 列中无效。在 id 列上有唯一索引。

2.27 SYSDISTRIB

sysdistrib 系统目录表存储数据分发信息以供查询优化器使用。数据分发为优化器提供了详细的表和列信息以改进 SELECT 语句的执行路径的选择。

sysdistrib 表具有以下列。

表 15. SYSDISTRIB 表列描述

列	类型	解释
tabid	INTEGER	标识从中收集数据值的表的代码
colno	SMALLINT	源表中的列号
seqno	INTEGER	多个条目的顺序号码
constructed	DATETIME YEAR TO FRACTION(5)	创建数据分发时的日期
mode	CHAR(1)	优化级别：M = 中 H = 高
resolution	SMALLFLOAT	在 UPDATE STATISTICS 语句中指定
confidence	SMALLFLOAT	在 UPDATE STATISTICS 语句中指定
encdat	STAT	统计信息
type	CHAR(1)	统计信息的类型： A = **encdat** 在固定长度字符字段中具有 ASCII 编码的直方图 S = **encdat** 具有用户定义的统计信息
smplsize	SMALLFLOAT	大于 0 且不超过 1.0 的值表示 UPDATE STATISTICS 对表中总行数采样的比例。大于 1.0 的值表示 UPDATE STATISTICS 所采样的实际使用行数。值 0 表示未指定采样大小。UPDATE STATISTICS HIGH 总是更新所有行的统计信息。
rowssampled	FLOAT	样本中的行数
constr_time	DATETIME YEAR TO FRACTION(5)	记录分发的时间
ustnrows	FLOAT	计算分发时分段中的行。

列	类型	解释
ustbuildduration	INTERVAL HOUR TO FRACTION(5)	计算此列的分发统计信息所花费的时间
nupdates	FLOAT	表的更新数
ndeletes	FLOAT	表的删除数
ninserts	FLOAT	表的插入数

当以方式 MEDIUM 或 HIGH 对某张表执行 UPDATE STATISTICS 语句时，会将信息存储在 **sysdistrib** 表中。（UPDATE STATISTICS LOW 不会将值插入到 **mode** 列中。）

只有用户 **gbasedbt** 才能选择 **encdat** 列。

sysdistrib 系统目录表中的每一行都使用 **tabid** 和 **colno** 列（为它们收集统计信息）作为键。

对于内置数据类型的列，将**类型**字段设置为 A。**encdat** 列存储用 ASCII 编码的直方图，此直方图分为多行，每行包含 256 个字节。

在 GBase 8s 中，对于用户定义的数据类型的列，**type** 字段设置为 S。**encdat** 列以多重表示格式存储由 **statcollect** 用户定义的例程收集的统计信息。对每个 **tabid** 和 **colno** 对只存储一行。**tabid**、**colno** 和 **seqno** 列的组合索引只允许这些值的唯一组合。

以下三个 DML 计数器列会记录在生成列分发统计信息时对表执行修改数据行的 DML 操作的计数：

nupdates 中的 UPDATE 操作数

ndeletes 中的 DELETE 操作数

和 ninserts 中的 INSERT 操作数

这些计数还可包括 MERGE 语句修改的行。

这些 DML 计数器列会存储在生成分发统计信息时存在的服务器分区中的计数器值。如果 AUTO_STAT_MODE 配置参数、AUTO_STAT_MODE 会话环境设置或 UPDATE STATISTICS 语句的 AUTO 关键字已启用数据分发统计信息的选择性更新，那么 ninserts、ndeletes 和 ninserts 值可影响 UPDATE STATISTICS 操作是否刷新现有数据分发统计信息。当以 MEDIUM 或 HIGH 方式对表运行 UPDATE STATISTICS 语句时，数据库服务器会比较这些列中的已存储值与分区中的当前值。如果已存储值之和不同于分区页面中的这些当前 sysdistrib DML 计数器值之和，少于 STATCHANGE 表属性或 STATCHANGE 配置参数的设置所指定的阈值，那么不会更新表的列分发统计信息。

2.28 SYSDOMAINS

不使用 sysdomains 视图。它显示其他系统目录表的列。它具有以下列。

表 16. SYSDOMAINS 表列描述

列	类型	解释
id	SERIAL	标识域的唯一代码
owner	CHAR(32)	域所有者的名称
name	VARCHAR(128)	域的名称
type	SMALLINT	标识域的类型的代码

此视图没有索引。

2.29 SYSERRORS

syserrors 系统目录表存储有关错误、警告和参考消息（由使用 **mi_db_error_raise()** DataBlade API 函数的 DataBlade 模块和用户定义的例程返回）的信息。

syserrors 表具有以下列。

列	类型	解释
sqlstate	CHAR(5)	与错误关联的 SQLSTATE 值。
locale	CHAR(36)	与此版本的消息相关联的语言环境（例如：en_us.8859-1）
level	SMALLINT	保留供将来使用
seqno	SMALLINT	保留供将来使用
message	VARCHAR(255)	消息文本

要创建新的消息，将一行直接插入 **syserrors** 表中。缺省情况下，所有用户都可以查看此表，但只有具有 DBA 特权的用户才能对其进行修改。

sqlstate、**locale**、**level** 和 **seqno** 列的组合索引只允许使用唯一值。

2.30 SYSEXTCOLS

sysextcols 系统目录表包含一行，该行描述格式类型 (fmttype) 为 FIXED 的外部表 tabid 中的每个内部列。

sysextcols 表具有以下列。

列	类型	解释
tabid	INTEGER	表的唯一标识代码
colno	SMALLINT	标识列的代码
exttype	SMALLINT	标识外部列类型的代码
extstart	SMALLINT	外部数据文件中列的起始位置
extlength	SMALLINT	外部列长（以字节计）

列	类型	解释
nullstr	CHAR(256)	在外部数据中表示 NULL
decprec	SMALLINT	外部小数的精度
extstype	VARCHAR(128,0)	外部类型名

对于 DELIMITED 或 GBase 8s 格式外部文件，sysextcols 中没有存储任何条目。

可使用 DBSCHEMA 实用程序来编写外部表的描述。要查询有关外部表的这些系统目录表，请使用存储在 systables 中并且 tabtype ="E"的 tabid。

tabid 列的索引允许重复值。

2.31 SYSEXTDFILES

sysextdfiles 系统目录表包含外部表的标识代码和路径。

对于每个外部表，具有以下列的 sysextdfiles 系统目录表中至少存在一行。

列	类型	解释
tabid	INTEGER	外部表的唯一标识代码
dfentry	CHAR(598)	源或目标文件绝对路径
blobdir	CHAR(473)	绝对或相对目录名称
clobdir	CHAR(473)	绝对或相对目录名称

可使用 DBSCHEMA 来编写外部表的描述。要查询有关外部表的这些系统目录表，请使用存储在 **systables** 中并且 **tabtype** ="E"的 **tabid**。

tabid 列的索引允许重复值。

2.32 SYSEXTERNAL

对于每个外部表，**sysexternal** 系统目录表中都存在单个行。

tabid 列使此系统目录表中的外部表记录与 **systables** 中的条目相关联。

列	类型	解释
tabid	INTEGER	外部表的唯一标识代码
fmttype	CHAR(1)	格式的类型：D =（定界）F =（固定）I = (GBase 8s)
recdelim	VARCHAR(128)	记录定界符
flddelim	CHAR(4)	字段定界符
datefmt	CHAR(8)	保留供将来使用
moneyfmt	CHAR(20)	保留供将来使用
maxerrors	INTEGER	允许的错误数

列	类型	解释
rejectfile	CHAR(464)	拒绝文件的名称
flags	INTEGER	可选**装入**标志
ndfiles	INTEGER	**sysextdfiles** 中的数据文件数

可使用 **dbschema** 实用程序来编写外部表的描述。要查询有关外部表的这些系统目录表，请使用存储在 **systables** 中并且 **tabtype** = “E” 的 **tabid**。

tabid 列的索引只允许唯一值。

2.33 SYSFRAGAUTH

sysfragauth 系统目录表存储有关对表分段授予的特权的信息。此表具有以下列。

表 17. SYSFRAGAUTH 表列描述

列	类型	解释
grantor	CHAR(32)	特权授权者的名称
grantee	CHAR(32)	特权被授权者的名称
tabid	INTEGER	标识分段表的代码
fragment	VARCHAR(128)	存储分段的数据库空间的名称
fragauth	CHAR(6)	指定分段特权的 6 字节模式（包括保留供将来使用的 3 个字节）： u *或* U = Update i *或* I = Insert d *或* D = Delete

在 **fragauth** 列中，大写代码（例如：U 表示 Update）意味着被授权者可以将该特权授予其他用户；小写（例如：u 表示 Update）意味着用户不能将该特权授予他人。连字符（ - ）指示 tabauth 模式内该位置缺少对应的特权。

tabid、grantor、grantee 和 **fragment** 列的组合索引只允许唯一值。**tabid** 和 **grantee** 列的组合索引允许重复值。

以下示例显示了一个基本表的分段级别特权（这些特权存在于 **sysfragauth** 表中）。在此示例中，被授权者 **rajesh** 可将 Update、Delete 和 Insert 特权授予其他用户。

grantor	grantee	tabid	fragment	fragauth
dba	omar	101	dbsp1	-ui---
dba	jane	101	dbsp3	--i---

grantor	grantee	tabid	fragment	fragauth
dba	maria	101	dbsp4	--id--
dba	rajesh	101	dbsp2	-UID--

2.34 SYSFRAGDIST

sysfragdist 系统目录表会存储分段表和索引的分段级别列统计信息。对于每个表分段或索引分段，都存在一行。

此处仅描述分段表中的列。(对于表级别列统计信息,请参阅 **sysdistrib** 系统目录表。)

sysfragdist 表具有以下列。

列	类型	解释
tabid	INTEGER	表的唯一标识代码 (= systables.tabid)
fragid	INTEGER	分段的唯一标识代码 (= sysfragments.partnum)
colno	SMALLINT	列的唯一标识代码 (= syscolumns.colno)
seqno	SMALLINT	序号（用于跨多行的分发）
mode	CHAR(1)	UPDATE STATISTICS 方式（H = 高，或 M = 中）
resolution	SMALLFLOAT	每个二进制文件中样本的平均百分比
confidence	SMALLFLOAT	MEDIUM 方式样本值相当于确切 HIGH 方式结果的估算可能性
rowssampled	FLOAT	样本中的行数
ustbuildduration	INTERVAL HOUR TO FRACTION(5)	计算此列分发所花费的时间
constr_time	DATETIME YEAR TO FRACTION(5)	记录分发的时间
ustnrows	FLOAT	计算分发时分段中的行。
minibinsize	FLOAT	仅供内部使用
nupdates	FLOAT	表的更新数
ndeletes	FLOAT	表的删除数
ninserts	FLOAT	表的插入数
encdist	BLOB	加密分段分发
sbsnum	INTEGER	存储 encdist 的智能大对象空间的唯一标识代码
version	INTEGER	保留供将来使用

具有给定 **tabid**、**fragid** 和 **colno** 值组合的行集用于标识该表分段的列统计信息。通

过将 **seqno** 列用于排序号，这些统计信息可跨多行。

在计算分段列统计信息的 UPDATE STATISTICS MEDIUM 或 HIGH 语句中指定的 *mode*、*resolution* 和 *confidence* 值会记录在相同名称的 **sysfragdist** 列中。要使用现有分段统计信息来构建表统计信息，这三个参数在引用相同表的分段的 UPDATE STATISTICS 语句之间不应该更改。唯一例外是“H”方式分段统计信息可用于构建“M”方式表统计信息。

分段的列分发统计信息存储在 BLOB 列 **encdist** 中。**sbsnum** 列存储了描述此分段的 **encdist** 对象所存储在的智能 blob 空间的标识代码。缺省情况下，SBSPACENAME 配置参数设置是标识代码在 **sbsnum** 列中的智能大对象空间的标识。

以下三列会记录生成列分发统计信息时对分段执行修改数据行的 DML 操作的计数：

nupdates 中的 UPDATE 操作数

ndeletes 中的 DELETE 操作数

和 ninserts 中的 INSERT 操作数

这些计数还可包括 MERGE 语句修改的行。

这些 DML 计数器列会存储在生成分发统计信息时存在的服务器分区中的计数器值。当以 MEDIUM 或 HIGH 方式对具有分段级别统计信息的分段表运行 UPDATE STATISTICS 时，数据库服务器会比较这些列中的已存储值与分区中的当前值。

如果 AUTO_STAT_MODE 配置参数、AUTO_STAT_MODE 会话环境设置或 UPDATE STATISTICS 语句的 AUTO 关键字已启用数据分发统计信息的选择性更新，那么 **ninserts**、**ndeletes** 和 **ninserts** 值可影响 UPDATE STATISTICS 操作是否刷新分段的现有数据分发统计信息。如果已存储值之和不同于分区页面的这些当前 DML 计数器值之和，少于 STATCHANGE 表属性或 STATCHANGE 配置参数的设置所指定的阈值，那么与 **sysfragdist** 表中行对应的分段列统计信息不会更新。

2.35 SYSFRAGMENTS

sysfragments 系统目录表存储表和索引的个别分段的分段存储信息和 LOW 方式统计信息分发。对于每个表分段或索引分段，都存在一行。

sysfragments 表具有以下列。

列	类型	解释
fragtype	CHAR(1)	指示分段对象类型的代码： I = 原始索引分段 T = 原始表分段
tabid	INTEGER	表的唯一标识代码
indexname	VARCHAR(128)	索引的名称

列	类型	解释
colno	INTEGER	TEXT 或 BYTE 列的标识代码或滚动窗口分段数上限
partn	INTEGER	物理存储位置的标识代码
strategy	CHAR(1)	分段分布策略的类型的代码： R = 循环分段存储策略 E = 基于表达式的分段存储策略 I = IN DBSPACE 子句指定作为分段存储策略一部分的存储位置 N = 时间间隔（或滚动窗口）分段存储策略 N = 时间间隔分段存储策略 L = 列表分段存储策略 T = 基于表的分段存储策略 H = 表是表层次结构内的子表
location	CHAR(1)	保留供将来使用；对于本地，显示 L
servername	VARCHAR(128)	保留供将来使用
evalpos	INTEGER	分段存储列表中的分段位置。 对于按 INTERVAL 的分段存储，为指示 exprtext 字段中信息类型的以下某个值： -1 = 时间间隔分段的数据库空间列表 -2 = 时间间隔值 -3 = 分段存储键 -4 = 滚动窗口分段 按 LIST 的分段存储也使用值 -3。
exprtext	TEXT	分段存储策略的表达式 对于按 INTERVAL、LIST 或滚动窗口进行的分段存储，提供与 evalpos 字段值对应的信息。 对于按 INTERVAL 或 LIST 的分段存储，提供与 evalpos 字段值对应的信息。
exprbin	BYTE	表达式的二进制版本

列	类型	解释
exprarr	BYTE	用于优化范围表达式分段存储策略中的表达式的范围分区数据
flags	INTEGER	供内部使用
dbspace	VARCHAR(128)	存储此分段的数据库空间的名称
levels	SMALLINT	B 型树索引级别数
npused	FLOAT	对于表分段存储策略：数据页数 对于索引分段存储策略：叶子页数 对于滚动窗口表：nrows 中存储大小限制的单位数
nrows	FLOAT	对于表：分段中的行数。 对于索引：唯一键数。 对于滚动窗口表：清除策略中的存储大小上限。
clust	FLOAT	索引集群的程度；较小的数字对应较大的集群
partition	分段的名称	可匹配存储分段的 GBase 8s 数据库空间的名称，也可以为不同名称
version	SMALLINT	更新分段统计信息时递增的数字
nupdates	FLOAT	分段的更新数
ndeletes	FLOAT	分段的删除数
ninserts	FLOAT	分段的插入数

每个分段在此表中都有一行。evalpos 和 evaltext 字段包含有关个别分段的信息。

使用按 INTERVAL 或 LIST 的分段存储创建的表和索引具有包含有关分段存储策略信息的其他行。

strategy 类型 T 用于连接索引。(这是其分段存储策略与表分段存储相同的分段索引。)

对表的分发统计信息最近一次重新计算以来，有关该表的 **sysfragments** 制表 DML 操作中 **nupdates、ndeletes** 和 **ninserts** 列的信息，请参阅 SYSDISTRIB系统目录表中具有相同名称的三个列的描述。

在 GBase 8s 中，**fragtype、tabid、indexname** 和 **evalpos** 列上的组合索引允许重复值。

2.36 SYSINDEXES

sysindexes 表是基于 **sysindices** 表的视图。它对数据库中的每个索引包含一行。

sysindexes 表具有以下列。

表 18. SYSINDEXES 表列描述

列	类型	解释
idxname	VARCHAR(128)	索引名
owner	VARCHAR(32)	索引的所有者（系统目录表的用户 gbasedbt 和数据库表的 *username*）
tabid	INTEGER	表的唯一标识代码
idxtype	CHAR(1)	索引类型：U = 唯一 D = 允许重复 G = 非位图通用键索引 g = 位图通用键索引 u = 唯一，位图 d = 非唯一，位图
集群	CHAR(1)	集群或非集群索引（C = 集群）
part1	SMALLINT	单个索引或组合索引的第 1 个组件的列号（colno）
part2	SMALLINT	组合索引的第 2 个组件
part3	SMALLINT	组合索引的第 3 个组件
part4	SMALLINT	组合索引的第 4 个组件
part5	SMALLINT	组合索引的第 5 个组件
part6	SMALLINT	组合索引的第 6 个组件
part7	SMALLINT	组合索引的第 7 个组件
part8	SMALLINT	组合索引的第 8 个组件
part9	SMALLINT	组合索引的第 9 个组件
part10	SMALLINT	组合索引的第 10 个组件
part11	SMALLINT	组合索引的第 11 个组件
part12	SMALLINT	组合索引的第 12 个组件
part13	SMALLINT	组合索引的第 13 个组件
part14	SMALLINT	组合索引的第 14 个组件
part15	SMALLINT	组合索引的第 15 个组件
part16	SMALLINT	组合索引的第 16 个组件
levels	SMALLINT	B 型树级别的数目
leaves	INTEGER	叶的数目
nunique	INTEGER	第一列中唯一键的数目
clust	INTEGER	集群的程度；较小的数字对应较大的集群
idxflags	INTEGER	存储索引的当前锁定方式的位图

对于大多数系统目录表，仅当运行了 UPDATE STATISTICS 语句之后才会在此表中反映影响现有索引的更改。

此表中 **part1** 至 **part16** 列中的每一列都保存组合索引的 16 个可能部件之一的列号（**colno**）。如果以降序顺序对组件排序，那么 **colno** 输入为负值。对于不使用用户定义的数据类型或函数索引的 B 型树索引填充此列。对于类属 B 型树和所有其他访问方法，**part1** 至 **part16** 列都包含零。

在对表运行 UPDATE STATISTICS 语句之前，**clust** 列是空白的。最大值是表中的行数，最小值是表中的数据页数。

2.37 SYSINDICES

sysindices 系统目录表描述数据库中的索引。它存储所有索引的 LOW 方式统计信息，且对在数据库中定义的每个索引包含一行。

表 19. sysindices 系统目录表列

列	类型	解释
idxname	VARCHAR(128)	索引的名称
owner	VARCHAR(32)	索引所有者的名称（系统目录表的用户 gbasedbt 和数据库表的 *username*）
tabid	INTEGER	表的唯一标识代码
idxtype	CHAR(1)	唯一性状态 U = 需要唯一值 D = 允许重复
集群	CHAR(1)	集群或非集群状态（C = 集群）
levels	SMALLINT	树的层数
leaves	FLOAT	叶的数目
nunique	FLOAT	第一列中唯一键的数目
clust	FLOAT	集群的程度；较小的数字对应较大的集群。最大值是表中的行数，最小值是表中的数据页数。对表运行 UPDATE STATISTICS 之前，此列是空白的。
nrows	FLOAT	表中的估计行数（在对表运行 UPDATE STATISTICS 之前为零）

列	类型	解释
indexkeys	INDEXKEYARRAY	索引键的内部表示法。列最多可有三个字段，格式为：**procid, (***col1,col2, . . . ,coln***),opclassid**，其中 1 < *n* < 341
amid	INTEGER	实现此索引的访问方法的唯一标识代码。（对于 **sysams** 表中的访问方法，值 = **am_id**。）
amparam	LIST(VARCHAR(255))	用于定制 **amid** 访问方法行为的参数列表
collation	CHAR(32)	创建索引时使用其整理顺序的数据库语言环境
pagesize	INTEGER	存储此索引的页面的大小（以字节计）
nhashcols	SMALLINT	FOT 索引中散列的列数
nbuckets	SMALLINT	森林树 (FOT) 索引中子树（存储区）数
ustlowts	DATETIME YEAR TO FRACTION	上次记录的索引统计信息的日期和时间
ustbuildduration	INTERVAL HOUR TO FRACTION(5)	计算索引统计信息所需的时间
nupdates	FLOAT	表的更新数
ndeletes	FLOAT	表的删除数
ninserts	FLOAT	表的插入数
fextsize	INT	索引的第一个扩展数据块的大小 (KB)
nextsize	INT	索引的下一个扩展数据块的大小 (KB)

只有在运行了 UPDATE STATISTICS 语句后，影响现有索引的更改才会在此系统目录表中反映出来。

indexkeys 列中的字段具有以下有效值：

procid（在 **sysprocedures** 中）仅为对表列定义的函数的返回值的函数索引存在。第二个字段中列（*col1, col2, ... , coln*）的列表标识被定义索引的列。

最大值与语言相关：对于 SPL 或 Java(TM) UDR，最大为 341；对于 C UDR，最大为 102。

opclassid 标识数据库服务器用于构建和搜索索引的辅助访问方法。这与访问方法的 **sysopclasses.opclassid** 值相同。

对索引的分发统计信息最近一次重新计算以来，有关该索引的 **sysindices** 制表 DML 操作中 **nupdates、ndeletes** 和 **ninserts** 列的信息，请参阅 SYSDISTRIB系统目录表中具有相同名称的三个列的描述。

fextsize 列显示可选 EXTENT SIZE 子句在定义索引的 CREATE INDEX 语句中所指定的、用户定义的第一个扩展数据块大小（以千字节计）。同样，**nextsize** 列显示可选 NEXT SIZE 子句在 CREATE INDEX 语句中所指定的、用户定义的下一个扩展数据块大小(以千字节计)。如果创建索引时省略了对应的 EXTENT SIZE 或 NEXT SIZE 子句，那么上述每个列会显示值零 (0)。

如果定义新索引的 CREATE INDEX 语句不包括任何显式扩展数据块大小规范，那么数据库服务器会自动计算第一个和下一个扩展数据块大小，但 **fextsize** 和 **nextsize** 列值会设置为 0。

建立了 **tabid** 列的索引并且此索引允许重复值。**idxname、owner** 和 **tabid** 列只允许唯一值。

2.38 SYSINHERITS

sysinherits 系统目录表存储有关表层次结构和命名 ROW 类型继承的信息。数据库中的每个超类型、子类型、超表和子表在 **sysinherits** 表中具有相应的行。

列	类型	解释
child	INTEGER	子表或子类型的标识代码
parent	INTEGER	超表或超类型的标识代码
class	CHAR(1)	继承类：t = 命名 ROW 类型 T = 表

child 和 **parent** 值都来自命名 ROW 类型的 **sysxtdtypes.extended_id** 或来自表的 **systables.tabid**。**child** 和 **parent** 列的简单索引允许重复值。

2.39 SYSLANGAUTH

syslangauth 系统目录表包含有关用于编写用户定义的例程 (UDR) 的计算机语言的权限信息。

表 20. SYSLANGAUTH 表列描述

列	类型	解释
grantor	VARCHAR(32)	语言权限授权者的名称
grantee	VARCHAR(32)	语言权限被授权者的名称
langid	INTEGER	标识 sysroutinelangs 表中的语言的代码

列	类型	解释
langauth	CHAR(1)	语言权限：u = 授予 Usage 特权 U = 授予 Usage 特权 (WITH GRANT OPTION)

langid、grantor 和 **grantee** 列的组合索引只允许唯一值。langid 和 grantee 列的组合索引允许重复值。

2.40 SYSLOGMAP

syslogmap 系统目录表包含分段存储信息。

表 21. SYSLOGMAP 表列描述

列	类型	解释
tabloc	INTEGER	表在其他数据库中位置的代码
tabid	INTEGER	表的唯一标识代码
fragid	INTEGER	分段的标识代码
flags	INTEGER	分段声明中修饰符的位图

tabloc 列的简单索引以及 **tabid** 和 **fragid** 列的组合索引不允许重复值。

2.41 SYSOBJSTATE

sysobjstate 系统目录表存储有关数据库对象的状态（对象方式）的信息。列示在此表中的数据库对象的类型包括索引、触发器和约束。

数据库中的每个索引、触发器和约束在 **sysobjstate** 表中都有相应的行（如果用户创建该对象的话）。数据库服务器对系统目录表创建的索引不会列示在 **sysobjstate** 表中，原因是不能更改这些索引的对象方式。

sysobjstate 表具有以下列。

表 22. SYSOBJSTATE 表列描述

列	类型	解释
objtype	CHAR(1)	数据库对象类型的代码： C = 约束 I = 索引 T = 触发器

列	类型	解释
owner	VARCHAR(32)	数据库对象所有者的权限标识
name	VARCHAR(128)	数据库对象的名称
tabid	INTEGER	对其定义对象的表的标识代码
state	CHAR(1)	数据库对象的当前状态（对象方式）。这些值可以是以下某个代码： D = 禁用 E = 启用 F = 过滤，没有完整性违例错误 G = 过滤，有完整性违例错误

objtype、**name**、**owner** 和 **tabid** 列的组合索引仅允许这些值的唯一组合。**tabid** 列的简单索引允许重复值。

2. 42 SYSOPCLASSES

sysopclasses 系统目录表包含有关与辅助访问方法相关联的运算符类的信息。它对已在数据库中定义的每个运算符类包含一行。**sysopclasses** 表具有以下列。

列	类型	解释
opclassname	VARCHAR(128)	运算符类的名称
owner	VARCHAR(32)	运算符类所有者的名称
amid	INTEGER	与此运算符类相关联的辅助访问方法的标识代码
opclassid	SERIAL	运算符类的标识代码
ops	LVARCHAR	属于此运算符类的运算符的名称列表
support	LVARCHAR	对此运算符类定义的支持函数的名称列表

opclassid 值对应于 **sysams.am_defopclass** 值，该值为 **amid** 列指定的辅助访问方法指定缺省运算符类。

sysopclasses 表具有 **opclassname** 和 **owner** 列的组合索引和 **opclassid** 列的索引。两个索引都只允许唯一值。

2.43 SYSOPCLSTR

sysopclstr 系统目录表定义数据库中的每个光学集群。在该表中，每个光学集群对应一行。

sysopclstr 表具有以下列。

列	类型	解释
owner	VARCHAR(32)	光学集群所有者的名称
clstrname	VARCHAR(128)	光学集群的名称
clstrsize	INTEGER	光学集群的大小
tabid	INTEGER	表的唯一标识代码
blobcol1	SMALLINT	BYTE 或 TEXT 列号 1
blobcol2	SMALLINT	BYTE 或 TEXT 列号 2
blobcol3	SMALLINT	BYTE 或 TEXT 列号 3
blobcol4	SMALLINT	BYTE 或 TEXT 列号 4
blobcol5	SMALLINT	BYTE 或 TEXT 列号 5
blobcol6	SMALLINT	BYTE 或 TEXT 列号 6
blobcol7	SMALLINT	BYTE 或 TEXT 列号 7
blobcol8	SMALLINT	BYTE 或 TEXT 列号 8
blobcol9	SMALLINT	BYTE 或 TEXT 列号 9
blobcol10	SMALLINT	BYTE 或 TEXT 列号 10
blobcol11	SMALLINT	BYTE 或 TEXT 列号 11
blobcol12	SMALLINT	BYTE 或 TEXT 列号 12
blobcol13	SMALLINT	BYTE 或 TEXT 列号 13
blobcol14	SMALLINT	BYTE 或 TEXT 列号 14
blobcol15	SMALLINT	BYTE 或 TEXT 列号 15
blobcol16	SMALLINT	BYTE 或 TEXT 列号 16
clstrkey1	SMALLINT	集群键号 1
clstrkey2	SMALLINT	集群键号 2
clstrkey3	SMALLINT	集群键号 3
clstrkey4	SMALLINT	集群键号 4
clstrkey5	SMALLINT	集群键号 5
clstrkey6	SMALLINT	集群键号 6
clstrkey7	SMALLINT	集群键号 7
clstrkey8	SMALLINT	集群键号 8
clstrkey9	SMALLINT	集群键号 9
clstrkey10	SMALLINT	集群键号 10

列	类型	解释
clstrkey11	SMALLINT	集群键号 11
clstrkey12	SMALLINT	集群键号 12
clstrkey13	SMALLINT	集群键号 13
clstrkey14	SMALLINT	集群键号 14
clstrkey15	SMALLINT	集群键号 15
clstrkey16	SMALLINT	集群键号 16

此表的内容对 CREATE OPTICAL CLUSTER、ALTER OPTICAL CLUSTER 和 DROP OPTICAL CLUSTER 语句敏感，这些语句已在支持光学集群子系统的数据库上执行。只有在运行 UPDATE STATISTICS 语句后，影响现有光学集群的更改才会在此表中反映出来。

clstrname 和 **owner** 列的组合索引只允许唯一值。**tabid** 列的简单索引允许重复值。

2.44 SYSPROCAUTH

sysprocauth 系统目录表描述对过程或函数授予的特权。它对授予的每一组特权包含一行。**sysprocauth** 表具有以下列。

表 23. SYSPROCAUTH 表列描述

列	类型	解释
grantor	VARCHAR(32)	访问例程的特权授权者的名称
grantee	VARCHAR(32)	访问例程的特权被授权者的名称
procid	INTEGER	例程的唯一标识代码
procauth	CHAR(1)	授予对例程的特权的类型：e = 对例程的 Execute 特权 E = Execute 特权 (WITH GRANT OPTION)

procid、**grantor** 和 **grantee** 列的组合索引只允许唯一值。**procid** 和 **grantee** 列的组合索引允许重复值。

2.45 SYSPROCBODY

sysprocbody 系统目录表描述数据库中每个过程或函数的已编译版本。因为 **sysprocbody** 表存储例程的文本，所以每个例程可以有多行。**sysprocbody** 表具有以下列。

表 24. SYSPROCBODY 表列描述

列	类型	解释

列	类型	解释
procid	INTEGER	例程的唯一标识代码
datakey	CHAR(1)	data 列中的信息的类型：A = 例程更改 SQL（在更新统计信息后不会更改该值）D = 例程用户文档文本 E = 创建时间信息 L = 文字值（即，文字数字或带引号字符串）P = 解释器指令代码 (p-code) R = 例程返回值类型列表 S = 例程符号表 T = 例程文本创建 SQL
seqno	INTEGER	例程内的行号
data	CHAR(256)	例程的实际文本

A 标志表示过程修饰符已更改。ALTER ROUTINE 语句仅更新修饰符而不是例程主体。UPDATE STATISTICS 更新查询计划而不是例程修饰符，并且数据键的值不会更改成 A 以外的值。A 标志标记所有已更改修饰符的过程和函数，包括超负荷的过程和函数。T 标志用于例程创建文本。

data 列包含实际数据，这些数据可以使用下列格式之一：

编码返回值列表

编码符号表

文字数据

例程的 P-code

例程的已编译代码

例程及其文档的文本

procid、**datakey** 和 **seqno** 列只允许唯一值。

2.46 SYSPROCCOLUMNS

sysproccolumns 系统目录表存储有关 SYSPROCEDURES 中所有 UDR 的返回类型和参数名称的信息。

此表中 **procid** 和 **paramid** 列的组合索引只允许唯一值。

表 25. SYSPROCCOLUMNS 表列描述

列	类型	解释
procid	INTEGER	例程的唯一标识代码
paramid	INTEGER	参数的唯一识别代码
paramname	VARCHAR (IDENTSIZE)	参数的名称
paramtype	SMALLINT	标识参数的类型
paramlen	SMALLINT	指定参数的长度

列	类型	解释
pxid	INTEGER	指定参数的扩展类型标识
paramattr	INTEGER	0 = 参数是未知类型 1 = 参数处于 INPUT 方式 2 = 参数处于 INOUT 方式 3 = 参数是多个返回值 4 = 参数处于 OUT 方式 5 = 参数是一个返回值

2.47 SYSPROCEDURES

sysprocedures 系统目录表列出了数据库中注册的每个函数和过程的特征。它对每个例程包含一行。

sysprocedures 中的每个函数具有唯一值 **procid**，称为例程标识符。在整个系统目录中，函数由其例程标识符而不是其名称标识。

sysprocedures 表具有以下列。

表 26. SYSPROCEDURES 表列描述

列	类型	解释
procname	VARCHAR(128)	例程的名称
owner	VARCHAR(32)	所有者的名称
procid	SERIAL	例程的唯一标识代码
mode	CHAR(1)	方式类型： D *或* d = DBA O *或* o = 所有者 P *或* p = 受保护 R *或* r = 受限制 T *或* t = 触发器
retsize	INTEGER	返回值的已编译大小（以字节计）
symsize	INTEGER	符号表的已编译大小（以字节计）
datasize	INTEGER	常量数据的已编译大小（以字节计）
codesize	INTEGER	例程代码的已编译大小（以字节计）
numargs	INTEGER	例程的自变量数
isproc	CHAR(1)	指定例程是过程还是函数： t = 过程 f = 函数
specificname	VARCHAR(128)	指定例程的名称
externalname	VARCHAR(255)	外部例程的位置。此项在内容和格式上都是特定于语言的。
paramstyle	CHAR(1)	参数样式：I = GBase 8s
langid	INTEGER	语言代码（在 sysroutinelangs 表中）
paramtypes	RTNPARAMTYPES	描述例程参数的信息

列	类型	解释
variant	BOOLEAN	例程是否是 VARIANT： t = 是 VARIANT f = 不是 VARIANT
client	BOOLEAN	保留供将来使用
handlesnulls	BOOLEAN	NULL 处理指示符： t = 处理 NULLf = 不处理 NULL
percallcost	INTEGER	每个调用的 CPU 量 执行 UDR 的整数成本：成本 / 调用 - 0 -(2^31-1)
commutator	VARCHAR(128)	换向函数的名称
negator	VARCHAR(128)	否定函数的名称
selfunc	VARCHAR(128)	用于估计 UDR 的选择性的函数的名称
internal	BOOLEAN	指定是否可以从 SQL 调用例程：t = 例程是内部的，不能从 SQL 调用 f = 例程是外部的，可以从 SQL 调用
class	CHAR(18)	执行例程应依据的 CPU 类
stack	INTEGER	每个调用所需的堆栈大小（以字节计）
parallelizable	BOOLEAN	UDR 的并行化指示符：t = 可并行化 f = 不可并行化
costfunc	VARCHAR(128)	UDR 的成本函数的名称
selconst	SMALLFLOAT	UDR 的选择性常量
flags	INTEGER	仅供内部使用

在 **mode** 列中，R 方式是 O 方式的特殊情况。如果例程是以不同于例程创建者的指定所有者创建的，那么例程处于受限 (R) 方式。如果执行了涉及远程数据库的例程语句，那么数据库服务器使用执行例程的用户的访问特权而不是例程所有者的特权。在所有其他情况下，R 方式例程与 O 方式例程行为相同。

数据库服务器可以创建受保护的例程供内部使用。**sysprocedures** 表在 **mode** 列中用字母 P 或 p 标识这些受保护的例程，其中 p 指示 SPL 例程。受保护例程具有以下限制：
您无法使用 ALTER FUNCTION、ALTER PROCEDURE 或 ALTER ROUTINE 语句修改受保护的例程。
您无法使用 DROP FUNCTION、DROP PROCEDURE 或 DROP ROUTINE 语句注销受保护的例程。
您无法使用 dbschema 实用程序显示受保护的例程。

在较早的版本中，受保护的 SPL 例程用小写 p 指示。从 V9.0 开始，受保护的 SPL

例程将被视为 DBA 例程，而不能是所有者例程。因此，D 和 O 表示 DBA 例程和所有者例程，而 d 和 o 表示受保护的 DBA 例程和受保护的所有者例程。

触发器方式指定用户定义的 SPL 例程，只能从触发操作的 FOR EACH ROW 部分调用该例程。

重要： 在发出 SET SESSION AUTHORIZATION 语句后，数据库服务器为所有使用新标识时创建的所有者例程指定限制方式。

唯一索引定义于 **procid** 列。**procname**、**isproc**、**numargs** 和 **owner** 列的组合索引允许重复值，**specificname** 和 **owner** 列的组合索引也允许重复值。

2.48 SYSPROCPLAN

sysprocplan 系统目录表描述每个例程内的数据操作语句的查询执行方案和依赖性列表。由于可以在不同的日期创建例程方案的不同部分，所以此表可以包含每个例程的多个行。

表 27. SYSPROCPLAN 表列描述

列	类型	解释
procid	INTEGER	例程的标识代码
planid	INTEGER	计划的标识代码
datakey	CHAR(1)	data 列中存储的信息的类型：D = 依赖性列表 I = 信息记录 Q = 执行计划
seqno	INTEGER	方案内的行号
created	DATE	创建方案的日期
datasize	INTEGER	列表或方案的大小（以字节计）
data	CHAR(256)	已编码（编译）的列表或计划
collation	CHAR(32)	创建例程时的排列顺序

在运行例程之前，会检查 **data** 列中例程的依赖性列表。如果方案存取的表的主版本号已更改，或者如果例程使用的任何对象自从优化方案以来已被修改（例如：如果已删除索引），那么会再次优化方案。当 **datakey** 为 I 时，**data** 列存储有关 UPDATE STATISTICS 和 PDQPRIORITY 的信息。

可通过对 **sysprocplan** 使用 DELETE 语句来删除给定例程的所有方案。当后来执行例程时，新的方案就会自动生成并记录在 **sysprocplan** 中。UPDATE STATISTICS FOR PROCEDURE 语句也会更新此表。

procid、**planid**、**datakey** 和 **seqno** 列的组合索引只允许唯一值。

2.49 SYSREFERENCES

sysreferences 系统目录表列出了各列的所有引用约束。它对数据库中的每个引用约束包含一行。

表 28. SYSREFERENCES 表列描述

列	类型	解释
constrid	INTEGER	唯一地标识约束的代码
主	INTEGER	相应主键的标识代码
ptabid	INTEGER	作为主键的表的标识代码
updrule	CHAR(1)	保留供将来使用；显示 R
delrule	CHAR(1)	约束使用级联删除还是限制规则：C = 级联删除 R = 限制（缺省值）
matchtype	CHAR(1)	保留供将来使用；显示 N
pendant	CHAR(1)	保留供将来使用；显示 N

建立了 constrid 列的索引并且此索引只允许唯一值。建立了 primary 列的索引并且此索引允许重复值。

2.50 SYSROLEAUTH

sysroleauth 系统目录表描述授予用户的角色。它对在数据库中授予用户的每个角色包含一行。**sysroleauth** 表具有以下列。

表 29. SYSROLEAUTH 表列描述

列	类型	解释
rolename	VARCHAR(32)	角色的名称
grantee	VARCHAR(32)	角色被授权者的名称
is_grantable	CHAR(1)	指定角色是否可授予：Y = 可授予 N = 不可授予

is_grantable 列指示该角色是否是使用 GRANT 语句的 WITH GRANT OPTION 授予的。**rolename** 和 **grantee** 列的组合索引只允许唯一值。

2.51 SYSROUTINELANGS

sysroutinelangs 系统目录表列出了用户定义的例程 (UDR) 的受支持编程语言。它具有以下列。

列	类型	解释
langid	SERIAL	唯一地标识受支持语言的代码
langname	CHAR(30)	语句的名称，例如：C 或 SPL
langinitfunc	VARCHAR(128)	语言的初始化函数的名称
langpath	CHAR(255)	UDR 语言的目录路径
langclass	CHAR(18)	UDR 语言的类的名称

langname 列的索引允许重复值。

2.52 SYSSECLABELAUTH

sysseclabelauth 系统目录表记录已授予用户的 LBAC 标号。它具有以下列。

列	类型	解释
GRANTEE	CHAR(32)	被授予标号者的名称
secpolicyid	INTEGER	安全标号所属的安全策略的标识
readseclabelid	INTEGER	已被授予读访问权的安全标号的标识
writeseclabelid	INTEGER	已授予写访问权的安全标号的标识

2.53 SYSSECLABELCOMPONENTS

sysseclabelcomponents 系统目录表记录安全标号组件。它具有以下列。

列	类型	解释
compname	VARCHAR(128)	组件名称
compid	SERIAL	组件标识
comptype	CHAR(1)	组件类型：A = 数组 S = 集 T = 树
numelements	INTEGER	组件中的元素数量
coveringinfo	VARCHAR(128)	内部编码信息
numalters	SMALLINT	在组件上已执行的变更操作数量

2.54 SYSSECLABELCOMPONENTELEMENTS

sysseclabelcomponentelements 系统目录表记录安全标号的组件元素的值。它具有以下列。

列	类型	解释
compid	INTEGER	组件标识
元素 (element)	VARCHAR(32)	元素名称
elementencoding	CHAR(8)	元素的编码格式

列	类型	解释
parentelement	VARCHAR(32)	树组件的父元素的名称。对于以下项，值为 NUll： 集组件 数组组件 树组件的根节点
alterversion	SMALLINT	在添加元素时变更操作的数量。此值由 dbexport 和 dbimport 命令使用。

2.55 2.48 SYSSECLABELNAMES

sysseclabelnames 系统目录表记录安全标号名称。它具有以下列。

列	类型	解释
secpolicyid	INTEGER	安全标号所属的安全策略的标识
seclabelname	VARCHAR(128)	安全标号名称
seclabelid	INTEGER	安全标号的标识

2.56 SYSSECLABELS

sysseclabels 系统目录表记录安全标号编码。它具有以下列。

列	类型	解释
secpolicyid	INTEGER	安全标号所属的安全策略的标识
seclabelid	INTEGER	安全标号标识
syssecla belnames	VARCHAR(128)	安全标号编码

2.57 SYSSECPOLICIES

syssecpolicies 系统目录表记录它具有的这些列的安全策略。

列	类型	解释
secpolicyname	VARCHAR(128)	安全策略名称
secpolicyid	SERIAL	安全策略标识
numcomps	SMALLINT	安全策略中的安全标号组件的数量
comptypelist	CHAR(16)	策略中每个组件类型的排序列表。A = 数组 S = 集 T = 树 - = 超越 NUMCOMPS

列	类型	解释
overrideseclabel	CHAR(1)	在用户的安全标号和豁免凭证不允许其以标号提供的对 INSERT 或 UPDATE SQL 语句的安全性来插入或更新数据行时，指示该行为。 Y：提供的安全标号被忽略并由用户写访问权的安全标号所取代。 N：在未得到授权以编写安全标号时返回一个错误。

2.58 SYSSECPOLICYCOMPONENTS

syssecpolicycomponents 系统目录表记录每个安全策略的组件。它具有以下列。

列	类型	解释
secpolicyid	INTEGER	安全策略标识
compid	INTEGER	标号安全策略的组件标识
compno	SMALLINT	安全标号组件在安全策略中存在的位置，从第 1 个位置开始。

2.59 SYSSECPOLICYEXEMPTIONS

syssecpolicyexemptions 系统目录表记录已经授予用户的豁免权。它具有以下列。

列	类型	解释
grantee	CHAR(32)	具有此豁免权的用户
secpolicyid	INTEGER	授予豁免权的策略标识
豁免权	CHAR(6)	授予在 GRANTEE 列中标识的用户的豁免权。这六个字符具有以下含义：1 = 读数组 2 = 读集 3 = 读树 4 = 写数组 5 = 写集 6 = 写树 每个字符具有以下某个值： E = 豁免D = 减记豁免U = 增记豁免 - = 无豁免

2.60 SYSSEQUENCES

syssequences 系统目录表列出了数据库中存在的序列对象。**syssequences** 表具有以下列。

列	类型	解释
seqid	SERIAL	唯一地标识序列对象的代码
tabid	INTEGER	作为表对象的序列的标识代码
start_val	INT8	序列的开始值
inc_val	INT8	连续值之间的增量的值
max_val	INT8	序列的最大可能值
min_val	INT8	序列的最小可能值
cycle	CHAR(1)	零表示 NOCYCLE，1 表示 CYCLE
cache	INTEGER	在序列高速缓存中预先分配的值的数目
order	CHAR(1)	零表示 NOORDER，1 表示 ORDER

