HashData

目录

管理指南

欢迎来到 HashData 数据仓库集群管理指南。 HashData 数据仓库是一个高性能，完全托管的 PB 级数据仓库服务。

创建数据仓库的第一步是启动一个由一组节点组成的 HashData 数据仓库集群。当你创建的 HashData 数据仓库集群启动后，你可以加载数据，然后执行数据分析查询。无论数据量大小，HashData 数据仓库都让你能够通过基于 SQL 的工具和商业智能应用获得极速的查询性能。

管理概述

HashData 数据仓库服务将管理所有与数据仓库创建、运维和扩容相关的工作。这些任务包括了准备计算和存储资源，监控和备份集群，以及对引擎打补丁和升级。

集群管理

一个 HashData 数据仓库集群由一组节点组成。这些节点中包括一个主节点和一个或者多个计算节点。你应该根据你的数据集大小和查询的复杂度以及数量来选择需要的计算节点类型和数量。

HashData 数据仓库服务有几个与集群访问和安全相关的功能。

一个数据仓库是一个集群，由一组称为计算节点的计算资源组成。每个数据仓库集群包含一个 HashData 数据仓库引擎和一个或多个数据库。

集群和节点

一个 HashData 数据仓库集群由一组节点组成。每个集群包含一个主节点和一个或多个计算节点。主节点接受来自客户应用的查询，解析查询，以及生成执行计划。然后，主节点调度和协调计算节点并行执行查询计划，汇总中间结果，然后将最终结果返回给客户应用。计算节点执行查询计算并且相互间传递数据来完成查询。中间结果将发送到主节点进行汇总聚合，然后由主节点返回给客户应用。更多关于主节点和计算节点的细节，请参考《开发指南》的 HashData 数据仓库架构 章节。

更改集群配置

当你创建初始集群后，可以改变每个节点的计算资源（CPU数量和内存大小）和存储资源（硬盘空间大小）的方式对集群进行纵向扩容和缩容。

关闭和删除集群

当你希望暂停集群运行和产生新费用的时候，你可以选择将集群关闭。对于关闭的集群，下次你需要使用的时候，可以将它重新启动。如果你的集群和存在数据仓库里面的数据都不需要了，你可以选择删除整个集群。

集群状态

集群状态反应了集群的当前状态。下面的表格详细描述了每种集群状态。

状态	说明
活跃	服务正在运行，并且可用
等待中	创建 HashData 数据仓库请求已经提交了，排队等待创建。
创建中	正在创建 HashData 数据仓库服务。
关闭中	关闭服务时的状态，服务正在关闭。
已关闭	服务已经关闭。
删除中	删除服务时的状态，服务正在被删除。
已删除	集群删除后，在回收站的状态。
更新中	数据仓库正在更新配置。

集群配置

目前 HashData 数据仓库已经在青云的第三方应用商店上线。您可以根据您的使用需求选择不同配置类型和节点数量。

推荐场景	CPU 数量	内存	适合场景
标准版	2 个	4GB	标准版用于开发或者小规模部署
企业版	4 个	8GB	适用于大部分企业的生产环境。
自定义	自定义	自定义	适用于用户特定的生产环境。

存储说明：

存储类型	详细说明
性能型	普通 SATA 存储
超高性能版	SSD 磁盘

集群性能监控

概述

HashData 数据仓库提供了两大类与集群运行状态相关的信息：（1）活跃会话、运行查询等用户级别的信息；（2）节点（包括主节点和计算节点）的资源使用情况。用户级别的信息，您可以通过查看数据库活动情况中的系统视图获取。在这一章节中，我们重点看看如何获得节点级别的信息。通过分析这两类数据，您可以判断系统的性能瓶颈在哪个环节，以及数据仓库集群的类型和大小是否与工作负载相匹配，从而决定是否需要进行横向和纵向的扩缩容。