2.61 SYSSURROGATEAUTH

syssurrogateauth 系统目录表会存储可信用户和代理用户信息。

运行 GRANT SETSESSIONAUTH 语句时会填充 **syssurrogateauth** 系统目录表。TO 子句中指定的用户或角色将添加到 **trusteduser** 列中。ON 子句中指定的用户将添加到 **surrogateuser** 列中。

例如，考虑如下语句：

```
GRANT SETSESSIONAUTH ON bill, john TO mary, peter;
```

syssurrogateauth 表中的条目将按如下所示创建：

trusteduser	surrogateuser
mary	bill
mary	john
peter	bill
peter	john

syssurrogateauth 表具有以下列。

表 30. SYSSURROGATEAUTH 表列描述

列	类型	解释
trusteduser	CHAR(32)	可信用户名或角色。
surrogateuser	CHAR(32)	代理用户名。

2.62 SYSSYNONYMS

syssynonyms 系统目录表未在使用。 syssyntable 表描述同义词。syssynonyms 系统目录表具有以下列。

表 31. SYSSYNONYMS 表列描述

列	类型	解释
owner	VARCHAR(32)	同义词所有者的名称
synname	VARCHAR(128)	同义词的名称
created	DATE	创建同义词时的日期
tabid	INTEGER	标识表、序列或视图的代码

owner 和 **synonym** 列的组合索引只允许唯一值。建立了 **tabid** 列的索引并且此索引允许重复值。

2.63 SYSSYNTABLE

syssyntable 系统目录表概述了每个公共或专用同义词与它表示的数据库对象（表、序列或视图）之间的映射。它为 **systables** 表中 **tabtype** 值为 P 或 S 的每个条目包含一行。**syssyntable** 表具有以下列。

列	类型	解释
tabid	INTEGER	标识公共同义词的代码
servername	VARCHAR(128)	外部数据库服务器的名称
dbname	VARCHAR(128)	外部数据库的名称
owner	VARCHAR(32)	外部对象的所有者的名称
tabname	VARCHAR(128)	外部表或视图的名称
btabid	INTEGER	基本表、序列或视图的标识代码

符合 ANSI 标准的数据库不支持公共同义词，它们的 **syssyntable** 表只能描述 **syssyntable.tabtype** 值为 P 的同义词。

如果为当前数据库中的对象定义同义词，那么只能使用 **tabid** 和 **btabid** 列。如果为当前数据库外部的表定义同义词，那么不使用 btabid 列，而是使用 **tabid**、**servername**、**dbname**、**owner** 和 **tabname** 列。

tabid 列映射至 **systables.tabid**。借助 **tabid** 信息，可以确定有关 **systables** 中的同义词的其他方面。

tabid 列的索引只允许唯一值。**btabid** 列的索引建立为允许重复值。

2.64 SYSTABAMDATA

systabamdata 系统目录表存储使用主访问方法创建的表的特定于表的散列参数。

systabamdata 表具有以下列。

表 32. SYSTABAMDATA 表列描述

列	类型	解释
tabid	INTEGER	表的标识代码
am_param	CHAR(256)LVARCHAR(8192)	访问方法参数选项
am_space	VARCHAR(128)	保存数据值的存储空间的名称

am_param 列存储确定主要访问方法如何访问给定表的配置参数。**am_param** 列表中的每个配置参数都具有 *keyword=value* 或 *keyword* 格式。

am_space 列指定表的位置。它可能在数据库服务器内的熟文件、另一个数据库或智能大对象空间中。

tabid 列是 **systables** 表的主键。此列具有索引并且必须包含唯一值。

2.65 SYSTABAUTH

systabauth 系统目录表描述对表、视图、序列或同义词授予的每一组特权。它对在数据库中授予的每一组表特权包含一行；REVOKE 语句可以修改行。**systabauth** 表具有以下列。

表 33. SYSTABAUTH 表列描述

列	类型	解释
grantor	VARCHAR(32)	特权授权者的名称
grantee	VARCHAR(32)	特权被授权者的名称
tabid	INTEGER	数据库对象的 systables.tabid 中的值
tabauth	CHAR(9) CHAR(8)	指定对表、视图、同义词或序列上的特权的模式：s *或* S = Selectu *或* U = Update* = 列级别特权 i *或* I = Insertd *或* D = Deletex *或* X = Indexa *或* A = Alterr *或* R = Referencesn *或* N = Under 特权

如果 **tabauth** 列显示大写的特权代码（例如：S 表示选择），那么这指示用户还可以选择将该特权授予他人。用小写列示的特权代码（例如：s 表示选择）指示用户具有指定的特权，但不能将该特权授予他人。

连字符 (-) 指示 **tabauth** 模式内该位置缺少对应的特权。

带星号 (*) 的 tabauth 值意味着存在列级别特权；另请参阅 **syscolauth**。（在 DB-Access 中，指定表的 **Info** 命令的 **Privileges** 选项可以显示对该表的列级别特权。）

tabid、**grantor** 和 **grantee** 的组合索引只允许唯一值。**tabid** 和 **grantee** 的组合索引允许重复值。

2.66 SYSTABLES

systables 系统目录表对在数据库(包括系统目录的表和视图)中定义的每个表对象(表、视图、同义词或 GBase 8s 中的序列）包含一行。

表 34. SYSTABLES 表列描述

列	类型	解释
tabname	VARCHAR(128)	表、视图、同义词或序列的名称
owner	CHAR(32)	表的所有者（系统目录表的用户 gbasedbt 和数据库表的 *username*）
partnum	INTEGER	物理存储位置代码
tabid	SERIAL	系统指定的顺序标识编号
rowsize	SMALLINT	最大行大小，以字节计 （≤ 40 MB）
ncols	SMALLINT	表中的列数
nindexes	SMALLINT	表的索引数
nrows	FLOAT	表中的行数
created	DATE	创建或上次修改表时的日期
version	INTEGER	改变表时更改的数字
tabtype	CHAR(1)	指示表对象类型的代码： T = 表 E = 外部表 V = 视图 Q = 序列 P = 专用同义词 S = 公共同义词 （类型 S 在符合 ANSI 标准的数据库中不可用。）
locklevel	CHAR(1)	表的锁定方式： B = 页面和行级别 P = 页面级别 R = 行级别
npused	FLOAT	数据库服务器曾经在 tablespace 中启动过的数据页数
fextsize	INTEGER	初始扩展数据块的大小 (KB)
nextsize	INTEGER	所有后续扩展数据块的大小 (KB)
flags	SMALLINT	用于对永久表分类的代码： **ROWID** 1 - 已定义行标识列 **UNDER** 2 - 在超表之下创建的表 **VIEWREMOTE** 4 - 视图基于远程表 **CDR**

列	类型	解释
		8 - 已定义 CDRCOLS **RAW** 16 - (GBase 8s) RAW 表 **EXTERNAL** 32- 外部表 **AUDIT** 64 - 审计表属性 - FGA **AQT** 128 - 视图是用于卸载 DWA 的 AQT **VIRTAQT** 256 - 视图是虚拟 AQT
site	VARCHAR(128)	保留供将来使用
dbname	VARCHAR(128)	保留供将来使用
type_xid	INTEGER	**sysxtdtypes.extended_id** 中的代码（对于类型表），或者 0（对于无类型表）
am_id	INTEGER	访问方法代码（**sysams** 表的关键字） NULL 或 0 表示内置存储管理器
ustlowts	DATETIME YEAR TO FRACTION (5)	上次记录表、行和页计数统计信息的时间
secpolicyid	INTEGER	已与表连接的安全策略的标识。NULL 表示不受保护的表
protgranularity	CHAR(1)	LBAC 粒度级别： R：行级别粒度 C：列级别粒度 B：行和列的粒度 “空白”表示不受保护的表
statlevel	CHAR(1)	统计信息级别 T = 表 F = 分段 A = 自动
statchange	SMALLINT	仅供内部使用

为 **systables** 表中记录的每个表、视图、序列和同义词指定 **tabid**，它是系统指定的、唯一标识该对象的 SERIAL 值。保留前 99 个 **tabid** 值用于系统目录。数据库中第一个用

户定义的表对象的 **tabid** 始终为 100。

对 **tabid** 列建立了索引，且该列只包含唯一值。**tabname** 和 **owner** 列的组合索引也需要唯一值。

version 列包含创建新表时存储在 **systables** 中的已编码数字。当对表执行数据定义的语句（例如，ALTER INDEX、ALTER TABLE、DROP INDEX 和 CREATE INDEX）时，此值的一部分会增大。

在 **flags** 列中，ST_RAW 表示支持事务日志记录的数据库中的非日志记录永久表。

SQL_LOGICAL_CHAR 参数的设置会编码到描述 VERSION 表的行中的 **systables.flags** 列值。注意此由系统生成的表的标识中有一个前导空格。

要确定数据库是否启用可将逻辑字符语义应用到字符列声明的 SQL_LOGICAL_CHAR 配置参数，您可以执行以下查询：

```
SELECT flags INTO $value FROM 'gbasedbt'.systables WHERE tabname = '
VERSION';
```

由于 SQL_LOGICAL_CHAR 设置以“VERSION.flags”值的两个最不重要数位编码，因此您可以依据以下公式从返回的 flags 值计算其设置：

SQL_LOGICAL_CHAR = (value & 0x03) + 1此处的 & 是位 AND 运算符。任何大于 1 的 SQL_LOGICAL_CHAR 设置表示创建数据库时已启用 SQL_LOGICAL_CHAR，且字符列的显式或缺省最大大小规范将乘以该设置。

当执行引用数据库表的预编译语句时，会检查版本值以确保自预编译语句以来没有进行任何更改。如果修改了表模式的 DDL 操作更改了版本值并且 SET ENVIRONMENT 语句的 IFX_AUTO_REPREPARE 设置已禁用了自动重新编译，那么不会执行预编译语句，并且必须再次预编译该语句。

npused 列既不反映用于 BYTE 或 TEXT 数据的页数，也不反映 DELETE 或 TRUNCATE 操作中释放的页数。

nrows 列和 **npused** 列可能无法准确反映由外部表使用的行数和数据页数，除非在创建外部表时指定了 NUMROWS 子句。请参阅《*GBase 8s 管理员指南*》以获取更多信息。

systables 表有两行用来存储有关数据库语言环境的信息：GL_COLLATE（其 **tabid** 为 90）和 GL_CTYPE（其 **tabid** 为 91）。要查看这些行，请输入以下 SELECT 语句：

```
SELECT * FROM systables WHERE tabid=90 OR tabid=91;
```

2.67 SYSTRACECLASSES

systraceclasses 系统目录表包含跟踪类的名称和标识符。**systraceclasses** 具有以下列。

表 35. SYSTRACECLASSES 表列描述

列	类型	解释

列	类型	解释
name	CHAR(18)	跟踪消息的类的名称
classid	SERIAL	跟踪类的标识代码

*跟踪类*是可以在开发和测试新的 DataBlade 模块和用户定义的例程过程中使用的跟踪消息类别。开发者可以通过在其代码中调用适当的 DataBlade API 例程来使用跟踪设施。

要创建新的跟踪类，将一行直接插入 **systraceclasses**表中。缺省情况下，所有用户都可以查看此表，但只有具有 DBA 特权的用户才能对其进行修改。

除非没有定义 MITRACE_OFF 配置参数，否则数据库不支持跟踪。

name 列的唯一索引要求每个跟踪类都具有唯一名称。数据库服务器为每个类指定唯一顺序代码。此 **classid** 列的索引也只允许唯一值。

2.68 SYSTRACEMSGS

systracemsgs 系统目录表存储可在调试用户定义的例程时使用的国际化跟踪消息。

systracemsgs 表具有以下列。

表 36. SYSTRACEMSGS 表列描述

列	类型	解释
name	VARCHAR(128)	消息的名称
msgid	SERIAL	消息模板的标识代码
locale	CHAR(36)	与此版本的消息相关联的语言环境（例如：en_us.8859-1）
seqno	SMALLINT	保留供将来使用
message	VARCHAR(255)	消息文本

DataBlade 模块开发者通过将一行直接插入 **systracemsgs** 表来创建跟踪消息。创建了消息后，开发团队就可以使用 DataBlade API 提供的跟踪除非没有定义 MITRACE_OFF 配置参数，否则数据库不支持跟踪。

语句按名称或 msgid 代码指定该消息。

要创建跟踪消息，必须指定消息的名称、语言环境和文本。缺省情况下，所有用户都可以查看 **systracemsgs** 表，但只有具有 DBA 特权的用户才能对其进行修改。

对 **name** 和 **locale** 列定义了唯一组合索引。对 **msgid** 列定义了其他唯一索引。

2.69 SYSTRIGBODY

systrigbody 系统目录表包含触发器定义的 ASCII 文本和触发器的线性化代码。*线性化代码*是以 ASCII 格式表示的二进制数据和代码。

要点： 数据库服务器使用存储在 **systrigbody** 中的线性化代码。一定不要改变包含线性化代码的行的内容。

systrigbody 表具有以下列。

表 37. SYSTRIGBODY 表列描述

列	类型	解释
trigid	INTEGER	触发器的标识代码
datakey	CHAR(1)	指定数据类型的代码：A = 主体（触发操作）的 ASCII 文本 B = 主体的线性化代码 D = 页眉（触发器定义）的英文文本 H = 页眉的线性化代码 S = 符号表的线性化代码
seqno	INTEGER	此数据段的页号
data	CHAR(256)	英文文本或线性化代码
collation	CHAR(32)	创建触发器时的排列顺序

trigid、**datakey** 和 **seqno** 列的组合索引只允许唯一值。

2.70 SYSTRIGGERS

systriggers 系统目录表包含有关数据库中 SQL 触发器的信息。此信息包括触发事件和触发器的相关引用规范。**systriggers** 表具有以下列。

表 38. SYSTRIGGERS 表列描述

列	类型	解释
trigid	SERIAL	触发器的标识代码
trigname	VARCHAR(128)	触发器的名称
owner	VARCHAR(32)	触发器所有者的名称
tabid	INTEGER	触发表的标识代码
event	CHAR(1)	触发事件类型的代码：D = 删除触发器 I = 插入触发器 U = 更新触发器 S = 选择触发器 d = INSTEAD OF 删除触发器 i = INSTEAD OF 插入触发器 u = INSTEAD OF 更新触发器
old	VARCHAR(128)	更新之前值的名称
new	VARCHAR(128)	更新之后值的名称
mode	CHAR(1)	保留供将来使用

trigname 和 **owner** 列的组合索引只允许唯一值。**trigid** 列的索引也需要唯一值。**tabid** 列的索引允许重复值。

2.71 SYSUSERS

sysusers 系统目录表列出每单个用户的权限标识，或列出拥有数据库级别访问特权的 PUBLIC 组的公共权限标识。此表还会列出拥有数据库中任何对象访问特权的每个角色的名称。

此系统目录表具有以下列：

表 39. SYSUSERS 表列描述

列	类型	解释
username	VARCHAR(32)	数据库用户或角色的名称。username 的索引只允许唯一值。username 值可以是用户的登录名或角色的名称。
usertype	CHAR(1)	指定 username 所拥有最高数据库级别特权的代码，其中 username 是单个用户、PUBLIC 组或角色名称。有效代码是：D = DBA（所有特权）R = 资源（创建 UDR、UDT、永久表和索引）C = 连接（使用现有表）G = 角色 U = 缺省角色。为用户指定缺省角色时，会将数据库的隐式连接授予用户。这是用户在被授予 C、D 或 R 角色之前的角色。
priority	SMALLINT	保留供将来使用。
password	CHAR(16)	保留供将来使用。
defrole	VARCHAR(32)	缺省角色的名称。

2.72 SYSVIEWS

sysviews 系统目录表描述数据库中的每个视图。因为它存储创建视图的 SELECT 语句，所以对于每个视图，**sysviews** 可包含多行。它具有以下列。

列	类型	解释
tabid	INTEGER	视图的标识代码
seqno	SMALLINT	SELECT 语句的行号
viewtext	CHAR(64)	用于创建视图的实际 SELECT 语句

tabid 和 **seqno** 列的组合索引只允许唯一值。

2.73 SYSVIOLATIONS

sysviolations 系统目录表存储有关基本表的约束违例的信息。

此表会在 DELETE、INSERT、MERGE 或 UPDATE 语句检测到以下违例时进行更新：在 SQL 的 START VIOLATIONS TABLE 语句已为其创建关联违例表（对于 GBase 8s，还创建了诊断表）的数据库表中，违反了启用的约束或唯一索引。对于具有活动违例表的每个基本表，**sysviolations** 表都有对应的行，同时具有以下各列。

列	类型	解释
targettid	INTEGER	*目标表*（对其定义违例表和诊断表的基本表）的标识代码
viotid	INTEGER	违例表的标识代码
diatid	INTEGER	诊断表的标识代码
maxrows	INTEGER	在其中已定义过滤方式对象的目标表上，通过单一插入、更新或删除操作可插入诊断表中的最大行数。

maxrows 列还表示在启用已禁用的对象或将禁用的对象设置为过滤方式的单个操作期间可插入到诊断表中的最大行数（前提是该目标表存在诊断表）。如果没有为诊断或违例表指定最大值，那么 **maxrows** 包含 NULL 值。

此表的主键是 **targettid** 列。还对 **viotid** 列定义了附加唯一索引。

GBase 8s 还具有 **diatid** 列的唯一索引。

2.74 SYSXADATASOURCES

sysxadatasources 系统目录表存储 XA 数据源。

sysxadatasources 表具有以下列。

列	类型	解释
xa_datasrc_owner	CHAR(32)	XA 数据源所有者的用户标识
xa_datasrc_name	VARCHAR(128)	XA 数据源的名称
xa_datasrc_rmid	SERIAL	XA 数据源的唯一 RMID
xa_source_typeid	INTEGER	XA 数据源类型标识

2.75 SYSXASOURCETYPES

sysxasourcetypes 系统目录表存储 XA 数据源类型。**sysxasourcetypes** 表具有以下列。

列	类型	解释
xa_source_typeid	SERIAL	源类型的唯一标识
xa_source_owner	CHAR(32)	所有者的用户标识

列	类型	解释
xa_source_name	VARCHAR(128)	源类型名称
xa_flags	INTEGER	
xa_version	INTEGER	
xa_open	INTEGER	xa_open_entry 的 UDR 标识
xa_close	INTEGER	xa_close_entry 的 UDR 标识
xa_end	INTEGER	xa_end_entry 的 UDR 标识
xa_rollback	INTEGER	xa_rollback_entry 的 UDR 标识
xa_prepare	INTEGER	xa_prepare_entry 的 UDR 标识
xa_commit	INTEGER	xa_commit_entry 的 UDR 标识
xa_recover	INTEGER	xa_recover_entry 的 UDR 标识
xa_forget	INTEGER	xa_forget_entry 的 UDR 标识
xa_complete	INTEGER	xa_complete_entry 的 UDR 标识

2.76 SYSXTDDESC

sysxtddesc 系统目录表提供了在数据库中定义的每个用户定义的数据类型 (UDT) 的文本描述。**sysxtddesc** 表具有以下列。

列	类型	解释
extended_id	INTEGER	唯一地标识扩展数据类型的代码
seqno	SMALLINT	要对 UDT 描述的一行进行排序和标识的值 仅当剩余的文本字符串大于 255 个字节时才会创建新的行。
description	CHAR(256)	扩展数据类型的文本描述

extended_id 和 **seqno** 的组合索引允许重复值。

2.77 SYSXTDTYPEAUTH

sysxtdtypeauth 系统目录表标识每个 UDT（用户定义的数据类型）上的特权。

sysxtdtypeauth 表对授予的每组特权包含一行，且具有以下列：

列	类型	解释
grantor	VARCHAR(32)	特权授权者的名称
grantee	VARCHAR(32)	特权被授权者的名称
type	INTEGER	标识 UDT 的代码
auth	CHAR(2)	标识针对 UDT 的特权的代码：n *或* N = Under 特权 u *或* U = Usage 特权

如果 **auth** 列中的特权代码是大写的（例如，“U”表示使用），那么具有此特权的用户还可以将此特权授予他人。如果代码是小写的，那么具有该特权的用户不能将此特权授予他人。

type、grantor 和 grantee 的组合索引只允许唯一值。**type 和 grantee** 列的组合索引允许重复值。

2.78 SYSXTDTYPES

在 sysxtdtypes 系统目录表中，在数据库中定义的每个 UDT（用户定义的数据类型）对应一个条目，这些类型包括不透明和单值数据类型以及复杂数据类型（命名 ROW 类型、未命名 ROW 类型和 COLLECTION 类型）。

sysxtdtypes 表具有以下列。

表 40. SYSXTDTYPES 表列描述

列	类型	解释
extended_id	SERIAL	扩展数据类型的唯一标识代码
domain	CHAR(1)	UDT 的域的代码
mode	CHAR(1)	对 UDT 进行分类的代码： B = 基本（不透明）类型 C = 集合类型或未命名 ROW 类型 D = 单值类型 R = 命名 ROW 类型 ’’（空白）= 内置类型
owner	VARCHAR(32)	UDT 所有者的名称
name	VARCHAR(128)	UDT 的名称
type	SMALLINT	对 UDT 分类的代码
source	INTEGER	sysxtdtypes 引用（仅适用于单值类型） 零 (0) 指示从内置数据类型创建了单值 UDT。
maxlen	INTEGER	可变长度数据类型的最大长度 零指示固定长度 UDT。
length	INTEGER	固定长度数据类型的长度（以字节计） 零指示可变长度 UDT。

列	类型	解释
byvalue	CHAR(1)	“T” = UDT 通过值传递 “F” = UDT 不通过值传递
cannothash	CHAR(1)	“T” = UDT 可通过缺省散列函数散列 “F” = UDT 不可通过缺省函数散列
align	SMALLINT	此 UDT 的对齐方式（= 1、2、4 *或* 8）
locator	INTEGER	未命名 ROW 类型的定位器键

每个扩展数据类型都用唯一标识符（称为扩展标识符（**extended_id)**）、数据标识符（**type**）以及长度和数据库类型的描述来作为特征。

对于使用内置数据类型创建的单值类型，**type** 列代码对应于 SYSCOLUMNS 中列出的 **syscolumns.coltype** 列（指示源类型）的值，但要加上一个十六进制值 0x0000800。 文件 $GBASEDBTDIR/incl/esql/sqltypes.h 包含有关 **sysxtdtypes.type** 和 **syscolumns.coltype** 代码的信息。

extended_id 列的索引只允许唯一值。**locator** 列的索引允许重复值，**name** 和 **owner** 列的组合索引也一样。**type** 和 **source** 列的组合索引也允许重复值。

2.79 DUAL

DUAL 系统目录表是一个全局表。它具有以下列：

列	类型	解释
dummy	INTEGER	值为 1。

DUAL 表中只有一行数据：‘1’。任何用户都可以访问 DUAL 表，它只返回一行数据。可以使用它选择系统变量或求一个表达式的值。

2.80 信息模式

“信息模式”由一些只读视图组成，它们提供有关当前数据库服务器中您可以存取的所有表、视图和列的信息。这些视图还提供了有关 SQL 方言（例如 GBase 8s、Oracle 或 Sybase）和 SQL 标准的信息。请注意，与系统目录（其表描述个别数据库）不同，这些视图描述 GBase 8s 实例，而不是单个数据库。

此版本的“信息模式”视图符合 X/Open CAE 标准。提供这些标准的目的是，允许其他数据库系统上开发的应用程序不必直接访问 GBase 8s 系统目录表，即可获取 GBase 8s 系统目录信息。

重要： 因为 X/Open CAE 标准“信息模式”视图不同于符合 ANSI 的“信息模式”视

图，所以建议您不要在符合 ANSI 标准的数据库上安装 X/Open CAE“信息模式”视图。

下列“信息模式”视图可用：

tables

列

sql_languages

server_info

2.80.1 生成信息模式视图

“信息模式”视图是在您作为 DBA 运行以下 DB-Access 命令时自动生成的：

dbaccess *database-name* $GBASEDBTDIR/etc/xpg4_is.sql

视图显示系统目录表中的数据。如果表、视图或例程存在并具有与“信息模式”视图相同的任何名称，那么必须重命名这些数据库对象或者在脚本中重命名视图之后才能安装视图。可以对每个视图使用 DROP VIEW 语句来删除视图。要重新创建视图，重新运行脚本。

要点： 除了为每个“信息模式”视图指定的列之外，个别供应商可能会包含附加列或更改列的顺序。建议应用程序不要使用格式 SELECT * 或 SELECT table-name* 来访问“信息模式”视图。

2.80.2 访问信息模式视图

所有“信息模式”视图都会将 Select 特权授予 PUBLIC WITH GRANT OPTION，以便所有用户都可查询这些视图。因为未对“信息模式”视图授予任何其他特权，所以不能更新它们。

可以如同查询数据库中的任何其他表或视图那样查询“信息模式”视图。

2.80.3 信息模式视图的结构

本节中描述下列“信息模式”视图：

tables

columns

sql_languages

server_info

为了接受长标识名称，视图中的大部分列被定义为具有很大的最大大小的 VARCHAR 数据类型。

表“信息模式”视图

tables 信息模式视图对您可以访问的每个表包含一行。它包含以下列。

列	数据类型	解释
table_schema	VARCHAR(32)	表所有者的名称
table_name	VARCHAR(128)	表或视图的名称

列	数据类型	解释
table_type	VARCHAR(128)	BASE TABLE 表示表，或者 VIEW 表示视图
remarks	VARCHAR(255)	保留供将来使用

tables 视图中的可视行取决于您的特权。例如，如果您对表具有一种或多种特权（例如，对一个或多个列的 Insert、Delete、Select、References、Alter、Index 或 Update 特权），或者如果将特权授予 PUBLIC，那么可以看到描述该表的行。

列“信息模式”视图

columns 信息模式视图对每个可访问列包含一行。它包含以下列。

表 41. 列信息模式视图的描述

列	数据类型	解释
table_schema	VARCHAR(128)	表所有者的名称
table_name	VARCHAR(128)	表或视图的名称
column_name	VARCHAR(128)	表或视图中列的名称
ordinal_position	INTEGER	列在其表中的位置 ordinal_position 值是一个顺序编号，从 1 开始（表示第一列）。这是对 XPG4 的 GBase 8s 扩展。
data_type	VARCHAR(254)	列的数据类型名称，例如，CHARACTER 或 DECIMAL
char_max_length	INTEGER	字符数据类型的最大长度（以字节计）；其他情况为 NULL
numeric_precision	INTEGER	使用下列值之一： 精确数字数据类型（DECIMAL、INTEGER、MONEY 和 SMALLINT）的总位数 近似数据类型（FLOAT 和 SMALLFLOAT）的尾数精度（与机器相关）的位数 对于所有其他数据类型为 NULL。
numeric_prec_radix	INTEGER	使用下列值之一： 2 = 近似数据类型（FLOAT 和 SMALLFLOAT） 10 = 精确数字数据类型（DECIMAL、INTEGER、MONEY 和 SMALLINT） 对于所有其他数据类型为 NULL

列	数据类型	解释
numeric_scale	INTEGER	对于 DECIMAL 和 MONEY 数据类型为小数点右边的有效位数 对于 INTEGER 和 SMALLINT 类型为 0，对于所有其他数据类型为 NULL
datetime_precision	INTEGER	对于 DATE 和 DATETIME 列为秒的小数部分中的位数；其他情况为 NULL 此列是对 XPG4 的 GBase 8s 扩展。
is_nullable	VARCHAR(3)	指示列是否允许为 NULL 值；为 YES 或 NO
remarks	VARCHAR(254)	保留供将来使用

sql_languages 信息模式视图

sql_languages 信息模式视图对符合当前数据库服务器支持的标准的每个实例包含一行。**sql_languages** 视图包含以下列。

列	数据类型	解释
source	VARCHAR(254)	定义此 SQL 版本的组织
source_year	VARCHAR(254)	批准源文档的年份
conformance	VARCHAR(254)	服务器遵循的标准
integrity	VARCHAR(254)	指示这是否为完整性增强功能部件；可为 YES 或 NO
implementation	VARCHAR(254)	标识供应商的 SQL 产品
binding_style	VARCHAR(254)	直接、模块或其他绑定样式
programming_lang	VARCHAR(254)	绑定样式适合的主语言

sql_languages 视图对于所有用户都完全可视。

server_info 信息模式视图

server_info 信息模式视图描述应用程序当前连接至的数据库服务器。它包含两列。

列	数据类型	解释
server_attribute	VARCHAR(254)	数据库服务器的属性
attribute_value	VARCHAR(254)	server_attribute 应用于当前数据库服务器时的值

此视图中的每一行都提供了有关一个属性的信息。符合 X/Open 的数据库必须提供具有关于数据库服务器的某些必需信息的应用程序。

server_info 视图包括以下 **server_attribute** 信息。

server_attribute	解释

server_attribute	解释
identifier_length	用户定义的标识符的最大字节数
row_length	行中的最大字节数
userid_length	用户名中的最大字节数
txn_isolation	数据库服务器的初始事务隔离级别： 读取未落实（= 不带事务日志记录的数据库的缺省隔离级别；也称为脏读取） 读取已落实（= 不符合 ANSI 但支持显式事务日志记录的数据库的缺省隔离级别） 可序列化（= 符合 ANSI 标准的数据库的缺省隔离级别；也称为可重复读）
collation_seq	数据库服务器采用的字符集排序 有可能是以下值： ISO 8859-1 EBCDIC 缺省 GBase 8s 表示法显示 ISO 8859-1。

server_info 视图对于所有用户都完全可视。

3 数据类型

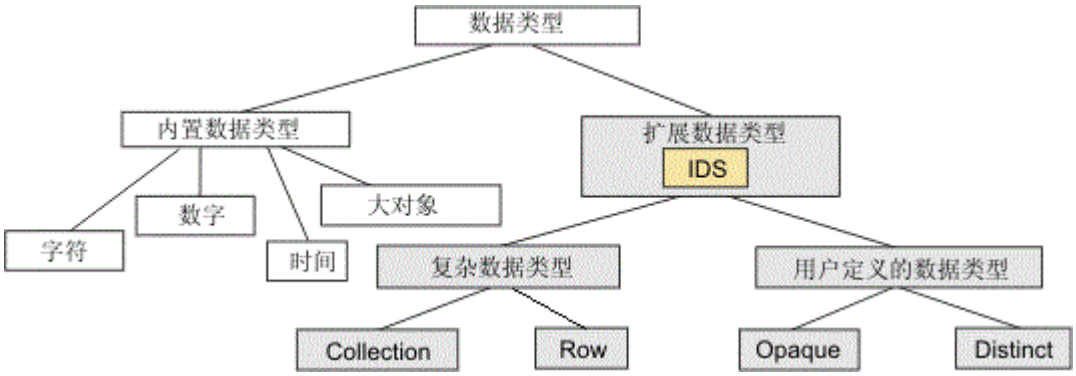
为数据库中的表的每一列指定数据类型。数据类型精确地定义可在该列中存储的值的种类。

这些主题描述内置和扩展数据类型、两种数据类型之间的强制转型和运算符优先级。

3.1 数据类型的摘要

GBase 8s 支持最常见内置数据类型集。此外，GBase 8s 上还支持扩展数据类型集。

下图显示 GBase 8s 支持的数据类型的逻辑类别。有阴影的类别表示仅在 GBase 8s 上受支持的其他数据类型。图 1. 受支持数据类型的概述



对此图作了简化；一些内置类型是作为不透明类型实现的，并且仅在 GBase 8s 上受支持。即，*不透明*和*内置*不是分离的类别，虽然大部分内置数据类型都不是不透明数据类型。

*内置*数据类型（系统定义的）和*扩展*数据类型（您可以定义）都具有下列特征。您可以：

使用它们来在数据库表中创建列。

将它们声明为自变量和例程的返回类型。

将它们用作从中创建 DISTINCT 数据类型的基本类型。

将它们强制转型为其他数据类型。

用 SPL 和 ESQL/C 声明和存取这些类型的主变量。

有关例外情况，请参阅每种数据类型的描述。要了解概述，请参阅内置数据类型和扩展数据类型。

可以使用 CREATE TABLE 语句将数据类型指定给列并使用 ALTER TABLE 语句更改数据类型。当更改现有列数据类型时，如果可能的话，会将所有数据转换为新数据类型。

有关 ALTER TABLE 和 CREATE TABLE 语句、创建特定数据类型及创建和删除强制转型

的 SQL 语句以及其他数据类型主题的信息，请参阅 *GBase 8s SQL 指南：语法*。

有关如何创建和使用 GBase 8s 支持的复杂数据类型的信息，请参阅《*GBase 8s 数据库设计和实现指南*》。有关如何创建用户定义的数据类型的信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

GBase 8s 支持的数据类型

下表列出 GBase 8s 支持的所有内置数据类型。

表 1. GBase 8s 支持的数据类型

数据类型	解释
BIGINT 数据类型	存储 8 个字节的整数值，从 -(263 -1) 到 263 -1
BIGSERIAL 数据类型	存储 8 个字节的顺序整数值，范围是 1 到 263 -1
BYTE 数据类型	存储任何种类的二进制数据，最长 231 个字节
CHAR(n) 数据类型	存储字符串；整理使用代码集顺序
CHARACTER(n) 数据类型	CHAR 的同义词
CHARACTER VARYING(m,r) 数据类型	存储可变长度字符串（符合 ANSI）；以代码集顺序整理
DATE 数据类型	存储日历日期
DATETIME 数据类型	存储日历日期和一天中的时间
DEC 数据类型	DECIMAL 的同义词
DECIMAL	存储精度可定义的浮点数；如果数据库符合 ANSI，那么小数位为零
DECIMAL (p,s) 定点	存储已定义小数位和精度的定点数
DOUBLE PRECISION 数据类型	FLOAT 的同义词
FLOAT(n)	存储相应于 C 中的 double 数据类型的双精度浮点数
INT 数据类型	INTEGER 的同义词
INT8	存储 8 个字节的整数值，从 -(263 -1) 到 263 -1
INTEGER 数据类型	存储从 -2,147,483,647 到 +2,147,483,647 的整数
INTERVAL 数据类型	以*年*和*月*为单位存储时间范围（或工作的级别）。
INTERVAL 数据类型	以连续的单位*日*、*小时*、*分钟*、*秒*和*秒的若干分之几*存储时间范围
MONEY(p,s) 数据类型	存储货币总额

NCHAR(n) 数据类型	与 CHAR 相同，但可以支持本地化的整理
NUMERIC(p,s) 数据类型	DECIMAL(*p, s*) 的同义词
NVARCHAR(m,r) 数据类型	与 VARCHAR 相同，但可以支持本地化整理
REAL 数据类型	SMALLFLOAT 的同义词
SERIAL(n) 数据类型	在 INT 的正数范围内存储连续整数 (> 0)
SERIAL8(n) 数据类型	在 INT8 的正数范围内存储连续整数 (> 0)
SMALLFLOAT	存储对应于 C 语言的 **float** 数据类型的单精度浮点数
SMALLINT 数据类型	存储从 -32,767 到 +32,767 的整数
TEXT 数据类型	存储任何种类的文本数据，最长 231 个字节
VARCHAR(m,r) 数据类型	存储可变长度字符串（最长 32,765 个字节）；以代码集顺序整理

这些内置 SQL 数据类型在所有 GBase 8s SQL 事务中都有效，包括这些类型的数据操作语言 (DML) 操作：

对本地数据库中对象的操作

对本地服务器实例数据库中对象的跨数据库操作

对两个或多个数据库服务器实例的数据库中对象的跨服务器操作

在跨服务器 MERGE 操作中，源表（而不是目标表）可以位于远程 GBase 8s 服务器的数据库中。

对于字符数据类型（CHAR、CHAR VARYING、LVARCHAR、NCHAR、NVARCHAR 和 VARCHAR），数据字符串可以包含数据库语言环境代码集中的字母、数字、标点符号、空格、区分标记、连字符和其他可打印符号。对于 **UTF-8** 和某些东亚语言环境的代码集，在数据字符串内支持多字节字符。

GBase 8s 支持的其他数据类型

下表列出 GBase 8s 支持的其他数据类型。

表 2. GBase 8s 支持的其他数据类型

数据类型	解释
BLOB 数据类型	以随机存取块的形式存储二进制数据
binary18	存储 18 字节二进制编码的字符串
binaryvar	存储最大长度为 255 个字节的二进制编码的字符串
BOOLEAN 数据类型	存储布尔值 true 和 false
CLOB 数据类型	以随机存取块的形式存储文本数据

数据类型	解释
DISTINCT 数据类型	以用户定义的类型存储数据，该类型与它所基于的源类型具有相同的格式，但该类型的强制转型和函数与源类型的不同
日历	存储 TimeSeries 数据类型的日历
日历模式	存储日历数据类型的日历模式的结构
IDSSECURITYLABEL 数据类型	存储 LBAC 安全标号对象。
LIST(e) 数据类型	存储元素的按顺序排序的集合，所有元素都具有同一数据类型 *e*；允许重复值
lld_locator	存储大对象标识
lld_lob_data	存储智能大对象的位置，并指定对象是否包含二进制或字符数据
LVARCHAR(m) 数据类型	存储可变长度字符串，可多至 32,739 个字节
MULTISET(e) 数据类型	存储值的非排序集合，所有元素都具有同一数据类型 *e*；允许重复值
节点 (node)	存储最多 256 个字符的可变长度的整数和小数点组合（用于表示分层关系）
OPAQUE 数据类型	存储用户定义的数据类型，其内部结构对于数据库服务器不可存取
ROW 数据类型，已命名	存储命名 ROW 类型
ROW 数据类型，未命名	存储未命名 ROW 类型
SET(e) 数据类型	存储元素的非排序集合，所有元素都具有同一数据类型 *e*；不允许重复值
ST_LineString	存储一维对象作为定义线性插值路径的点序列
ST_MultiLineString	存储 ST_LineString 数据类型的集合
ST_MultiPoint	存储 ST_Point 数据类型的集合
ST_MultiPolygon	存储 ST_Polygon 数据类型的集合
ST_Point	存储占用坐标空间中单个位置的零维几何
ST_Polygon	存储二维表面，并且作为定义外部边界环以及 0 或 0 个以上内部环的点序列存储
TimeSeries	存储行子类型的集合

其他主题中会单独描述 GBase 8s 的这些扩展数据类型。这些数据类型在定义数据类型的数据库上的本地操作中有效。

跨数据库分布式 SQL 事务中的扩展数据类型

对相同 GBase 8s 实例的其他数据库的分布式操作可访问 BOOLEAN、BLOB、CLOB 和 LVARCHAR 数据类型，这些数据类型是作为内置不透明类型来实现的。如果 UDT 和 DISTINCT 类型显式地强制转型为内置类型，并且如果所有的 UDT、强制转型和 DISTINCT

类型在所有参与的数据库中都进行了定义，这样的操作还能访问 DISTINCT 类型（它的基本类型是内置类型）和用户定义类型 (UDT)。

然而，您无法引用跨数据库事务（这些事务访问本地 GBase 8s 实例的多个数据库）中的以下扩展数据类型：

没有强制转型为内置数据类型的 UDT

没有强制转型为内置数据类型的 DISTINCT 类型

集合数据类型

命名或未命名 ROW 数据类型

跨服务器分布式 SQL 事务中的扩展数据类型

访问其他 GBase 8s 实例数据库的分布式 SQL 事务和函数调用无法返回复杂或智能大对象数据类型的值，也无法返回大部分单值或内置不透明数据类型。跨服务器 SQL 操作中只可访问以下数据类型：

任何透明的内置数据类型

BOOLEAN

透明的内置类型的 DISTINCT

BOOLEAN 的 DISTINCT

LVARCHAR 的 DISTINCT

以上所列出的任何 DISTINCT 类型的 DISTINCT

IDSSECURITYLABEL

LVARCHAR

只有 DISTINCT 数据类型显式地强制转型为内置类型，且所有 DISTINCT 类型、其数据类型层次结构和其强制转型在参与分布式操作的每个数据库中采用相同的定义时，跨服务器分布式 SQL 事务才可支持 DISTINCT 数据类型。对于使用上面列表中的数据类型作为参数或返回数据类型的跨服务器 UDR 中的查询或其他 DML 操作，UDR 在每个参与数据库中的定义也必须相同。

拥有足够安全凭证的用户可以在对受保护数据的跨服务器和跨数据库操作中，访问存储安全标号对象的内置 DISTINCT 数据类型 IDSSECURITYLABEL。如对受保护数据的本地操作，在数据库服务器在对保护数据安全的安全标号和发出查询的用户的安全凭证进行比较后，访问安全策略保护的远程表的分布式查询只可以返回 IDSLBACRULES 允许的符合条件的行。

3.2 ANSI 到 GBase 8s 的数据类型映射

GBase 8s 具有与大多数 ANSI 数据类型等效的数据类型。

下表显示 ANSI 数据类型和等效的 GBase 8s 数据类型。

表 3. ANSI 到 GBase 8s 的数据类型映射

ANSI 数据类型	GBase 8s 数据类型
CHARACTER(n) 或 CHAR(n)	CHARACTER(n) 或 CHAR(n)
CHARACTER VARYING(n) 或 VARCHAR(n)	CHARACTER VARYING(n)、VARCHAR(m,r) 或 LVARCHAR(n)
NATIONAL CHARACTER(n) 或 NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) 或 NVARCHAR(n)	NVARCHAR(m,r)
INTEGER	INTEGER 或 INT
SMALLINT	SMALLINT
FLOAT	FLOAT(n)
REAL	REAL 或 SMALLFLOAT
DOUBLE PRECISION	DOUBLE PRECISION 或 FLOAT(n)
NUMERIC(p,s) 或 DECIMAL(p,s)	NUMERIC(p,s) 或 DECIMAL(p,s)
DATE	DATE
TIMESTAMP	DATETIME YEAR TO FRACTION(n)

3.3 数据类型的描述

3.3.1 BIGINT 数据类型

BIGINT 数据类型存储从 $-(2^{63}-1)$ 到 $2^{63}-1$，即 -9,223,372,036,854,775,807 到 9,223,372,036,854,775,807 的 8 个字节的整数值。

该数据类型相比 INT8 具有一定的存储优势，且相比 INT8 和 DECIMAL 数据类型，它在一些算术运算和排序比较方面也具有优势。

3.3.2 BIGSERIAL 数据类型

BIGSERIAL 数据类型存储 BIGINT 数据类型的顺序整数（在插入新行时由数据库服务器自动指定）。BIGSERIAL 数据类型的行为与 SERIAL 数据类型相似，但范围更大。

缺省 BIGSERIAL 起始号为 1，但可以在创建或改变表时指定初始值 *n*。*n* 值必须为 1 至 9,223,372,036,854,775,807 范围内的正整数。如果将值零 (0) 插入到 BIGSERIAL 列，使用的值是 BIGSERIAL 列中已存在的最大正值 +1。如果您插入任何非零值，该值会原样插入。

由于所有串行数据类型都如此，因此 BIGSERIAL 数据类型会存储您提供的负值。然而，

生成的值通常是从 1 到 2^{63} -1 的正数。

BIGSERIAL 数据类型可以存储从 -(2^{63} -1) 到 2^{63} -1 的值，即 -9,223,372,036,854,775,807 到 9,223,372,036,854,775,807 的 8 个字节的值。

一个表只能有一个 SERIAL 列，但是它可以具有一个 SERIAL 列和一个 SERIAL8 列或 BIGSERIAL 列。

将 SERIAL8 和 BIGSERIAL 与 INT8 或 BIGINT 配合使用

对 INT8 和 BIGINT 有效的所有算术运算符（例如：+、-、* 和 /）以及对 INT8 和 BIGINT 有效的所有 SQL 函数（例如，ABS、MOD 和 POW 等等）对 SERIAL8 和 BIGSERIAL 值同样有效。

适用于 INT8 和 BIGINT 的数据转换规则也适用于 SERIAL8 和 BIGSERIAL，但对于 SERIAL8 或 BIGSERIAL 具有 NOT NULL 约束。

一个表的 SERIAL8 或 BIGSERIAL 列的值可存储在另一个表的 INT8 或 BIGINT 列中。但是，在第二个表中，INT8 或 BIGINT 值不遵从原始 SERIAL8 或 BIGSERIAL 列的约束。

3.3.3 BLOB 数据类型

BLOB 数据类型以随机存取块（称为智能大对象空间）的形式存储任何种类的二进制数据。二进制数据通常由已保存的电子表格、程序装入模块和数字化声音模式等等组成。数据库服务器不会对 BLOB 列的内容进行解释。

BLOB 列最长可为 4 太字节（4*2^{40} 个字节），不过系统资源可能会强加较低的实际限值。分配给智能大对象数据类型的最小磁盘空间量为 512 字节。

术语*智能大对象*指的是 BLOB 和 CLOB 数据类型。将 CLOB 数据类型（请参阅 CLOB 数据类型页）用于随机访问文本数据。有关 BLOB 和 CLOB 数据类型的一般信息，请参阅智能大对象。

您可以使用以下 SQL 函数对 BLOB 列执行操作：

FILETOBLOB 将文件复制到 BLOB 列。

LOTOFILE 将 BLOB（或 CLOB）值复制到操作系统文件中。

LOCOPY 将现有智能大对象复制至新的智能大对象。

有关这些 SQL 函数的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

在 SQL 中，只能对 BLOB 数据执行等于（ = ）比较运算以及使用加密和解密函数。（*GBase 8s SQL 指南：语法* 中描述了加密和解密函数。）要执行其他运算，必须使用客户机应用程序中的某种应用程序编程接口 (API)。

可以通过下列方法将数据插入到 BLOB 列：

使用 dbload 或 onload 实用程序

使用 LOAD 语句 (DB-Access)

使用 FILETOBLOB 函数

从 BLOB (**ifx_lo_t**) 主变量 (GBase 8s ESQL/C)

如果您使用 DB-Access 来选择 BLOB 列，那么只返回字符串 <SBlob 值>；不会显示任何实际值。

3.3.4 BOOLEAN 数据类型

BOOLEAN 数据类型将 TRUE 或 FALSE 数据值作为单字节存储。

下表显示了 BOOLEAN 数据类型的内部和文字表示法。

逻辑值	内部表示法	文字表示法
TRUE	\0	't'
FALSE	\1	'f'
NULL	仅供内部使用	NULL

可以比较两个 BOOLEAN 值来测试相等或不相等。还可以将 BOOLEAN 值与布尔文字 't' 和 'f' 进行比较。BOOLEAN 值是不区分大小写的；'t' 等同于 'T'，而 'f' 等同于 'F'。

可使用 BOOLEAN 列存储布尔表达式返回的内容。在下面的示例中，如果 **column1** 小于 **column2**，那么 boolean_column 的值为 't'；如果 **column1** 大于或等于 **column2**，就为 'f'；如果 **column1** 或 **column2** 的值未知，就为 NULL：

```
UPDATE my_table SET boolean_column = lessthan(column1, column2)
```

3.3.5 BYTE 数据类型

BYTE 数据类型以无差别字节流的形式存储任何种类的二进制数据。二进制数据通常由数字化的信息（如，电子表格、程序装入模块和数字化声音模式等等）组成。

术语*简单大对象*指的是 BYTE 和 TEXT 数据类型。可以将相同表的不超过 195 列声明为 BYTE 和 TEXT 数据类型。

BYTE 数据类型不具有最大大小。BYTE 列具有 2^{31} 个字节的理论限制和磁盘容量确定的实际限制。

可以存储、检索、更新或删除 BYTE 列的内容。但是，不能在算术或字符串运算中使用 BYTE 操作数，也不能使用 UPDATE 语句的 SET 子句将文字指定给 BYTE 列。也不能通过以下任何方法使用 BYTE 项：

使用聚集函数

使用 IN 子句

使用 MATCHES 或 LIKE 子句

使用 GROUP BY 子句

使用 ORDER BY 子句

仅当您正在使用 IS NULL 或 IS NOT NULL 运算符测试 NULL 值时，BYTE 操作数在布尔表达式中才有效。

您可以使用以下可装入行或更新字段的方法来插入 BYTE 数据：

使用 **dbload** 或 **onload** 实用程序

使用 LOAD 语句 (DB-Access)

从 BYTE 主变量 (GBase 8s ESQL/C)

不能使用带引号的文本字符串、数字或任何其他实际值来插入或更新 BYTE 列。

当您选择 BYTE 列时，可以接收该列的全部或一部分。要检索整个列，使用选择列的常规语法。还可以通过使用下标来选择 BYTE 列的任何部分，如下一示例所示，该示例读取与目录号 10001 相关联的 cat_picture 列的前 75 个字节：

```
SELECT cat_picture [1,75] FROM catalog WHERE catalog_num = 10001
```

内置强制转型将 BYTE 值转换为 BLOB 值。有关更多信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

如果使用 DB-Access Interactive Schema Editor 选择 BYTE 列，那么仅返回字符串“<BYTE值>”；不显示任何数据值。

要点： 如果您尝试从子查询中返回 BYTE 列，那么即使既未在布尔表达式中也未在聚集中使用该列，也会导致错误。

3.3.6 CHAR(n) 数据类型

CHAR 数据类型存储任何字母、数字和符号组成的字符串。它可以根据数据库语言环境存储单字节和多字节字符。

CHAR(*n*) 列的长度为 *n* 个字节，其中 1 ≤ *n* ≤ 32,767。如果未指定 *n*，那么 CHAR(1) 是缺省长度。字符列通常存储字母数字字符串，例如：姓名、地址和电话号码等等。当将值作为 CHAR(*n*) 检索或存储时，只传输 *n* 个字节的数据。如果字符串短于 *n* 个字节，那么用空格将该字符串延长到声明的长度。如果数据值大于 *n* 个字节，那么插入或检索已从右边截断至长度为 *n* 的数据字符串，并且数据库服务器不会发生异常。

这在多字节语言环境中不会创建部分字符。在从右至左语言环境（如阿拉伯语、希伯来语或波斯语）中，是从左边截断的。

CHAR 数据类型声明中的大小规范可受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 功能影响。

将 CHAR 值看作数字值

CHAR 数据类型存储数字值算术字符串操作数如果您打算对存储在列中的数字执行计

算，那么应将数字数据类型指定给该列。虽然可以在 CHAR 列中存储数字，但也许不能在某些算术运算中使用这些数字。例如：如果将总和插入 CHAR 列中，并且如果 CHAR 列太小而无法保存该值，那么可能会遇到溢出问题。在这种情况下，插入操作将失败。如果将有前导零的数字（如一些邮政编码）作为数字类型 INTEGER 或 SMALLINT 存储，那么会除去前导零。相反，可以将这些数字存储在 CHAR 列中。

排序和关系比较

CHAR 数据类型整理整理CHAR 数据类型一般而言，对 CHAR 值进行排序的整理顺序就是代码集中字符的顺序。(一个例外情况是有范围的 MATCHES 运算符；请参阅整理 VARCHAR 值。）有关整理顺序的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

代码集东亚语言环境多字节对于多字节语言环境，数据库支持代码集中的任何多字节字符。当以 CHAR 数据类型存储多字节字符时，务必计算所需字节的数目。有关多字节字符和语言环境的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

关系运算符()，空格填充 CHAR 值可将 CHAR 值与其他 CHAR 值进行比较，方法是用空格填充右边长度较小的值直到这些值具有相同的长度，然后使用整理的代码集顺序来比较这两个值。

CHAR 类型的不可打印字符

CHAR 值可以包括制表符、换行符、空格和不可打印的字符。但是，必须使用应用程序来将不可打印的字符插入主变量中，并将主变量插入数据库中。在将不可打印字符传递至数据库服务器之后，就可以存储或检索这些字符。在选择了不可打印字符之后，将其访存到主变量中并使用您的显示机制来显示这些字符。

一个重要的异常是 ASCII 代码中的首值在 CHAR 数据类型的列中被用作数据结束终止符。由于这个原因，不能从 CHAR 列检索相同字符串中的任何后续字符，因为数据库服务器仅读取此空终止符前面的字符（如果有）。例如，不能将以下 7 字节的字符串用作长度为 7 个字节的 CHAR 数据类型：

```
abc\0def
```

如果尝试使用 DB-Access 来显示不可打印字符，那么屏幕会返回不一致的结果。（哪些字符是不可打印字符是特定于语言环境的。有关更多信息，请参阅《*GBase 8s GLS 用户指南*》中客户机和数据库服务器之间的代码集转换的论述。）

3.3.7 CHARACTER(n) 数据类型

CHARACTER 数据类型是 CHAR 的同义词。

3.3.8 CHARACTER VARYING(m,r) 数据类型

CHARACTER VARYING 数据类型存储字母、数字和可变长度的符号组成的字符串，其中 *m* 是列的最大大小（以字节计），而 *r* 是为该列保留的最小字节数。

CHARACTER VARYING 数据类型符合 SQL 的 ANSI/ISO 标准；非 ANSI VARCHAR 数据类型支持相同的功能。有关更多信息，请参阅 VARCHAR(m,r) 数据类型中对 VARCHAR 类型的描述。

3.3.9 CLOB 数据类型

CLOB 数据类型以随机存取块(称为智能大对象空间)的形式存储任何种类的文本数据。如果此信息也是文本的（例如，PostScript(TM)、“超文本标记语言”(HTML)、“标准图形标记语言”(SGML) 或“可扩展标记语言”(XML) 数据），那么文本数据可包括文本格式的信息。

术语智能大对象指的是 CLOB 和 BLOB 数据类型。CLOB 数据类型支持对不适合 BLOB 值的字符串进行特殊运算。CLOB 值最长可为 4 太字节（$4*2^{40}$ 个字节）。分配给智能大对象数据类型的最小磁盘空间量为 512 字节。

将 BLOB 数据类型(请参阅 BLOB 数据类型)用于随机访问二进制数据。有关 CLOB 和 BLOB 数据类型的常规信息，请参阅智能大对象。

以下 SQL 函数可以对 CLOB 列执行操作：

FILETOCLOB 将文件复制到 CLOB 列。

LOTOFILE 将 CLOB（或 BLOB）值复制到文件中。

LOCOPY 将 CLOB（或 BLOB）值复制到新的智能大对象。

ENCRYPT_DES 或 ENCRYPT_TDES 从纯文本 CLOB 参数创建一个加密的 BLOB 值。

DECRYPT_BINAR 或 DECRYPT_CHAR 从加密的 BLOB 参数返回未加密的 BLOB 值（该 ENCRYPT_DES 或 ENCRYPT_TDES 从纯文本 CLOB 值中创建）。

CLOB 数据没有任何强制转型。因此，除非使用这些加密和解密函数返回一个 BLOB，否则数据库服务器无法将 CLOB 类型的数据转换为任何其他数据类型。在 SQL 中，只能对 CLOB 数据执行等于（ = ）比较运算。要执行其他运算，必须从客户机应用程序中使用某种应用程序编程接口。

CLOB 类型的多字节字符

可以通过下列方法将数据插入到 CLOB 列：

使用 dbload 或 onload 实用程序

使用 LOAD 语句 (DB-Access)

从 CLOB (**ifx_lo_t**) 主变量 (ESQL/C)

有关 CLOB 类型的示例，请参阅 *GBase 8s SQL 指南：教程* 和《*GBase 8s 数据库设计和实现指南*》。

对于 GLS，以下规则适用：

必须在数据库语言环境中定义多字节 CLOB 字符。

用代码集顺序整理 CLOB 数据类型。

数据库服务器处理 CLOB 数据的代码集转换。

有关数据库语言环境、整理顺序和代码集转换的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

3.3.10 DATE 数据类型

DATE 数据类型存储日历日期。DATE 数据类型需要四个字节。日历日期在内部存储为等于自 1899 年 12 月 31 日以来的天数的整数值。

因为 DATE 值是作为整数存储的，所以可在算术表达式中使用这些值。例如，可以从另一个 DATE 值中减去 DATE 值。结果（正或负 INTEGER 值）指示两个日期之间经过的天数。（您可以使用 UNITS DAY 表达式来将结果转换成 INTERVAL DAY TO DAY 数据类型。）

以下示例显示 DATE 列的缺省显示格式：

mm/dd/yyyy

在此示例中，*mm* 是月份 (1-12)，*dd* 是月中某日 (1-31)，*yyyy* 是年份 (0001-9999)。可以通过设置 **DBDATE** 环境变量来指定不同的时间单位顺序和不同于 / 的时间单位分隔符（或无分隔符）。有关更多信息，请参阅 DBDATE 环境变量。

在非缺省语言环境中，可以用特定于文化的格式显示日期。语言环境以及 **GL_DATE** 和 **DBDATE** 环境变量（如下一章中所述）影响 DATE 值的显示格式。但是，它们不影响数据库中 DATE 列的内部存储格式。有关更多信息，请参阅《*GBase 8s GLS 用户指南*》。

3.3.11 DATETIME 数据类型

DATETIME 数据类型会存储以日历日期和一天中的时间表示的瞬间时刻。

您选择存储 DATETIME 值的精确度；其精度范围可从一年到几分之一秒。

DATETIME 将数据值存储为在数据类型声明中表示每个时间单位（*年*、*月*和*日*等等）的连续字段序列。

用于指定 DATETIME 数据类型的字段限定符具有以下格式：

DATETIME *largest_qualifier* TO *smallest_qualifier*

这类似于 INTERVAL 字段限定符，但 DATETIME 表示时间点而不是时间范围（INTERVAL 表示时间范围）。DATETIME 和 INTERVAL 限定符之间存在如下差异：

DATETIME 关键字替换 INTERVAL 关键字。

DATETIME 字段限定符不能指定 *largest_qualifier* 时间单位的非缺省精度。

DATETIME 数据类型的字段限定符可以包含 YEAR、MONTH 和更小的时间单位，而包含 DAY 字段限定符（或更小时间单位）的 INTERVAL 数据类型不能还包含 YEAR 或 MONTH 字段限定符。

如果 *smallest_qualifier* 未指定大于 *largest_qualifier* 的时间单位，那么

DATETIME 数据类型的 *largest_qualifier* 和 *smallest_qualifier* 可以是下表列出的任何字段。（最大和最小时间单位可以相同；例如，DATETIME YEAR TO YEAR。）

表 4. DATETIME 字段限定符

限定符字段	有效条目
YEAR	从 1 到 9,999（公元）编号的年份
MONTH	从 1 到 12 编号的月份
DAY	从 1 到 31 编号的日（适合于月）
HOUR	从 0（午夜）到 23 编号的小时
MINUTE	从 0 到 59 编号的分钟
SECOND	从 0 到 59 编号的秒
FRACTION	最多具有 5 位小数位的秒的十进制小数。缺省小数位是 3 位（千分之一秒）。对于 *smallest_qualifier*，要指定另一个小数位，请写 FRACTION(*n*)，其中 *n* 是位数（从 1 到 5）。

DATETIME 列的声明不需要包括全部 YEAR 至 FRACTION 时间单位范围。它可以包括这些时间单位的任何连续子集，或者甚至仅包括单个时间单位。

例如，如果每个输入的值都包含连续的时间单位系列的信息，就可以在声明为 YEAR TO MINUTE 的列中输入 MONTH TO HOUR 值。但是，不能只输入 MONTH 和 HOUR 的值；条目中还必须包括 DAY 的值。

如果使用 DB-Access TABLE 菜单，并且未指定 DATETIME 限定符，那么会指定缺省 DATETIME 限定符 YEAR TO YEAR。

有效的 DATETIME 文字必须包含 DATETIME 关键字、要输入的值和字段限定符。如前面说明的那样，由于输入的值可包含比对列声明的字段数少的字段，所以必须包括这些限定符。第一个字段和最后一个字段的可接受限定符与表 1列出的有效 DATETIME 字段的列表完全相同。

将字段限定符的值用整数表示并用定界符隔开这些限定符。下表列出了在缺省的美国英语语言环境中与 DATETIME 值配合使用的定界符。（这些定界符是在 INTERVAL 值中使用的定界符的超集。）

表 5. 与 DATETIME 配合使用的定界符

定界符	在 DATETIME 文字中的位置
连字符（-）（=）	在 YEAR、MONTH 和 DAY 时间单位值之间
空格（ ）	在 DAY 与 HOUR 时间单位值之间
冒号（：）	在 HOUR、MINUTE 与 SECOND 时间单位值之间
小数点（.）	在 SECOND 与 FRACTION 时间单位值之间

下图显示具有定界符的 DATETIME YEAR TO FRACTION(3) 值。图 2. 具有定界符的 DATETIME 值示例

当输入具有比列中的时间单位字段少的时间单位字段的值时，就会自动扩展输入的值以填满所有声明的时间单位字段。如果漏掉任何较高有效字段（即，时间单位大于包含的任何时间单位），那么会用系统时钟日历中的那些时间单位的当前值自动填充这些字段。如果漏掉任何不太重要的字段，那么在条目中会用零（或对 MONTH 和 DAY 使用 1）填充这些字段。

还可以将 DATETIME 值作为字符串输入。字符串必须包括 DATETIME 列中定义的每个字段的信息。下面示例中的 INSERT 语句显示了作为字符串输入的 DATETIME 值：

```
INSERT INTO cust_calls (customer_num, call_dtime, user_id,
            call_code, call_descr)
    VALUES (101, '2001-01-14 08:45', 'maryj', 'D',
'Order late - placed 6/1/00');
```

如果将 **call_dtime** 声明为 DATETIME YEAR TO MINUTE，那么字符串必须包含*年*、*月*、*日*、*小时*和*分钟*字段的值。

如果字符串不包含所有已声明字段的信息（或如果它添加了其他字段），那么数据库服务器会返回错误。

除了 *year* 和 *fraction* 字段之外，DATETIME 列的所有字段都是 2 位数。*year* 字段存储为 4 位。当在年份字段中输入两位值时，缩写的年份扩展为四位的方式取决于 DBCENTURY 环境变量的设置。

例如：如果输入 02 作为 *year* 值，那么将该年份是解释为 1902、2002 还是 2102 取决于 DBCENTURY 的设置和执行时系统时钟日历的值。如果不设置 DBCENTURY，那么在缺省情况下，将追加当前年份的前导数字。

*小数*字段需要 *n* 位数，其中 1 ≤ *n* ≤ 5，上舍入为偶数。可以使用以下公式（上舍入为整数个字节）来计算 DATETIME 值需要的字节数：

（所有字段的总位数）/2 + 1

例如，YEAR TO DAY 限定符总共需要八位（四位用于*年*，两位用于*月*，两位用于*日*）。根据公式，此数据值需要 5（即 (8/2) + 1）个字节的存储器。

当数据库服务器在 SQL 语句中从操作系统获取当前时间时，USEOSTIME 配置参数可能会影响亚秒级粒度。有关详细信息，请参阅 *GBase 8s 管理员参考*。

使用 ESQL API 时，DBTIME 环境变量会影响 DATETIME 格式。GL_DATE 和 DBDATE 环境变量的非缺省语言环境和设置还会影响日期时间数据的显示。但是，它们不会影响 DATETIME 列的内部存储格式。

如果指定的语言环境不是美国英语，那么该语言环境为 DATETIME 值定义特定于文化的显示格式。要更改缺省显示格式，更改 GL_DATETIME 环境变量的设置。当具有非缺省语言环境的数据库使用非缺省 GL_DATETIME 设置时，USE_DTENV 环境变量必须设置为 1，数据库服务器才能在以下运算中正确处理本地化的 DATETIME 值：

使用 DB-Access 的 LOAD 或 UNLOAD 功能

使用 dbexport 或 dbimport 迁移实用程序

对 CREATE EXTERNAL TABLE 语句定义的数据库表或对象使用 SQL 的 DML 语句。

3.3.12 DEC 数据类型

DEC 数据类型与 DECIMAL 同义。

3.3.13 DECIMAL 数据类型

DECIMAL 数据类型可采用两种格式：DECIMAL (*p*) 浮点和 DECIMAL (*p,s*) 定点。

在符合 ANSI 标准的数据库中，所有 DECIMAL 数字都是定点。

缺省情况下，数据库服务器将包括小数点 (.) 的文字数值解释为 DECIMAL 值。

DECIMAL(p) 浮点

DECIMAL 数据类型存储最多 32 个有效位的浮点十进制数字，其中 *p* 是总有效位数（*精度*）。

可以选择指定精度。如果未指定任何精度 (*p*)，那么将 DECIMAL 视为 DECIMAL(16)（具有 16 位精度的浮点十进制）。DECIMAL(*p*) 具有在 10^{-130} 与 10^{124} 之间的绝对指数范围。

如果在符合 ANSI 标准的数据库中声明 DECIMAL(*p*) 列，那么小数位的缺省值为 DECIMAL(*p*, 0) 意味着只能将整数值存储在此数据类型中。

在不符合 ANSI 标准的数据库中，DECIMAL(*p*) 是一个数值范围大到足以存储值的指数符号表示法的浮点数据类型。

例如：以下计算说明 DECIMAL(5) 列在缺省语言环境中需要多少存储字节（其中小数点占据一个单独的字节）： 数据值的符号为 1 字节

第一个数字为 1 字节

小数点为 1 字节

其余数字为 4 字节（精度 5 - 1）

e 符号为 1 字节

指数符号为 1 字节

指数为 3 字节

总共 12 个字节 因此，DECIMAL(5) 列中的“12345”在不符合 ANSI 标准的数据库内显示为“12345.00000”（即，有六位小数）。

DECIMAL (p,s) 定点

在定点数字 DECIMAL(*p, s*) 中，不管数字值如何，小数点都固定在特定位置。当您指定此类型的列时，就将其精度 (*p*) 声明为它可存储的总位数（从 1 到 32）。将其*小数位* (*s*) 声明为小数部分的总位数（即，小数点右边的位数）。

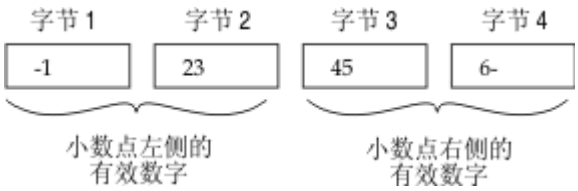
绝对值小于 $0.5 * 10^{-s}$的所有数都具有值零。可存储而不会发生溢出错误的 DECIMAL(*p, s*) 数据类型的最大绝对值是 $10^{p-s} - 10^{-s}$。DECIMAL 列通常存储的是必须正确存储和显示的带小数部分的数（例如，比率或百分率）。在符合 ANSI 标准的数据库中，所有 DECIMAL 数的绝对值的范围必须在 10^{-32} 至 10^{+31} 内。

DECIMAL 存储器

数据库服务器使用一个字节的磁盘存储空间来存储两位十进制数，并加上一个字节来存储指数和符号（第一个字节用 excess-65 格式表示一个符号位和 7 位指数）。余下的字节将尾数表示为 base-100 位。小数点左边的有效位和小数点右边的有效位存储在独立的字节组中。当使用最大*精度*规范时，DECIMAL(32,s) 数据类型可在小数点右边存储 s-1 个小数位数（如果 s 是奇数）。

在下面的示例中说明了数据库服务器存储小数的方式。如果指定 DECIMAL(6,3)，那么数据类型由整数部分中的三个有效位和小数部分中的三个有效位组成（例如：123.456）。小数点左边的三位存储在 2 个字节上（其中一个字节只保存一位），小数点右边的三位存储在另外 2 个字节上，如图 1所示。

（未显示指数字节。）由于指数和符号需要额外字节，所以 DECIMAL(6,3) 总共需要 5 个字节的存储空间。 图 3. 说明 Decimal (p,s) 值中数位存储的示意图



可使用以下公式（下舍入为整数字节数）来计算 DECIMAL(*p, s*) 数据类型的字节存储 (N)（其中 N 包括存储指数和符号所需的字节）：

如果*小数位*为奇数：N = (*精度* + 4) / 2

如果*小数位*为偶数：N = (*精度* + 3) / 2

例如，数据类型 DECIMAL(5,3) 需要 4 个字节的存储空间（9/2 下舍入等于 4）。

使用这些公式时要注意一点。数据库服务器用来存储小数值的最大字节数是 17。一个字节用来存储指数和符号，其余的 16 个字节用来存储最多 32 位精度。但是，如果指定精度 32 和*奇数*小数位，那么会丢失 1 位精度。例如，考虑数据类型 DECIMAL(32,31)。将此小数定义为小数点左边 1 位，小数点右边 31 位。小数点左边的 1 位需要 1 个字节的存储器。这使得只留下 15 个字节的存储器给小数点右边的位。15 个字节只能容纳 30 位，因此丢失 1 位精度。

3.3.14 DISTINCT 数据类型

DISTINCT 类型是从源类型（称为基本类型）派生的数据类型。

源类型可以是：

内置类型

现有 DISTINCT 类型

现有命名 ROW 类型

现有不透明类型

DISTINCT 类型继承其源类型在磁盘上的长度和对齐方式。因此，DISTINCT 类型高效地使用数据库服务器预先存在的功能。

创建 DISTINCT 数据类型时，数据库服务器将自动创建两种显式强制转型：一种从 DISTINCT 类型强制转型为其源类型，另一种是从源类型强制转型为 DISTINCT 类型。基于内置源类型的 DISTINCT 类型不继承为内置类型提供的内置强制转型。但是，DISTINCT 类型却继承已对源类型定义的任何用户定义的强制转型。

DISTINCT 类型不能直接与其源类型进行比较。要比较这两种类型，首先必须显式地将一种类型强制转型为另一种类型。

必须在数据库中定义 DISTINCT 类型。DISTINCT 类型的定义存储在 **sysxtdtypes** 系统目录表中。以下 SQL 语句会维护数据库中 DISTINCT 类型的定义：

CREATE DISTINCT TYPE 语句将 DISTINCT 类型添加至数据库。

DROP TYPE 语句从数据库中除去先前定义的 DISTINCT 类型。

有关上面提到的 SQL 语句的更多信息，请参阅 *GBase 8s SQL 指南：语法*。有关对 DISTINCT 数据类型强制转型的信息，请参阅单值类型的强制转型。有关显示如何创建和注册 DISTINCT 类型的强制转型函数的示例，请参阅《*GBase 8s 数据库设计和实现指南*》。

基本类型为内置字符类型的 DISTINCT 类型的声明中的大小规范可受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 功能影响。

3.3.15 DOUBLE PRECISION 数据类型

DOUBLE PRECISION 关键字是 FLOAT 关键字的同义词。

3.3.16 FLOAT(n) 数据类型

FLOAT 数据类型存储最多 17 个有效位的双精度浮点数。FLOAT 对应于 IEEE 4 字节浮点，并对应于 C 中的 **double** 数据类型。FLOAT 数据类型的值范围与计算机上 C **double** 数据类型的值范围相同。

您可以使用 *n* 来指定 FLOAT 数据类型的精度，但 SQL 会忽略该精度。值 *n* 必须是 1 与 14 之间的整数。

具有 FLOAT 数据类型的列通常存储只能近似计算的科学数字。由于浮点数只保留其大部分有效数字，所以您在此类型的列中输入的数字与数据库服务器显示的数字可能会稍有不同。

两个值之间的差别取决于计算机内部存储浮点数的方式。例如：您可能会将值 1.1000001 输入 FLOAT 字段中，那么在处理了 SQL 语句之后，数据库服务器可能会将此值显示为 1.1。当值具有的位数比浮点数可存储的位数多时，就会发生这种情况。在这种情况下，用值的近似形式存储值，将其余有效数位视为零。

对于 FLOAT 数据类型，每个值通常需要 8 个字节的存储器。将 FLOAT 值转换为 DECIMAL 值产生 17 位精度。

3.3.17 IDSSECURITYLABEL 数据类型

IDSSECURITYLABEL 类型将安全标号存储在受基于标号的访问控制 (LBAC) 安全策略保护的表中。

只有具有 **DBSECADM** 角色的用户才能创建、改变或删除此数据类型的列。IDSSECURITYLABEL 是内置的 DISTINCT OF VARCHAR(128) 数据类型。 具有安全策略的表可以只有一个 IDSSECURITYLABEL 列。 不具有安全策略的表可以没有该列。不能在类型为 IDSSECURITYLABEL 的列中对安全标号进行加密。

3.3.18 INT 数据类型

INT 数据类型与 INTEGER 同义。

3.3.19 INT8 数据类型

INT8 数据类型存储从值为范围 －9,223,372,036,854,775,807 到 9,223,372,036,854,775,807［或 $-(2^{63}-1)$ 到 $2^{63}-1$］的整数（18 或 19 位精度）。

数字 －9,223,372,036,854,775,808 是保留值，不能使用。INT8 数据类型通常用来存储很大的计数和数量等等。

GBase 8s 使用可能需要多达 10 个字节存储空间的内部格式来存储 INT8 数据。

对整数数据执行算术运算和排序比较比对浮点或定点小数数据执行的效率要高，但 INT8 不能存储绝对值超过 ｜ $2^{63}-1$ ｜ 的数据。如果值超出 INT8 的数字范围，那么数据库

服务器不存储该值。

3.3.20 INTEGER 数据类型

INTEGER 数据类型对 9 或 10 位精度存储范围为 -2,147,483,647 到 2,147,483,647 的整数。

数字 2,147,483,648 是保留值，不能使用。INTEGER 值存储为有符号的二进制整数，并且通常用来存储计数和数量等等。

对整数数据执行算术运算和排序比较比对浮点或小数数据执行效率要高。但是，INTEGER 列不能存储范围超出 $(2^{31}-1)$ 的绝对值。如果数据值超出 INTEGER 的数字范围，那么数据库服务器不存储该值。

INTEGER 数据类型的每个值需要 4 个字节的存储器。

3.3.21 INTERVAL 数据类型

INTERVAL 数据类型存储表示时间范围的值。INTERVAL 类型分为两类：*year-month 时间间隔*和 *day-time 时间间隔*。

year-month 时间间隔可以表示年和月的范围，day-time 时间间隔可以表示天、小时、分钟、秒和秒的小数的范围。

INTERVAL 值总是由表示时间单位的一个值或一串值组成。在数据定义语句（例如：定义 INTERVAL 数据类型精度的 CREATE TABLE 或 ALTER TABLE）中，限定符必须使用以下格式：

INTERVAL *largest_qualifier*(*n*) TO *smallest_qualifier*

如表 1中所示，此处 *largest_qualifier* 和 *smallest_qualifier* 关键字取自两个 INTERVAL 类中的一个。

如果 SECOND（或更大的时间单位）是 *largest_qualifier*，那么 INTERVAL 数据类型的声明可有选择地指定最大时间单位的精度 *n*(*n* 的范围为 1 到 9)；这不是 DATETIME 数据类型的功能。

如果 *smallest_qualifier* 是 FRACTION，那么还可以指定范围为 1 到 5 的小数位。对于 FRACTION TO FRACTION 限定符，*n* 的上限是 5 而不是 9。有两类不可比较的 INTERVAL 数据类型：

smallest_qualifier 大于 DAY

largest_qualifier 小于 MONTH

表 6. 时间间隔类

时间间隔类	时间单位	有效条目
YEAR-MONTH INTERVAL	YEAR	年数

时间间隔类	时间单位	有效条目
YEAR-MONTH INTERVAL	MONTH	月数
DAY-TIMEINTERVAL	DAY	天数
DAY-TIMEINTERVAL	HOUR	小时数
DAY-TIMEINTERVAL	MINUTE	分钟数
DAY-TIMEINTERVAL	SECOND	秒数
DAY-TIMEINTERVAL	FRACTION	秒的十进制小数，最多为 5 位。缺省小数位是 3 位（千分之一秒）。要指定非缺省小数位，请写 FRACTION(*n*)，其中 1 ≤ *n* ≤ 5。

如同 DATETIME 数据类型一样，可以定义 INTERVAL，从而只包括所需的部分时间单位。但由于“月份”的构造（如在日历日期中使用时）不是具有固定天数的时间单位，所以单个 INTERVAL 值不能组合月份和日；不支持涉及两种不同的 INTERVAL 类的操作数的算术。

输入到 INTERVAL 列中的值不需要包括在列的数据类型声明中指定的全部时间单位。例如，可以将具有 HOUR TO SECOND 精度的值输入定义为 DAY TO SECOND 的列中。但是，值必须总是由连续的时间单位组成。在前面的示例中，不能只输入 HOUR 和 SECOND 值；还必须包括 MINUTE 值。

有效的 INTERVAL 文字包含 INTERVAL 关键字、要输入的值和字段限定符。（请参阅 *GBase 8s SQL 指南：语法* 中对文字时间间隔的论述。）当值只包含一个字段时，最大和最小字段相同。

当在 INTERVAL 列中输入值时，必须指定值中的最大和最小字段，就如对待 DATETIME 值那样。另外，可以有选择地指定第一个字段的精度（如果最后一个字段是 FRACTION，那么还可以指定最后一个字段的小数位）。如果最大和最小字段限定符都是 FRACTION，那么只能在最后一个字段中指定小数位。

最大和最小字段的可接受限定符与表 1显示的 INTERVAL 字段的列表完全相同。

如果使用 DB-Access**TABLE** 菜单但没有指定任何 INTERVAL 字段限定符，那么指定缺省 INTERVAL 限定符 YEAR TO YEAR。

INTERVAL 值中的 *largest_qualifier* 最多可以为 9 位数（FRACTION 除外，它不能超过 5 位数），但如果想要输入的值大于该字段允许的缺省位数，那么必须显式标识输入的值中的有效位数。例如，要定义最多可存储 999 天的 DAY TO HOUR 类型的 INTERVAL，可以按以下方式进行指定：

```
INTERVAL DAY(3) TO HOUR
```

INTERVAL 文字值使用 DATETIME 文字所用的定界符（在同一 INTERVAL 值中无效的 MONTH 和 DAY 时间单位除外）。下表显示了 INTERVAL 定界符。

表 7. INTERVAL 定界符

定界符	在 INTERVAL 文字中的位置
连字符	在值的 YEAR 与 MONTH 部分之间
空格	在值的 DAY 与 HOUR 部分之间
冒号	在值的 HOUR、MINUTE 与 SECOND 部分之间
小数点	在值的 SECOND 与 FRACTION 部分之间

还可以将 INTERVAL 值作为字符串输入。字符串必须包括在列的数据类型声明中指定的同一时间单位的信息。下面示例中的 INSERT 语句显示了作为字符串输入的 INTERVAL 值：

```
INSERT INTO manufact (manu_code, manu_name, lead_time)
    VALUES ('BRO', 'Ball-Racquet Originals', '160')
```

由于将 **lead_time** 列定义为 INTERVAL DAY(3) TO DAY，所以此 INTERVAL 值只需要一个字段，即前导时间所需的日范围。如果字符串不包含所有字段的信息(或添加其他字段)，那么数据库服务器返回错误。有关将 INTERVAL 值作为字符串来输入的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

缺省情况下，INTERVAL 列的所有字段都是 2 位数，年份和小数字段除外。年份字段存储为 4 位数。小数字段需要 *n* 位数，其中 1 ≤ *n* ≤ 5，上舍入为偶数。可以使用以下公式（上舍入为整数个字节）来计算 INTERVAL 值所需的字节数：

（所有字段的总位数）/2 + 1

例如，INTERVAL YEAR TO MONTH 需要 6 位数（4 位用于*年份*，2 位用于*月份*）并需要 4（即 (6/2) + 1）个字节的存储空间。

有关将 INTERVAL 用作常量表达式的信息，请参阅 *GBase 8s SQL 指南：语法* 中对 INTERVAL 字段限定符的描述。

3.3.22 LIST(e) 数据类型

LIST 数据类型是可存储相同 SQL 数据类型的有序非 NULL 元素的集合类型。

LIST 数据类型支持（但不需要）重复元素值。LIST 数据类型的元素具有顺序位置。LIST 对象必须具有第一元素，可后跟第二元素，以此类推。

有关不支持顺序位置的无序集合数据类型，请参阅 MULTISET(e) 数据类型和 SET(e) 数据类型。有关可以存储一组包含不同 SQL 数据类型值的复杂数据类型，请参阅 ROW 数据类型。

缺省情况下，数据库服务器将新元素插入元素集末尾的插入 LIST 对象中。为了支持 LIST 的顺序位置，INSERT 语句提供了 AT 子句。此子句允许您指定要插入 LIST 元素值的位置。有关更多信息，请参阅 *GBase 8s SQL 指南：语法* 中的 INSERT 语句。

LIST 对象中的所有元素都具有相同的元素类型。要指定元素类型，使用以下语法：

LIST(*element_type* NOT NULL)

LIST 的 *element_type* 可以是下列数据类型之一：

内置类型（SERIAL、SERIAL8、BIGSERIAL、BYTE 和 TEXT 除外）

DISTINCT 类型

未命名或命名 ROW 类型

其他集合类型

不透明类型

必须对 LIST 元素指定 NOT NULL 约束。其他约束对 LIST 列无效。有关 LIST 数据类型语法的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

可以在任何其他数据类型有效的地方使用 LIST。例如：

在用来搜索匹配的 LIST 值的 SELECT 语句 WHERE 子句中的 IN 谓词后面

用作确定 LIST 列中的元素数目的 CARDINALITY 或 **mi_collection_card( )** 函数的自变量

*不能*将 LIST 值用作聚集函数（例如，AVG、MAX、MIN 或 SUM）的自变量。

与使用其他集合数据类型一样，必须在数据类型声明中使用括号（( )）来为 LIST 数据类型的元素集定界：

```
CREATE FUNCTION   update_nums( list1 LIST (ROW (a VARCHAR(10),
                                               b VARCHAR(10),
                                               c INT) NOT NULL ));
```

但是，在包含文字 LIST 值的 SQL 表达式中，必须使用花括号（{ }）来对 LIST 对象的元素集定界，如以下示例中所示。

如果两个 LIST 值具有相同顺序的相同元素，那么它们相等。以下是 LIST 对象的两个示例，但是它们的值不相等。：

```
LIST{"blue", "green", "yellow"}
LIST{"yellow", "blue", "green"}
```

上面的表达式不相等，原因是值顺序不同。要使其相同，第二条语句必须为：

```
LIST{"blue", "green", "yellow"}
```

3.3.23 LVARCHAR(m) 数据类型

使用 LVARCHAR 数据类型来创建用于存储可变长度字符串（其上限（*m*）最多可为 32,739 个字节）的列。

LVARCHAR 数据类型作为内置不透明数据类型实现。您可以在相同或不同 GBase 8s 实例的数据库中使用分布式查询，以访问远程表中的 LVARCHAR 列。

缺省情况下，数据库服务器将加引号的字符串解释为 LVARCHAR 类型。它还将 LVARCHAR 用于不透明数据类型的输入和输出强制转型。

LVARCHAR 数据类型以字符串（外部）格式存储不透明数据类型。每个不透明类型都具有输入支持函数和强制转型，从而将其从 LVARCHAR 转换为数据库服务器可以处理的形式。每个不透明类型还具有输出支持函数和强制转型，从而将不透明类型从其内部表示法转换为 LVARCHAR。

要点： 在将 LVARCHAR 声明为数据库表中的列的数据类型（没有大小规范）时，缺省最大大小是 2 KB（2048 个字节），但可以指定最多 32,739 个字节的显式最大长度。当在对不透明数据类型的 I/O 操作中使用 LVARCHAR 时，最大大小仅受操作系统限制。

LVARCHAR 数据类型声明中的大小规范可受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 功能影响。

有关 LVARCHAR 的更多信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

3.3.24 MONEY(p,s) 数据类型

MONEY 数据类型存储货币金额。

与 DECIMAL(*p*,*s*) 数据类型类似，MONEY 可以存储最多 32 个有效数字的定点数，其中 *p* 是有效数字总数（精度），而 *s* 是小数点右边的位数（小数位）。

与 DECIMAL 数据类型不同，总是将 MONEY 数据类型视为定点小数处理。数据库服务器将数据类型 MONEY(*p*) 定义为 DECIMAL(*p*,2)。如果未指定精度和小数位，那么数据库服务器将 MONEY 列定义为 DECIMAL(16,2)。

可以使用以下公式（下舍入为整数字节）来计算 MONEY 数据类型的字节存储器：

如果*小数位*为奇数：N = (*精度* + 4) / 2

如果*小数位*为偶数：N = (*精度* + 3) / 2

例如，精度为 16 并且小数位为 2 的 MONEY 数据类型 (MONEY(16,2)) 需要 10（或者 (16 + 3)/2）个字节的存储空间。

在缺省语言环境中，客户机应用程序使用下列货币表示法来确定 MONEY 列中值的格式：

货币符号：值前面的美元符号（ **$** ）

千位分隔符：逗号（ , ），它在值的整数部分中每三位分隔一次

小数点：句点（ .)位于值的整数部分与小数部分之间

要更改 MONEY 值的格式，请更改 **DBMONEY** 环境变量。有关有效的 **DBMONEY** 设置，请参阅 DBMONEY 环境变量。

数据库服务器用于小数位的缺省值与语言环境相关。缺省语言环境指定两位缺省小数位。对于非缺省语言环境，如果声明中省略了小数位，那么数据库服务器使用特定于语言

环境的小数位来创建 MONEY 值。

客户机应用程序使用的货币表示法与语言环境相关。如果指定非缺省语言环境，那么客户机将特定于文化的格式用于 MONEY 值，根据语言环境文件所指定的内容，这些 MONEY 值的前导（或结尾）货币符号、千位分隔符和小数分隔符可能与缺省的美国英语格式不同。有关语言环境依赖性的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

3.3.25 MULTISET(e) 数据类型

MULTISET 数据类型是存储无序集的集合类型，它能包含重复的元素值。

MULTISET 中的元素没有顺序位置。也就是说，在 MULTISET 中不存在第一元素、第二元素或第三元素的概念。

MULTISET 中的所有元素都具有相同的元素类型。要指定元素类型，使用以下语法：

MULTISET(*element_type* NOT NULL)

集合的 *element_type* 可以是下列任何类型：

任何内置类型（SERIAL、SERIAL8、BIGSERIAL、BYTE 和 TEXT 除外）

未命名或命名 ROW 类型

另一个集合类型或不透明类型

除非另外指示，否则可以在使用任何其他数据类型的地方使用 MULTISET。例如：

在用来搜索匹配的 MULTISET 值的 SELECT 语句 WHERE 子句中的 IN 谓词后面

作为用来确定 LIST 列中的元素数目的 CARDINALITY 或 **mi_collection_card( )** 函数的自变量

*不能*将 MULTISET 值用作聚集函数（例如，AVG、MAX、MIN 或 SUM）的自变量。

必须对 MULTISET 元素指定 NOT NULL 约束。其他约束对 MULTISET 列无效。有关 MULTISET 集合类型的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

如果两个 multiset 数据值具有相同的元素，那么它们相等，即使各元素在集中的位置不同。下列两个示例都是 multiset 值但不相等：

MULTISET {"blue", "green", "yellow"}

MULTISET {"blue", "green", "yellow", "blue"}

下列 multiset 值是相等的：

MULTISET {"blue", "green", "blue", "yellow"}

MULTISET {"blue", "green", "yellow", "blue"}

可以将相同表的不超过 97 列声明为 MULTISET 数据类型。（同一限制适用于 SET 和 LIST 集合类型。）

命名 ROW

请参阅 ROW 数据类型，已命名。

3.3.26 NCHAR(n) 数据类型

NCHAR 数据类型存储固定长度字符数据。该数据可以是单字节或多字节字母、数字和受数据库语言环境的代码集支持的其他符号组成的字符串。

CHAR 与 NCHAR 数据类型之间的主要差别在于整理顺序。

CHAR 数据类型的整理顺序遵循代码集顺序，但 NCHAR 数据类型的整理顺序可以是本地化的顺序，前提是 DB_LOCALE（或 SET COLLATION）指定定义了本地化整理顺序的语言环境。

NCHAR 数据类型声明中的大小规范可能受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 配置参数影响。

在使用 NLSCASE INSENSITIVE 属性创建的数据库中，对 NCHAR 字符串的操作会忽略字母大小写，从而在对数据值排序时不管字母大小写。例如，在查询返回的整理列表中，NCHAR 字符串 "IDS" 可能位于 "IdS" 或 "iDs" 之前或之后，具体取决于检索这些数据字符串的顺序，因为以下所有 NCHAR 字符串会被视为重复值：

"ids" "IDS" "idS" "IDs" "IdS" "iDs" "iDS" "Ids"

3.3.27 NUMERIC(p,s) 数据类型

NUMERIC 数据类型与定点 DECIMAL 同义。

3.3.28 NVARCHAR(m,r) 数据类型

NVARCHAR 数据类型存储可变长度字符串。字符串可以包括数字、符号以及单字节和（在某些语言环境中）多字节字符。

VARCHAR 与 NVARCHAR 数据类型之间的主要差别在于整理顺序。VARCHAR 数据的整理遵循代码集顺序，但如果 DB_LOCALE（或 SET COLLATION）指定定义了本地化整理顺序的语言环境，那么 NVARCHAR 整理可以是特定于语言环境的。（整理 VARCHAR 值一节描述了例外情况。）

声明为 NVARCHAR 的列，无圆括号或参数，具有一个字节的最大大小并且保留的大小置零。

NVARCHAR 数据类型声明中的第一个参数可能受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 配置参数影响。

在使用 NLSCASE INSENSITIVE 属性创建的数据库中，对 NVARCHAR 字符串的操作会忽略字母大小写，从而在对数据值排序时不管字母大小写。例如，在查询返回的整理列表中，NVARCHAR 字符串 "GBASE" 可能位于 "GbasE" 或 "gBase" 之前或之后，具体取决于检索这些数据字符串的顺序，因为以下所有 NVARCHAR 字符串会被视为重复值：

"gbase" "GBASE" "gbasE" "GBase" "GbasE" "Gbase" "GBAse" 等等

3.3.29 OPAQUE 数据类型

OPAQUE 类型是必须为其向数据库服务器提供信息的数据类型。

您必须提供此信息：

如何在磁盘上存储数据值的数据结构

确定如何在磁盘存储格式与用于数据输入和显示的用户格式之间转换的支持函数

确定如何为此数据类型构建、使用和处理索引的辅助访问方法

使用数据类型的用户函数

用于在数据库中注册 OPAQUE 类型的系统目录条目

OPAQUE 类型的内部结构对于数据库服务器不可视，只能通过用户定义的例程访问。OPAQUE 类型的定义存储在 **sysxtdtypes** 系统目录表中。以下 SQL 语句会维护数据库中 OPAQUE 类型的定义：

CREATE OPAQUE TYPE 语句在数据库中注册了一个新的 OPAQUE 类型。

DROP TYPE 语句从数据库中除去先前定义的 OPAQUE 类型。

有关上面提到的 SQL 语句的更多信息，请参阅 *GBase 8s SQL 指南：语法*。有关如何创建 OPAQUE 类型和 OPAQUE 类型示例的信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

3.3.30 REAL 数据类型

REAL 数据类型与 SMALLFLOAT 同义。

3.3.31 ROW 数据类型，已命名

命名的 ROW 数据类型必须使用名称进行声明。此 SQL 标识在同一个数据库内的数据类型名称中必须唯一。（未命名 ROW 类型是包含字段但没有任何用户定义的名称的 ROW 类型。）只有命名 ROW 类型支持数据类型继承。有关更多信息，请参阅 ROW 数据类型。

定义命名 ROW 类型

必须通过使用 SQL 的 CREATE ROW TYPE 语句在数据库中声明并注册新的命名 ROW 类型。命名 ROW 类型的定义存储在 **sysxtdtypes** 系统目录表中。

ROW 数据类型的字段可以是任何内置数据类型或 UDT，但 ROW 类型的 TEXT 或 BYTE 字段只在类型表中有效。如果要在 CREATE TABLE 或 ALTER TABLE 语句中将 ROW 类型指定给列，那么其元素不能是 TEXT 或 BYTE 数据类型。

一般来说，ROW 类型字段的数据类型可以是以下任何类型：

内置类型（TEXT 或 BYTE 数据类型除外）

集合类型（LIST、MULTISET 或 SET）

单值类型

其他命名或未命名 ROW 类型

不透明类型

以下 SQL 语句会维护命名 ROW 数据类型的定义：

CREATE ROW TYPE 语句将命名 ROW 类型添加至数据库。

DROP ROW TYPE 语句从数据库中除去先前定义的命名 ROW 类型。

有关这些 SQL 语法语句的详细信息，请参阅《*GBase 8s SQL 指南：语法*》。有关如何创建和使用命名 ROW 类型的示例，请参阅《*GBase 8s 数据库设计和实现指南*》。

等价和命名 ROW 类型

即使两个命名 ROW 类型具有完全相同的结构，因为其具有不同的名称，所以它们也不能相等。例如，以下命名 ROW 类型具有相同的结构（字段数相同并且行中字段的数据类型的顺序相同）但不相等：

name_t (lname CHAR(15), initial CHAR(1), fname CHAR(15))

emp_t (lname CHAR(15), initial CHAR(1), fname CHAR(15))

如果两个操作数是不同的命名 ROW 类型，那么布尔等式条件（例如 name_t = emp_t）的求值结果始终是 FALSE。

命名 ROW 类型和继承

命名 ROW 类型可以是类型继承层次结构的一部分。一种命名 ROW 类型可以是另一种命名 ROW 类型的父代（或超类型）。层次结构中的子类型继承其超类型的所有属性。《*GBase 8s SQL 指南：语法*》和《*GBase 8s 数据库设计和实现指南*》中的 CREATE ROW TYPE 语句中说明了类型继承。

类型表

作为继承层次结构一部分的表必须是类型表。类型表是已指定命名 ROW 类型的表。有关用来创建类型表的语法，请参阅《*GBase 8s SQL 指南：语法*》中的 CREATE TABLE 语句。在该节中还说明了表继承及其与类型继承的关系。有关如何创建和使用类型表的信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

3.3.32 ROW 数据类型，未命名

未命名 ROW 类型包含字段，但没有任何用户声明的名称。未命名 ROW 类型通过其结构来定义。

如果两个未命名 ROW 类型具有相同的结构（意味着字段的数据类型的排序列表），那么它们相同。如果两种未命名 ROW 类型具有相同数目的字段，并且如果一种 ROW 类型中每个字段的数据类型顺序与另一种 ROW 类型中对应字段的数据类型顺序相匹配，那么这两个未命名 ROW 数据类型等同。

例如，下列未命名 ROW 类型等同：

ROW (lname char(15), initial char(1) fname char(15))

ROW (dept char(15), rating char(1) name char(15))

下列 ROW 类型具有相同的字段数和相同的数据类型，但它们不相等，因为其字段顺序不同：

ROW (x integer, y varchar(20), z real)

ROW (x integer, z real, y varchar(20))

未命名 ROW 类型字段可以是下列任何数据类型：

内置类型

集合类型

单值类型

其他 ROW 类型

不透明类型

不能在类型表或类型继承层次结构中使用未命名 ROW 类型。有关未命名 ROW 类型的更多信息，请参阅《*GBase 8s SQL 指南：语法*》和《*GBase 8s 数据库设计和实现指南*》。

创建未命名 ROW 类型

您可以使用数种方法来创建未命名 ROW 类型：

可以使用 ROW 关键字来声明未命名 ROW 类型。ROW 中的每个字段可具有不同的字段类型。要指定字段类型，使用以下语法：

ROW(*field_name field_type*, ...)

field_name 必须遵从 SQL 标识的规则。（请参阅 *GBase 8s SQL 指南：语法* 中的 Identifier 一节。）

要生成未命名 ROW 类型，请使用 ROW 关键字作为具有一系列值的构造函数。使用具有指定值的缺省数据类型来创建对应的未命名 ROW 类型。

例如：以下声明：

```
ROW(1, 'abc', 5.30)
```

定义下面的未命名 ROW 数据类型：

```
ROW (x INTEGER, y VARCHAR, z DECIMAL)
```

可通过隐式或显式从命名 ROW 类型或从另一种未命名 ROW 类型的强制转型来创建未命名 ROW 类型。

任何表（对命名 ROW 类型定义的表除外）的行都是未命名 ROW 类型。

同一个表中不超过 195 列可以是未命名 ROW 类型。

将值插入到未命名 ROW 类型列

当为未命名 ROW 类型指定字段值时，在构造函数后面列出字段值并将这些值列在圆括号中。例如，假设您具有未命名 ROW 类型的列。以下 INSERT 语句将一组字段值添加至此

行列：

```
INSERT INTO table1 VALUES (ROW(4, 'abc'))
```

可在 SELECT 语句 WHERE 子句的 IN 谓词中指定行列以搜索匹配行值。有关更多信息，请参阅 *GBase 8s SQL 指南：语法* 中的 Condition 一节。

3.3.33 SERIAL(n) 数据类型

SERIAL 数据类型存储 INT 数据类型的顺序整数（在插入新行时由数据库服务器自动指定）。

缺省顺序起始号为 1，但可以在创建或改变表时指定初始值 *n*。

您可以指定正数或负数作为起始号。

如果您指定零 (0) 作为起始号，那么使用的值是 SERIAL 列中已存在的最大正值 + 1。

SERIAL 的最大值是 2,147,483,647。如果指定大于 2,147,483,647 的数，那么会接收到语法错误。如果需要更大的范围，那么使用 SERIAL8 或 BIGSERIAL 数据类型，而不是 SERIAL。

一个表只能有一个 SERIAL 列，但是它可以具有一个 SERIAL 列和一个 SERIAL8 列或 BIGSERIAL 列。

列中的 SERIAL 值并不是自动唯一。必须对此列应用唯一索引或主键约束以防止重复的 SERIAL 数。如果您在 DB-Access 中使用 Interactive Schema Editor 来定义该表，那么唯一索引将自动应用于 SERIAL 列。

SERIAL 数可能不连续，原因是存在并发的用户、回滚和其他因素。

如果 *column* 是 SERIAL 数据类型，那么用于间接输入的 SPL 的 DEFINE *variable* LIKE *column* 语法声明 INTEGER 数据类型的变量。

在指定了数字之后，就不能进行更改。您可以将值插入 SERIAL 列中（使用 INSERT 语句），或如果新值与列中的任何现有值都不重复，也可以重置 serial 列（使用 ALTER TABLE 语句）。要将值插入到 SERIAL 列中，数据库服务器会使前面的值增加 1（如果复位值较大的话，那么将复位值增加 1）并将结果指定为输入值。但是，如果 ALTER TABLE 已经将 SERIAL 列的下一个值复位为小于该列中已存在的值，那么下一个值将遵循以下公式：

（SERIAL 列中的最大现有值） + 1

例如，当最大现有值为 128 时，如果将 **customer.customer_num** 的序号值复位为 50，那么下一个指定数将为 129。有关 SERIAL 数据条目的更多详细信息，请参阅 *GBase 8s SQL 指南：语法*。

SERIAL 列可以存储唯一代码，例如，订单、发票或客户编号。SERIAL 数据值需要 4 个

字节的存储空间，并且具有与 INTEGER 数据类型相同的精度。有关将唯一整数指定给数据库表中每一行的其他方法的详细信息，请参阅 *GBase 8s SQL 指南：语法* 中的 CREATE SEQUENCE 语句。

3.3.34 SERIAL8(n) 数据类型

SERIAL8 数据类型存储 INT8 数据类型的顺序整数（在插入新行时由数据库服务器自动指定）。

SERIAL8 数据类型的行为与 SERIAL 数据类型相似，但范围更大。有关如何将值插入 SERIAL8 列的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

SERIAL8 数据列通常用于存储大型的唯一数字代码，例如，订单、发票或客户编号。SERIAL8 数据值具有与 INT8 值相同的精度和存储器需求（INT8页）。

缺省顺序起始号为 1，但可以在创建或改变表时指定初始值 *n*。

您可以指定正数或负数作为起始号。

如果您指定零 (0) 作为起始号，那么使用的值是 SERIAL8 列中已存在的最大正值 + 1。

一个表只能有一个 SERIAL 列，但是它可以具有一个 SERIAL 列和一个 SERIAL8 列或 BIGSERIAL 列。

列中的 SERIAL8 值并不是自动唯一。必须对此列应用唯一索引或主键约束以防止重复的 SERIAL 数。如果您在 DB-Access 中使用交互式模式编辑器来定义该表，那么唯一索引将自动应用于 SERIAL8 列。

SERIAL8 数可能不连续，原因是存在并发的用户、回滚和其他因素。

如果 *column* 是 SERIAL8 数据类型，那么用于间接输入的 SPL 的 DEFINE *variable* LIKE *column* 语法声明 INTEGER 数据类型的变量。

有关更多信息，请参阅指定 SERIAL8 的开始值。有关将 SERIAL8 数据类型与 INT8 或 BIGINT 数据类型配合使用的信息，请参阅将 SERIAL8 和 BIGSERIAL 与 INT8 或 BIGINT 配合使用。

指定 SERIAL8 的开始值

缺省顺序起始号为 1，但可以在创建或改变表时指定初始值 *n*。要使表的 SERIAL8 列从值 1 开始，请在将行插入该表中时为 SERIAL8 列提供值 0。数据库服务器将把值 1 指定给该表第一行的 SERIAL8 列。可以指定的最大 SERIAL8 值是 $2^{63}-1$ (9,223,372,036,854,775,807)。如果指定大于这个的值，那么会接收到语法错误。当数据库服务器生成最大的 SERIAL8 值时，它就会回绕并从 1 开始生成值。

在指定了非零 SERIAL8 数之后，就不能更改它。但是，可以将值插入 SERIAL8 列中（使用 INSERT 语句），或如果 SERIAL8 值 *n* 与列中的任何现有值都不重复，也可以重

置该值（使用 ALTER TABLE 语句）。

当将数字插入 SERIAL8 列中或复位 SERIAL8 列的下一个值时，数据库服务器就会按输入数字的顺序指定下一个数。但是，如果将 SERIAL8 列的下一个值复位为比该列中已存在的值要小的值，那么使用以下公式来计算下一个值：

SERIAL8 *列中的最大现有值* + 1

例如：当最高指定的客户号为 128 时，如果将 **customer** 表中 **customer_num** 列的 SERIAL8 值重新设置为 50，那么指定的下一个客户号为 129。

3.3.35 SET(e) 数据类型

SET 数据类型是存储唯一元素的无序集合类型

如 *GBase 8s SQL 指南：语法* 中所述，重复的元素值无效。（有关支持重复值的集合类型，请参阅 MULTISET(e) 数据类型中对 MULTISET 的描述。）

可以将相同表的不超过 97 列声明为 SET 数据类型。（同一限制还适用于 MULTISET 和 LIST 集合类型。）

SET 中的元素没有按顺序的位置。也就是说，在 SET 中不存在第一、第二或第三元素的构造。（有关具有元素的顺序位置的集合类型，请参阅 LIST(e) 数据类型。）SET 中的所有元素都具有相同的元素类型。要指定元素类型，使用以下语法：

SET(*element_type* NOT NULL)

集合的 *element_type* 可以是下列任何类型：

内置类型（SERIAL、SERIAL8、BIGSERIAL、BYTE 和 TEXT 除外）

命名或未命名 ROW 类型

其他集合类型

不透明类型

必须对 SET 元素指定 NOT NULL 约束。对于 SET 列，没有任何其他约束有效。有关 SET 集合类型语法的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

除非另有指示，否则可以在使用任何其他数据类型的地方使用 SET。例如：

在用来搜索匹配的 SET 值的 SELECT 语句 WHERE 子句中的 IN 谓词后面

作为用来确定 SET 列中的元素数目的 CARDINALITY 或 **mi_collection_card( )** 函数的自变量

不能将 SET 值用作聚集函数（例如，AVG、MAX、MIN 或 SUM）的自变量。有关更多信息，请参阅 *GBase 8s SQL 指南：语法* 中的 Condition 和 Expression 两节。

下面的示例声明两个集合。第一条语句声明一组整数，第二条语句声明一组字符元素。

```
SET(INTEGER NOT NULL)
```

SET(CHAR(20) NOT NULL)

下面的示例从值列表中构造相同的集合：

SET{1, 5, 13}

SET{"Oakland", "Menlo Park", "Portland", "Lenexa"}

在以下示例中，SET 构造函数是 CREATE TABLE 语句的一部分：

```
CREATE TABLE tab
(
      c CHAR(5),
      s SET(INTEGER NOT NULL)
);
```

下列 set 值相等：

SET{"blue", "green", "yellow"}

SET{"yellow", "blue", "green"}

3.3.36 SMALLFLOAT 数据类型

SMALLFLOAT 数据类型会存储具有大约九个有效数字的单精度浮点数。

SMALLFLOAT 对应于 C 中的 float 数据类型。SMALLFLOAT 数据类型的值范围与计算机上 C float 数据类型的值范围相同。

SMALLFLOAT 数据类型列通常存储只能近似计算的科学数字。由于浮点数只保留它们的大部分有效数位，所以在此类型的列中输入的数和数据库显示的数可能稍有不同，这取决于计算机内部存储浮点数的方式。

例如，您可能会在 SMALLFLOAT 字段中输入值 1.1000001，在处理了 SQL 语句之后，应用程序可能会将此值显示为 1.1。当值具有的位数比浮点数可存储的多时，就会出现这种差别。在这种情况下，用值的近似形式存储值，将其余有效数位视为零。

SMALLFLOAT 数据类型通常需要 4 个字节的存储空间。将 SMALLFLOAT 值转换为 DECIMAL 值会产生 9 位精度。

3.3.37 SMALLINT 数据类型

SMALLINT 数据类型存储范围在 －32,767 到 32,767 的小整数。最大负数 －32,768 是保留值，不能使用。

SMALLINT 值作为有符号二进制整数存储。

整数列通常存储计算和数量等等。由于 SMALLINT 数据类型的每个值只需要两个字节，所以可以有效地执行算术运算。但是，与其他内置数字数据类型相比，SMALLINT 只存储有限范围的值。如果数字在最小和最大 SMALLINT 值的范围之外，那么数据库服务器不会存储该数据值，而是会发出错误消息。

3.3.38 TEXT 数据类型

TEXT 数据类型存储所有类型的文本数据。它可以同时包含语言环境支持的单字节字符和多字节字符。术语*简单大对象*指的是 TEXT 和 BYTE 数据类型。

TEXT 列具有 2^{31} 个字节(两千兆字节)的理论限制和可用磁盘存储器确定的实际限制。可以将相同表的不超过 195 列声明为 TEXT 数据类型。同一限制还适用于 BYTE 数据类型。

可以存储、检索、更新或删除 TEXT 列中的值。

仅当正在使用 IS NULL 或 IS NOT NULL 运算符测试 NULL 值时，才能在布尔表达式中使用 TEXT 操作数。

您可以使用以下可装入行或更新字段的方法来插入 TEXT 数据：

使用 **dbload** 或 **onload** 实用程序

使用 LOAD 语句 (DB-Access)

从 TEXT 主变量 (ESQL)

存在内置强制转型将 TEXT 对象转换为 CLOB 对象。有关更多信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

TEXT 数据类型的字符串使用代码集顺序整理。有关整理顺序的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

在 TEXT 列中选择数据

当您选择 TEXT 列时，可以接收该列的全部或一部分。要检索整个列，使用选择列的常规语法。还可通过使用下标来选择 TEXT 列的任何部分，如以下示例所示：

```
SELECT cat_descr [1,75] FROM catalog WHERE catalog_num = 10001
```

SELECT 语句读取与 **catalog_num** 值 10001 相关联的 **cat_descr** 列的前 75 个字节。

将数据装入到 TEXT 列

您可以使用 LOAD 语句将数据插入表中。例如，inp.txt 文件包含以下信息：

```
|aaaaa|
2|bbbbb|
3|cccccc| 要将此数据装入 blobtab 表中，请使用以下语句：
LOAD FROM inp.txt INSERT INTO blobtab;
```

限制

不能在算术或字符串表达式中使用 TEXT 操作数，也不能使用 UPDATE 语句的 SET 子句将字面值指定给 TEXT 列。

也不能用以下任何方法使用 TEXT 值：

使用聚集函数

使用 IN 子句

使用 MATCHES 或 LIKE 子句

使用 GROUP BY 子句

使用 ORDER BY 子句

不能使用带引号的文本字符串、数字或任何其他实际值来插入或更新 TEXT 列。

要点：如果您尝试从子查询中返回 TEXT 列，那么会产生错误，即使没有任何 TEXT 列用于比较条件或与 IN 谓词一起使用也是这样。

TEXT 值中的不可打印字符

TEXT 列通常存储文档和程序源文件等等。在缺省“美国英语”语言环境中，类型为 TEXT 的数据对象可以包含可打印 ASCII 字符与下列控制字符的组合：

制表符 (CTRL-I)

换行符 (CTRL-J)

换页符 (CTRL-L)

同时可以在 text 列中插入可打印字符和不可打印字符。GBase 8s 产品不会对插入到具有 TEXT 数据类型的列中的任何数据值进行检查。（但是，应用程序可能无法显示包括不可打印字符的 TEXT 值。）有关输入和显示不可打印字符的详细信息，请参阅 CHAR 类型的不可打印字符。

3.3.39 TIMESTAMP 数据类型

TIMESTAMP 与 DATETIME 同义。唯一不同的是，其中指定小数秒精度的 FRACTION 时间单位值是 1 到 6 范围内的数字，缺省值为 0。

3.3.40 VARCHAR(m,r) 数据类型

VARCHAR 数据类型存储包含单字节和（如果语言环境支持）多字节字符的可变长度字符串，其中 *m* 是列的最大大小（以字节计），*r* 是为该列保留的最小字节数。

声明为 VARCHAR 的列（无圆括号或参数）具有一个字节的最大大小并且保留的大小置零。

VARCHAR 数据类型是字符可变数据类型的 GBase 8s 实现。可变长度字符串的 ANSI 标准数据类型是 CHARACTER VARYING。

VARCHAR 列的最大大小 (*m*) 参数的大小范围可以从 1 到 32765 个字节。

指定最小保留空间 (*r*) 参数是可选的。此值可以在从 0 到 32765 个字节的范围内，但必须小于 VARCHAR 列的最大大小 (*m*)。如果您未指定任何最小值，缺省值是 0。当起初打算在列中插入具有短字符串或 NULL 字符串的行，但后来想要用较长的值更新此数据时，应指定此参数。

当为 VARCHAR 列设置索引时，其最大长度不能超过 *Maxlen* 值：

Maxlen =((PAGESIZE-93)/5-1)

其中 PAGESIZE 是数据库服务器的页面大小。可以使用 onstat －d 命令查看 dbspace

的 PAGESIZE 值。

同一表上不超过 151 列可以是 VARCHAR 数据类型。

ROWTYPE 类型中包含 VARCHAR 列时其最大长度不能超过 32747 个字节。

将表中的字段修改为 VARCHAR 列或 Alter Type Varchar 列时，此 varchar 列长度需要小于页面的可用空间。具体最大长度如下表所示：

数据库页面大小	可修改的最大长度
2K	1958
4K	4006
8K	8102
16K	16294

在基于 VARCHAR 列（或 NVARCHAR 列）的索引中，每个索引键都具有基于实际输入的数据值的长度，而不是声明的列的最大大小。

当您在 VARCHAR 列中存储字符串时，只会存储实际数据字符。数据库服务器不会除去任何用户输入的结尾空格的 VARCHAR 字符串，也不会将 VARCHAR 值填充至列的声明长度。如果指定保留空间 (*r*)，但某些数据字符串比 *r* 字节要短，那么为行保留的某些空间变得无用。

用比较 CHAR 值的相同方式将 VARCHAR 值与其他 VARCHAR 值（和其他字符串数据类型）进行比较。用空格在右边填充较短的值直到这些值具有相同的长度；然后对其比较全长。

VARCHAR 类型的不可打印字符

用处理 CHAR 值中的不可打印字符的相同方法输入、显示和处理不可打印 VARCHAR 字符。有关详细信息，请参阅 CHAR 类型的不可打印字符。

存储 VARCHAR 列中的数字值

当在 VARCHAR 列中插入数字值时，不会用结尾空格将存储的值填充至列的最大长度。数字 VARCHAR 值中的位数是存储该值所需要的字符数。例如：在下一个示例中，存储在表 **mytab** 中的值为 1。

```
create table mytab (col1 varchar(10));
insert into mytab values (1);
```

技巧： VARCHAR 将 C 语言的*null*（二进制 0）和字符串终止符看作不可打印字符的终止字符。

在某些东亚语言环境中，如果数据库语言环境支持多字节代码集，那么 VARCHAR 数据类型能够存储多字节字符。如果存储多字节字符，务必要计算所需的字节数。有关更多信息，请参阅《*GBase 8s GLS 用户指南*》。

VARCHAR 类型的多字节字符

VARCHAR 数据类型声明中的第一个参数可受字符类型声明中的逻辑字符语义一节中描述的 SQL_LOGICAL_CHAR 功能影响。

整理 VARCHAR 值

NVARCHAR 与 VARCHAR 数据类型之间的主要差别（就如 CHAR 与 NCHAR 之间的差别）是整理顺序的差别。通常，VARCHAR（与 CHAR 和 LVARCHAR 一样）值是按其在代码集中存在的字符顺序进行整理的。

例外情况是 MATCHES 运算符，当 **DB_LOCALE**（或 SET COLLATION）指定了本地化整理顺序时，如果使用方括号（[]）符号来定义范围，那么它会将本地化整理应用于 NVARCHAR 和 VARCHAR 值（并应用于 CHAR、LVARCHAR 和 NCHAR 值）。有关更多信息，请参阅《*GBase 8s GLS 用户指南*》。

3.4 内置数据类型

GBase 8s 支持以下内置数据类型。

分类	数据类型
字符	CHAR、CHARACTER VARYING、LVARCHAR、NCHAR、 NVARCHAR、VARCHAR、IDSSECURITYLABEL
大对象	简单大对象类型：BYTE 和 TEXT 智能大对象类型：BLOB 和 CLOB
逻辑	BOOLEAN
数字	BIGINT、BIGSERIAL、DECIMAL、FLOAT、 INT8、INTEGER、MONEY、SERIAL、SERIAL8、 SMALLFLOAT 和 SMALLINT
时间	DATE、DATETIME 和 INTERVAL
游标	SYS_RECURSOR

有关对字符、数字和其他数据类型的描述，请参阅数据类型的描述中适当的条目。页码索引在表 1中的字母顺序列表中。

3.4.1 字符数据类型

字符数据类型存储字符串值。

1. 内置字符类型

表 1. 内置字符数据类型的属性

	大小（以字节计）	缺省值	保留	整理	长度
CHAR(n)	1 至 32,767	1 个字节	无	代码集	固定
NCHAR(n)	1 至 32,767	1 个字节	无	本地化	固定

	大小（以字节计）	缺省值	保留	整理	长度
VARCHAR(m, r)	1 到 32,765	对于 r 为 0	0 至 32,765 个字节	代码集	可变
NVARCHAR(m, r)	1 到 32,765	对于 r 为 0	0 至 32,765 个字节	本地化	可变
LVARCHAR(m)	1 至 32,739	2048 个字节	无	代码集	可变

数据库服务器还可以使用 LVARCHAR 表示不透明数据类型的外部格式。在数据库服务器的 I/O 操作中，除文件大小限制、操作系统或硬件资源的限制之外，LVARCHAR 数据值的大小没有上限。

2.字符类型声明中的逻辑字符语义

GBase 8s 支持配置参数 SQL_LOGICAL_CHAR，其设置可指示 SQL 解析器将 CREATE TABLE 或 ALTER TABLE 语句数据类型声明中字符列的最大大小解释为逻辑字符，而不是以字节为单位。

创建数据库之后，数据库服务器的当前 SQL_LOGICAL_CHAR 设置会记录在系统目录的 systables 表中。设置为 OFF 或 1 时，该功能对随后在数据库中创建或变更的表没有影响。

然而，在 SQL_LOGICAL_CHAR 设置为 ON 或为 2、3 或 4 之间的数字的数据库中，针对以下字符类型，SQL 解析器会将显式和隐式大小声明解释为 SPL 变量声明和数据库表中的列声明中的逻辑字符：

CHAR 和 CHARACTER

CHARACTER VARYING 和 VARCHAR

LVARCHAR

NCHAR

NVARCHAR

以上所列出数据类型的 DISTINCT 类型

那些 DISTINCT 类型的 DISTINCT 类型

以上所列出类型的 ROW 数据类型字段。

以上所列出类型的 LIST、MULTISET 和 SET 元素。

此功能对之前表中列出的字符类型的最大存储大小限制没有影响。然而，对于使用多字节语言环境的数据库，可在将字符串插入到字符列或指定给字符变量时，降低数据截断的风险。

例如，如果数据库的 SQL_LOGICAL_CHAR 设置是 4，那么 VARCHAR(10, 5) 规范会解释为要求最大 40 个字节的存储，保留这些字节中的 5 个，在标准 SQL 表示法中创建 VARCHAR(40, 5) 数据类型，而不是声明中指定的规范。

由于多字节字符的最小大小是 1 个字节，因此 VARCHAR 和 NVARCHAR 数据类型的保留大小参数不受 SQL_LOGICAL_CHAR 设置影响。在此示例中，5 多字节字符的最小大小是 5 个字节，大小保持不变。

请参阅《*GBase 8s 管理员参考*》中对 SQL_LOGICAL_CHAR 配置参数的描述，以获取有关 SQL_LOGICAL_CHAR 设置对 **DB_LOCALE** 指定多字节语言环境的数据库的影响的更多信息。有关多字节语言环境和逻辑字符的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

3.IDSSECURITYLABEL

GBase 8s 还支持将 IDSSECURITYLABEL 数据类型用于实现基于标号的访问控制 (LBAC) 的系统。此内置数据类型可正式归类为字符类型，因为其定义为 VARCHAR(128) 的 DISTINCT 数据类型，但只有拥有 **DBSECADM** 角色的用户可在 DDL 操作中声明此数据类型。它支持 LBAC 安全性功能，而不是作为通用字符类型发挥功能。

4.数据类型提升

对于 GBase 8s 的一些字符串操纵操作，上面列出的五个内置字符数据类型支持数据类型提升，以降低由于返回的字符串太大而无法存储在 NVARCHAR 或 VARCHAR 列或程序变量中，从而导致字符串操作失败的风险。请参阅 *GBase 8s SQL 指南：语法* 中的“从 CONCAT 和 String 操纵函数返回的类型”主题，以获取在 GBase 8s 字符类型之间进行数据类型提升的详细信息。

5.本地语言支持

有时，NCHAR 和 NVARCHAR 类型称为本地语言支持数据类型，因为其支持本地化的整理。由于 VARCHAR 或 NVARCHAR 类型的列没有缺省大小，因此您必须在声明中指定大小（不大于 32765 ）。

6.NLSCASE lNSENSITIVE 数据库

在使用 NLSCASE INSENSITIVE 关键字选项创建的数据库中，对 NCHAR 或 NVARCHAR 类型数据字符串的操作不区分相同字母的大小写变体。行使用装入字符时所用字母大小写存储在 NCHAR 或 NVARCHAR 列中，但由相同字母以相同顺序组成的数据字符串将求值为重复，即使某些字母的大小写不同。例如，三个 NCHAR 字符串 "ABC"、"AbC" 和 "abC" 将视为重复。其他内置字符类型（包括 CHAR、LVARCHAR 和 VARCHAR）遵循缺省的区分大小写规则，以便 CHAR 列中的三个相同字符串求值为不同值。

使用 NLSCASE INSENSITIVE 属性的数据库还忽略基本类型为 NCHAR 或 NVARCHAR 的 DISTINCT 数据类型的字母大小写，以及已命名或未命名 ROW 类型中的 NCHAR 或 NVARCHAR 字段、COLLECTION 数据类型的 NCHAR 或 NVARCHAR 元素，包括 LIST、SET 或

MULTISET。

在不区分 NCHAR 或 NVARCHAR 值的字母大小写的数据库中，字符串操纵操作可能会生成意外的结果（如果它们将 CHAR、LVARCHAR 或 VARCHAR 操作数或参数隐式强制转型为 NCHAR 或 NVARCHAR 数据类型）。在一些上下文中，操作可能返回重复字符串，尽管数据库服务器不会将这些字母大小写变体视为原始数据类型的重复值。

3.4.2 大对象数据类型

大对象数据类型已定义表与大对象存储分开大对象是逻辑存储在表列中但物理存储与列无关的数据对象。大对象是独立于表存储的，因为它们通常存储大量的数据。将此数据与表分开可提高性能。

图 1 显示了大对象数据类型。

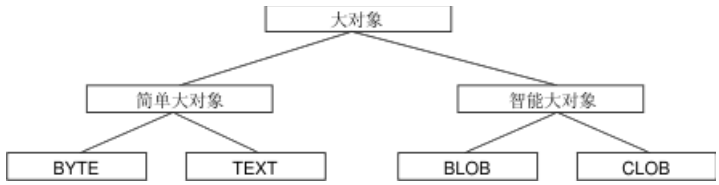


图 1. 大对象数据类型

只有 GBase 8s 支持 BLOB 和 CLOB 数据类型。

有关简单大对象与智能大对象的相对优点和缺点，请参阅《*GBase 8s 数据库设计和实现指南*》。

简单大对象

简单大对象是具有理论大小限制 2^{31} 个字节和磁盘容量确定的实际限制的大对象类别。GBase 8s 支持以下简单大对象数据类型：

BYTE

存储二进制数据。有关此数据类型的更详细的信息，请参阅 BYTE 数据类型页上的描述。

TEXT

存储文本数据。有关此数据类型的更详细的信息，请参阅 TEXT 数据类型页上的描述。

可以将相同表的不超过 195 列声明为 BYTE 或 TEXT 数据类型。与智能大对象不同，简单大对象不支持随机存取数据。在客户机应用程序和数据库服务器之间传输简单大对象时，您必须传输完整的 BYTE 或 TEXT 值。如果数据不适合于内存，那么必须将数据值存储在操作系统文件中，然后从该文件中检索数据值。

Blob 空间已定义块数据库服务器将简单大对象存储在 *Blob 空间*中。*Blob 空间*是包含只存储 BYTE 和 TEXT 数据的一个或多个块的逻辑存储器区域。有关如何定义 Blob 空间的信息，请参阅《*GBase 8s 管理员指南*》。

智能大对象

智能大对象是支持对数据进行随机访问并且通常可恢复的一类大对象。

随机存取功能允许您查找和读取智能大对象，就好像它是操作系统文件一样。

智能大对象对于具有大量存储器需求的不透明数据类型也很有用。（请参阅不透明数据类型中对不透明数据类型的描述。）它们具有 2^{42} 个字节的理论大小限制和磁盘容量确定的实际限制。

GBase 8s 支持下列智能大对象数据类型：

BLOB

存储二进制数据。有关此数据类型的更多信息，请参阅 BLOB 数据类型页上的描述。

CLOB

存储文本数据。有关此数据类型的更多信息，请参阅 CLOB 数据类型。

GBase 8s 将智能大对象存储在*智能大对象空间*中。*智能大对象空间*是包含只存储 BLOB 和 CLOB 数据的一个或多个块的逻辑存储器区域。有关如何定义智能大对象空间的信息，请参阅《*GBase 8s 性能指南*》。

当定义 BLOB 或 CLOB 列时，可以确定下列大对象的特征：

LOG 和 NOLOG：数据库服务器是否应该根据当前数据库日志记录方式记录智能大对象。

KEEP ACCESS TIME 和 NO KEEP ACCESS TIME：数据库服务器是否应该跟踪智能大对象的最后访问时间。

HIGH INTEG 和 MODERATE INTEG：数据库服务器是应该使用智能大对象空间页眉和页脚来检测数据损坏 (HIGH INTEG)，还是仅使用页眉来检测 (MODERATE INTEG)。

使用这些特征可能会影响性能。有关信息，请参阅《*GBase 8s 性能指南*》。

当 SQL 语句访问智能大对象时，数据库服务器不会发送实际的 BLOB 或 CLOB 数据。相反，它建立指向数据的指针并返回此指针。然后，客户机应用程序可以在对智能大对象的打开、读或写操作中使用此指针。

要从客户机应用程序中访问 BLOB 或 CLOB 列，请使用以下某个应用程序编程接口 (API)：

在 GBase 8s ESQL/C 程序中，使用智能大对象 API。(有关更多信息，请参阅 *GBase 8s ESQL/C 程序员手册*。)

在 DataBlade 模块中，使用客户机和服务器 API。(有关更多信息，请参阅 *GBase 8s DataBlade API 程序员指南*。)

有关智能大对象的信息，请参阅 *GBase 8s SQL 指南：语法* 和《*GBase 8s 数据库设计和实现指南*》。

3.4.3 时间数据类型

DATE 、DATETIME 和 TIMESTAMP 数据值表示零维时间点；INTERVAL 数据值表示 1 维时间范围（具有正值或负值）。DATE 精度总是天的整数计数，但各种字段限定符可以定义 DATETIME 和 INTERVAL 精度。您可使用算术和关系表达式中的 DATE、DATETIME、TIMESTAMP 和 INTERVAL 数据。可以使用另一个 DATETIME 值、INTERVAL 值和当前时间（由关键字 CURRENT 指定）或某个时间单位（通过使用关键字 UNITS）来处理 DATETIME 值。

可以在 DATETIME 值有效的大多数上下文中使用 DATE 值，反之亦然。还可以在 DATETIME 值有效的算术运算中使用 INTERVAL 操作数。此外，可以使两个 INTERVAL 值相加，并可以用一个数来乘或除 INTERVAL 值。

INTERVAL 列可以保存表示两个 DATETIME 值之差或两个 INTERVAL 值之差（或之和）的值。在这两种情况下，结果都是时间范围，该时间范围是一个 INTERVAL 值。相反，如果从 DATETIME 值中加上或减去 INTERVAL 值，那么会生成另一个 DATETIME 值，因为结果是特定时间。

表 1列出了可以对 DATE、DATETIME 和 INTERVAL 操作数执行的二进制算术运算以及由算术表达式返回的数据类型。

表 1. 对 DATE、DATETIME、TIMESTAMP 和 INTERVAL 值的算术运算

操作数 1	运算符	操作数 2	结果
DATE	-	DATETIME	INTERVAL
DATETIME	-	DATE	INTERVAL
DATE	+或-	INTERVAL	DATETIME
DATETIME	-	DATETIME	INTERVAL
DATETIME	+或-	INTERVAL	DATETIME
INTERVAL	+	DATETIME	DATETIME
INTERVAL	+或-	INTERVAL	INTERVAL
DATETIME	-	CURRENT	INTERVAL
CURRENT	-	DATETIME	INTERVAL
INTERVAL	+	CURRENT	DATETIME
CURRENT	+或-	INTERVAL	DATETIME
DATETIME	+或-	UNITS	DATETIME
INTERVAL	+或-	UNITS	INTERVAL
INTERVAL	* 或/	NUMBER	INTERVAL

不允许其他组合。不能使两个 DATETIME 值或两个 TIMESTAMP 值相加，原因是此操作不会生成特定时间或时间范围。例如：不能将 12 月 25 日与 1 月 1 日相加，但您可以从一个值中减去另一个以了解两者之间的时间跨度。

与数字常量运算

可以将 DATETIME、TIMESTAMP 值加或减数字常量。该数字常量可以是整数和浮点数。

GBase 8s 数据库将运算日期表达式中的数字常量解释为天数。例如 SYSDATE +1 是明天，SYSDATE +(10/1440)表示从现在起十分钟。

DATETIME 值或 TIMESTAMP 值可以来自下列源：

类型为 DATETIME (或 TIMESTAMP) 的列或程序变量

SYSDATE 关键字

CURRENT 关键字

TO_DATE() 函数

浮点数不能大于11574。

例如,

```
SELECT TO_CHAR(
    TO_DATE('20170101 08:00:00','YYYY-MM-DD HH24:MI:SS')+1.25,
                YYYY-MM-DD HH24:MI:SS')
    FROM DUAL;
```

Result: 2017-01-02 14:00:00

DATETIME 值或 TIMESTAMP 值运算

可以将 DATETIME 值（或 TIMESTAMP 值）互相进行减法运算。

日期可以是任何顺序。DATETIME 值（或 TIMESTAMP 值）可以来自下列源：

类型为 DATETIME (或 TIMESTAMP) 的列或程序变量

SYSDATE 关键字

CURRENT 关键字

TO_DATE() 函数

其运算结果可以是以下两种类型：

- 正的或负的 INTERVAL 值。第一个 DATETIME 值确定结果的精度，它包含与第一个操作数相同的时间单位。
- 正的或负的浮点数（以天为单位）。这种情况只能通过 JDBC 接口返回。

例如，假定当前系统时间为 2017 年 1 月 3 日 14:00:00 ，执行以下语句，用户通过 JDBC 接口得到一个以天为单位的浮点数:

```
SELECT SYSDATE-TO_DATE('20170102 08:00:00','YYYY-MM-DD HH24:MI:SS')
FROM DUAL;
```

Result: 1.25（通过 JDBC 接口返回）

如果第二个 DATETIME 值（或 TIMESTAMP 值）具有的字段比第一个值少，那么会自动增加第二个操作数的精度以与第一个值相匹配。

在下面的示例中，从 DATETIME YEAR TO MINUTE 值中减去 DATETIME YEAR TO HOUR 值，从而得到正的时间间隔值 60 天 1 小时 30 分钟。由于第二个操作数中不包括分钟，所以

在执行减法之前，数据库服务器将第二个操作数的分钟值设置为 0。

DATETIME (2003-9-30 12:30) YEAR TO MINUTE

- DATETIME (2003-8-1 11) YEAR TO HOUR

Result: INTERVAL (60 01:30) DAY TO MINUTE

如果第二个 DATETIME 操作数的字段比第一个操作数多（不管额外字段的精度是大于还是小于第一个操作数中的字段），在计算过程中都会忽略第二个值中附加的时间单位字段。这指示第二个日期比第一个晚。

DATETIME (2005-9-30) YEAR TO DAY

- DATETIME (10-1) MONTH TO DAY

Result: INTERVAL (-1) DAY TO DAY [假定当前年为 2005]

您可以使用 mi_datetime_compare() 函数比较两个 DATETIME 值。

用 INTERVAL 值处理 DATETIME

算术INTERVAL 操作数可以将 INTERVAL 值与 DATETIME 值相加或从 DATETIME 值中减去。在这两种情况下，结果都是 DATETIME 值。如果将 INTERVAL 值与 DATETIME 值相加，那么值的顺序不重要；但是，如果进行减法运算，那么 DATETIME 值必须在前面。加或减正的 INTERVAL 值会使 DATETIME 结果时间推后或提前。下面示例中的表达式使日期提前 3 年 5 个月：

DATETIME (2000-8-1) YEAR TO DAY

+ INTERVAL (3-5) YEAR TO MONTH

Result: DATETIME (2004-01-01) YEAR TO DAY

要点： 估计加或减的逻辑。请注意，有 28、29、30 或 31 天的月份以及有 365 或 366 天的年份。

EXTEND 函数DATETIME 数据类型EXTEND 函数在大多数情况下，当操作数没有相同的精度时数据库服务器自动调整计算。但是，在某些上下文中，必须显式调整一个值的精确才能执行计算。如果加上或减去的 INTERVAL 值具有未包括在 DATETIME 值中的字段，那么必须使用 EXTEND 函数来增加 DATETIME 值的精度。（有关 EXTEND 函数的更多信息，请参阅 *GBase 8s SQL 指南：语法* 中的 Expression 段。）

TO 关键字EXTEND 函数YEAR 关键字EXTEND 函数例如，不能从上面示例中具有 YEAR TO DAY 字段限定符的 DATETIME 值中减去 INTERVAL MINUTE TO MINUTE 值。但是，如下面的示例所示，可以使用 EXTEND 函数来执行此计算：

EXTEND (DATETIME (2008-8-1) YEAR TO DAY, YEAR TO MINUTE)

- INTERVAL (720) MINUTE(3) TO MINUTE

Result：DATETIME (2008-07-31 12:00) YEAR TO MINUTE

字段限定符EXTEND 函数EXTEND 函数允许您显式将 DATETIME 精度从 YEAR TO DAY 增加至 YEAR TO MINUTE。这使得数据库服务器能够执行计算，并且生成 YEAR TO MINUTE 的扩展精度。

使用 DATETIME 和 INTERVAL 值处理 DATE

如表 2 所示，可将某些算术表达式中的 DATE 操作数与 DATETIME 或 INTERVAL 操作数配合使用（通过编写表达式来执行处理）。

表 2. 用 DATETIME 或 INTERVAL 值处理 DATE 的表达式的结果

表达式	结果
DATE - DATETIME	INTERVAL
DATETIME - DATE	INTERVAL
DATE *+或-* INTERVAL	DATETIME

在表 2 所示的情况下，第一次将 DATE 值转换为其对应的 DATETIME 等值，然后使用算术规则对表达式进行求值。

虽然在许多情况下，您可以交换 DATE 与 DATETIME 值，但是必须指定值是 DATE 还是 DATETIME 数据类型。DATE 值可以来自下列源：

类型为 DATE 的列或程序变量

TODAY 关键字

DATE( ) 函数

MDY 函数

DATE 文字

DATETIME 值可以来自以下源：

类型为 DATETIME 的列或程序变量

DATETIME 数据类型源数据

CURRENT 关键字

EXTEND 函数

DATETIME 文字

数据库语言环境定义缺省 DATE 和 DATETIME 格式。对于缺省语言环境“美国英语”，这些格式以“*mm/dd/yy*”表示 DATE 值，“*yyyy-mm-ddhh*:MM:*ss*”表示 DATETIME 值。

带引号字符串 DATE 和 DATETIME 文字字符串要将 DATE 和 DATETIME 值表示为字符串，字符串中字段的顺序必须为所需顺序。换句话说，当期望 DATE 值时，字符串必须使用 DATE 格式，当期望 DATETIME 值时，字符串必须使用 DATETIME 格式。例如，可以将字符串 10/30/2008 用作 DATE 字符串，但不能用作 DATETIME 字符串。相反，必须将 2008-10-30 或 08-10-30 用作 DATETIME 字符串。

在非缺省语言环境中，文字 DATE 和 DATETIME 字符串必须与语言环境定义的格式相匹配。有关更多信息，请参阅《*GBase 8s GLS 用户指南*》。

可以使用 **DBDATE** 和 **GL_DATE** 环境变量来定制数据库服务器期望的 DATE 格式。可以使用 **DBTIME** 和 **GL_DATETIME** 环境变量来定制数据库服务器期望的 DATETIME 格式。有关更多信息，请参阅 DBDATE 环境变量和 DBTIME 环境变量。有关所有这些环境变量的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

UNITS 运算符还可以从一个 DATE 值中减去另一个 DATE 值，但结果是正或负的 INTEGER 天数计数，而不是 INTERVAL 值。如果需要 INTERVAL 值，那么可以使用 UNITS DAY 运算符来将 INTEGER 值转换为 INTERVAL DAY TO DAY 值，否则在相减之前使用 EXTEND 来将其中一个 DATE 值转换为 DATETIME 值。

限定符字段UNITS(/)，斜杠DATE 分隔符例如，以下表达式使用 **DATE**( ) 函数将字符串常量转换为 DATE 值，计算它们的差，然后使用 UNITS DAY 关键字将 INTEGER 结果转换为 INTERVAL 值：

```
(DATE ('5/2/2007') - DATE ('4/6/1968')) UNITS DAY
Result: INTERVAL (12810) DAY(5) TO DAY
```

要点： 由于 UNITS 相对于其他 SQL 运算符具有较高的优先级，通常应将作为 UNITS 的操作数的任何算术表达式括在圆括号中，就象先前的示例那样。(( ))，圆括号表达式中的定界符

限定符字段EXTEND如果需要 YEAR TO MONTH 精度，那么可以在第一个 DATE 操作数中使用 EXTEND 函数，如下面的示例所示：

```
EXTEND (DATE ('5/2/2007'), YEAR TO MONTH) - DATE ('4/6/1969')

Result: INTERVAL (39-01) YEAR TO MONTH
```

DATETIME 数据类型在表达式中因为 DATETIME 值先出现，所以结果 INTERVAL 精度是 YEAR TO MONTH。如果 DATE 值先出现，那么结果 INTERVAL 精度将是 DAY(5) TO DAY。

处理 INTERVAL 值

仅当两个 INTERVAL 值都来自同一个类时（即两个值都是 year-month 或都是 day-time）才能将这两个值相加或相减。

在下面的示例中，从 MINUTE TO FRACTION 值中减去了 SECOND TO FRACTION 值：

```
INTERVAL (100:30.0005) MINUTE(3) TO FRACTION(4)
        - INTERVAL (120.01) SECOND(3) TO FRACTION
Result: INTERVAL (98:29.9905) MINUTE TO FRACTION(4)
```

使用数字限定符警告数据库服务器：第一个值中的 MINUTE 和 FRACTION 以及第二个值中 SECOND 超出了缺省位数。

当加上或减去 INTERVAL 值时，第二个值的字段精度不能比第一个值的字段精度高。

但是，第二个 INTERVAL 可具有比第一个精度小的字段。例如，当第一个 INTERVAL 为 DAY TO HOUR 时，第二个 INTERVAL 可以是 HOUR TO SECOND。在计算过程中会忽略第二个 INTERVAL 值中的附加字段（在此情况下为 MINUTE 和 SECOND）。

您可以使用 mi_interval_compare() 函数比较两个 INTERVAL 值。

乘或除 INTERVAL 值

(*)，星号乘运算符可以用数字乘或除 INTERVAL 值。但是，计算中的任何余数将被忽略，并且结果会截断为 INTERVAL 的精度。下面的表达式用一个具有小数部分的文字数字乘以 INTERVAL 值：

```
INTERVAL (15:30.0002) MINUTE TO FRACTION(4) * 2.5
Result: INTERVAL (38:45.0005) MINUTE TO FRACTION(4)
```

INTERVAL 数据类型在表达式中在此示例中，15 * 2.5 = 37.5 分钟，30 * 2.5 = 75 秒以及 2 * 2.5 = 5 FRACTION (4)。将 0.5 分钟转换为 30 秒，将 60 秒转换为 1 分钟，这产生最后结果 38 分钟 45 秒又 0.0005 秒。任何计算结果都具有与原始 INTERVAL 操作数相同的精度。

3.4.4 SYS_REFCURSOR 游标类型

本版 GBase 8s 支持 SYS_RECURSOR 游标类型，用于定义游标变量、游标类型的参数和返回类型。

游标类型 SYS_RECURSOR 定义的变量为游标变量。作为 SPL_var 的一种，支持现有 SPL_var 在 LET、RETURN 中的用法，用于游标变量赋值、返回游标变量。

游标变量只支持在 SPL、JDBC 端中使用，不支持在 ODBC、ESQL、DB-Access 等其它客户端中使用。

3.5 扩展数据类型

GBase 8s 使您能够创建*扩展数据类型*来表示用内置数据类型不易表示的数据。（但是不能在查询外部表的分布式事务中使用扩展数据类型。）可以创建下列类别的扩展数据类型：

- 复杂数据类型
- 单值数据类型
- 不透明数据类型

下面各节提供了这些数据类型中的每一种的概述。

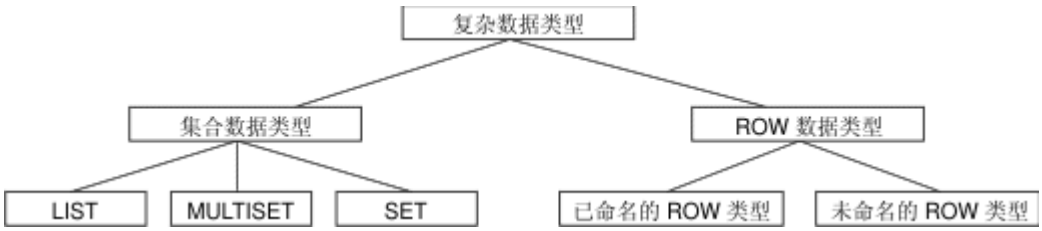
有关扩展数据类型的更多信息，请参阅《GBase 8s 数据库设计和实现指南》和 GBase 8s 用户定义的例程与数据类型开发者指南。

3.6 复杂数据类型

*复杂数据类型*可以存储一个或多个其他内置或扩展数据类型的值。

图 1 显示了 GBase 8s 支持的复杂数据类型。

图: GBase 8s 的复杂数据类型



下表总结了复杂数据类型的结构。

表 1. 集合类型是由元素组成的复杂数据类型，每个元素都具有相同的数据类型。

集合类型	描述
LIST	一组有序元素，每个元素在组中不必是唯一的。
MULTISET	一组元素，每个元素不必是唯一的。忽略元素的顺序。
SET	一组元素，每个元素都是唯一的。忽略元素的顺序。

表 2. ROW 类型是由字段组成的复杂数据类型。

ROW 类型	描述
命名 ROW 类型	用其名称标识的行类型。
未命名 ROW 类型	用其结构标识的行类型。

以嵌套复杂数据类型。例如，可以构造其字段包括一个或多个 set、multiset、ROW 类型和列表的 ROW 类型。同样，集合类型也可以具有数据类型为 ROW 类型或集合类型的元素。

包括不透明类型的复杂类型继承以下支持函数。

- **input**
- **output**
- **send**
- **recv**
- **import**
- **export**
- **import_binary**
- **export_binary**
- **assign**
- **destroy**
- **LO_handles**
- **hash**
- **lessthan**
- **equal**
- **lessthan**（仅适用于 ROW 类型）

接下来的各个主题对复杂数据类型进行了概述。有关更多信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

集合数据类型

集合数据类型是由一个或多个元素（所有元素都具有相同的数据类型）组成的复杂类型。集合元素可以为除 BYTE、TEXT、SERIAL、SERIAL8 或 BIGSERIAL 之外的任何数据类型（包括其他复杂类型）。

要点： 元素不能具有 *NULL* 值。必须对集合元素指定 *NOT NULL* 约束。没有其他约束对于集合有效。

GBase 8s 支持三种类型的内置集合类型：LIST、SET 和 MULTISET。用来声明这些集合的关键字是*类型构造函数*的名称或只是*构造函数*。有关集合类型的语法，请参阅 *GBase 8s SQL 指南：语法*。可以将相同表的不超过 97 列声明为集合数据类型。

当为集合指定元素值时，将元素值列示在构造函数后面并用花括号（ { } ）括起来。例如，假设您具有含以下 MULTISET 数据类型的集合列：

```
CREATE TABLE table1
(
        mset_col MULTISET(INTEGER NOT NULL)
)
```

下一个 INSERT 语句向此列添加一组元素值。（这两个示例中的单词 MULTISET 是 MULTISET 构造函数。）

```
INSERT INTO table1 VALUES (MULTISET{5, 9, 7, 5})
```

可使花括号保留为空来指示空集：

```
INSERT INTO table1 VALUE (MULTISET{})
```

空集合不等同于列的 NULL 值。

访问集合数据

要存取集合列的元素，必须将集合访存到集合变量中并修改集合变量的内容。集合变量可以具有任何下列类型：

- SPL 例程中的变量

有关更多信息，请参阅 *GBase 8s SQL 指南：教程*。

- GBase 8s ESQL/C 程序中的主变量

有关更多信息，请参阅 *GBase 8s ESQL/C 程序员手册*。

还可以使用嵌套点表示法来存取集合数据。有关访问集合元素的更多信息，请参阅 *GBase 8s SQL 指南：教程*。

重要： 集合数据类型作为用于函数索引的函数的自变量无效。

ROW 数据类型

ROW 数据类型是一个或多个元素（称为*字段*）的有序集合。每个字段都具有名称和数据类型。行的字段与表列差不多，但有一些重要的差别：

- 字段没有缺省子句。
- 不能对字段定义约束。
- 只能将字段与行类型一起使用，而不能将其与表一起使用。

存在两种类型的 ROW 数据类型：

- *命名 ROW 数据类型*是用其名称标识的。
- *未命名 ROW 数据类型*是用其结构标识的。

未命名 ROW 数据类型的*结构*是其字段的数目（和数据类型的顺序）。

可以将同一个表的不超过 195 列声明为 ROW 数据类型。有关 ROW 数据类型的更多信息，请参阅 ROW 数据类型，已命名和 ROW 数据类型，未命名。

您可以在命名与未命名 ROW 数据类型之间进行强制转型；《*GBase 8s 数据库设计和实现指南*》中对此进行了描述。

3.6.1 单值数据类型

单值数据类型具有与数据库中某些其他源数据类型相同的内部结构。源类型可以是内置或扩展数据类型。单值类型与其源类型的区别在于对单值类型定义的支持函数。

同一个表中 DISTINCT 集合数据类型（SET、LIST 和 MULTISET）的列不超过 97 列。同一个表中类型为 DISTINCT（基于 BYTE、TEXT、ROW、或 LVARCHAR 源类型）的列不超过 195 列。（此处的 195 列接近基页大小为 2 Kb 的平台适用的下限。对于基页大小为 4 Kb 的平台，例如 AIX® 系统，这些数据类型的列上限约为 450 列。）有关更多信息，请参阅 DISTINCT 数据类型一节。另请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

3.6.2 不透明数据类型

不透明数据类型是完全封装的用户定义的数据类型。对于数据库服务器，不透明数据类型的内部结构是未知的。

除 DISTINCT 类型的用户定义类型 (UDT) 之外，源类型是内置类型的 UDT 为不透明数据类型。

内置数据类型 BLOB、BOOLEAN、CLOB 和 LVARCHAR 是作为不透明数据类型来实现的。您无法在跨服务器的分布式操作中访问这些内置不透明数据类型，但是您可以在相同 GBase 8s 实例的其他数据库中进行访问。

您必须为不透明数据类型的数据库服务器提供以下信息：

- 如何在磁盘上存储数据值的数据结构
- 确定如何在磁盘存储格式与用于数据输入和显示的用户格式之间转换的支持函数
- 确定如何为此数据类型构建、使用和处理索引的辅助访问方法
- 使用数据类型的用户函数
- 用于在数据库中注册不透明类型的系统目录条目

不透明类型的内部结构对数据库服务器不可视，只能通过用户定义的例程存取。不透明类型的定义存储在 sysxtdtypes 系统目录表中。以下 SQL 语句会维护数据库中不透明类型的定义：

- CREATE OPAQUE TYPE 语句在数据库中注册了一个新的不透明类型。
- DROP TYPE 语句从数据库中除去先前定义的不透明类型。

有关更多信息，请参阅 OPAQUE 数据类型一节。另请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

数据类型强制转型和转换

有时，使用 CREATE TABLE 语句指定给列的数据类型不适用。当需要存储比当前数据类型所能容纳的值更大的值时，您可以更改列的数据类型。数据库服务器允许您使用下列方法之一来更改列的数据类型或者将列的值强制转型为另一数据类型：

- 使用 ALTER TABLE 语句来修改列的数据类型。

例如，如果您创建了 SMALLINT 列，后来发现必须存储大于 32,767 的整数，那么必须更改该列的数据类型才能存储更大的值。可以使用 ALTER TABLE 来将数据类型更改为 INTEGER。该转换会更改当前存在于列中的所有值和可能添加的任何新值的数据类型。

- 使用 CAST AS 关键字或双冒号 (::) 强制转型运算符来将值强制转型为另一种数据类型。

强制转型并不会永久地改变值的数据类型；它只是以更方便的形式表示值。将用户定义的数据类型强制转型为内置类型使客户机程序能够在不知道数据类型的内部结构的情况下处理这些数据类型。

如果更改数据类型，那么新数据类型必须能够存储所有旧的值。

数据类型转换和强制转型都取决于在 syscasts 系统目录表中注册的强制转型。有关 syscasts 的信息，请参阅 SYSCASTS。

强制转型是内置的或用户定义的。存在强制转型单值和扩展数据类型的准则。有关对不透明数据类型强制转型的更多信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。有关对其他扩展数据类型进行强制转型的信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

使用内置强制转型

用户 gbasedbt 拥有内置强制转型。他们控制从一种内置数据类型到另一种内置数据类型的转换。内置强制转型允许数据库服务器尝试以下数据类型转换：

- 字符类型至任何其他字符类型
- 字符类型至另一种内置类型或从另一种内置类型至字符类型
- 数字类型至任何其他数字类型

必要时，数据库服务器自动调用适当的内置强制转型。对于时间数据类型，DATE 与 DATETIME 数据类型之间的转换需要使用 EXTEND 函数进行显式强制转型，而对于数字与 INTERVAL 之间的转换，需要使用 UNITS 运算符进行显式强制转型。内置强制转型不可用于将大型（BYTE、BLOB、CLOB 和 TEXT）内置类型转换为其他内置数据类型。

当将列从一种内置数据类型转换为另一种时，数据库服务器将适当的内置强制转型应用于列中已存在的每个值。如果新的数据类型不能存储任何生成的值，那么 ALTER TABLE 语句会失败。

例如，如果您尝试将列从 INTEGER 数据类型转换为 SMALLINT 数据类型，并且 INTEGER 列中存在以下值，那么数据库服务器不会更改数据类型，因为 SMALLINT 列不能容纳大于 32,767 的数字：

```
400    700    50000      700
```

如果您尝试将数据从 FLOAT 或 SMALLFLOAT 列传送至 INTEGER、SMALLINT 或 DECIMAL 列，那么会发生相同的情况。在数据类型转换期间可能会发生溢出、下溢或截断错误。

下面各节描述某些类型的强制转型和转换期间的数据库服务器行为。

数字到数字的转换

当将数据从一种数字数据类型转换为另一种数字数据类型时，偶尔会发现舍入错误。

下表指示哪些数字数据类型转换是可接受的，以及当在某些数字数据类型之间转换时可能遇到哪些种类的错误。 在表中，使用了以下代码：

OK

无错误

P

可能会出错，这取决于小数的精度

E

可能会出错，这取决于数据值

D

无错误，但可能会丢失较少的有效位

表 1. 可接受的转换和可能的错误

目标类型	SMALL INT	INTEGER	INT8	SMALL FLOAT	FLOAT	DECIMAL
SMALLINT	OK	OK	OK	OK	OK	OK
INTEGER	E	OK	OK	E	OK	P
INT8	E	E	OK	D	E	P
SMALLFLOAT	E	E	E	OK	OK	P
FLOAT	E	E	E	D	OK	P
DECIMAL	E	E	E	D	D	P

例如，如果将 FLOAT 值转换为 DECIMAL(4,2)，那么数据库服务器在将浮点数存储为

DECIMAL 之前先对它进行舍入。

此转换可能会导致错误，这取决于指定给 DECIMAL 列的精度。

在数字与字符之间转换

可以将字符列（具有诸如 CHAR、NCHAR、NVARCHAR 或 VARCHAR 等数据类型）转换为数字列。但是，如果数据字符串包含数字列中任何无效字符(例如：字母 *1* 而不是数字 *1*)，那么数据库服务器会返回错误。

还可以将数字列转换为字符列。但是，如果字符列不够大，不足以接收数字，那么数据库服务器会生成错误。如果数据库服务器生成错误，那么它不能完成 ALTER TABLE 语句或强制转型，并使列值保留为字符。您就会接收到错误消息，并且会自动回滚语句（不管您是否在事务中）。

INTEGER 和 DATE 之间的转换

您可以将整数列（SMALLINT、INTEGER 或 INT8）转换为 DATE 值。数据库服务器将整数解释为内部格式的 DATE 列的值。您还可以将 DATE 列转换为整数列。数据库服务器将内部格式的 DATE 列存储为表示儒略日期的整数。

DATE 和 DATETIME 之间的转换

可以将 DATE 列转换为 DATETIME 列。但是，如果 DATETIME 列包含的字段比 DATE 列多，那么数据库服务器要么忽略这些字段，要么用零填充这些字段。以下列表中的说明显示了如何转换这两种数据类型（假设缺省数据格式为 *mm/dd/yyyy*）：

- ✓ 如果将 DATE 转换为 DATETIME YEAR TO DAY，那么数据库服务器会将现有的 DATE 值转换为 DATETIME 值。例如：值 08/15/2002 变为 2002-08-15。
- ✓ 如果将 DATETIME YEAR TO DAY 转换为 DATE 格式，那么值 2002-08-15 将变为 08/15/2002。
- ✓ 如果将 DATE 转换为 DATETIME YEAR TO SECOND，那么数据库服务器会将现有 DATE 值转换为 DATETIME 值并用零填充其他的 DATETIME 字段。例如：08/15/2002 变为 2002-08-15 00:00:00。
- ✓ 如果将 DATETIME YEAR TO SECOND 转换为 DATE，那么数据库服务器会将现有 DATETIME 转换为 DATE 值，但删除小于 DAY 的时间单位的字段。例如：2002-08-15 12:15:37 变为 08/15/2002。

使用用户定义的强制转型

创建隐式和显式强制转型的用户拥有这些强制转型。他们控制用户定义的数据类型与其他数据类型之间的强制转型和转换。用户定义的数据类型的开发者必须创建某些隐式和显式强制转型以及用来实现这些强制转型的函数。这些强制转型使用户定义的类型可以表示为客户机可以处理的格式。

有关如何注册和使用隐式和显式强制转型的信息，请参阅 *GBase 8s SQL 指南：语法* 和

《*GBase 8s 数据库设计和实现指南*》中的 CREATE CAST 语句。

隐式强制转型

隐式强制转型允许您将用户定义的数据类型转换为内置类型，反之亦然。数据库服务器在必须对表达式进行求值和比较或传递自变量时自动调用单个隐式强制转型。需要多个隐式强制转型的操作会失败。

用户可以使用 CAST AS 关键字或双冒号（::）强制转型运算符来显式调用隐式强制转型。

显式强制转型

与隐式强制转型或内置强制转型不同，数据库服务器*决不会*自动调用显式强制转型。用户必须使用 CAST AS 关键字或使用双冒号（ :: ）强制转型运算符来显式调用它们。

确定应用哪种强制转型

数据库服务器使用下列规则来确定在特定情况下应用哪种强制转型：

1. 为了比较两种内置类型，数据库服务器自动调用适当的内置强制转型。
2. 数据库服务器仅对每个操作数应用一次隐式强制转型。如果需要两个或更多强制转型才能将操作数转换为指定类型，那么用户必须显式调用其他强制转型。

在下面的示例中，将文字值 5.55 隐式强制转型为 DECIMAL，然后将它显式强制转型为 MONEY，最后强制转型为 yen：

```
CREATE DISTINCT TYPE yen AS MONEY
. . .
INSERT INTO currency_tab
        VALUES (5.55::MONEY::yen)
```

要将单值类型与其源类型进行比较，用户必须将其中一种类型显式强制转型为另一种类型。

为了将单值类型与不是其源的类型进行比较，数据库服务器将查找源类型与指定类型之间的隐式强制转型。

如果这两种强制转型均未注册，那么用户必须调用单值类型与指定类型之间的显式强制转型。如果此强制转型未注册，那么数据库服务器自动调用从源类型到指定类型的强制转型。

如果没有定义这些强制转型，那么比较会失败。

要将不透明类型与内置类型进行比较，用户必须将不透明类型显式强制转型为数据库服务器可以理解的数据类型（如 LVARCHAR、SENDRECV、IMPEX 或 IMPEXBIN）。然后，数据库服务器调用内置强制转型来将结果转换为指定的内置类型。

要比较两种不透明类型，用户必须将其中一种不透明类型显式地强制转型为数据

库服务器可以理解的形式（例如：LVARCHAR、SENDRECV、IMPEX 或 IMPEXBIN），然后将此类型显式地强制转型为第二种不透明类型。

有关强制转型和 IMPEX、IMPEXBIN、LVARCHAR 和 SENDRECV 类型的信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

单值类型的强制转型

根据内置类型或现有不透明类型或 ROW 类型来定义单值类型。虽然单值类型的数据与源类型的数据具有相同的长度、对齐方式和数据传递方式，但是不能将这两种类型直接进行比较。要将单值类型与其源类型进行比较，必须将其中一种类型显式强制转型为另一种类型。

当创建新的单值类型时，数据库服务器会自动注册两种显式强制转型：

从单值类型到其源类型的强制转型

从源类型到单值类型的强制转型

可以创建单值类型与其源类型之间的隐式强制转型。但是，要创建隐式强制转型，首先必须删除单值类型与其源类型之间的缺省显式强制转型。

还可以使用已对源类型注册的所有强制转型而不必对单值类型进行修改。还可以创建和注册只适用于单值类型的新强制转型和支持函数。

有关显示如何为单值类型创建强制转型函数以及将函数注册为强制转型的示例，请参阅《*GBase 8s 数据库设计和实现指南*》。

可以对哪些扩展数据类型进行强制转型？

下表显示了可以进行强制转型的扩展数据类型组合。

表 2. 扩展数据类型组合

目标类型	不透明类型	单值类型	命名 ROW 类型	未命名 ROW 类型	集合类型	内置类型
不透明类型	显式或隐式	显式	显式	无效	无效	显式或隐式 [3]
单值类型	显式 [3]	显式	显式	无效	无效	显式或隐式
命名 ROW 类型	显式 [3]	显式	显式 [3]	显式 [1]	无效	无效
未命名 ROW 类型	无效	无效	显式 [1]	隐式 [1]	无效	无效
集合类型	无效	无效	无效	无效	显式 [2]	无效
内置类型	显式或隐式 [3]	显式或隐式	无效	无效	无效	系统定义（隐式）

[1] 当两种 ROW 类型在结构上等价，或存在强制转型来处理相应字段类型不同的数据转换时适用。[2] 当存在强制转型来在各个集合类型的元素类型之间进行转换时适用。[3] 当存在用户定义的强制转型来在两种数据类型之间进行转换时适用。

该表只显示了源类型与目标类型之间的强制转型是否可能。在某些情况下，首先必须创建用户定义的强制转型，然后才能执行两种数据类型之间的转换。在其他情况下，数据库服务器提供隐式强制转型或必须显式调用的内置强制转型。

3.7 运算符优先级

*运算符*是可以在 SQL 表达式中出现的符号或关键字。大多数 SQL 运算符都被限制为其操作数和返回值的数据类型。某些运算符只支持具有内置数据类型的操作数；其他一些运算符可以支持内置和扩展数据类型作为操作数。

下表以优先级的降序（从最高到最低）排列显示了 GBase 8s 支持的运算符的优先级。具有相同优先级的运算符列出在同一行中。

运算符优先级	在表达式中的示例
. (*membership*) [] (*substring*)	customer.phone[1, 3]
UNITS	x UNITS DAY
+ - (*unary*)	-y
:: (*cast*)	NULL::TEXT
*/	x/y
+- (*binary*)	x-y
|| (*concatenation*)	customer.fname || customer.lname
ANY ALL SOME	orders.ship_date > SOME (SELECT paid_date FROM orders)
NOT	NOT y
<<==>>=!= <>	x>= y
IN BETWEEN ... AND LIKE MATCHES	customer.fname MATCHES y
AND	x AND y
OR	x OR y

请参阅 *GBase 8s SQL 指南：语法* 以获取这些 SQL 运算符的语法和语义。

4 环境变量

各种*环境变量*都会影响 GBase 8s 产品的功能。可设置环境变量来识别终端、指定软件的位置以及定义其他参数。

一些环境变量是必需的，其他则是可选的。必须设置必需的环境变量或接受它们的缺省设置。

这些主题描述如何使用应用于一个或多个 GBase 8s 产品的环境变量并显示如何进行设置。

4.1 环境变量的类型

本章中说明了两种类型的环境变量：

特定于 GBase 8s 的环境变量

要使用 GBase 8s 产品时，设置 GBase 8s 环境变量。每个 GBase 8s 产品出版物都会指定一些环境变量，您必须设置它们才能使用该产品。

与特定操作系统一起使用的环境变量

GBase 8s 产品依赖于某些标准操作系统环境变量的正确设置。例如：必须总是设置 PATH 环境变量。

在 UNIX™ 环境中，可能还需要设置 TERMCAP 或 TERMINFO 环境变量才能有效地使用某些产品。

《*GBase 8s GLS 用户指南*》中描述了支持非缺省语言环境的 GLS 环境变量。表 1 的环境变量列表中包含了 GLS 变量。

数据库服务器使用在初始化该数据库服务器时生效的环境变量。

onstat - g env 命令列示生效的环境设置。

提示： 特定于客户机应用程序或 SQL API 的其他环境变量可能在该产品的出版物中说明。

重要： 在初始化数据库和创建 sysmaster 数据库时，不要在用户 gbasedbt 的主目录（也不要在该目录的 .gbasedbt 文件中）设置任何环境变量。

4.2 环境变量的限制

环境变量块的大小

会话开始时，客户机会分组服务器将使用的所有环境变量，并将环境变量作为单一块发送到服务器。此块的最大大小是 32K。如果环境变量块大于 32K，错误 -1832 会返回到应用程序。此错误的文本是“环境块大于 32K”。

要解决此错误，您可以取消设置一个或多个环境变量，或减少一些环境变量的大小。

4.3 在 UNIX 上使用环境变量

您可以设置、取消设置、修改和查看环境变量。如果已经使用 GBase 8s 产品，那么可能需要设置某些或全部相应的环境变量。

在 UNIX™ 上，可以在下列位置设置环境变量：

在命令行上的系统提示符处

如果在系统提示符处设置某个环境变量，必须在下一次登录到系统时重新指定该变量。

在环境配置文件中

环境配置文件是一个公共文件或专用文件，可在其中设置 GBase 8s 产品使用的所有环境变量。使用这种文件可减少在命令行或 shell 文件中必须设置的环境变量的数目。

在登录文件中

在 .login、.cshrc 或 .profile 文件中设置的环境变量值是在您每次登录系统时自动指定的。

在 SQL 的 SET ENVIRONMENT 语句中

某些环境变量的值可通过 SET ENVIRONMENT 语句重新设置。新设置的作用域通常是执行 SET ENVIRONMENT 语句的例程，但对于 GBase 8s 的 OPTCOMPIND 环境变量（如 OPTCOMPIND 环境变量一节中所述），它是当前会话。有关这些例程和 SET ENVIRONMENT 语句的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

在 GBase 8s ESQL/C 中，可以使用 putenv( ) 系统调用在应用程序内设置受支持的环境变量，并使用 getenv( ) 系统调用来检索值（如果 UNIX 系统支持这些函数）。有关 putenv( ) 和 getenv( ) 的更多信息，请参阅 *GBase 8s ESQL/C 程序员手册* 和 C 文档。

4.3.1 在配置文件中设置环境变量

随 GBase 8s 产品提供的公共（共享）环境配置文件位于 **$GBASEDBTDIR/etc/gbasedbt.rc** 中。此共享文件的许可权必须设置为 644。

用户可通过在专用环境配置文件中设置变量来覆盖系统或共享环境变量。此文件必须具有下列所有特征：

- 存储在用户的主目录中
- 命名 **.gbasedbt**
- 许可权设置为用户可读取

环境配置文件可包含注释行（前面加上 # 注释指示符）和设置各个值（用空格或制表符隔开）的变量定义行，如以下示例所示：

```
# This is an example of an environment-configuration file
#
DBDATE DMY4-
#
# These are ESQL/C environment variable settings
#
GBASEDBTC gcc
CPFIRST TRUE
```

可使用 **ENVIGNORE** 环境变量（在 ENVIGNORE 环境变量 (UNIX) 中作了描述）来覆盖环境配置文件中的一个或多个条目。使用 GBase 8s **chkenv** 实用程序（在使用 chkenv 实用程序检查环境变量中进行了描述）来对环境配置文件的内容进行完整性检查。如果该文件包含错误的环境变量或者如果该文件太大，那么 **chkenv** 实用程序返回错误消息。

第一次在 shell 文件或环境配置文件中设置环境变量时，必须告诉 shell 进程读取您的输入，然后才能使用 GBase 8s 产品。如果使用 C shell，那么将该文件当作**源文件**；如果使用 Bourne 或 Korn shell，那么使用句点 (**.**)来执行该文件。

4.3.2 设置登录时的环境变量

将设置环境变量的命令添加至适当的登录文件：

对于 C shell

.login 或 **.cshrc**

对于 Bourne shell 或 Korn shell

.profile

4.3.3 设置环境变量的语法

使用标准 UNIX™ 命令来设置环境变量。下表中的示例显示如何将 ABCD 环境变量设置为 C shell、Bourne shell 和 Korn shell 的*值*。Korn shell 还支持快捷方式，如最后一行所示。环境变量区分大小写。

Shell	命令
C	setenv ABCD value
Bourne	ABCD=value export ABCD
Korn	ABCD=value export ABCD
Korn	export ABCD=value

下图显示在本章中如何表示设置环境变量的语法。这些图指示 C shell 的设置；对于 Bourne 或 Korn shell，使用先前表中说明的语法。

4.3.4 取消设置环境变量

要取消设置环境变量，输入以下命令。

Shell	命令
C	unsetenv ABCD
Bourne 或 Korn	unset ABCD

4.3.5 修改环境变量设置

有时必须向已设置的环境变量添加信息。例如，在 UNIX™ 中始终会设置 **PATH** 环境变量。在使用 GBase 8s 产品时，必须向 **PATH** 设置添加存储 GBase 8s 产品的可执行文件的目录的名称。

在以下示例中，GBASEDBTDIR 为 /usr/gbasedbt。（即，在安装期间，GBase 8s 产品安装在 /usr /gbasedbt 目录中。)可执行文件在 bin 子目录中，即 /usr/gbasedbt/bin。要将此目录添加在 C shell PATH 环境变量之前，使用以下命令：

```
setenv PATH /usr/gbasedbt/bin:$PATH
```

可使用 GBASEDBTDIR 环境变量的值(表示为 $GBASEDBTDIR)而不是输入显式路径名，如以下示例所示：

```
setenv GBASEDBTDIR /usr/gbasedbt
setenv PATH $GBASEDBTDIR/bin:$PATH
```

您可能更倾向于使用此版本以确保 PATH 条目不会与在 GBASEDBTDIR 中设置的搜索路径相冲突，这样您无需在每次更改 GBASEDBTDIR 时重置 PATH。如果在 C shell 命令行上设置 PATH 环境变量，那么可能需要为现有的 GBASEDBTDIR 和 PATH 加上花括号（{}），如以下命令所示：

```
setenv PATH ${GBASEDBTDIR}/bin:${PATH}
```

有关如何设置和修改环境变量的更多信息，请参阅操作系统的出版物。

4.3.6 查看环境变量设置

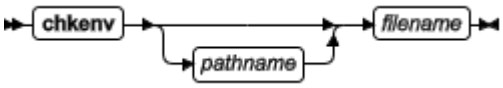
安装一个或多个 GBase 8s 产品后，在系统提示时输入以下命令来查看当前环境设置。

UNIX™ 版本	命令

BSD UNIX	env
UNIX System V	printenv

4.3.7 使用 chkenv 实用程序检查环境变量

chkenv 实用程序检查共享或专用环境配置文件的有效性。它验证该文件中各个环境变量的名称，但不验证它们的值。当您在环境配置文件中定义 GBase 8s 产品使用的所有环境变量时，使用 **chkenv** 来提供调试信息。



filename

是要调试的环境配置文件的名称。

pathname

是环境变量文件所在的完整目录路径。

4.3.8 环境变量的优先顺序规则

当 GBase 8s 产品访问环境变量时，通常下列优先级规则适用：

1. 最高优先级授予在环境 (shell) 中定义的值（通过在 shell 提示符下显式设置该值）。
2. 第二优先级授予在用户主目录 (**~/.gbasedbt**) 中的专用环境配置文件中定义的值。
3. 下一优先级授予在公共环境配置文件 (**$GBASEDBTDIR/etc/gbasedbt.rc**) 中定义的值。
4. 最低优先级授予缺省值（如果存在的话）。

有关 GLS 环境变量的优先级信息，请参阅《*GBase 8s GLS 用户指南*》。

要点： 如果在启动数据库服务器之前设置了一个或多个环境变量，而且未对客户机产品显式设置相同的环境变量，那么客户机将采用原始设置。

环境设置

可以下列方式为命令提示符实用程序设置环境变量：

使用“控制面板”中的“系统”applet

在命令行会话中

使用系统 applet 来更改环境变量

要使用控制面板中的“系统”applet 更改环境变量

1. 从“控制面板”窗口中双击“系统”applet 图标。

2. 单击靠近窗口顶部的“环境”选项卡。

两个列表框显示“系统环境变量”和“用户环境变量”。“系统环境变量”适用于整个系统，而“用户环境变量”仅适用于个别用户的会话。

3.

4. 要更改现有变量的值，选择该变量。 该变量的名称及其当前值在窗口底部的框中。

5. 要添加新变量，突出显示现有变量并在窗口底部的框中输入新的变量名。

6. 接着，在窗口底部输入新变量的值并单击**设置**。

7. 要删除变量，请选择该变量并单击**删除**。

要点： 为了使用“系统”applet 来更改“系统”环境变量，您必须属于 Administrators 组。有关将用户指定给组的信息，请参阅操作系统文档。

4.4 在 Windows 上使用环境变量

以下各节讨论为 Windows™ 应用程序设置、查看、取消设置和修改环境变量。

- **在 Windows 上设置环境变量的位置**
- **环境设置**
- **Windows 环境变量的优先顺序规则**

4.4.1 在 Windows 上设置环境变量的位置

根据您所使用的 GBase 8s 应用程序，您可以在 Windows™ 上的几个位置设置环境变量。

如环境设置中所述，可以几种方式设置环境变量。

SQL 的 SET ENVIRONMENT 语句可设置某些特定于例程的环境选项。有关更多信息，请参阅 *GBase 8s SQL 指南：语法* 中对 SET ENVIRONMENT 的描述。

要在 Windows 环境中使用客户机应用程序（例如 GBase 8s ESQL/C 或“模式工具”），请使用 Setnet32 实用程序来设置环境变量。有关 Setnet32 实用程序的信息，请参阅您的操作系统的《*GBase 8s 客户机产品安装指南*》。

在 GBase 8s ESQL/C 中，可使用 ifx_putenv() 函数在应用程序内设置受支持的环境变量，还可使用 ifx_getenv() 函数来检索值（如果 Windows 系统支持这些函数）。

有关 ifx_putenv( ) 和 ifx_getenv( ) 的更多信息，请参阅《*GBase 8s ESQL/C 程序员手册*》。

4.4.2 环境设置

可以下列方式为命令提示符实用程序设置环境变量：

- 使用“控制面板”中的“系统”applet
- 在命令行会话中

使用系统 applet 来更改环境变量

“系统”applet 提供图形界面来创建、修改和删除系统范围内的和特定于用户的变量。使用“系统”applet 设置的环境变量对所有命令提示符会话都可见。

要使用控制面板中的“系统”applet 更改环境变量

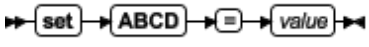
1. 从“控制面板”窗口中双击“系统”applet 图标。
2. 单击靠近窗口顶部的“环境”选项卡。
3. 两个列表框显示“系统环境变量”和“用户环境变量”。“系统环境变量”适用于整个系统，而“用户环境变量”仅适用于个别用户的会话。
4. 要更改现有变量的值，选择该变量。该变量的名称及其当前值在窗口底部的框中。
5. 要添加新变量，突出显示现有变量并在窗口底部的框中输入新的变量名。
6. 接着，在窗口底部输入新变量的值并单击**设置**。
7. 要删除变量，请选择该变量并单击**删除**。

要点： 为了使用“系统”applet 来更改“系统”环境变量，您必须属于 Administrators 组。有关将用户指定给组的信息，请参阅操作系统文档。

使用命令提示符来更改环境变量

您可以在命令提示符中更改环境变量的设置。

下图显示在 Windows™ 中的命令提示符下设置环境变量的语法。



如果未指定任何 *value*，那么表明取消设置此环境变量，就好像它不存在一样。

要在安装一个或多个 GBase 8s 产品后查看当前设置，在命令提示符下输入以下命令。

有时必须向已设置的环境变量添加信息。例如，在 Windows 环境中始终会设置 PATH 环境变量。当您使用 GBase 8s 产品时，必须将存储 GBase 8s 产品的可执行文件的目录的名称添加至 PATH。

在以下示例中，GBASEDBTDIR 为 d:\gbasedbt（即，在安装过程中，GBase 8s 产品安装在 d:\gbasedbt 目录中）。可执行文件在 bin 子目录中，即 d:\gbasedbt\bin。要在 **PATH** 环境变量值的开头添加此目录，使用以下命令：

```
set PATH=d:\gbasedbt\bin;%PATH%
```

可使用 GBASEDBTDIR 环境变量的值（表示为 **%GBASEDBTDIR%**）而不是输入显式路径名，如以下示例所示：

```
set GBASEDBTDIR=d:\gbasedbt
set PATH=%PATH%
```

您可能更倾向于使用此版本以确保 PATH 条目不会与在 GBASEDBTDIR 中设置的搜索路径冲突，同时不必在每次更改 GBASEDBTDIR 时重置 PATH。

有关设置和修改环境变量的更多信息，请参阅操作系统出版物。

使用 dbservername.cmd 来初始化命令提示符环境

每次打开 Windows™ 命令提示符时，它都会作为一个独立的环境运行。因此，您在其中设置的环境变量仅对特定命令提示符实例有效。

例如：如果打开一个命令窗口并设置变量 GBASEDBTDIR，然后打开另一个命令窗口并输入 set 以检查环境，那么会发现并未在新的命令提示符会话中设置 GBASEDBTDIR。

数据库服务器安装程序会创建一个批处理文件，您可以使用它来配置命令提示符实用程序，以确保每次运行命令提示符会话时都能正确地初始化命令提示符环境。批处理文件 dbservername.cmd 位于 %GBASEDBTDIR% 中，是一个可使用任何文本编辑器修改的纯文本文件。如果已经在 %GBASEDBTDIR% 中安装了多个数据库服务器，那么将会有多个具有 .cmd 扩展名的批处理文件，每一个都包含与其相关联的数据库服务器的名称。

要从命令提示符运行 dbservername.cmd，请输入 dbservername 或配置命令提示符以便它在启动时自动运行 dbservername.cmd。

4.4.3 Windows 环境变量的优先顺序规则

当 GBase 8s 产品访问环境变量时，通常下列优先级规则适用：

1. 具有选中的 **Use my settings** 框的 Setnet32 中的设置
2. 具有未选中的 **Use my settings** 框的 Setnet32 中的设置
3. 在运行应用程序之前对命令行的设置。
4. 作为用户变量在 Windows™ 中的设置。
5. 作为系统变量在 Windows 中的设置。
6. 最低优先级授予缺省值。

在应用程序启动时它检验最初的五个值。除非另有说明，否则在运行应用程序之后再更改环境变量将不会起任何作用。

4.5 GBase 8s 产品中的环境变量

下面的主题讨论（以字母顺序）GBase 8s 数据库服务器产品以及它们的实用程序使用的环境变量。

重要： 下列环境变量的描述包括在 UNIX™ 上设置环境变量的语法。

4.5.1 ANSIOWNER 环境变量

在符合 ANSI 标准的数据库中，您可通过将 ANSIOWNER 环境变量设置为 1 来防止发生在没有用引号定界的所有者名称中小写字母转换成大写字母的缺省行为。

要防止符合 ANSI 标准的数据库中所有者名称的小写字母转换成大写字母，必须在初始化 GBase 8s 之前设置 **ANSIOWNER**。

下表显示了符合 ANSI 标准的 GBase 8s 数据库如何存储或读取称为 oblong 的数据库对象的指定名称（如果您是 oblong 的所有者并且您的 userid（全部用小写字母）为 owen）：

表 1. 隐式的、未加引号的和加引号的所有者名称的字母大小写，有或者无 ANSIOWNER

所有者格式	规范	ANSIOWNER = 1	未设置 ANSIOWNER
隐式：	oblong	owen.oblong	OWEN.oblong
未加引号：	owen.oblong	owen.oblong	OWEN.oblong
加引号：	'owen'.oblong	owen.oblong	owen.oblong

因为它们不匹配您的 userid 的字母大小写，所以指定了格式（存储为 OWEN.oblong）的任何 SQL 语句都将失败并出错。

4.5.2 CPFIRST 环境变量

使用 CPFIRST 环境变量，可以为您的编程环境中的所有 GBase 8s ESQL/C 源文件指定缺省编译顺序。

在未设置 CPFIRST 的情况下编译 GBase 8s ESQL/C 程序时，在缺省情况下，GBase 8s ESQL/C 预处理器首先在程序源文件上运行，然后将结果文件传递给 C 语言预处理器和编译器。但是，可以以下顺序编译 GBase 8s ESQL/C 程序源文件：

1. 运行 C 预处理器
2. 运行 GBase 8s ESQL/C 预处理器
3. 运行 C 编译器和链接程序

要对特定程序使用非缺省编译顺序，可对程序源文件指定 .ecp 扩展名，对具有 .ec 扩展名的程序源文件运行 **esql** 命令的 -cp 选项，或设置 CPFIRST。

将 CPFIRST 设置为 TRUE（仅大写），以便在对您所在环境中的所有 GBase 8s ESQL/C 源文件运行 GBase 8s ESQL/C 预处理器之前运行 C 预处理器，而不管 -cp 选项是否已传递到 **esql** 命令，或者这些源文件是否具有 .ec 或 .ecp 扩展名。

要对其中 CPFIRST 环境变量已设置为 TRUE 的系统复原缺省顺序，可将 CPFIRST 设置为 FALSE。在支持 C shell 的 UNIX™ 系统上，以下命令具有相同的作用：

Unsetenv CPFIRST

4.5.3 CMCONFIG 环境变量

设置 CMCONFIG 环境变量可以指定连接管理器配置文件的位置。使用该配置文件可以指定服务级别协议和其他连接管理器配置选项。

path/file_name 是连接管理器配置文件的完整路径和文件名。

如果未设置 CMCONFIG 环境变量，且未在 oncmsm 实用程序命令行上指定配置文件名，那么连接管理器将尝试装入具有以下路径和文件名的文件：

$GBASEDBTDIR/etc/cmsm.cfg

示例

假定 CMCONFIG 环境变量指向连接管理器配置文件的有效路径和文件名。要使用 shell 环境中指定的配置文件重新装入连接管理器实例，请输入以下命令：

./oncmsm -r

要使用 shell 环境中指定的配置文件关闭连接管理器实例，请输入以下命令：

./oncmsm -k

4.5.4 DBACCNOIGN 环境变量

使用 DBACCNOIGN 环境变量，可以指定在发生指定的错误时 DB-Access 实用程序的行为。

如果在以下某种情况下发生错误，那么 DBACCNOIGN 环境变量会影响 DB-Access 实用程序的行为：

- 以非菜单方式运行 DB-Access。
- 仅在 GBase 8s 中，以菜单方式通过 DB-Access 执行 LOAD 命令。

如果在上述任一情况下运行 DB-Access 实用程序时发生错误，请将 DBACCNOIGN 环境变量设置为 1，以回滚未完成的事务。

例如，假定 DB-Access 运行以下 SQL 命令：

```
DATABASE mystore
BEGIN WORK

INSERT INTO receipts VALUES (cust1, 10)
INSERT INTO receipt VALUES (cust1, 20)
INSERT INTO receipts VALUES (cust1, 30)

    UPDATE customer
        SET balance =
            (SELECT (balance-60)
            FROM customer WHERE custid = 'cust1')
        WHERE custid = 'cust1
COMMIT WORK
```

此处有一条语句的表名拼写错误：**receipt** 表不存在。如果未在环境中设置 DBACCNOIGN，那么 DB-Access 会将两条记录插入到 **receipts** 表中，并更新 **customer** 表。现在 **customer** balance 的减少量超过了插入的 receipts 的总和。

但如果将 DBACCNOIGN 设置为 1，那么会打开消息，指示 DB-Access 回滚了所有 INSERT 和 UPDATE 语句。这些消息还会标识错误原因，以便您能够解决问题。

设置 DBACCNOIGN 时的 LOAD 语句示例

可在执行 LOAD 语句期间设置 DBACCNOIGN 环境变量来保护数据完整性，即使 DB-Access 以菜单方式运行 LOAD 语句也是如此。

假定您从 DB-Access SQL 菜单执行 LOAD 语句。正确装入了 49 行数据，但第 50 行数据包含无效值，因而导致错误。如果将 DBACCNOIGN 设置为 1，那么数据库服务器

不会将先前的 49 行插入到数据库中。如果未设置 DBACCNOIGN，那么数据库服务器会插入前 49 行。

4.5.5 DBANSIWARN 环境变量

使用 DBANSIWARN 环境变量，可以指示您希望执行 ANSI 标准 SQL 语法的 GBase 8s 扩展检查。

与大多数环境变量不同，您不需要为 DBANSIWARN 设置值。您可以将其设置为任何值，也可以不设置任何值。

在设置了 DBANSIWARN 的情况下运行 DB-Access，与在从命令行调用 DB-Access（或可识别 **-ansi** 标志的任何 GBase 8s 产品）时加上 **-ansi** 标志具有同等功效。如果在运行 DB-Access 之前设置了 DBANSIWARN，那么任何语法扩展警告都将显示在屏幕上的 SQL 菜单内。

在运行时，如果所执行的语句被识别为包含 SQL 语法 ANSI/ISO 标准的任何 GBase 8s 扩展，那么 DBANSIWARN 环境变量会导致 SQL 通信区 (SQLCA) 中 **sqlwarn** 数组的第六个字符设置为 W。

有关 SQLCA 的详细信息，请参阅 *GBase 8s ESQL/C 程序员手册*。

在设置 DBANSIWARN 之后，GBase 8s 扩展检查会自动进行，直到您注销或取消设置 DBANSIWARN。要关闭 GBase 8s 扩展检查，可通过以下命令禁用 DBANSIWARN：

```
unsetenv DBANSIWARN
```

4.5.6 DBBLOBBUF 环境变量

使用 DBBLOBBUF 环境变量，可以控制在 UNLOAD 语句处理 TEXT 或 BYTE 值时，这些值是临时存储在内存中还是存储在文件中。DBBLOBBUF 仅影响 UNLOAD 语句。

size 表示 TEXT 或 BYTE 数据的最大大小 (KB)。

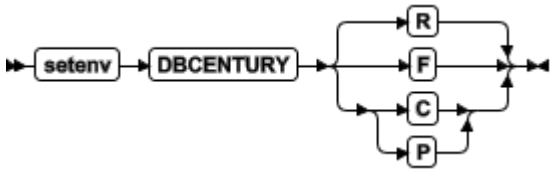
如果 TEXT 或 BYTE 数据大小小于缺省值 10 KB（或 DBBLOBBUF 的设置），那么 TEXT 或 BYTE 值临时存储在内存中。如果数据大小大于缺省值或 DBBLOBBUF 设置，那么会将该数据值写入临时文件。例如，要将缓冲区大小设置为 15 KB，请按以下示例所示设置 DBBLOBBUF：

```
setenv DBBLOBBUF 15
```

此处小于 15 KB 的任何 TEXT 或 BYTE 值都将临时存储在内存中。大于 15 KB 的值临时存储在文件中。

4.5.7 DBCENTURY 环境变量

使用 DBCENTURY 环境变量，可以指定如何展开使用缩写年份值输入的字面 DATE 和 DATETIME 值。为避免展开缩写年份时出现问题，应用程序应要求输入 4 位年份，且应始终将年份显示为 4 位。



当未设置 **DBCENTURY**（或设置为 R）时，当前年份的前两位被用来展开两位年份值。例如，如果今天的日期为 09/30/2003，那么缩写日期 12/31/99 扩展为 12/31/2099，而缩写日期 12/31/00 扩展为 12/31/2000。

R、P、F 和 C 设置确定用于展开两位数年份的算法。

设置	算法
R = 当前®	使用当前年份的前两位展开年份值。
P = 过去	通过对缩写年份值加上前缀 19 和 20 来创建展开的日期。将这两个日期与当前日期进行比较，使用早于当前日期的最新日期。
F = 将来	通过对缩写年份值加上前缀 20 和 21 来创建展开的日期。将这两个日期与当前日期进行比较，使用晚于当前日期的最早日期。
C = 最接近	通过对缩写年份值加上前缀 19、20 和 21 来创建展开的日期。将这三个日期与当前日期进行比较，使用最接近当前日期的日期。

设置是区分大小写的，不会对无效设置发出任何错误。如果输入 f（作为示例），那么缺省 (R) 设置生效。P 和 F 设置不能返回当前日期，因为当前日期既不是过去也不是将来。

为以单个位输入的年份被加上前缀 0，然后展开。不展开三位年份。用前导零填充早于 100 的年份。

展开年份值的示例

本主题中的示例说明 DBCENTURY 的各种设置如何使缩写年份以 DATE 和 DATETIME 值的形式展开。

DBCENTURY = P

示例数据类型：DATE

当前日期：4/6/2003

用户输入：1/1/1

加前缀“19”的展开项：1/1/1901

加前缀“20”的展开项：1/1/2001

分析：两项都早于当前日期，但 1/1/2001 更接近当前日期

当前日期。

要点： DBCENTURY 的结果取决于系统时钟日历的当前日期。因此，如果当前日期为 1/1/2001 且 DBCENTURY = P，那么此示例的缩写日期 1/1/1 将改而扩展成 1/1/1901。

DBCENTURY = F

示例数据类型：DATETIME year to month

当前日期：5/7/2005

用户输入：1-1

加前缀“20”的展开项：2001-1

加前缀“21”的展开项：2101-1

分析：只有日期 2101-1 在当前日期之后，所以选择它。

DBCENTURY = C

示例数据类型：DATE

当前日期：4/6/2000

用户输入：1/1/1

加前缀“19”的展开项：1/1/1901

加前缀“20”的展开项：1/1/2001

加前缀“21”的展开项：1/1/2101

分析：此处 1/1/2001 最接近当前日期，所以选择它。

DBCENTURY = R 或未设置 DBCENTURY

示例数据类型：DATETIME year to month

当前日期：4/6/2000

用户输入：1-1

加前缀“20”的展开项：2001-1

示例数据类型：DATE

当前日期：4/6/2003

用户输入：0/1/1

加前缀“20”的展开项：2000/1

分析：在这两个示例中，都使用加前缀“20”的算法。

当语言环境指定非格列高利历（如希伯莱或伊斯兰历）时，**DBCENTURY** 设置不会影响 GBase 8s 产品。缩写年份时，当前年份的前导位用于备用日历系统。

数据库对象中的缩写年份和表达式

当数据库对象中的表达式(包括检查约束、分段存储表达式、SPL 例程、触发器或 UDR)包含年份为 1 位或 2 位数字的字面日期值或 DATETIME 值时，数据库服务器将使用在创建（或上次修改）数据库对象时 DBCENTURY（和其他相关环境变量）所具有的设置对表达式求值。

如果 DBCENTURY 已重新设置为新值，那么在展开缩写年份时会忽略新值。

例如：假定用户创建了一个表并对名为 **birthdate** 的列定义以下检查约束：

birthdate < '09/25/50'

表达式是根据定义约束时的 DBCENTURY 值解释的。如果包含 **birthdate** 列的表是在 09/23/2000 创建的且 DBCENTURY =C，那么在对 **birthdate** 列执行插入或更新操作时检查约束表达式始终解释为 `birthdate < '09/25/1950'`。即使用户对 birthdate 列执行插入或更新操作时对 **DBCENTURY** 设置了不同的值，约束表达式仍然根据定义（或上次修改）检查约束时的设置进行解释。

在一些较早版本的 GBase 8s 上创建的数据库对象不支持创建时设置的优先级。

对于要获取此功能的旧对象

1. 删除这些对象。
2. 重新创建这些对象(或对于分段存储表达式，拆离这些对象然后重新连接)。

重新定义这些对象之后，对象的表达式内的日期文字将根据创建或上次修改对象时的环境进行解释。否则，它们的行为将取决于运行时环境，如果环境更改，行为有可能变得不一致。

同时包含旧对象和新对象的数据库的管理可能会很复杂，这是因为对日期表达式求值的新旧行为之间存在差异。为避免这一情况，建议重新定义所有旧对象。

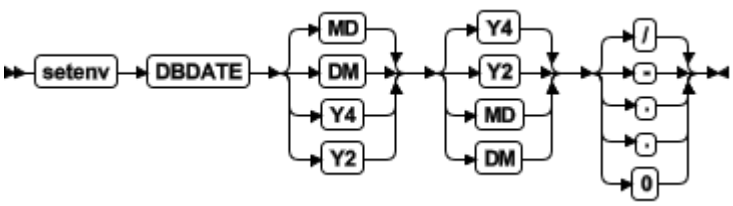
DBCENTURY 的值和当前日期并非确定数据库服务器解释 date 和 DATETIME 值的方式的唯一因子。DBDATE、DBTIME、GL_DATE 和 GL_DATETIME 环境变量也能够影响解释日期的方式。有关 GL_DATE 和 GL_DATETIME 的信息，请参阅《*GBase 8s GLS 用户指南*》。

重要： GBase 8s 的 DBCENTURY 的行为与先前版本不兼容。

4.5.8 DBDATE 环境变量

使用 **DBDATE** 环境变量可以指定 DATE 值的最终用户格式。

在使用 C shell 的 UNIX™ 系统上，使用以下语法设置 **DBDATE**。



下列格式化符号在 **DBDATE** 设置中有效：

-./

是在日期格式中可作为分隔符存在的字符。

0

指示时间单位之间不显示任何分隔符。

D 和 M

是表示日和月的字符。

Y2 和 Y4

是表示年份和年份精度的字符。

一些东亚语言环境支持基于纪元的日期的附加语法。有关基于纪元的格式的详细信息，请参阅《*GBase 8s GLS 用户指南*》。

DBDATE 可指定显示格式的下列属性：

- 日期中时间单位的顺序（月、日和年）
- 年份是以两位数字 (Y2) 还是四位数字 (Y4) 的形式显示
- 月、日和年时间单位之间的分隔符

对于“美国英语”语言环境，**DBDATE** 的缺省值为 MDY4/，其中 M 表示月，D 表示日，Y4 表示四位的年份，而斜杠 (/) 表示时间单位分隔符（例如，01/08/2011）。可充当分隔符的其他有效字符包括连字符 (-)、句点 (.) 或零 (0)。要指示没有分隔符，请使用零。如果尝试指定连字符、句点或零以外的字符作为分隔符，或者未在 **DBDATE** 规范中加入任何分隔符，那么缺省情况下会使用斜杠 (/)。

如果未在客户机上设置 **DBDATE**，那么数据库服务器上的任何 **DBDATE** 设置会覆盖客户机上的 MDY4/ 缺省值。如果在客户机上设置了 **DBDATE**，那么客户机会使用该值（而不是数据库服务器上的设置）。

下表显示了有效 DBDATE 设置的一些示例以及它们对应于日期 2011 年 1 月 8 日的显示：

DBDATE 设置	2011 年 1 月 8 日的表示法：	DBDATE 设置	2011 年 1 月 8 日的表示法：
MDY4/	01/08/2011	Y2DM.	11.08.01
DMY2-	08-01-11	MDY20	010811
MDY4	01/08/2011	Y4MD*	2011/01/08

格式 Y4MD*（因为星号不是有效分隔符）和 MDY4（未定义任何分隔符）都会显示缺省符号（斜杠）作为分隔符。

要点： 如果使用 Y2 格式，那么 DBCENTURY 环境变量的设置还会影响在数据输入时对文字 DATE 值求值的方式。

另外，GBase 8s ESQL/C 调用的某些例程可使用 **DBTIME** 变量（而不是 **DBDATE**）来将 DATETIME 格式设置为国际规范。有关更多信息，请参阅 DBTIME 环境变量和 *GBase 8s ESQL/C 程序员手册* 中对 **DBTIME** 环境变量的论述。

DBDATE 变量的设置优先于 **GL_DATE** 环境变量的设置，并且优先于 **CLIENT_LOCALE** 指定的任何缺省 DATE 格式。有关 **GL_DATE** 和 **CLIENT_LOCALE** 的信息，请参阅《*GBase 8s GLS 用户指南*》。

最终用户格式影响下列上下文：

- 在显示 DATE 值时，GBase 8s 产品使用 DBDATE 环境变量来确定输出的格式。
- 在 DATE 值的数据输入期间，GBase 8s 产品使用 DBDATE 环境变量来解释输入。

例如，如果在 INSERT 语句中指定文字 DATE 值，那么数据库服务器期望此文字值与 **DBDATE** 指定的格式兼容。同样，数据库服务器将您指定为 **DATE( )** 函数的自变量的日期解释为 **DBDATE** 格式。

数据库对象中的 DATE 表达式

当数据库对象中的表达式(包括检查约束、分段存储表达式、SPL 例程、触发器或 UDR)包含字面日期值时，数据库服务器使用在创建（或上次修改）数据库对象时 DBDATE（或其他相关环境变量）所具有的设置对表达式求值。如果 DBDATE 已复位为新值，那么对文字 DATE 求值时会忽略该新值。

例如：假如 DBDATE 设置为 MDY2/，用户创建了一个表且 orderdate 列具有以下检查约束：

```
orderdate < '06/25/98'
```

则先前表达式的日期根据定义约束时 **DBDATE** 的值来进行格式编排。检查约束表达式解释为 `orderdate < '06/25/98'`，不管对 **orderdate** 列执行插入或更新操作期间 **DBDATE** 的值如何。假设当用户将值 `'30/01/98'` 插入 **orderdate** 列中时，**DBDATE** 复位为 DMY2/。插入的日期值使用日期格式 DMY2/，而检查约束表达式使用日期格式 MDY2/。

有关较早版本的 GBase 8s 中始终根据运行时环境求值的旧对象的讨论，请参阅数据库对象中的缩写年份和表达式。该节描述如何重新定义对象，以便根据定义对象（或上次修改对象）时生效的环境变量设置解释日期。

要点： GBase 8s 的 DBDATE 的行为与先前版本不兼容。

4.5.9 DBDELIMITER 环境变量

设置 DBDELIMITER 环境变量，可以指定与 dbexport 实用程序和 LOAD 及 UNLOAD 语句配合使用的字段定界符。

delimiter 是未装入的数据文件的字段定界符。

delimiter 可以是任何单个字符，但以下列表中的字符除外：

- 十六进制数字（0 至 9，a 至 f，A 至 F）
- 换行符或 CTRL-J
- 反斜杠 (\) 符号

竖线（ | = ASCII 124）是缺省值。例如，要将字段定界符更改为加号（ + ），可按如下方式设置 DBDELIMITER：

```
setenv DBDELIMITER '+'
```

4.5.10 DBEDIT 环境变量

使用 DBEDIT 环境变量，可以指定要与 DB-Access 中的 SQL 语句和命令文件配合使用的文本编辑器。

如果设置了 DBEDIT，那么指定文本编辑器是自动调用的。如果未设置 DBEDIT，系统将提示您指定一个文本编辑器作为会话剩余部分的缺省编辑器。

editor 是您想要使用的文本编辑器的名称。

对于大多数 UNIX™ 系统，缺省文本编辑器为 **vi**。如果使用另一文本编辑器，请确保它创建的是平面 ASCII 文件。某些*文档方式*的字处理程序引入的打印机控制字符可能会干扰 GBase 8s 产品的运行。

要指定 EMACS 文本编辑器，请通过以下命令设置 DBEDIT：

```
setenv DBEDIT emacs
```

4.5.11 DBFLTMASK 环境变量

DB-Access 实用程序在 14 字符缓冲区内显示数据类型为 FLOAT、SMALLFLOAT 和 DECIMAL(*p*) 的浮点值。缺省情况下，DB-Access 显示小数点右边尽可能多的位数（在此字符缓冲区能容纳的情况下）。因此，DB-Access 显示的实际小数位数取决于浮点值的大小。

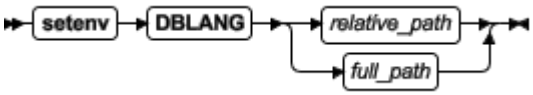
要降低浮点值中小数点右边显示的位数，请将 DBFLTMASK 设置为指定的位数。

scale 是想要 GBase 8s 客户机应用程序在浮点值中显示的小数位数。此处 *scale* 必须小于 16（显示的缺省位数）。

如果浮点值包含的小数点右边位数超过 **DBFLTMASK** 指定的位数，DB-Access 会将该值舍入为指定位数。如果浮点值包含的小数点右边位数少于指定位数，那么 DB-Access 会使用零来填充该值。但是，如果将 **DBFLTMASK** 设置为 14 个字符的缓冲区不能容纳的值，DB-Access 会将该值舍入为可容纳的位数。

4.5.12 DBLANG 环境变量

使用 DBLANG 环境变量，可以指定 $GBASEDBTDIR 的子目录或 GBase 8s 产品使用的已编译消息文件所在目录的完整路径名。



relative_path 是 **$GBASEDBTDIR** 的子目录。

full_path 是已编译消息文件的路径名。

缺省情况下，GBase 8s 产品将已编译消息放在 **$GBASEDBTDIR/msg** 目录的特定于语言环境的子目录中。这些已编译消息文件具有文件扩展名 **.iem**。如果想要使用不同于 **$GBASEDBTDIR/msg** 的消息目录，以便可在其中存储创建的消息文件，必须执行下列步骤：

要使用不同于 $GBASEDBTDIR/msg 的消息目录

1.使用 **mkdir** 命令来为消息文件创建适当的目录。

可将此目录放在 **$GBASEDBTDIR** 或 **$GBASEDBTDIR/msg** 目录下，也可以将它放在任何其他目录下。

2.将新目录的所有者和组设置为 **gbasedbt**，并将此目录的存取许可权设置为 755。

3.将 **DBLANG** 环境变量设置为新目录。如果这是 **$GBASEDBTDIR** 或 **$GBASEDBTDIR/msg** 的子目录，那么您只需要列示新目录的相对路径。否则，必须指定该目录的全路径名。

4.将 **.iem** 文件或您创建的消息文件复制至 **$DBLANG** 指定的新消息目录。

消息目录中的所有文件应具有所有者和组 **gbasedbt** 以及存取许可权 644。

使用缺省美国英语语言环境的 GBase 8s 产品按以下顺序搜索消息文件：

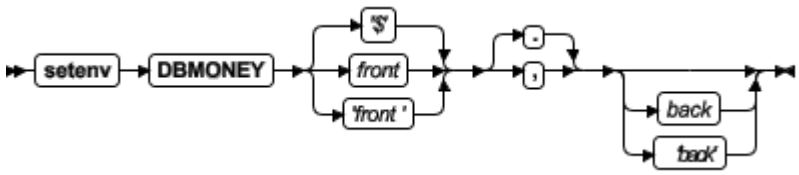
1. 在 $DBLANG 中，如果 DBLANG 设置为全路径名的话
2. 在 $GBASEDBTDIR/msg/$DBLANG 中，如果 DBLANG 设置为相对路径名的话
3. 在 $GBASEDBTDIR/$DBLANG 中，如果 DBLANG 设置为相对路径名的话
4. 在 $GBASEDBTDIR/msg/en_us/0333 中
5. 在 $GBASEDBTDIR/msg/en_us.8859-1 中

6. 在 $GBASEDBTDIR/msg 中
7. 在 $GBASEDBTDIR/msg/english 中

有关消息的搜索路径的更多信息，请参阅《*GBase 8s GLS 用户指南*》中对 **DBLANG** 的描述。

4.5.13 DBMONEY 环境变量

使用 DBMONEY 环境变量，可以指定 smallfloat、FLOAT、DECIMAL 或 MONEY 数据类型列中值的显示格式，以及从这些数据类型中的任何一种派生出来的复杂数据类型列中值的显示格式。



$ 是缺省语言环境中 MONEY 值之前的货币符号（如果未指定任何其他 *front* 符号或未设置 **DBMONEY**）。

, *或* **.** 是一个逗号或句点（缺省值），它将 FLOAT、DECIMAL 或 MONEY 值的整数部分与小数部分隔开。您未指定的符号成为千位分隔符。

back 是跟在 MONEY 值后面的货币符号。

front 是 MONEY 值之前的货币符号。

back 符号最多可有 7 个字符，可包含语言环境支持的任何字符，但数字、逗号 (**,**) 或句点 (**.**) 符号除外。*front* 符号最多可有 7 个字符，可包含语言环境支持的任何字符，但数字、逗号 (,) 或句点 (.) 符号除外。如果对 *front* 或 *back* 指定字母表中的字母之外的任何字符，必须用单引号（' ）将 *front* 或 *back* 设置括起来。

在显示 MONEY 值时，GBase 8s 产品使用 **DBMONEY** 设置来确定输出的格式。但是，**DBMONEY** 对存储在数据库中各列的数据值的内部格式不起任何作用。

如果未设置 **DBMONEY**，那么缺省语言环境（美国英语）的 MONEY 值的格式将是在 MONEY 值前加上美元符号 (**$**)，用句点 (**.**) 将 MONEY 值的整数部分与小数部分隔开，且没有 *back* 符号。例如：100.50 的格式为 $100.50。

假定您想要将 MONEY 值显示为 DM（德国马克）单位，使用 货币符号 DM 和逗号 (,) 作为小数分隔符。输入以下命令以设置 DBMONEY 环境变量：

```
setenv DBMONEY DM,
```

此处 DM 是 MONEY 值之前的 *front* 货币符号，而逗号将 MONEY 值的整数部分与小数部分隔开。因此，值 100.50 显示为 DM100,50。

有关 **DBMONEY** 在非缺省语言环境中如何确定 MONEY 值的格式的更多信息，请参阅《*GBase 8s GLS 用户指南*》。

4.5.14 DBONPLOAD 环境变量

使用 DBONPLOAD 环境变量,可以指定 High Performance Loader (HPL) 的 onpload 实用程序使用的数据库的名称。

如果设置了 DBONPLOAD，那么 onpload 将指定名称用作数据库的名称；否则数据库的缺省名称为 onpload。

dbname 指定 onpload 实用程序使用的数据库的名称。

例如：要将名称 load_db 指定为数据库的名称，输入以下命令：

setenv DBONPLOAD load_db

有关更多信息，请参阅 *GBase 8s High-Performance Loader 用户指南*。

4.5.15 DBPATH 环境变量

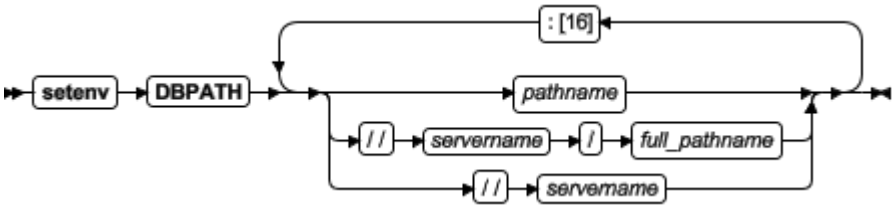
使用 DBPATH 环境变量可以标识包含数据库的数据库服务器。DBPATH 还可指定 DB-Access 在其中查找命令脚本（.sql 文件）的目录（除当前目录之外）的列表。

CONNECT DATABASE、START DATABASE 和 DROP DATABASE 语句使用 DBPATH 在以下两种情况下查找该数据库：

- 如果数据库的位置未显式声明
- 如果不能在缺省服务器中定位该数据库

CREATE DATABASE 语句不使用 **DBPATH**。

要向现有条目添加新 DBPATH 条目，请参阅修改环境变量设置。



full_pathname 是从**根目录**开始的，存储 **.sql** 文件的目录的全路径。

pathname 是存储 **.sql** 文件的目录的有效相对路径。

servername

是存储数据库的 GBase 8s 服务器的名称。不能使用 *servername* 引用数据库文件。

DBPATH 可包含最多 16 个条目。每个条目必须少于 128 个字符。此外，**DBPATH** 的最大长度取决于在其上设置 **DBPATH** 的硬件平台。

在使用 CONNECT DATABASE、START DATABASE 或 DROP DATABASE 语句访问数据库时，首先在该语句中指定的目录或数据库服务器中搜索该数据库。如果未指定任何数据库服务器，那么使用 **GBASEDBTSERVER** 环境变量指定的缺省数据库服务器。

如果在初次搜索期间找不到该数据库，同时又设置了 **DBPATH**，那么会在指定数据库中搜索 **DBPATH** 中的数据库服务器和目录。这些条目是按它们列示在 **DBPATH** 设置中的顺序搜索的。

将 DBPATH 与 DB-Access 配合使用

如果在尚未选择数据库的情况下使用 DB-Access 并从 **SQL** 菜单选择中**选择**选项，就会看到 **DBPATH** 中列出的目录中所有 **.sql** 文件的列表。选择了数据库之后，就不会使用 **DBPATH** 来查找 **.sql** 文件。仅显示当前工作目录中的 **.sql** 文件。

搜索本地目录

使用不带数据库服务器名的路径名搜索本地计算机上的 .sql 脚本。在以下示例中，DBPATH 设置让 DB-Access 先在当前目录中搜索这些数据库文件，然后在本地计算机上的 Joachim 和 Sonja 目录中搜索它们：

```
setenv DBPATH /usr/joachim:/usr/sonja
```

如先前示例所示，如果该路径名指定目录名但未指定数据库服务器名，那么在运行 GBASEDBTSERVER 指定的缺省数据库服务器的计算机上搜索该目录；请参阅 GBASEDBTSERVER 环境变量。例如，如以下示例所示，对于先前示例，如果将 GBASEDBTSERVER 设置为 quality，DBPATH 值已作了*解释*，其中双斜杠放在数据库服务器名称之前：

```
setenv DBPATH //quality/usr/joachim://quality/usr/sonja
```

搜索联网计算机以查找数据库

如果使用多个数据库服务器，可将 DBPATH 显式设置为包含想要对其搜索数据库的数据库服务器和目录名。例如，如果将 GBASEDBTSERVER 设置为 quality，但您还想搜索 marketing 数据库服务器以查找 /usr/joachim，那么按以下示例所示设置 DBPATH：

```
setenv DBPATH //marketing/usr/joachim:/usr/sonja
```

指定服务器名

可将 **DBPATH** 设置为仅包含数据库服务器名。此功能允许您仅定位数据库；不能使用它来定位命令文件。

数据库管理员必须将 **DBPATH** 提及的每个数据库服务器包括在 **$GBASEDBTDIR/etc/sqlhosts** 文件中。有关通信配置文件和数据库服务器名称的信息，请参阅《*GBase 8s 管理员指南*》和 *GBase 8s 管理员参考*。

例如：如果将 GBASEDBTSERVER 设置为 quality，那么可以通过设置 DBPATH 来先在 quality 数据库服务器上搜索数据库，然后在 marketing 数据库服务器上搜索数据库，如以下示例所示：

```
setenv DBPATH //marketing
```

如果在此示例中使用 DB-Access，那么使用 DATABASE 菜单的**选择**选项来显示 **quality** 和 **marketing** 数据库服务器上所有数据库的名称。

4.5.16 DBPRINT 环境变量

使用 DBPRINT 环境变量可以指定缺省打印程序。

program 任何生成标准 ASCII 输出的命令、shell 脚本或 UNIX™ 实用程序。

如果未设置 DBPRINT，那么缺省 *program* 可在下列两个位置之一中找到：

- 对于大多数 BSD UNIX 系统，缺省程序为 **lpr**。
- 对于 UNIX 系统 V，缺省程序通常为 **lp**。

输入以下命令来设置 DBPRINT 环境变量以将 **myprint** 指定为打印程序：

```
setenv DBPRINT myprint
```

4.5.17 DBREMOTECMD 环境变量 (UNIX)

使用 DBREMOTECMD 环境变量，可以覆盖缺省远程 shell 以使用数据库服务器执行远程磁带操作。

可将 DBREMOTECMD 设置为简单命令或设置为完整路径名。

command 覆盖缺省远程 shell 的命令。

pathname 覆盖缺省远程 shell 的路径名。

如果不指定完整路径名，数据库服务器会搜索您的 PATH 以获取指定的 *command*。您应在交互式 UNIX™ 平台上使用完整路径名语法，以避免其他目录中类似命名的程序带来的问题以及与*受限 shell* (/usr/bin/rsh) 之间可能产生的混淆。

以下命令对简单命令名设置 DBREMOTECMD：

```
setenv DBREMOTECMD rcmd
```

用于设置 DBREMOTECMD 的下一个命令会指定完整路径名：

```
setenv DBREMOTECMD /usr/bin/remsh
```

有关使用远程磁带设备进行备份的更多信息，请参阅指定远程设备。

4.5.18 DBSPACETEMP 环境变量

DBSPACETEMP 环境变量指定在其中构建临时表的数据库空间。该列表可能包含标准数据库空间和/或临时数据库空间。

dbspace 是现有的标准或临时数据库空间的名称。

您可以列出数据库空间（用冒号（ : ）或逗号（ , ）符号来分隔），以便为跨物理存储设备的临时表指定空间。例如：用来设置 DBSPACETEMP 环境变量的以下命令为临时表指定三个数据库空间：

setenv DBSPACETEMP sorttmp1:sorttmp2:sorttmp3

DBSPACETEMP 覆盖 DBSPACETEMP 参数在数据库服务器配置文件中指定的任何缺省数据库空间。对于 UPDATE STATISTICS 操作，只有在指定 HIGH 关键字选项时才使用 DBSPACETEMP。

在 UNIX™ 平台上，如果 **DBSPACETEMP** 中的数据库空间列表是由作为原始设备来分配的块所组成的，那么就可能获得更好的性能。

就象操作系统定义的那样，环境变量的最大大小限制数据库空间的数目。如果该环境变量指定的数据库空间不存在，那么数据库服务器不会创建该数据库空间。

两类临时表包括用户创建的*显式*临时表和数据库服务器创建的*隐式*临时表。使用 **DBSPACETEMP** 为这两种类型的临时表指定数据库空间。

如果使用 CREATE TEMP TABLE 语句创建显式临时表，且不在 IN *dbspace* 子句或 FRAGMENT BY 子句中为该表指定数据库空间，那么数据库服务器使用 **DBSPACETEMP** 中的设置来确定在何处创建该表。

如果使用 SELECT INTO TEMP 语句创建显式临时表，那么数据库服务器使用 **DBSPACETEMP** 中的设置来确定在何处创建该表。

如果设置了 **DBSPACETEMP**，并且它列出的数据库空间包括日志记录和非日志记录数据库空间，那么数据库服务器会把临时表（该表隐式或显式支持事务日志记录）存储在日志记录数据库空间中，并把非日志记录临时表存储在非日志记录数据库空间中。

数据库服务器在执行连接操作、带有 GROUP BY 子句的 SELECT 语句、带有 ORDER BY 子句的 SELECT 语句和索引构建时会创建隐式临时表以供其自身使用。

当数据库服务器创建显式或隐式临时表时，它会使用磁盘空间来写临时数据。如果临时表位置的设置或语句规范之间有冲突，那么以降序（从最高到最低）的优先顺序解决这些冲突：

1. 在 UNIX 平台上，环境变量 PSORT_DBTEMP（如果设置了该环境变量的话）指定的一个或多个操作系统目录
2. 环境变量 DBSPACETEMP 指定的一个或多个数据库空间（如果设置了该环境变量的话）
3. ONCONFIG 参数 DBSPACETEMP 指定的数据库空间。
4. DUMPDIR 配置参数指定的操作系统文件空间
5. 目录 $GBASEDBTDIR/tmp (UNIX)。

要点： 如果将 DBSPACETEMP 环境变量设置为无效值，那么对于显式临时表，数据库服务器缺省值为根数据库空间，而对于隐式临时表，缺省值为 /tmp，而不是 DBSPACETEMP

配置参数的设置。在此情况下，数据库服务器可能会填充 /tmp 直至达到限制，最终使得数据库服务器关闭或杀死文件系统。

4.5.19 DBTEMP 环境变量

DBTEMP 环境变量由 DB-Access、GBase 8s Enterprise Gateway 产品使用。DBTEMP 与 DBSPACETEMP 类似，它指定放置临时文件和临时表的目录。

pathname 临时文件和表的目录的完整路径名。

对于使用 C shell 的 UNIX™ 系统，以下示例将 DBTEMP 设置为路径名 usr/magda/mytemp：

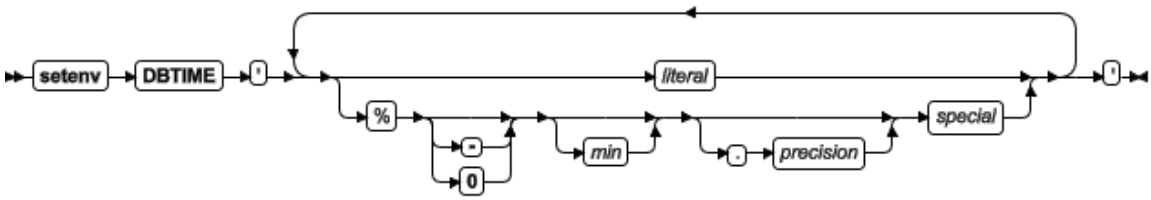
```
setenv DBTEMP usr/magda/mytemp
```

如果未设置 DBTEMP，那么数据库服务器会在 /tmp 目录中创建临时文件，而在 DBSPACETEMP 目录中创建临时表。如果未设置 DBSPACETEMP，请参阅 DBSPACETEMP 环境变量以获取缺省值。同样，如果未在客户机系统上设置 DBTEMP，那么会在 /tmp 目录中创建临时文件（例如：为滚动游标创建的那些临时文件）。

如果未设置 DBTEMP，那么在对大的或复杂的数据类型（如 BYTE 或 ROW）的值执行操作时可能会遇到意外行为或故障。

4.5.20 DBTIME 环境变量

DBTIME 环境变量对 DATETIME 值的显示和数据输入格式指定格式编排掩码。 只有在 DBTIME 要格式化的 DATETIME 数据值具有与指定的 DBTIME 设置相同精度的情况下，DBTIME 环境变量才有用。对于用不同 DATETIME 限定符声明的 DATETIME 值，您可能会遇到意外或无效的显示格式。



literal 是文字空格或任何可打印字符。

Min 是一个文字整数，设置 *special* 指定的值的子串中的最小字符数。

Precision 是任何时间单位的值的位数或月份名中的最大字符数。

Special 为以下列示的占位符之一。

这些术语和符号在接下来的几页中作了描述。

这个引号括起来的字符串可以包括字面的字符以及 DATETIME 值的个别时间单位和其他元素值的占位符。**DBTIME** 仅在您调用某些 GBase 8s ESQL/C DATETIME 例程时起

作用。（有关详细信息，请参阅 *GBase 8s ESQL/C 程序员手册*。）如果未设置 **DBTIME**，那么表示未定义这些例程的行为，且“YYYY-MM-DD hh:mm:ss.fffff”是缺省语言环境中的 DATETIME YEAR TO FRACTION(5) 文字值的缺省显示和输入格式。

百分比 (%) 符号给予后跟的 *special* 占位符很特殊的意义。如果没有前导 % 符号，格式化掩码内的任何字符都被解释为文字字符，即使它与以下列表中的占位符字符之一是相同字符。此外还需注意 *special* 占位符符号是区分大小写的。

DBTIME 格式字符串内的下列字符是 DATETIME 值内的时间单位（或其他功能）的占位符。

%b

由缩写月份名替换。

%B

由完整月份名替换。

%d

由十进制数形式的月份中的一天 [01,31] 替换。

%F*n*

由带有整数 *n* 指定的小数位的秒的小数替换。*n* 的缺省值为 2；*n* 的范围为 0 ≤ *n* ≤ 5。

%H

由小时（24 小时时钟）替换。

%I

由小时（12 小时时钟）替换。

%M

由十进制数形式的分钟 [00,59] 替换。

%m

由十进制数形式的月份 [01,12] 替换。

%p

由 A.M. 或 P.M.（或语言环境文件中的等效值）替换。

%S

由十进制数形式的秒 [00,59] 替换。

%y

由四位十进制数的年份替换。

%Y

由四位十进制数的年份替换。用户必须输入四位值。

%%

替换为 %（以允许格式字符串中出现 %）。

例如，考虑 DATETIME YEAR TO SECOND 的以下显示格式：

Mar 21, 2001 at 16 h 30 m 28 s

如果用户输入两位年份值，那么此值根据 **DBCENTURY** 环境变量设置展开为 4 位。如果未设置 **DBCENTURY**，那么在缺省情况下使用字符串 19 作为头两位。

按以下命令行（用于 C shell）所示设置 DBTIME：

setenv DBTIME '%b %d, %Y at %H h %M m %S s'

缺省 DBTIME 生成以下 ANSI SQL 字符串格式：

2001-03-21 16:30:28

可按以下示例所示设置缺省 DBTIME：

setenv DBTIME '%Y-%m-%d %H:%M:%S'

可选字段宽度和精度规范 (*w.p*) 可直接跟在百分比 (%) 字符之后。它解释为如下所示：

w

指定最小字段宽度。该值是右对齐的，左边填充空格。

-w

指定最小字段宽度。该值是左对齐的，右边填充空格。

0*w*

指定最小字段宽度。该值是靠右对齐的，并在左侧使用零填充。

p

指定d、H、I、m、M、S、y 和 Y 时间单位值的精度或 b 和 B 月份名中的最大字符数。

下列限制适用于字段宽度和精度规范：

如果数据值提供的位数少于 *precision* 指定的位数，那么使用前导零填充该值。

如果数据值提供的字符数超过 *precision* 指定的字符数，那么从右边截断超出的字符。

如果没有为 d、H、I、m、M、S 或 y 占位符指定字段宽度或精度，那么缺省值为 0.2 或 0.4（对于 Y 占位符）。

仅当将 DATETIME 值转换为 ASCII 字符串时，*precision* 规范才有意义，但反过来却不是这样。

F 占位符不支持此文件宽度和精度语法。

与 **DBDATE**、**GL_DATE** 或 **GL_DATETIME** 一样，**DBTIME** 设置仅控制数据值的字符串表示；它不能更改 DATETIME 列的内部存储格式。（有关确定 DATE 值的格式的信息，请参阅 DBDATE 环境变量页中对 **DBDATE** 的讨论。）

在支持基于纪元的日期的东亚语言环境中，**DBTIME** 还可指定日本或台湾纪元。请参阅《*GBase 8s GLS 用户指南*》以获取有关设置 **DBTIME** 以显示基于纪元的 DATETIME 值的更多占位符符号的详细信息以及对 **GL_DATETIME** 和 **GL_DATE** 环境变量的描述。

4.5.21 DBUPSPACE 环境变量

使用 DBUPSPACE 环境变量，可以指定和限制 UPDATE STATISTICS 语句在尝试同时构造多列分布时可使用的系统磁盘空间量。

max 是一个正整数，指定要为 UPDATE STATISTICS 操作中排序而分配的最大磁盘空间 (KB)。

default 为一个正整数，指定不使用 PDQ 时要分配的最大内存数量 (4-50 MB)。

选项

无符号整数：

1：不使用任何索引来排序。在 sqexplain.out 中打印更新统计信息的整个计划。

2：不使用任何索引来排序。不打印更新统计信息的计划。

3 以上：使用可用的索引来排序。在说明输出文件中打印更新统计信息的整个计划。

例如，要将 DBUPSPACE 设置为 2,500 KB 的磁盘空间和 1 兆字节的内存，请输入以下命令：

setenv DBUPSPACE 2500:1

在设置此值后，数据库服务器可在执行 UPDATE STATISTICS 语句期间使用不超过 2,500 KB 的磁盘空间。如果表需要 5 兆字节的磁盘空间用于排序，那么 UPDATE STATISTICS 分两次完成该任务；每次构造一半列的分布。

如果未设置 **DBUPSPACE**，那么对于 *max*，缺省值为 1 兆字节 (1,024 KB)，而对于 *default*，缺省值为 15 兆字节。如果尝试将 **DBUPSPACE** 设置为小于 1,024 KB 的任何值，它会自动设置为 1,024 KB，但不会返回任何错误消息。如果此值尚未大到足以允许一次构造多个分布，那么至少会完成一个分布，即使完成此任务所需的磁盘空间量超过 **DBUPSPACE** 指定的磁盘空间量也是如此。

4.5.22 DEFAULT_ATTACH 环境变量

如果 **DEFAULT_ATTACH** 环境变量设置为 1，那么缺省情况下非分段表上的非分段 B 型树索引页面会存储在存储表数据页面的同一分区（和同一数据库空间）中。CREATE INDEX 语句的 IN TABLE 关键字不是必需的（但也不会返回错误）。

但是，对于索引页面始终存储在不同于索引表数据页面的单独分区中的任何其他索引，将 **DEFAULT_ATTACH** 设置为 1 将对这些索引没有任何作用。存储分发始终与表的存储分发不同的这些索引类型包括

- R 型树索引，
- 函数索引，
- 森林树索引，
- 分段索引，
- 以及分段表上的索引。

仅非分段表上的非分段 B 型树索引支持将索引存储数据页面所在的同一分区中。

如果未设置 **DEFAULT_ATTACH**，那么缺省情况下，未指定 IN TABLE 作为其 Storage Options 子句的任何 CREATE INDEX 语句都会创建将页面存储在不同于数据页面的分区的索引。

要点： GBase 8s 将来的发行版可能不继续支持 DEFAULT_ATTACH。建议不要开发依赖于该不推荐使用的功能的新应用程序。

4.5.23 DELIMIDENT 环境变量

DELIMIDENT 环境变量指定括在双引号（ ″ ）之间的字符串是定界数据库标识。

DELIMIDENT 环境变量在客户机系统上也是受支持的，在客户机系统中它可以设置为 y、n 或不设置。

y 指定客户机应用程序必须使用单引号 (') 来对字符串定界，且必须仅在定界的 SQL 标识（它可以支持比未定界的标识中有效的字符集更大的字符集）的两边使用双引号 (")。定界字符串或定界标识中的字母是区分大小写的。这是 OLE DB 和 .NET 的缺省值。

n 指定客户机应用程序可以使用双引号 (") 或单引号 (') 来对字符串定界，但是不对 SQL 标识定界。如果数据库服务器在需要 SQL 标识的上下文中遇到用双引号或单引号定界的字符串，将发出错误。符合 SQL 标识条件的所有者名称可以用单引号 (') 来定界。 必须使用一对相同的引号来对字符串定界。

这是 ESQL/C、JDBC 和 ODBC 的缺省值。具有 ESQL/C 作为底层的 API（如 GBase 8s 4GL、DataBlade® API (LIBDMI) 和 C++ API）具有与 ESQL/C 相同的行为，并使用“n”作为缺省值（如果在客户机系统上没有指定 DELIMIDENT 的值）。

在客户机系统上指定没有值的 DELIMIDENT 环境变量要求客户机应用程序使用 DELIMIDENT 设置，这是应用程序编程接口 (API) 的缺省设置。

不需要任何值；DELIMIDENT 生效（如果它存在的话），且当它在环境变量列表上时仍然有效。当 DELIMIDENT 在服务器级别设置时，除去此变量需要重新启动服务器。

定界标识可以包含空格（如短语 "**Vitamin E**"），也可以与 SQL 关键字完全相同（如 "**TABLE**" 或 "**USAGE**"）。还可使用定界标识来声明包含 SQL 标识的缺省字符集之外的字符的数据库标识（如“**Column #6**”）。在缺省语言环境中，此缺省字符集由字母、数字和下划线 (_) 符号组成。

即使设置了 DELIMIDENT，您也可以使用单引号（ ’ ）来将权限标识定界为数据库对象名称的所有者名称部分，如以下示例所示：

```
RENAME COLUMN 'Owner'.table2.collum3 TO column3;
```

此示例是一般规则的例外，一般规则是这样的：当设置了 DELIMIDENT 时，SQL 解析器将以单引号定界的字符串解释为文字串，而将以双引号（ ″ ）定界的字符串解释为 SQL 标识。

数据库标识符（也称为 *SQL 标识符*）是数据库对象（例如：表和列）的名称。*存储器标识*是存储器对象（例如，数据库空间、Blob 空间和智能大对象空间）的名称。不能使用 DELIMIDENT 来声明包含缺省 SQL 字符集以外字符的存储器标识。

定界标识是区分大小写的。要使用定界标识，用 GBase 8s ESQL/C 编写的应用程序必须在编译时和运行时设置 DELIMIDENT。

重要： 如果尚未设置 DELIMIDENT，应注意对它进行设置可能会导致在定界 SQL 标识（例如，字符串文字的定界符）之外的上下文中使用双引号（ ″ ）的现有 .sql 脚本或客户机应用程序出错。如果设置了 DELIMIDENT，那么必须对不是 SQL 标识的定界构造使用单引号（ ’ ）而不是双引号。

在使用 C shell 并已设置了 DELIMIDENT 的 UNIX™ 系统上，可通过以下命令禁用此功能（它会导致双引号之间的任何内容都解释为 SQL 标识）：

```
unsetenv  DELIMIDENT
```

4.5.24 ENVIGNORE 环境变量 (UNIX™)

ENVIGNORE 环境变量可取消激活公共（共享）配置文件 gbasedbt.rc 和专用环境配置文件 .gbasedbt 中的指定环境变量设置。

variable 要取消激活的环境变量的名称。

在连续 *variable* 名称之间使用冒号（ : ）。例如：要忽略环境配置文件中的 DBPATH 和 DBMONEY 条目，输入以下命令：

```
setenv ENVIGNORE DBPATH:DBMONEY
```

公共环境配置文件存储在 $GBASEDBTDIR/etc/gbasedbt.rc 中。

专用环境配置文件以 .gbasedbt 的形式存储在用户的主目录中。

有关创建或修改环境配置文件的信息，请参阅在配置文件中设置环境变量。

ENVIGNORE 本身不能在环境配置文件中进行设置。

4.5.25 FET_BUF_SIZE 环境变量

FET_BUF_SIZE 环境变量可覆盖所有数据类型的访存缓冲区的大小的缺省设置，但 BYTE 和 TEXT 值除外。 对于 ANSI 数据库，您必须针对 FET_BUF_SIZE 环境变量将事务设置为 READ ONLY 来提高性能，否则会逐一返回行。

size 是一个正整数，它大于缺省缓冲区大小，但不超过 2147483648 (2 GB)，用于指定访存缓冲区的大小（以字节计）来保存查询检索的数据。

例如，要在使用 C shell 的 UNIX™ 系统上将缓冲区大小设置为 5,000 字节，请输入以下命令来设置 FET_BUF_SIZE：

setenv FET_BUF_SIZE 5000

在将 FET_BUF_SIZE 设置为有效值时，新值将覆盖缺省值（或先前对 FET_BUF_SIZE 设置的任何值）。访存缓冲区的缺省设置取决于行大小。

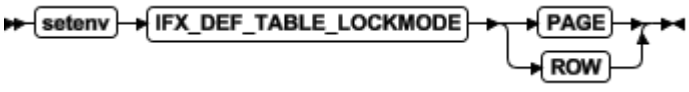
BYTE 和 TEXT 值的处理不会受 FET_BUF_SIZE 的影响。

如果将 FET_BUF_SIZE 设置为小于缺省大小的值或大于 2147483648 (2 GB) 的值，不会发生任何错误。但是，在这些情况下，无效访存缓冲区大小将被忽略，而缺省大小将生效。

有效的 FET_BUF_SIZE 设置不仅对本地数据库服务器有效，而且对从中通过分布式查询（其中，本地服务器作为协调程序而远程数据库作为下级）检索行的任何远程数据库服务器也有效。缓冲区的大小越大，可能返回的行越多，于是客户机应用程序必须等待数据库服务器返回行的时间也就越长。大型缓冲区可通过降低填充客户机端缓冲区的开销来改进性能。

4.5.26 IFX_DEF_TABLE_LOCKMODE 环境变量

IFX_DEF_TABLE_LOCKMODE 环境变量可在未显式指定 LOCKMODE PAGE 或 LOCKMODE ROW 关键字的情况下对后续创建的数据库表指定缺省锁定方式。 如果必须创建同一锁定方式的几个表，此功能是非常方便的。使用 C shell 的 UNIX™ 系统支持以下语法：



PAGE 缺省锁定方式是页级粒度的。该值禁用 COMMITTED READ 的 LAST COMMITTED 功能。

ROW 缺省锁定方式是行级别粒度的。

可通过将 ONCONFIG 文件的 DEF_TABLE_LOCKMODE 参数设置为 PAGE 或 ROW 来获取类似功能。 创建或修改表时，会根据以下优先级的降序（最高至最低）来解析所有冲突的锁定方式规范。

1. CREATE TABLE 或 ALTER TABLE 的显式 LOCKMODE 指定
2. IFX_DEF_TABLE_LOCKMODE 环境变量设置
3. ONCONFIG 文件中的 DEF_TABLE_LOCKMODE 参数设置
4. 系统缺省所订方式（= 页方式）

要将 DEF_TABLE_LOCKMODE 设置为缺省方式（或者要在未设置 DEF_TABLE_LOCKMODE 的情况下复原系统缺省值），请使用以下命令：

unsetenv IFX_DEF_TABLE_LOCKMODE

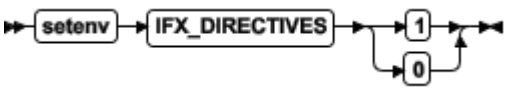
如果在运行 oninit 之前在数据库服务器的环境中设置了 IFX_DEF_TABLE_LOCKMODE，那么它的作用域是数据库服务器的所有会话（就如同在 ONCONFIG 文件中设置了 DEF_TABLE_LOCKMODE 一样）。如果在 shell 又或者在 $HOME/.gbasedbt 或 $GBASEDBTDIR/etc/gbasedbt.rc 文件中设置了 IFX_DEF_TABLE_LOCKMODE，那么作用域被限制为当前会话（如果在 shell 中设置了它的话）或个别用户。

要点： 这对现有表没有任何影响。如果指定 *ROW* 作为锁定方式，那么数据库将使用它来复原、恢复或复制数据。对于以 *PAGE* 方式创建的表，这可能会导致锁定表溢出或性能下降。

4.5.27 IFX_DIRECTIVES 环境变量

IFX_DIRECTIVES 环境变量设置确定优化器是否允许来自查询内的查询优化伪指令。**IFX_DIRECTIVES** 环境变量是在客户机上设置的。

可指定 ON 和 OFF 或 1 和 0 来设置环境变量。



1 接受优化器伪指令

0 不接受优化器伪指令

IFX_DIRECTIVES 环境变量的设置会覆盖为数据库服务器设置的 DIRECTIVES 配置参数的值。但是，如果未设置 **IFX_DIRECTIVES** 环境变量，那么所有客户机会话将继承 ONCONFIG 参数 DIRECTIVES 确定的伪指令的数据库服务器配置。

IFX_DIRECTIVES 环境变量的缺省设置为 ON。

有关 DIRECTIVES 参数的更多信息，请参阅《*GBase 8s 管理员参考*》。有关伪指令对性能影响的更多信息，请参阅《*GBase 8s 性能指南*》。

4.5.28 IFX_EXTDIRECTIVES 环境变量

IFX_EXTDIRECTIVES 环境变量指定查询优化器是否允许来自 **sysdirectives** 系统目录表的外部查询优化伪指令应用到现有应用程序的查询中。

您有两个用于设置 IFX_EXTDIRECTIVES 环境变量的选项：

全局，用于所有用户：

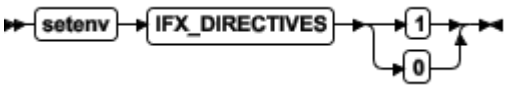
在服务器上，将环境中的 IFX_EXTDIRECTIVES 设置为用户 gbasedbt，然后运行 oninit 命令。

特定于客户机：

在客户机上，设置环境中的 IFX_EXTDIRECTIVES。在客户机环境中设置 IFX_EXTDIRECTIVES 时，无论使用哪种服务器（全局）设置，都会使用客户机设置。

您可以使用 onstat -g env 命令来确定服务器设置。

可指定 ON 和 OFF 或 1 和 0 来设置环境变量。



1　接受的外部优化器伪指令

0　不接受的外部优化器伪指令

如果数据库服务器配置文件中的参数 EXT_DIRECTIVES 和客户机系统上的环境变量设置 **IFX_EXTDIRECTIVES** 都已设置为 1 或 ON，那么给定客户机应用程序内的查询可使用外部伪指令。如果未设置 **IFX_EXTDIRECTIVES**，那么仅当 ONCONFIG 参数 EXT_DIRECTIVES 设置为 2 时才支持外部伪指令。下表概括了支持外部优化器伪指令的有效 IFX_EXTDIRECTIVES 和 EXT_DIRECTIVES 设置的效果。

表 1. 外部伪指令上的 IFX_EXTDIRECTIVES 和 EXT_DIRECTIVES 设置的效果

	EXT_DIRECTIVES = 0	EXT_DIRECTIVES = 1	EXT_DIRECTIVES = 2
IFX_EXTDIRECTIVES 无设置	OFF	OFF	ON
IFX_EXTDIRECTIVES0 = OFF	OFF	OFF	OFF
IFX_EXTDIRECTIVES1 = ON	OFF	ON	ON

数据库服务器将除了 1 或 2（或无设置）的任何 EXT_DIRECTIVES 设置解释为等同于 OFF，从而禁用外部伪指令的支持。IFX_EXTDIRECTIVES 除了 1 以外的任何值对客户机具有相同效果。

有关如何定义外部优化器伪指令的信息，请参阅 *GBase 8s SQL 指南：语法* 中对 SQL 的 SAVE EXTERNAL DIRECTIVES 语句的描述。有关 EXT_DIRECTIVES 配置参数的更多信息，请参阅《*GBase 8s 管理员参考*》。有关伪指令对性能影响的更多信息，请参阅《*GBase 8s 性能指南*》。

4.5.29 IFX_LARGE_PAGES 环境变量

IFX_LARGE_PAGES 环境变量指定数据库服务器是否可以在硬件和操作系统支持共享内存大页面的平台上使用大页面。如果在服务器环境中启用该环境变量，那么 GBase 8s 可以对物理内存中的非消息共享内存段使用大页面。

只有 AIX® 和 Solaris 操作系统支持 IFX_LARGE_PAGES 环境变量。如果操作系统不支持大页面或系统上没有配置大页面，那么 IFX_LARGE_PAGES 的设置对 GBase 8s 没有影响。

只有 AIX、Solaris 和 Linux™ 操作系统支持 IFX_LARGE_PAGES 环境变量。如果操作系统不支持大页面或系统上没有配置大页面，那么 IFX_LARGE_PAGES 的设置对 GBase 8s 没有影响。

您可以指定 1 或 0 设置此环境变量。

0　已禁用大页面。这是 AIX 系统上的缺省值。

1　已启用大页面。这是 Solaris 系统上的缺省值。

已启用大页面。这是 Solaris 和 Linux 系统上的缺省值。

DBSA 必须使用操作系统命令来配置大页面。请参阅操作系统文档以获取配置过程。

如果足够的大页面已配置且可用，GBase 8s 可以对物理内存中锁定的非消息共享内存段使用大页面。RESIDENT 配置参数控制在物理内存中共享内存段是否锁定，以使得段无法交换。如果没有足够的大页面保存段，段可能包含大页面和常规页面的组合。

在 AIX 上，GBase 8s 使用的大页面的大小为 16 MB。

在 Linux x86_64 上，GBase 8s 使用的大页面由 /proc/meminfo 文件中的 Hugepagesize 条目定义。

GBase 8s 会自动对齐段地址并向上舍入到段大小。除了有关舍入的消息之外，服务器还会在每次尝试使用大页面存储段时将参考消息写入到服务器日志文件。

启用 IFX_LARGE_PAGES 时，使用大页面可以显著提高大内存配置中的性能。

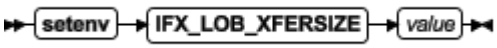
4.5.30 IFX_LOB_XFERSIZE 环境变量

使用 IFX_LOB_XFERSIZE 环境变量，可以指定将多少字节的 CLOB 或 BLOB 数据从客户机应用程序传输到数据库服务器之后执行错误检查。

每次传输指定的字节数时都会进行错误检查。如果发生错误，不会发送其余数据并报告错误。如果没有发生错误，文件传输会继续，直到传输完成。

例如，如果 **IFX_LOB_XFERSIZE** 的值设置为 10485760 (10 MB)，那么每发送 10485760 个字节的 CLOB 或 BLOB 数据之后就会执行错误检查一次。如果没有设置 **IFX_LOB_XFERSIZE**，那么会在全部 BLOB 或 CLOB 数据都传输完成之后执行错误检查。

IFX_LOB_XFERSIZE 的有效范围是从 1 到 9223372036854775808 个字节。**IFX_LOB_XFERSIZE** 环境变量是在客户机上设置的。

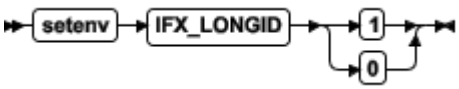


value 在检查是否发生错误之前，CLOB 或 BLOB 中从客户机应用程序传输到数据库服务器的字节数

您应该调整 **IFX_LOB_XFERSIZE** 的值以适合您的环境。将 **IFX_LOB_XFERSIZE** 设置为足够低，以便尽早检测到大 BLOB 或 CLOB 数据类型的传输错误，但不要过低，否则会使用过多的网络资源。

4.5.31 IFX_LONGID 环境变量

IFX_LONGID 环境变量设置和客户机应用程序的版本号确定给定客户机应用程序能否处理长标识。（较旧版本的 GBase 8s 将 SQL 标识限制为 18 字节或更少字节；在设置了 IFX_LONGID 的情况下，*长标识*最多可有 128 字节。）有效 IFX_LONGID 值包括 1 和 0。



1 客户机支持长标识。

0 客户机不能支持长标识。

当 IFX_LONGID 设置为零时，应用程序仅显示长标识符的前 18 个字节，而不（通过 +）指示发生了截断。

如果取消设置 IFX_LONGID 或将其设置为 1 或 0 之外的值，那么根据客户机应用程序的内部版本确定。如果（基于服务器的）版本不低于 9.0304 或处于（基于 CSDK 的）范围 2.90 ≤ *版本* < 4.0，那么认为该客户机能够处理长标识符。否则认为客户机应用程序不能处理长标识符。

IFX_LONGID 设置覆盖客户机应用程序的内部版本。如果客户机不能处理长标识符（尽管版本号较新），那么将 IFX_LONGID 设置为 0。如果客户机版本可以处理长标识符（尽管版本号较旧），那么将 IFX_LONGID 设置为 1。

如果在客户机上设置 IFX_LONGID，那么该设置仅影响该客户机。如果在设置了 IFX_LONGID 的情况下启动数据库服务器，那么所有客户机应用程序在缺省情况下使用该设置。但是，如果在客户机和数据库服务器上将 IFX_LONGID 设置为不同的值，那么客户机设置优先。

重要： 如果 ESQL 可执行文件是使用 -static 选项构建的且构建时使用的 libos.a 库版本不支持长标识，那么这些文件不能使用 IFX_LONGID 环境变量。必须使用新的包括对长标识的支持的 libos.a 库重新编译这类应用程序。使用共享库（没有 -static 选项）的可执行文件可使用 IFX_LONGID 而不进行重新编译（如果它们使用新的提供对长标识的支持的 libifos.so 的话）。有关详细信息，请参阅 ESQL 产品出版物。

4.5.32 IFX_NETBUF_PVTPOOL_SIZE 环境变量 (UNIX™)

使用 IFX_NETBUF_PVTPOOL_SIZE 环境变量，可以为每个数据库服务器会话指定可用（未使用）专用网络缓冲池的最大大小。

count 指定池中单元（缓冲区）数目的整数。

缺省大小为 1 个缓冲区。如果 IFX_NETBUF_PVTPOOL_SIZE 设置为零，那么每个会话从可用全局网络缓冲池获取缓冲区。必须以十进制格式指定该值。

4.5.33 IFX_NETBUF_SIZE 环境变量

使用 IFX_NETBUF_SIZE 环境变量可以将网络缓冲区配置为最佳大小。使用此环境变量可以针对每个数据库服务器会话为可用（未使用）全局池和专用网络缓冲池中的所有网络缓冲区指定大小。

size 为一个网络缓冲区的整数大小（以字节计）。

缺省大小为 4 KB（4,096 字节）。最大大小为 64 KB（65,536 字节），最小大小为 512 字节。可以十六进制或十进制格式指定该值。

技巧： 不能对每个会话设置不同大小。

4.5.34 IFX_NO_SECURITY_CHECK 环境变量 (UNIX™)

IFX_NO_SECURITY_CHECK 环境变量允许用户 **gbasedbt** 或 **root** 完成对数据库服务器实例的操作，即使 GBase 8s 实用程序检测到 $GBASEDBTDIR 路径不安全也是如此。请勿使用此环境变量，除非系统设置确实需要。

IFX_NO_SECURITY_CHECK 的目的是用于以下环境：数据库服务器已启动，但在运行时检测到运行时路径不再安全。在此情况中，超级用户可能需要停止数据库服务器，以纠正安全缺陷。使用此环境变量时，用户 **gbasedbt** 或 **root** 可以使用 onmode 实用程序关闭不安全的 GBase 8s 实例，而不使用此环境变量时则无法关闭不安全的实例，因为当 $GBASEDBTDIR 路径不安全时主要程序不会运行。

使用此环境变量有一些风险，但在一些情况中，可能需要纠正更大的安全性问题。由于仅用户 **gbasedbt** 或 **root** 可以调用 IFX_NO_SECURITY_CHECK，因此非法用户不大可能能够对其进行运行。

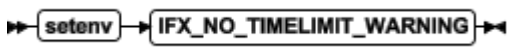
要使用此环境变量，请将其设置为任何非空字符串。

1 运行此环境变量时在此处输入的任何值都会禁用 onsecurity 实用程序。

重要： 完成安全性问题的故障诊断之后，关闭此环境变量。

4.5.35 IFX_NO_TIMELIMIT_WARNING 环境变量

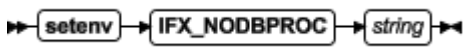
GBase 8s 软件产品的试用版或评估版(自安装软件起经过一段限制时间后会停止运作)在缺省情况下会发出警告消息，告诉用户许可证何时将到期。但是如果设置了环境变量 IFX_NO_TIMELIMIT_WARNING，那么限制时间的软件不会发出这些警告消息。



对于不喜欢查看警告消息的用户来说，此功能是重定向错误输出的另一种选择。但是，**IFX_NO_TIMELIMIT_WARNING** 设置对限制时间的许可证何时过期没有影响；软件会在未设置此环境变量而停止起作用的同一时间点停止起作用。如果的确设置了 **IFX_NO_TIMELIMIT_WARNING**，那么用户将不会看到可能令人生气的关于许可证即将到期的警告，但是当数据库服务器（或无论什么具有限制时间的许可证的软件）在没有任何警告的情况下停止起作用时，一些用户可能会对您恼怒的。

4.5.36 IFX_NODBPROC 环境变量

IFX_NODBPROC 环境变量使您能够阻止数据库服务器运行 sysdbopen( ) 或 sysdbclose( ) 过程。如果将此环境变量设置为任何值，那么这些过程无法运行。



string 设置任何值都将阻止数据库服务器运行 sysdbopen( ) 或 sysdblcose( )。

4.5.37 IFX_NOT_STRICT_THOUS_SEP 环境变量

GBase 8s 要求千位分隔符之后具有 3 位数字。例如，1,000 被认为是正确的，而 1,00 则是错误的。在前发行版中，这两种格式都被认为是正确的。

n 针对前发行版中的行为，将 n 设置为 1，即千位分隔符之后具有的数字可以少于三位。

4.5.38 IFX_ONTAPE_FILE_PREFIX 环境变量

当 TAPEDEV 和 LTAPEDEV 指定目录时，使用 **IFX_ONTAPE_FILE_PREFIX** 环境变量来指定备份文件名的前缀（替换 *hostname_servernum* 格式）。如果没有设置值，那么各个等级的文件名为 *hostname_servernum*_L*n*，而日志文件的文件名为 *hostname_servernum*_Log *nnnnnnnnn*。

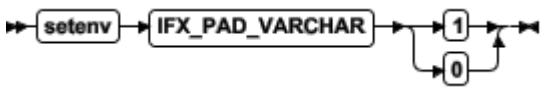
如果将 IFX_ONTAPE_FILE_PREFIX 的值设置为 My_Backup，那么备份文件名具有以下名称：

- My_Backup_L0
- My_Backup_L1
- My_Backup_L2
- My_Backup_Log0000000001
- My_Backup_Log0000000002

string 用于备份文件名的前缀。

4.5.39 IFX_PAD_VARCHAR 环境变量

IFX_PAD_VARCHAR 环境变量设置对数据库服务器发送与接收 VARCHAR 和 NVARCHAR 数据值的方式进行控制。有效 IFX_PAD_VARCHAR 值为 1 和 0。



1 发送整个结构，最高可达声明的 *max* 大小。

0 只发送包含数据的结构部分。

例如：当 **IFX_PAD_VARCHAR** 设置为 0 时，发送声明为 NVARCHAR(255) 的列中的“ABC”字符串时会发送 3 个字节。

但是，如果在先前示例中设置为 1，那么发送的字节的数目将会为 255 字节。

IFX_PAD_VARCHAR 结果是上下文相关的。在低带宽网络中，设置为 0 能够通过减少发送数据的总量改进性能。但是在高带宽网络中，如果处理可变长度数据包所需的 CPU 时间大于发送整个字符流所需的时间，那么设置为 1 可改进性能。在跨服务器分布的操作中，此设置无效，并且填充字符将从在数据库服务器之间传递的 VARCHAR 或 NVARCHAR 值中删除。

4.5.40 IFX_SMX_TIMEOUT 环境变量

使用 IFX_SMX_TIMEOUT 环境变量，可以指定高可用性复制 (HDR)、远程独立 (RS) 或共享磁盘 (SD) 辅助服务器在服务器多路复用器组 (SMX) 连接中等待主服务器的消息的最大秒数。

value 秒数的任何正数字值，或 -1 以禁用此环境变量。可指定的秒数没有上限。

缺省值

10 例如，要指示辅助服务器应等待不超出 60 秒，请指定：

setenv IFX_SMX_TIMEOUT 60

如果在 IFX_SMX_TIMEOUT 环境变量中指定的秒数后且在 IFX_SMX_TIMEOUT_RETRY 环境变量中指定的周期数完成后，辅助服务器未接收到任何消息，那么辅助服务器将打印 **online.log** 中的错误消息并关闭 SMX 连接。如果 SMX 超时消息在 **online.log** 中，那么可能需要增加 IFX_SMX_TIMEOUT 值和/或 IFX_SMX_TIMEOUT_RETRY 值，并重新启动辅助节点。

此环境变量仅适用于辅助服务器。如果在主服务器上设置此环境变量，那么仅当主服务器在发生故障之后变为辅助服务器时，它才会变为有效。

4.5.41 IFX_SMX_TIMEOUT_RETRY 环境变量

使用 IFX_SMX_TIMEOUT_RETRY 环境变量来指定在未接收到主服务器的响应的情况下，高可用性复制 (HDR)、远程独立 (RS) 或共享磁盘 (SD) 辅助服务器将重复 IFX_SMX_TIMEOUT 环境变量所指定等待周期的次数。

value 任何正数字值

缺省值 6

例如，要指示在未接收到主服务器的响应的情况下，IFX_SMX_TIMEOUT 配置参数中指定的时间量应该重复最多 20 次，请指定：

setenv IFX_SMX_TIMEOUT_RETRY 20

如果在 IFX_SMX_TIMEOUT 环境变量中指定的秒数后且在 IFX_SMX_TIMEOUT_RETRY 环境变量中指定的周期数完成后，辅助服务器未接收到任何消息，那么辅助服务器将打印 **online.log** 中的错误消息并关闭 SMX 连接。如果 SMX 超时消息在 **online.log** 中，那么可能需要增加 IFX_SMX_TIMEOUT 值和/或 IFX_SMX_TIMEOUT_RETRY 值，并重新启动辅助节点。

此环境变量仅适用于辅助服务器。如果在主服务器上设置此环境变量，那么仅当主服务器在发生故障之后变为辅助服务器时，它才会变为有效。

4.5.42 IFX_UNLOAD_EILSEQ_MODE 环境变量

要使用此环境变量，请将其设置为任何非空字符串。

value 任何字母或数字值。例如：yes、true 或 1。

从数据库访存或检索字符数据时，此环境变量会生效。

```
setenv IFX_UNLOAD_EILSEQ_MODE 1
setenv IFX_UNLOAD_EILSEQ_MODE yes
setenv IFX_UNLOAD_EILSEQ_MODE on
```

此环境变量类似于在 ONCONFIG 文件中设置 EILSEQ_COMPAT_MODE 配置参数。该配置参数会影响插入到数据库中的字符数据，而 IFX_UNLOAD_EILSEQ_MODE 环境变量会影响从数据库卸载的字符数据。

4.5.43 IFX_UPDDESC 环境变量

必须先在执行时设置 IFX_UPDDESC 环境变量，才能执行 UPDATE 语句的 DESCRIBE。

Value 是非 NULL 的值。

NULL 值（此处意味着未设置 **IFX_UPDDESC**）禁用 describe-for-update 功能。任何非 NULL 的值都会启用该功能。

4.5.44 IFX_XASTDCOMPLIANCE_XAEND 环境变量

IFX_XASTDCOMPLIANCE_XAEND 环境变量能够使用以下语法覆盖当前会话的配置参数。有效 IFX_XASTDCOMPLIANCE_XAEND 值 为 1 和 0。

0 仅在显式回滚后释放全局事务

1 在任何回滚后释放全局事务

当通过 DISABLE_B162428_XA_FIX 配置参数对新行为禁用服务器实例后，而一个客户机需要新行为的情况下，此环境变量可能特别有用。将此环境变量设置为零可支持当前会话中的新行为。

4.5.45 IFX_XFER_SHMBASE 环境变量

实用程序用于连接服务器共享内存段的备用基地址。

地址　十六进制的有效地址

在数据库服务器分配共享内存后，数据库服务器可分配多个邻近的操作系统共享内存段。连接共享内存的客户机实用程序也必须连续地连接所有这些操作系统分段。实用程序可能在服务器将共享内存段连接到的地址装入一些其他的共享对象（例如，onbar 上的 xbsa 库）。要在这种情况下进行变通，您可以在环境变量 IFX_XFER_SHMBASE 中为实用程序指定不同的基地址以连接共享内存段。onstat、onmode 和 oncheck 实用程序必须作为 oninit 连接来完全相同的共享内存库。设置 IFX_XFER_SHMBASE 不是这些实用程序的选项。

4.5.46 IFXRESFILE 环境变量 (Linux)

运行 RPM-method 安装命令之前，将 IFXRESFILE 环境变量设置为响应文件的路径和名称。如果要接受缺省的 GBase 8s 安装设置，请勿使用此环境变量。

path_filename 指定响应文件（.ini 文件）的路径和名称，在此文件中您更改了随安装介质一起提供的 bundle.ini 文件的缺省安装设置。

有关通过定制 bundle.ini 文件来创建响应文件的信息，请参阅《*UNIX™ 和 Linux™ 系统上 GBase 8s 安装指南*》。

4.5.47 IMCADMIN 环境变量

IMCADMIN 环境变量支持 imcadmin 管理工具，这是通过指定数据库服务器的名称（imcadmin 可通过该数据库服务器连接至 MaxConnect）实现的。要让 imcadmin 正常运行，必须在使用 GBase 8s 产品之前设置 IMCADMIN。

dbservername 是数据库服务器的名称。

此处 *dbservername* 必须列示在运行 MaxConnect 的计算机上的 sqlhosts 文件中。MaxConnect 使用此设置来从 sqlhosts 文件获取以下连接信息：

- 必须建立管理侦听器端口的位置
- 指定数据库服务器使用的网络协议
- 指定数据库服务器所在的系统的主机名

不能使用 **imcadmin** 工具，除非 **IMCADMIN** 设置为有效数据库服务器名。

有关使用 **IMCADMIN** 的更多信息，请参阅 *GBase 8s MaxConnect 用户指南*。

4.5.48 IMCCONFIG 环境变量

IMCCONFIG 环境变量对 MaxConnect 配置文件指定非缺省文件名和路径名(可选)。在支持 C shell 的 UNIX™ 系统上，可使用以下命令来设置此变量。

pathname 是全路径名或简单文件名。

如果该设置是未经全路径名限定的文件名，那么 MaxConnect 在 **$GBASEDBTDIR/etc/** 目录中搜索该指定文件。因此，如果将 **IMCCONFIG** 设置为 **IMCconfig.imc2**，MaxConnect 会搜索作为其配置文件的 **$GBASEDBTDIR/etc/IMCconfig.imc2**。

如果未设置 **IMCCONFIG** 环境变量，那么 MaxConnect 在缺省情况下搜索作为其配置文件的 **$GBASEDBTDIR/etc/IMCconfig**。

4.5.49 IMCSERVER 环境变量

IMCSERVER 环境变量指定了连接的相关信息所在 **sqlhosts** 文件中数据库服务器条目的名称。

数据库服务器可以是本地或远程的。在支持 C shell 的 UNIX™ 系统上，可使用该命令设置 IMCSERVER 环境变量。

dbservername 是数据库服务器的有效名称。

此处 *dbservername* 必须是 **sqlhosts** 文件中的数据库服务器的名称。有关使用 MaxConnect 的 **sqlhosts** 设置的更多信息，请参阅《*GBase 8s 管理员指南*》。除非 **IMCSERVER** 设置为有效的数据库服务器，否则不能使用 MaxConnect。

4.5.50 GBASEDBTC 环境变量 (UNIX™)

GBASEDBTC 环境变量指定要用来编译 GBase 8s ESQL/C 所生成文件的 C 编译器的文件名或路径名。 该设置仅在 C 编译阶段生效。

如果未设置 GBASEDBTC，那么大部分系统上的缺省编译器为 cc。

compiler C 编译器的文件名。

pathname C 编译器的完整路径名。

例如，要指定 GNU C 编译器，请输入以下命令：

setenv GBASEDBTC gcc

重要： 如果使用 gcc，请注意，数据库服务器假定字符串是可写的，所以您必须使用 -fwritable-strings 选项进行编译。任务失败可能会产生不可预测的结果（可能包括核心转储）。

4.5.51 GBASEDBTCMNAME 环境变量

如果连接管理器发出事件警报，那么会使用 GBASEDBTCMNAME 环境变量来存储发出警报的连接管理器实例的名称。该环境变量由连接管理器自动设置。

GBASEDBTCMNAME 环境变量对应于连接管理器配置文件中的 NAME 参数。CMALARMPROGRAM 程序使用该环境变量来确定对事件警报负责的连接管理器实例。您也可以在自己的连接管理器事件警报处理程序中使用该环境变量。

该环境变量由连接管理器自动设置，不应进行修改。

4.5.52 GBASEDBTCMCONUNITNAME 环境变量

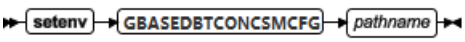
如果连接管理器发出事件警报，那么会使用 GBASEDBTCMCONUNITNAME 环境变量来存储发出警报的连接管理器连接单元的名称。该环境变量由连接管理器自动设置。

GBASEDBTCMCONUNITNAME 环境变量对应于连接管理器配置文件中的连接单元名称参数。CMALARMPROGRAM 程序使用该环境变量来确定对事件警报负责的连接管理器实例。您也可以在自己的连接管理器事件警报处理程序中使用该环境变量。

该环境变量由连接管理器自动设置，不应进行修改。

4.5.53 GBASEDBTCONCSMCFG 环境变量

使用 GBASEDBTCONCSMCFG 环境变量可以指定 concsm.cfg 文件的位置（该文件描述通信支持模块）。



pathname 指定 **concsm.cfg** 文件的全路径名。

以下命令指定 concsm.cfg 文件在 /usr/myfiles 中：

setenv GBASEDBTCONCSMCFG /usr/myfiles

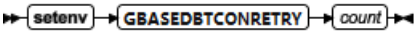
还可对该文件指定不同名称。以下示例在同一目录中指定 csmconfig 的文件名。

setenv GBASEDBTCONCSMCFG /usr/myfiles/csmconfig

concsm.cfg 文件的缺省位置位于 **$GBASEDBTDIR/etc** 中。有关通信支持模块和 **concsm.cfg** 文件内容的更多信息，请参阅 *GBase 8s 管理员参考*。

4.5.54 GBASEDBTCONRETRY 环境变量

GBASEDBTCONRETRY 环境变量设置客户机在 GBASEDBTCONTIME 指定的时间限制期间对每个数据库服务器所作的*附加*连接尝试的最大次数。



count 是连接至每个数据库服务器的附加尝试的次数。

例如：以下命令将 GBASEDBTCONRETRY 设置为指定三次附加连接尝试（在初次尝试之后）：

setenv GBASEDBTCONRETRY 3

GBASEDBTCONRETRY 的缺省值是初次连接尝试之后重试一次。**GBASEDBTCONTIME** 设置在下面一节中作了描述，它优先于 **GBASEDBTCONRETRY** 设置。

4.5.55 GBASEDBTCONTIME 环境变量

GBASEDBTCONTIME 环境变量指定 CONNECT 语句进行建立与数据库服务器的连接的每次尝试持续多少秒后返回错误。如果未设置任何值，那么缺省值 60 秒通常支持几百个并发客户机连接，但对于某些系统，如果设置很低的值（例如 15），可能极少会遇到连接错误。节点间的总距离、硬件速度、流量和网络并发性级别都会影响优化 **GBASEDBTCONTIME** 时应设置的值。

GBASEDBTCONTIME 和 GBASEDBTCONRETRY 环境变量允许您将客户机端连接功能配置为重试该连接而不是返回 -908 错误。

seconds 表示建立与数据库服务器的连接的尝试所花的最少秒数。

例如：输入以下命令以将 GBASEDBTCONTIME 设置为 60 秒：

setenv GBASEDBTCONTIME 60

如果 **GBASEDBTCONTIME** 设置为 60 而 **GBASEDBTCONRETRY** 设置为 3，那么在异常终止之前，连接数据库服务器的尝试会（在初次尝试 0 秒后）在第 20 秒、第 40 秒和第 60 秒进行（如有必要）。这一 20 秒时间间隔是 **GBASEDBTCONTIME** 除以 **GBASEDBTCONRETRY** 的结果。如果尝试将 **GBASEDBTCONTIME** 设置为零，那么数据库服务器自动将其复位为缺省值 60 秒。

如果 CONNECT 语句的执行涉及搜索 DBPATH，那么应用以下规则：

DBPATH 设置中的所有适当服务器至少都会被访问一次，即使可能超出 **GBASEDBTCONTIME** 值也是如此。因此，CONNECT 语句运行时可能会超出 **GBASEDBTCONTIME** 时间限制，并返回指示连接失败或找不到该数据库的错误。

GBASEDBTCONRETRY 指定应对 **DBPATH** 中的每个数据库服务器条目进行多少次附加连接尝试。

GBASEDBTCONTIME 值在 **DBPATH** 中指定数目的数据库服务器条目之间分配。因此，如果 **DBPATH** 包含大量服务器，那么应相应增加 **GBASEDBTCONTIME**

的值。例如：如果 **DBPATH** 包含三个条目，尝试每个连接至少花 30 秒，那么将 **GBASEDBTCONTIME** 设置为 90。

GBASEDBTCONTIME 优先于 **GBASEDBTCONRETRY** 设置。可以在超出 **GBASEDBTCONTIME** 值之后但达到 **GBASEDBTCONRETRY** 值之前结束重试。

如以下示例所示，可使用 onutil SET 命令修改 GBASEDBTCONTIME 和 GBASEDBTCONRETRY 环境变量：

```
% onutil
1> SET GBASEDBTCONTIME 120;
Dynamic Configuration completed successfully
2> SET GBASEDBTCONRETRY 10;
Dynamic Configuration completed successfully
```

4.5.56 GBASEDBTCPPMAP 环境变量

设置 GBASEDBTCPPMAP 环境变量以指定 C++ 程序的映射文件的全限定路径名。映射文件中的信息包括数据库服务器类型、支持该数据库对象或值对象类型的共享库的名称、该对象的库入口点以及对其构建对象的 C++ 库。

pathname 存储 C++ 映射文件的目录路径。

该映射文件是可具有任何文件名的文本文件。您可以指定数个映射文件，在 UNIX™ 上用冒号 (:) 分隔。

在 UNIX 上，缺省映射文件为 $GBASEDBTDIR/etc/c++map。

4.5.57 GBASEDBTDIR 环境变量

GBASEDBTDIR 环境变量指定包含安装产品文件的子目录的目录。总是必须设置 GBASEDBTDIR。 验证 GBASEDBTDIR 是否设置为安装数据库服务器的目录的全路径名。如果有多个版本的数据库服务器，那么将 GBASEDBTDIR 设置为想要访问的版本的适当目录名。有关何时设置 GBASEDBTDIR 的信息，请参阅《*GBase 8s 安装指南*》。

pathname 是安装产品文件的目录路径。

例如：要将 GBASEDBTDIR 设置为 usr/gbasedbt/ 以作为安装目录，输入以下命令：

```
setenv GBASEDBTDIR /usr/gbasedbt
```

4.5.58 GBASEDBTOPCACHE 环境变量

GBASEDBTOPCACHE 环境变量可对客户机应用程序的登台区域 Blob 空间指定内存高速缓存的大小。

kilobytes 指定您为光学内存高速缓存设置的值。

通过指定内存高速缓存的大小 (KB) 来设置 GBASEDBTOPCACHE 环境变量。指定大小必须等于或小于系统范围的配置参数 OPCACHEMAX 的大小。

如果未设置 GBASEDBTOPCACHE，那么缺省高速缓存大小为 128 千字节或在配置参数 OPCACHEMAX 中指定的大小。OPCACHEMAX 的缺省值是 0。如果将 GBASEDBTOPCACHE 设置为值 0，那么 Optical Subsystem 不使用高速缓存。

4.5.59 GBASEDBTSERVER 环境变量

GBASEDBTSERVER 环境变量指定 SQL API 客户机、DB-Access 实用程序或其他 GBase 8s 产品与之建立显式或隐式连接的缺省数据库服务器。

必须先设置此环境变量，才能使用 GBase 8s 客户机产品。它具有以下语法。

dbservername 是缺省数据库服务器的名称。

GBASEDBTSERVER 的值可以是本地或远程服务器，但必须对应于运行该应用程序的计算机上的 **$GBASEDBTDIR/etc/sqlhosts** 文件中的有效 *dbservername* 条目。*dbservername* 必须以小写字母开头，且不能超过 128 字节。它可以包括任何可打印字符，但大写字符、字段定界符（空格或制表符）、换行字符和连字符（或减号）除外。

例如：以下命令指定 coral 数据库服务器作为缺省值：

setenv GBASEDBTSERVER coral

GBASEDBTSERVER 指定执行 CONNECT DEFAULT 时应用程序连接的数据库服务器。它还定义在应用程序中的第一条语句不是 CONNECT 语句的情况下，应用程序与之建立初始隐式连接的数据库服务器。

要点： 必须设置 GBASEDBTSERVER，即使该应用程序或 DB-Access 不使用隐式或显式缺省连接。

4.5.60 GBASEDBTSHMBASE 环境变量 (UNIX™)

GBASEDBTSHMBASE 环境变量仅影响连接至使用进程间通信 (IPC) 共享内存 (ipcshm) 协议的 GBase 8s 数据库的客户机应用程序。

重要： 重新设置 GBASEDBTSHMBASE 要求您全面了解应用程序如何使用内存。一般您不会重新设置 GBASEDBTSHMBASE。

GBASEDBTSHMBASE 指定共享内存通信段连接至客户机进程的位置，以便客户机应用程序可避免与它使用的其他内存段冲突。如果不设置 GBASEDBTSHMBASE，那么通信段的内存地址缺省为特定于实现的值，如 0x800000。

value 是用于计算内存地址的整数 (KB)。

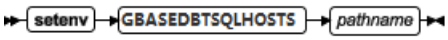
数据库服务器通过将 GBASEDBTSHMBASE 的值乘以 1,024 来计算连接各段的内存地址。例如：在使用 C shell 的系统上，可通过输入以下命令将内存地址设置为值 0x800000：

setenv GBASEDBTSHMBASE 8192

有关更多信息，请参阅《*GBase 8s 管理员指南*》和《*GBase 8s 管理员参考*》。

4.5.61 GBASEDBTSQLHOSTS 环境变量

GBASEDBTSQLHOSTS 环境变量指定 SQL 客户机或数据库服务器可找到连接信息的位置。



pathname 连接信息文件的完整路径名。

UNIX： 缺省值为 $GBASEDBTDIR/etc/sqlhosts

例如，以下命令会覆盖缺省位置，并指定 mysqlhosts 文件位于 /work/envt 目录中：

setenv GBASEDBTSQLHOSTS /work/envt/mysqlhosts

4.5.62 GBASEDBTSTACKSIZE 环境变量

GBASEDBTSTACKSIZE 环境变量指定适用于所有客户机进程的堆栈大小 (KB)。在客户机环境中为 GBASEDBTSTACKSIZE 设置的任何值将被数据库服务器忽略。

size 是一个整数，用于设置 SQL 客户机线程的堆栈大小 (KB)。

例如，要将 GBASEDBTSTACKSIZE 降至 20 KB，请输入以下命令：

setenv -STACKSIZE 20

如果未设置 GBASEDBTSTACKSIZE，那么从数据库服务器配置参数 STACKSIZE 获取堆栈大小，否则堆栈大小的缺省值为特定于平台的值。对于非递归数据库活动，SQL 客户机的主线程的缺省堆栈大小值为 32 KB。

警告： 有关设置此值的指示信息，请参阅《*GBase 8s 管理员参考*》。如果 GBASEDBTSTACKSIZE 的值设置得不正确，可能导致数据库服务器产生故障。

4.5.63 GBASEDBTTERM 环境变量 (UNIX)

GBASEDBTTERM 环境变量指定 DB-Access 是否应该使用 terminfo 目录或 termcap 文件中的信息。

在基于字符的系统上，terminfo 目录和 termcap 文件确定依赖于终端的键盘和屏幕功能，例如：功能键的操作、屏幕显示的色彩和强度属性以及窗口边界和图形字符的定义。

如果未设置 GBASEDBTTERM，那么缺省设置为 terminfo。

terminfo 目录对已定义的每个终端名称包含一个文件。GBASEDBTTERM 的 terminfo 设置仅在完全支持 UNIX™ System V terminfo 库的计算机上受支持。有关详细信息，请参阅产品的机器说明文件。

在系统上安装 DB-Access 时，termcap 文件放在 **$GBASEDBTDIR** 的 etc 子目录中。此文件是操作系统 **termcap** 文件的超集。可使用数据库服务器提供的 **termcap** 文件、系统 termcap 文件或您创建的 termcap 文件。如果不使用缺省 termcap 文件，那么必须设置 TERMCAP 环境变量。有关设置 TERMCAP 环境变量的信息，请参阅TERMCAP 环境变量 (UNIX)。

4.5.64 INF_ROLE_SEP 环境变量

在 UNIX™ 系统上安装或重新安装数据库服务器时，INF_ROLE_SEP 环境变量会配置角色分离的安全性功能。角色分离强制根据运行和审计数据库服务器的人员分隔管理任务。在完成安装后，INF_ROLE_SEP 没有效果。如果未设置 INF_ROLE_SEP，那么用户 gbasedbt（缺省值）可执行所有管理任务。

n 是任意正整数。

如果在 UNIX 平台上安装 GBase 8s 时设置了 INF_ROLE_SEP，那么会实施角色分离并会指定单独的组来负责以下每项责任：

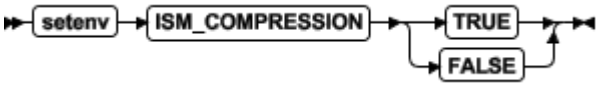
- 数据库服务器管理员 (DBSA)
- 审计分析员 (AAO)
- 标准用户

在 UNIX 上，您可以通过更改拥有 aaodir、dbsadir 或 etc 目录的组在安装完成后的任何时间内建立角色分隔。您可以通过将拥有这些目录的组重置为 **gbasedbt** 来禁用角色分离。例如：您能对审计分析员 (AAO) 启用角色分离，而不用对数据库服务器管理员 (DBSA) 启用角色分离。

有关角色分离的安全性功能的更多信息，请参阅 *GBase 8s 安全指南*。要了解如何在安装数据库服务器时配置角色分离，请参阅《*GBase 8s 安装指南*》。

4.5.65 ISM_COMPRESSION 环境变量

使用 ISM_COMPRESSION 环境变量，可以指定 Storage Manager 是否应该使用数据压缩算法来存储和检索数据。



如果 **ISM_COMPRESSION** 在发出请求的 ON-Bar 进程的环境中设置为 TRUE，那么 Storage Manager 服务器将使用数据压缩算法。

如果 **ISM_COMPRESSION** 设置为 FALSE 或未作设置，那么 Storage Manager 服务器不使用压缩。

4.5.66 ISM_DEBUG_FILE 环境变量

在 Storage Manager 服务器环境中使用 ISM_DEBUG_FILE 环境变量可以指定在何处写入 XBSA 消息。

pathname 指定 XBSA 消息日志文件的位置。

如果未设置 ISM_DEBUG_FILE，那么 XBSA 消息日志位于 $GBASEDBTDIR/ism/applogs/xbsa.messages 目录（在 UNIX™ 上）。

4.5.67 ISM_DEBUG_LEVEL 环境变量

在 ON-Bar 环境中使用 ISM_DEBUG_LEVEL 环境变量可以控制记录在 XBSA 消息日志中的报告详细信息的级别。XBSA 共享库会写入此日志。

value 指定报告详细信息的级别，其中 1 ≤ *value* ≤ 9。

如果 **ISM_DEBUG_LEVEL** 未设置、具有空值或具有此范围之外的值，那么缺省详细信息级别为 1。0 级的详细信息级别会使得所有 XBSA 调试记录无法显示。详细信息级别为 1 时，将仅报告 XBSA 故障。

4.5.68 ISM_ENCRYPTION 环境变量

在 ON-Bar 环境中使用 ISM_ENCRYPTION 环境变量可以指定 Storage Manager 是否使用数据加密。

ISM_ENCRYPTI 的三个设置是受支持的：

XOR 使用加密。

NONE 不使用加密。

TRUE 使用加密。

如果 ISM_ENCRYPTION 设置为 NONE 或未作设置，那么 ISM 服务器不使用加密。

如果 ISM_ENCRYPTION 在提出请求的 ON-Bar 进程的环境中设置为 TRUE 或 XOR，那么 Storage Manager 服务器使用加密来存储或检索在该请求中指定的数据。

4.5.69 ISM_MAXLOGSIZE 环境变量

在 Storage Manager 服务器环境中使用 ISM_MAXLOGSIZE 环境变量可以指定 Storage Manager 活动日志的大小阈值。

size 指定活动日志的大小阈值（以兆字节计）。

如果未设置 ISM_MAXLOGSIZE ，那么缺省大小限制为 1 兆字节。如果 ISM_MAXLOGSIZE 设置为空值，那么该阈值为 0 字节。

4.5.70 ISM_MAXLOGVERS 环境变量

在 Storage Manager 服务器环境中使用 ISM_MAXLOGVERS 环境变量可以指定要由 ISM 服务器保留的活动日志文件的最大数目。

value 指定要保存的文件的数目。

如果未设置 **ISM_MAXLOGVERS**，那么文件的缺省数目为四。如果该设置为空值，那么 ISM 服务器不保存任何活动日志文件。

4.5.71 JAR_TEMP_PATH 环境变量

设置 JAR_TEMP_PATH 变量以指定 jar 管理过程（例如：install_jar( ) 和 replace_jar( )）可存储 Java™ 虚拟机的临时 .jar 文件的非缺省本地文件系统位置。

pathname 指定临时 **.jar** 文件的本地目录。

对于启动数据库服务器的用户，必须具有此目录的读写许可权。如果未设置 **JAR_TEMP_PATH** 环境变量，那么 **.jar** 文件的临时副本存储在数据库服务器的本地文件系统的 **/tmp** 目录中。

4.5.72 JAVA_COMPILER 环境变量

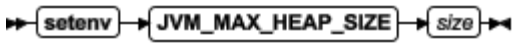
可在 Java™ 虚拟机环境中设置 JAVA_COMPILER 环境变量以禁用 JIT 编译。

NONE 和 none 设置是等效的。在支持 C shell 且 JAVA_COMPILER 已设置为 NONE 或 none 的 UNIX™ 系统上，可使用以下命令对 JVM 环境启用 JIT 编译器：

unset JAVA_COMPILER

4.5.73 JVM_MAX_HEAP_SIZE 环境变量

JVM_MAX_HEAP_SIZE 环境变量可对 Java™ 虚拟机设置堆大小的非缺省上限。



size 是指定最大大小的正整数（以兆字节计）。

例如，以下命令将最大堆大小设置为 12 MB：

set JVM_MAX_HEAP_SIZE 12

如果未设置 **JVM_MAX_HEAP_SIZE**，那么 16 MB 是缺省的最大大小。

4.5.74 LD_LIBRARY_PATH 环境变量 (UNIX™)

LD_LIBRARY_PATH 环境变量告诉 Solaris 系统上的 shell 要在哪些目录中搜索客户机或共享 GBase 8s 常规库。必须指定包含客户机库的目录才能使用本产品。

pathname 指定库的搜索路径。

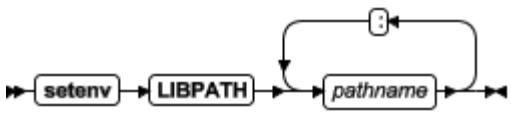
对于 AIX® 上的 INTERSOLV DataDirect ODBC Driver，设置 LIBPATH。对于 HP-UX 上的 INTERSOLV DataDirect ODBC Driver，设置 SHLIB_PATH。

以下示例将 LD_LIBRARY_PATH 环境变量设置为目录：

```
setenv LD_LIBRARY_PATH
${GBASEDBTDIR}/lib:${GBASEDBTDIR}/lib/esql:$LD_LIBRARY_PATH
```

4.5.75 LIBPATH 环境变量 (UNIX™)

LIBPATH 环境变量告诉 AIX® 系统上的 shell 在哪些目录中搜索 INTERSOLV DataDirect ODBC Driver 的动态链接库。必须指定本产品安装所在目录的完整路径名。



pathname 指定库的搜索路径。

在 Solaris 上，设置 LD_LIBRARY_PATH。在 HP-UX 上，设置 SHLIB_PATH。

4.5.76 NODEFDAC 环境变量

当 NODEFDAC 环境变量设置为 yes 时，在不符合 ANSI 标准的数据库中当前会话期间创建新表时，无法将缺省表特权（Select、Insert、Update 和 Delete）授予 PUBLIC。

yes 防止将不符合 ANSI 标准的数据库中新表的缺省表特权授予 PUBLIC。此设置还会在以“所有者”方式创建新的用户定义的例程时防止将对该例程的 Execute 特权授予 PUBLIC。

yes 设置是区分大小写的，并且还区分前导和尾部空格。在设置中包含大写字母或空格等同于未设置 **NODEFDAC**。当未设置 **NODEFDAC**，或者如果设置为任何值（除了 *yes*）时，表和所有者方式 UDR 的缺省特权授予 PUBLIC（缺省情况下，当表或 UDR 创建在不符合 ANSI 标准的数据库中时）。

4.5.77 ONCONFIG 环境变量

ONCONFIG 环境变量会指定活动文件（称为 onconfig 文件）的名称（该文件中保存数据库服务器的配置参数）。

此文件在初始化过程期间是作为输入读取的。在准备 onconfig 配置文件之后，将 ONCONFIG 环境变量设置为此文件的名称。

filename 是 $GBASEDBTDIR/etc/$ONCONFIG 目录中 onconfig 文件的名称

此文件包含数据库的配置参数。

要准备 onconfig 文件，请建立 onconfig.std 文件的副本并修改该副本。请为 onconfig 文件命名，使其能够与特定数据库服务器轻松关联。如果您拥有数据库服务器的多个实例，那么每个实例*必须*拥有其自己的唯一命名 onconfig 文件。

如果未设置 ONCONFIG 环境变量，那么数据库服务器将在初始化期间从 onconfig 文件读取配置值。

4.5.78 OPTCOMPIND 环境变量

可设置 OPTCOMPIND 环境变量以便优化器可选择适当的连接方法。

0 尽量使用嵌套循环连接而不是分类合并连接或散列连接。

1 当隔离级别*不是*“可重复读”时，优化器的行为如同设置 2 时一样；否则优化器的行为如同设置 0 时一样。

2 嵌套循环连接并不一定是首选的。优化器只根据成本来作决定，而不考虑事务隔离方式。

如果未设置 **OPTCOMPIND**，那么数据库服务器使用 ONCONFIG 配置文件中的 OPTCOMPIND 值。如果既未设置环境变量又未设置配置参数，那么缺省值为 2。

在 GBase 8s 上，SET ENVIRONMENT OPTCOMPIND 语句可以在运行时动态设置或重置 **OPTCOMPIND**。这仅覆盖当前用户会话的当前 **OPTCOMPIND** 值（或 ONCONFIG

配置参数 OPTCOMPIND）。有关 SQL 的 SET ENVIRONMENT OPCOMPIND 语句的更多信息，请参阅 *GBase 8s SQL 指南：语法*。

有关 ONCONFIG 配置参数 OPTCOMPIND 的更多信息，请参阅 *GBase 8s 管理员参考*。有关优化器使用的不同连接方法的更多信息，请参阅《*GBase 8s 性能指南*》。

4.5.79 OPTMSG 环境变量

启动 GBase 8s ESQL/C 应用程序之前，在运行时设置 OPTMSG 环境变量，可以为应用程序中的所有 SQL 语句启用（或禁用）优化的消息传送（消息链接）。

0　禁用优化消息传送。

1　启用优化消息传送并为任何后续连接实现该功能。

缺省值为 0（零），它将显式禁用消息链接。例如，您可能希望对需要即时应答的语句禁用优化的消息传送以便进行调试，或确保数据库服务器在应用程序终止前处理所有消息。

在应用程序内设置 OPTMSG 时，可为每个连接或在每个线程内激活或取消激活优化消息传送。要启用优化消息传送，必须在建立连接之前设置 OPTMSG。

有关设置 OPTMSG 和定义相关全局变量的更多信息，请参阅 *GBase 8s ESQL/C 程序员手册*。

4.5.80 OPTOFC 环境变量

使用 OPTOFC 环境变量可在 GBase 8s ESQL/C 应用程序或使用 DECLARE 和 OPEN 语句来建立游标的其他 API（例如，JDBC、ODBC、OLE DB、LIBDMI 和 Lib C++）中启用优化 OPEN-FETCH-CLOSE 功能。

0　对应用程序的所有线程禁用 **OPTOFC**。

1　对应用程序的每个线程中的每个游标启用 **OPTOFC**。

缺省值为 0（零）。

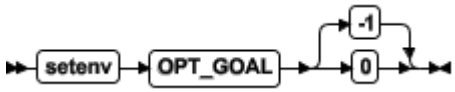
您可以在客户机或服务器上设置 **OPTOFC** 环境变量。如果在服务器上设置此环境变量，那么任何未显式设置此环境变量的应用程序会使用在服务器上设置的值。

OPTOFC 环境变量减少应用程序与数据库服务器之间的消息请求数。

如果从 shell 设置 **OPTOFC**，那么必须在启动 GBase 8s ESQL/C 应用程序之前设置它。有关启用 **OPTOFC** 和相关功能的更多信息，请参阅 *GBase 8s ESQL/C 程序员手册*。

4.5.81 OPT_GOAL 环境变量 (UNIX™)

在启动应用程序之前在用户环境中设置 OPT_GOAL 环境变量以对优化器指定查询性能目标。



0　指定用户响应时间优化。

-1　指定总查询时间优化。

优化器的缺省行为是使用优化总查询时间的查询方案。

还可使用优化器伪指令对个别查询指定优化目标或使用 SET OPTIMIZATION 语句对会话指定优化目标。

这两种方法优先于 OPT_GOAL 环境变量设置。还可对 GBase 8s 系统设置 OPT_GOAL 配置参数；此方法的优先级最低。

有关优化数据库服务器查询的更多信息，请参阅《*GBase 8s 性能指南*》。有关 SET OPTIMIZATION 语句的信息，请参阅 *GBase 8s SQL 指南：语法*。

4.5.82 PATH 环境变量

UNIX™ PATH 环境变量告诉 shell 要在哪些目录中搜索可执行程序。必须先将包含 GBase 8s 产品的目录添加至 PATH 设置，才能使用该产品。

pathname　指定可执行文件的搜索路径。

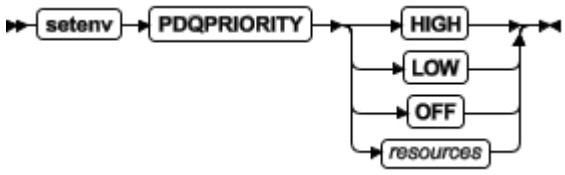
在 UNIX 系统上，路径名之间使用冒号 (:) 分隔符。

可以各种方式指定搜索路径。PATH 环境变量告诉操作系统在何处搜索可执行程序。必须将包含 GBase 8s 产品的目录包括在**路径**设置中才能使用该产品。此目录应位于 $GBASEDBTDIR/bin 之前，而 $GBASEDBTDIR/bin 也必须包括在内。

有关如何修改路径的更多信息，请参阅修改环境变量设置。

4.5.83 PDQPRIORITY 环境变量

PDQPRIORITY 环境变量确定数据库服务器使用的并行度，并影响数据库服务器分配资源（包括内存、处理器和磁盘读取）的方式。



resources 是范围从 0 至 100 的整数。值 1 相当于 LOW，而值 100 相当于 HIGH。小于 0 的值将设置为 0 (OFF)，而大于 100 的值将设置为 100 (HIGH)。

值 0 相当于 OFF（仅适用于 GBase 8s）。

此处 HIGH、LOW 和 OFF 关键字具有下列作用：

HIGH 当数据库服务器在所有用户间分配资源时，它将给予该查询尽可能多的资源。

LOW 以并行方式从分段表中访存数据值。

OFF 关闭了 PDQ 处理（仅适用于 GBase 8s）。

通常，数据库服务器使用的资源越多，它针对给定查询的性能也就越好。但是，如果服务器使用的资源过多，那么资源争用可能夺取其他查询的资源，从而使得性能下降。有关 **PDQPRIORITY** 的性能注意事项的更多信息，请参阅《*GBase 8s 性能指南*》。

如 *GBase 8s SQL 指南：语法* 所述，当应用程序发出 SQL 语句 SET PDQPRIORITY 时，会覆盖此环境变量的设置。

在 GBase 8s 中使用 PDQPRIORITY

*资源*值会指定查询优先级和数据库服务器用来处理该查询的资源量。

如果未设置 PDQPRIORITY，那么缺省值为 OFF。

当 PDQPRIORITY 设置为 HIGH 时，GBase 8s 根据几个标准确定用于 PDQPRIORITY 的适当值。这些标准包括可用处理器的数目、被查询的表的分段存储、该查询的复杂性和其他因子。

4.5.84 PLCONFIG 环境变量

PLCONFIG 环境变量会指定 High Performance Loader (HPL) 使用的配置文件的名称。此文件必须位于 $GBASEDBTDIR/etc 目录中。如果未设置 PLCONFIG 环境变量，那么 $GBASEDBTDIR/etc/plconfig 就是缺省配置文件。

filename 指定 High-Performance Loader 使用的配置文件的简单文件名。

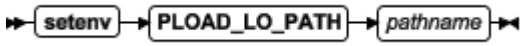
例如，要将 $GBASEDBTDIR/etc/custom.cfg 文件指定为 High-Performance Loader 的配置文件，请输入以下命令：

setenv PLCONFIG custom.cfg

有关更多信息，请参阅 *GBase 8s High-Performance Loader 用户指南*。

4.5.85 PLOAD_LO_PATH 环境变量

PLOAD_LO_PATH 环境变量允许您指定智能大对象句柄的路径名（它标识如 BLOB 和 CLOB 数据类型这样的智能大对象的位置）。



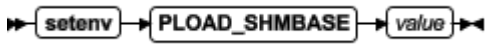
pathname 指定智能大对象句柄的目录。

如果未设置 **PLOAD_LO_PATH**，那么缺省目录为 **/tmp**。

有关更多信息，请参阅 *GBase 8s High-Performance Loader 用户指南*。

4.5.86 PLOAD_SHMBASE 环境变量

PLOAD_SHMBASE 环境变量允许您指定 High Performance Loader (HPL) onpload 进程将连接的共享内存地址。如果未设置 PLOAD_SHMBASE，那么 HPL 会确定要使用哪个共享内存地址。



value 用来计算共享内存地址。

如果 onpload 实用程序不能连接，那么会发出一个错误，且您必须指定新值。

onpload 实用程序尝试确定在哪个地址进行连接，如以下（降序）顺序所示：

1. 在数据库服务器所在的地址 (SHMBASE) 进行连接。
2. 在数据库服务器段之外进行连接。
3. 在 PLOAD_SHMBASE 中指定的地址进行连接。

提示： 建议让 HPL 决定在何处进行连接，且仅当需要避免 onpload 与数据库服务器之间的共享内存冲突时才设置 PLOAD_SHMBASE。

有关更多信息，请参阅 *GBase 8s High-Performance Loader 用户指南*。

4.5.87 PSM_ACT_LOG 环境变量

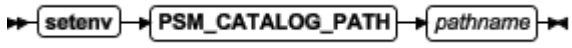
使用 PSM_ACT_LOG 环境变量，可以为您的环境（例如，单个会话）指定 GBase 8s Primary Storage Manager 活动日志的位置。

pathname $GBASEDBTDIR/psm_act.log 的位置的完整路径名。如果仅指定文件名，那么存储管理器将在您启动存储管理器的工作目录中创建活动日志。

PSM_ACT_LOG 环境变量会覆盖 PSM_ACT_LOG 配置参数的值。

4.5.88 PSM_CATALOG_PATH 环境变量

使用 PSM_CATALOG_PATH 环境变量，可以为您的环境（例如，单个会话）指定 GBase 8s Primary Storage Manager 目录表的位置。

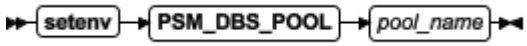


pathname 目录表的位置的完整路径名，该目录表中包含有关存储管理器管理的池、设备和对象的信息。

PSM_CATALOG_PATH 环境变量会覆盖 PSM_CATALOG_PATH 配置参数的值。

4.5.89 PSM_DBS_POOL 环境变量

使用 PSM_DBS_POOL 环境变量，可以为您的环境（例如，单个会话）更改 GBase 8s Primary Storage Manager 放置备份和复原数据库空间数据的池的名称。



pool_name 存储管理器池的名称。

PSM_DBS_POOL 环境变量会覆盖 PSM_DBS_POOL 配置参数的值。

4.5.90 PSM_DEBUG 环境变量

使用 PSM_DEBUG 环境变量，可以为您的环境（例如，单个会话）指定 GBase 8s Primary Storage Manager 调试日志中打印的调试信息量。

value

0 = 无调试消息。

1 = 仅打印内部错误。

2 = 打印有关函数入口和出口的信息，还打印内部错误。

3 = 打印 1-2 指定的信息以及其他详细信息。

4 = 打印有关并行操作的信息以及 1-3 指定的信息。

5 = 打印有关 GBase 8s Primary Storage Manager 中内部状态的信息。

6 = 打印 1-5 指定的信息以及其他详细信息。

7 = 打印 1-6 指定的信息以及其他详细信息。

8 = 打印 1-7 指定的信息以及其他详细信息。

9 = 打印所有调试信息。

PSM_DEBUG 环境变量会覆盖 PSM_DEBUG 配置参数的值。

4.5.91 PSM_DEBUG_LOG 环境变量

使用 PSM_DEBUG_LOG 环境变量，可以为您的环境（例如，单个会话）指定 GBase 8s Primary Storage Manager 调试日志的位置。

pathname $GBASEDBTDIR/psm_debug.log 的位置的完整路径名。如果仅指定文件名，那么存储管理器将在您启动存储管理器的工作目录中创建调试日志。

PSM_DEBUG_LOG 环境变量会覆盖 PSM_DEBUG_LOG 配置参数的值。

4.5.92 PSM_LOG_POOL 环境变量

使用 PSM_LOG_POOL 环境变量，可以为您的环境（例如，单个会话）更改 GBase 8s Primary Storage Manager 放置备份和复原日志数据的池的名称。

pool_name 存储管理器日志池的名称。

PSM_LOG_POOL 环境变量会覆盖 PSM_LOG_POOL 配置参数的值。

4.5.93 PSORT_DBTEMP 环境变量

PSORT_DBTEMP 环境变量指定数据库服务器将环境变量用于执行排序的临时文件写入的位置。

pathname 排序期间用于中间写操作的 UNIX™ 目录的名称。

要设置 PSORT_DBTEMP 环境变量以指定该目录（例如：/usr/leif/tempsort），输入以下命令：

setenv PSORT_DBTEMP /usr/leif/tempsort

要获取最佳性能，请指定在不同磁盘上的文件系统中的目录。

您还可能想要考虑设置环境变量 DBSPACETEMP 以放置在数据库空间中进行排序时使用的临时文件而不是操作系统文件。请参阅 DBSPACETEMP 环境变量中对 DBSPACETEMP 环境变量的讨论。

数据库服务器使用 PSORT_DBTEMP 指定的目录，即使未设置 PSORT_NPROCS 环境变量也是如此。有关 PSORT_DBTEMP 环境变量的更多信息，请参阅《*GBase 8s 管理员指南*》和《*GBase 8s 性能指南*》。

4.5.94 PSORT_NPROCS 环境变量

PSORT_NPROCS 环境变量使数据库服务器能够通过为进行排序分配更多线程来改进并行进程排序软件包的性能。

在排序软件包执行并行排序之前，确保数据库服务器有足够的内存可供排序使用。

threads 是一个整数，指定要用来对查询排序的最大线程数。此值不能超过 10。

以下命令将 PSORT_NPROCS 设置为 4：

setenv PSORT_NPROCS 4

要禁用并行排序，输入以下命令：

unsetenv PSORT_NPROCS

如果计算机有多个 CPU，建议最初将 PSORT_NPROCS 设置为 2。如果后续 CPU 活动的速度小于 I/O 活动，可增加 PSORT_NPROCS 的值。

技巧： 如果未设置 PDQPRIORITY 环境变量，那么数据库服务器分配最少量的内存来进行排序。这一最少内存即使启动两个排序线程都不够。如果未设置 PDQPRIORITY，那么在执行大规模排序（如索引构建）之前检查可用内存以确保您有足够内存。

拆离索引的缺省 PSORT_NPROCS 值

如果设置了 **PSORT_NPROCS** 环境变量，那么数据库服务器使用指定的排序线程数作为普通排序的上限。如果未设置 **PSORT_NPROCS**，那么不会进行并行排序。数据库服务器对该排序使用一个线程。如果 **PSORT_NPROCS** 设置为 0，那么数据库服务器对该排序使用三个线程。

连接的索引的缺省 PSORT_NPROCS 值

线程的缺省数目对各个连接索引是不同的。

如果设置了 **PSORT_NPROCS** 环境变量，那么正在构建的索引的每个分段具有指定数目的排序线程。

如果未设置 **PSORT_NPROCS**，或者如果其设置为 0，那么除非您有单 CPU 虚拟处理器，否则该索引的每个分段将有两个排序线程。如果您有单 CPU 虚拟处理器，那么您将获取索引的每个分段的一个排序线程。

有关 **PSORT_NPROCS** 环境变量的更多信息，请参阅《*GBase 8s 管理员指南*》和 *GBase 8s 性能指南*。

4.5.95 RTREE_COST_ADJUST_VALUE 环境变量

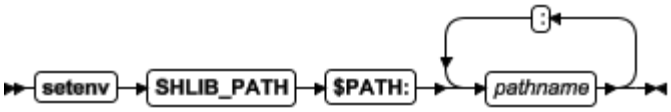
RTREE_COST_ADJUST_VALUE 环境变量指定一个系数，用户定义的数据类型的支持函数可使用该系数来估计对针对 UDT 列的查询使用 R 型树索引的成本。

value 为一个浮点数（其中 1 ≤ *value* ≤ 1000），它指定一个乘法器来估计对 UDT 列使用索引的成本。

对于空间查询，I/O 开销可能会超出 CPU 成本很多，所以通过用未更正的估计成本乘以此设置中的适当*值*，数据库服务器可以就如何执行对存在 R 型树索引的 UDT 列的查询作出更好的基于成本的决策。

4.5.96 SHLIB_PATH 环境变量 (UNIX™)

SHLIB_PATH 环境变量告诉 HP-UX 系统上的 shell 要在哪些目录中搜索动态链接库。例如：它可与 INTERSOLV DataDirect ODBC Driver 配合使用。必须对安装该产品的目录指定全路径名。



pathname 指定库的搜索路径。

在 Solaris 系统上，设置 LD_LIBRARY_PATH。在 AIX® 系统上，设置 LIBPATH。

4.5.97 SRV_FET_BUF_SIZE 环境变量

使用 SRV_FET_BUF_SIZE 环境变量，可以指定本地数据库服务器用于数据库服务器中分布式 DML 事务的访存缓冲区的大小。

size 是一个不超过 1048576 (1 MiB) 的正整数，指定用于保存服务器中分布式查询所检索数据的访存缓冲区的大小（以字节计）。

例如，要在使用 C shell 的 UNIX™ 系统上将缓冲区大小设置为 5,000 字节，请输入以下命令来设置 SRV_FET_BUF_SIZE：

setenv SRV_FET_BUF_SIZE 5000

将 SRV_FET_BUF_SIZE 设置为有效值后，新值将覆盖 SRV_FET_BUF_SIZE 的缺省值（或先前设置的任何值）。仅当数据库服务器的启动环境中设置了该设置时，它才生效。

如果未设置 SRV_FET_BUF_SIZE，那么访存缓冲区的缺省设置取决于行大小。

如果将 SRV_FET_BUF_SIZE 设置为小于缺省大小的值或大于 1048576 (1 MiB) 的值，那么不会发生任何错误。如果为 SRV_FET_BUF_SIZE 指定大于 1048576 的大小，那么该值将设置为 1048576。

有效的 SRV_FET_BUF_SIZE 设置仅在服务器上的 DML 事务中生效（在这些事务中，本地数据库服务器作为协调器或作为下级数据库服务器参与其中）。

但是，它对仅访问本地服务器实例的数据库的查询无效，并且不会影响客户机与本地服务器通信中的访存缓冲区的大小。

BYTE 和 TEXT 对象的处理不会受 SRV_FET_BUF_SIZE 设置的影响。

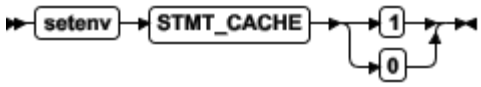
为本地数据库服务器环境设置 SRV_FET_BUF_SIZE 不会重置在服务器上的 DML 事务中协调本地服务器实例或参与其中的远程服务器实例的访存缓冲区大小。

缓冲区的大小越大，可返回的行就越多，因而本地服务器必须等待数据库服务器返回行的频率就越低。大型缓冲区可提高在服务器之间传输大量数据的性能。

4.5.98 STMT_CACHE 环境变量

使用 STMT_CACHE 环境变量来控制会话上的共享语句高速缓存的使用。

此功能可减少内存消耗，还可加速不同用户会话间的查询处理。有效 STMT_CACHE 值包括 1 和 0。



1 启用 SQL 语句高速缓存。

0 禁用 SQL 语句高速缓存。

对不使用 SET STMT_CACHE 语句的应用程序设置 STMT_CACHE 环境变量，可以控制对 SQL 语句高速缓存的使用。缺省情况下，语句高速缓存是禁用的，但是可以通过 onconfig.std 文件的 STMT_CACHE 参数或 SET STMT_CACHE 语句来启用该功能。

如果通过配置文件设置来禁用 SQL 语句高速缓存，那么此环境变量不起任何作用。应用程序中的 SET STMT_CACHE 语句设置的值会覆盖 STMT_CACHE 设置。

4.5.99 TERM 环境变量 (UNIX™)

TERM 环境变量用于终端处理。它允许 DB-Access（及其他基于字符的应用程序）识别您正在使用的终端并与其通信。

type 指定终端类型。

在 TERM 设置中指定的终端类型必须对应于 termcap 文件或 terminfo 目录中的某个条目。

必须先从数据库管理员处获取代表您终端的代码，才能设置 TERM 环境变量。

例如：要指定 vt100 终端，通过输入以下命令设置 TERM 环境变量：

setenv TERM vt100

4.5.100 TERMCAP 环境变量 (UNIX™)

TERMCAP 环境变量用于终端处理。它告诉 DB-Access（和其他基于字符的应用程序）与 termcap 文件通信而不是与 terminfo 目录通信。

pathname 指定 termcap 文件的位置。

termcap 文件包含各种类型的终端以及它们的特征的列表。例如，要提供 DB-Access 终端处理信息（该信息是在 /usr/gbasedbt/etc/termcap 文件中指定的），请输入以下命令：

setenv TERMCAP /usr/gbasedbt/etc/termcap

可以下列任一方式设置 TERMCAP：如果存在若干个 termcap 文件，那么它们具有以下优先级顺序（降序）：

1. 您创建的 termcap 文件
2. 数据库服务器提供的 termcap 文件（即 $GBASEDBTDIR/etc/termcap）

3. 操作系统 termcap 文件（即，/etc/termcap）

如果设置 TERMCAP 环境变量，应确保 GBASEDBTTERM 环境变量设置为 termcap。

如果不设置 TERMCAP 环境变量，那么在缺省情况下使用 terminfo 目录。

4.5.101 TERMINFO 环境变量 (UNIX)

TERMINFO 环境变量用于终端处理。

该环境变量仅在完全支持 System V 和 Solaris UNIX™ 系统提供的 terminfo 库的平台上受支持。

TERMINFO 告诉 DB-Access 与 terminfo 目录通信而不是与 termcap 文件通信。terminfo 目录有包含与终端及其特征有关的文件的子目录。

要设置 TERMINFO，输入以下命令：

setenv TERMINFO /usr/lib/terminfo

4.5.102 THREADLIB 环境变量 (UNIX™)

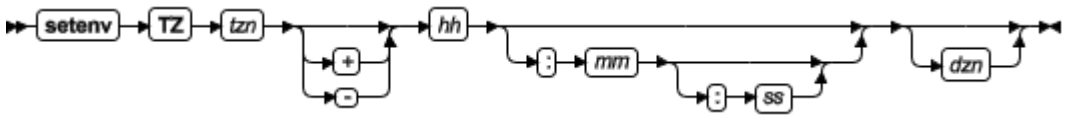
使用 THREADLIB 环境变量可编译多线程 GBase 8s ESQL/C 应用程序。多线程 GBase 8s ESQL/C 应用程序允许您与一个或多个数据库建立与线程数一样多的连接。当应用程序执行时，这些连接可以保持活动状态。

THREADLIB 环境变量指示在编译应用程序时使用哪些线程软件包。目前仅支持“分布式计算环境”(DCE)。

当编译多线程 GBase 8s ESQL/C 应用程序时，将 -thread 选项传递至 GBase 8s ESQL/C 脚本时会检查 THREADLIB 环境变量。当在编译时使用 -thread 选项的情况下，如果未设置 THREADLIB 或将 THREADLIB 设置为不受支持的线程软件包，那么 GBase 8s ESQL/C 脚本生成错误。

4.5.103 TZ 环境变量

TZ 环境变量用于设置时区。它由各种时间函数用于计算相对于全球标准时间 (UTC)（以前称为格林威治标准时间 (GMT)）的时间。格式由操作系统指定。



tzn 三个字母的时区名称，如 PST。您必须指定从本地时间到 UTC（全球标准时间）之间的正确偏移量。

hh UTC 与本地时间之间的小时数之差（一位数或两位数）。可选择带符号。

mm UTC 与本地时间之间的分钟数之差（两位数）。

ss UTC 与本地时间之间的秒数之差（两位数）。

dzn 三个字母的夏令时区域，如 PDT。如果本地从未采用夏令时，请在设置 **TZ** 时不使用 *dzn* 值。

例如，如果使用实施太平洋夏令时的太平洋标准时间，请将 **TZ** 环境变量设置为 PST8PDT。有关设置 **TZ** 环境变量的更多信息，请参阅操作系统文档。

4.5.104 USETABLENAME 环境变量

USETABLENAME 环境变量可防止用户使用同义词在 ALTER TABLE 或 DROP TABLE 语句中指定 *table*。 与大多数环境变量不同，USETABLENAME 不需要设置值。无论您将其设置为任何值或不对其设置值，它都会起作用。

缺省情况下，ALTER TABLE 或 DROP TABLE 语句接受表示要改变或删除的 *table* 名称的有效同义词。（相反，如果您指定同义词，RENAME TABLE 会报错，如果您尝试用同义词替代这些语句中的 *sequence* 名，那么 ALTER SEQUENCE、DROP SEQUENCE 和 RENAME SEQUENCE 语句也会报错。）

如果设置 **USETABLENAME**，那么同义词在 ALTER TABLE 或 DROP TABLE 语句中时，会导致错误。设置 **USETABLENAME** 对 DROP VIEW 语句没有任何影响，该语句将接受视图的有效同义词。

5 附录

5.1 stores_demo 数据库

stores_demo 数据库包含描述虚构业务的一组表，且 GBase 8s 文档中的许多示例基于此数据库。

stores_demo 数据库使用缺省（美国英语）语言环境并且不符合 ANSI 标准。

有关如何创建和填充 **stores_demo** 数据库的信息，请参阅《*GBase 8s DB-Access 用户指南*》。有关如何设计和实现关系数据库的信息，请参阅《*GBase 8s 数据库设计和实现指南*》。

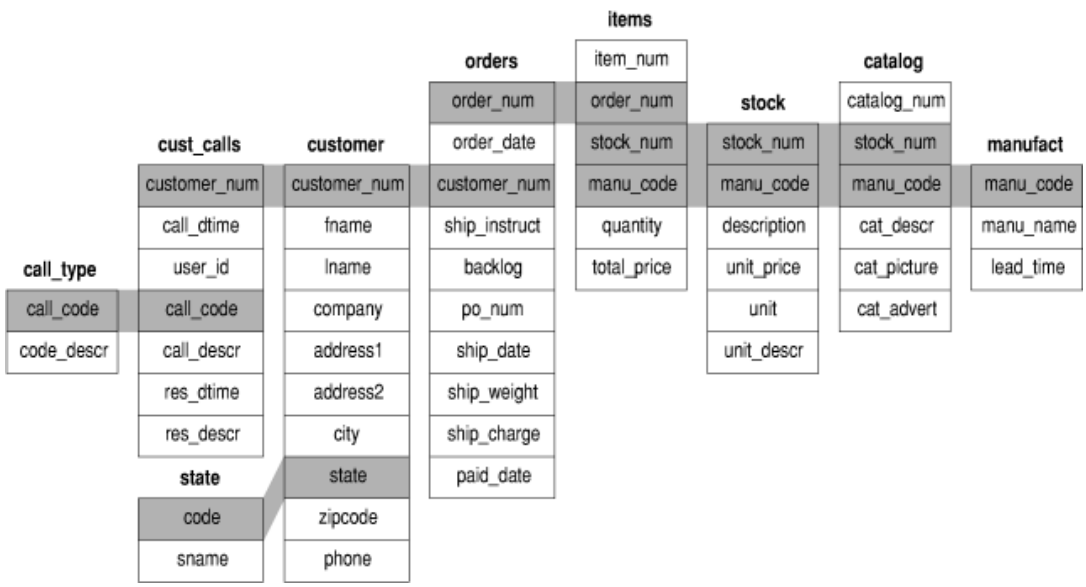
您可以在 OpenAdmin Tool (OAT) 的模式浏览器中查看表的结构以及它们的数据。

stores_demo 数据库映射

stores_demo 数据库中的一些表之间存在关系。

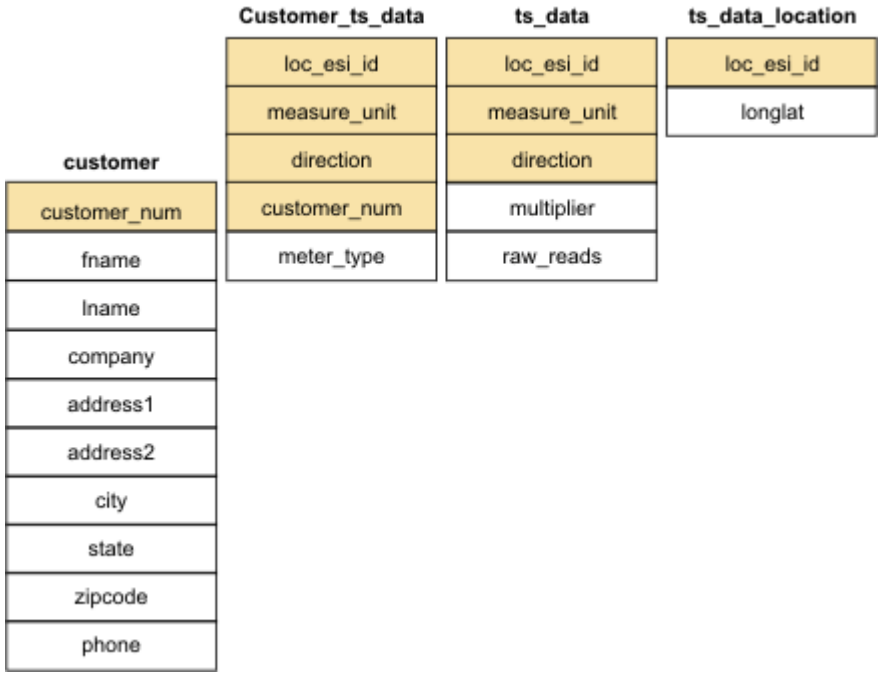
下图显示 **stores_demo** 数据库中客户、目录顺序和客户电话之间的连接。将一个表中的列与另一个表中同名列连接的阴影指示表之间的关系或*连接*。

图: 客户和目录顺序之间的连接



下图显示 **stores_demo** 数据库中客户、电表数据和位置之间的连接。**Customer_ts_data**、**ts_data** 和 **ts_data_location** 表包含时间系列数据。您可以阻止在创建演示数据库时创建这些时间系列表。

图: 客户、用电数据和位置之间的连接



5.2 superstores_demo 数据库

superstores_demo 数据库举例说明了对象关系模式。

随 DB-Access 提供的 SQL 文件和用户定义的例程 (UDR) 允许您派生 ***superstores_demo*** *对象关系数据库。*

superstores_demo *数据库使用缺省语言环境，但并不符合 ANSI 标准。*

有关如何创建和填充演示数据库的信息（包括相关 SQL 文件），请参阅《GBase 8s DB-Access 用户指南》。有关演示数据库的概念性信息，请参阅《GBase 8s 数据库设计和实现指南》。

superstores_demo 表的结构

尽管 superstores_demo 数据库中许多表具有与 stores_demo 表相同的名称，但它们并不相同。

superstores_demo 数据库包括下列表。这些表是按字母顺序（而不是按创建它们的先后顺序）列示的。

- **call_type**
- **catalog**
- **cust_calls**
- **customer**
  - **retail_customer**
  - **whlsale_customer**
- **items**

- **location**
  - **location_non_us**
  - **location_us**
- **manufact**
- **orders**
- **region**
- **sales_rep**
- **state**
- **stock**
- **stock_discount**
- **units**

您可以在 OpenAdmin Tool (OAT) 的模式浏览器中查看表的结构以及它们的数据。

用户定义的例程和扩展数据类型

superstores_demo 数据库使用用户定义的例程 (UDR) 和扩展数据类型。

UDR 是由您定义的例程，可在 SQL 语句或其他 UDR 中进行调用。UDR 可返回值，也可不返回值。

GBase 8s 的数据类型系统是可扩展和灵活的系统，支持创建下列各种数据类型：

- 通过重新定义数据库服务器提供的数据类型的某些行为来扩展现有数据类型。
- 由用户定制的数据类型的定义

有关创建和使用 UDR 和扩展数据类型的信息，请参阅 *GBase 8s 用户定义的例程与数据类型开发者指南*。

如下所示，superstores_demo 数据库在 UDR 中创建*单值*数据类型 percent：

```
CREATE DISTINCT TYPE percent AS DECIMAL(5,5);
DROP CAST (DECIMAL(5,5) AS percent);
CREATE IMPLICIT CAST (DECIMAL(5,5) AS percent);superstores_demo
```

数据库创建下列命名行类型：

- **location** 层次结构：
  - **location_t**
  - **loc_us_t**
  - **loc_non_us_t**
- **customer** 层次结构：
  - **name_t**
  - **customer_t**
  - **retail_t**
  - **whlsale_t**

- **orders** 表
  - **ship_t**

location_t 定义

location_id SERIALloc_type CHAR(2) company VARCHAR(20)

street_addr LIST(VARCHAR(25) NOT NULL)

city VARCHAR(25) country VARCHAR(25)

loc_us_t 定义

state_code CHAR(2) zipROW(code INTEGER, suffix SMALLINT)

phone CHAR(18)

loc_non_us_t 定义

province_code CHAR(2) zipcode CHAR(9) phone CHAR(15)

name_t 定义

first VARCHAR(15) **last** VARCHAR(15)

customer_t 定义

customer_num SERIALcustomer_type CHAR(1)

customer_name name_tcustomer_loc INTEGER

contact_dates LIST(DATETIME YEAR TO DAY NOT NULL)

cust_discount percentcredit_status CHAR(1)

retail_t 定义

credit_num CHAR(19) expiration DATE

whlsale_t 定义

resale_license CHAR(15) terms_net SMALLINT

ship_t 定义

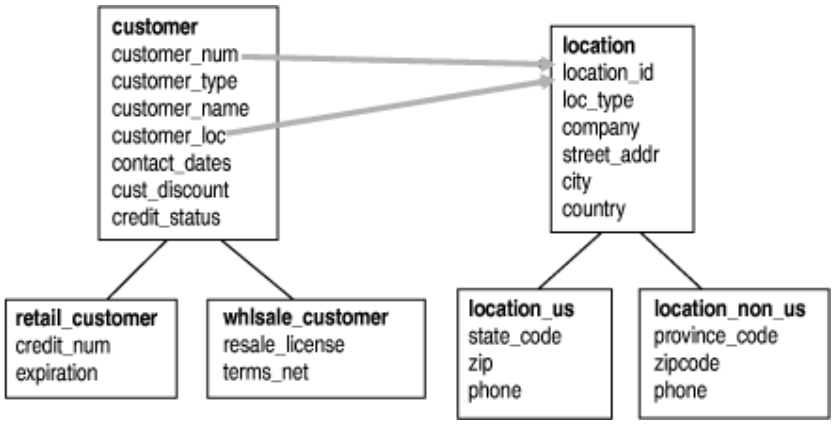
date DATE weight DECIMAL(8,2)

charge MONEY(6,2) instruct VARCHAR(40)

表层次结构

下图显示如何关联 superstores_demo 数据库的分层表。两个表之间的外键和主键关系由从 customer.custnum 和 customer.loc 列指向 location.location_id 列的阴影箭头指示。

图: superstores_demo 表的层次结构


customer
customer_num
customer_type
customer_name
customer_loc
contact_dates
cust_discount
credit_status
location
location_id
loc_type
company
street_addr
city
country
retail_customer
credit_num
expiration
whlsale_customer
resale_license
terms_net
location_us
state_code
zip
phone
location_non_us
province_code
zipcode
phone

GBASE®

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微信二维码

技术支持热线：400-013-9696