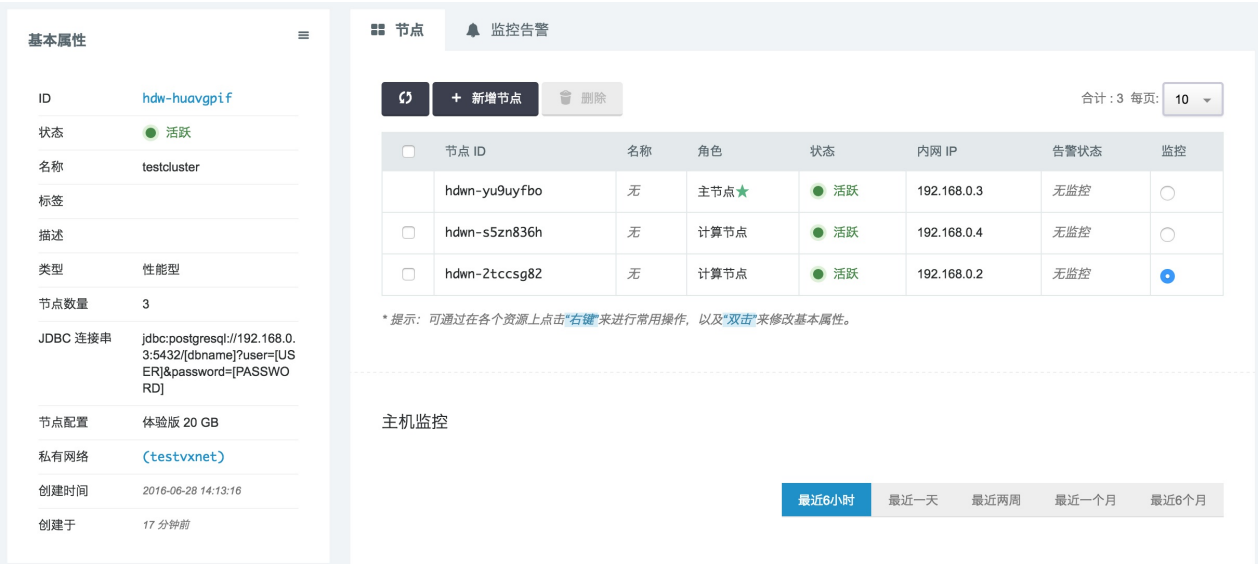
性能数据概要

目前， HashData 数据仓库提供三类节点物理资源监控数据：

- CPU使用率
- 内存使用率
- 硬盘使用率

节点物理资源监控数据

在数据仓库主控制页面，选择您需要感兴趣的数据仓库集群；进入集群详情页面后，选择需要监控的节点：



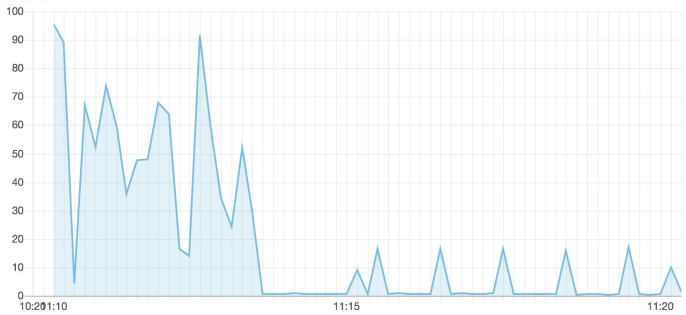
打开“主机监控”开关，您就能看到节点的物理资源监控数据：

CPU

单位: %　　间隔: 5分钟

监控项: CPU

开

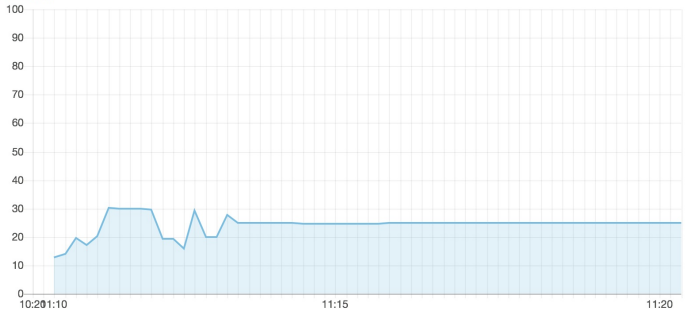


内存

单位: %　　间隔: 5分钟

监控项: 内存

开

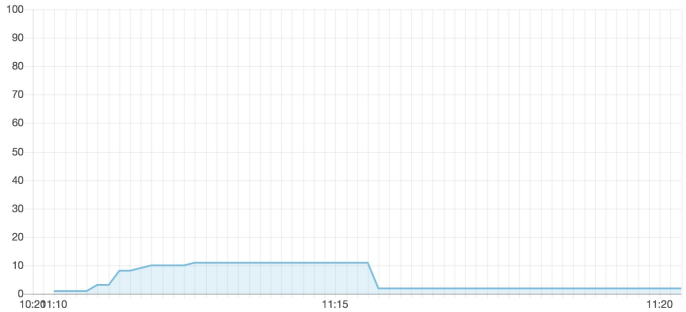


硬盘使用率

单位: %　　间隔: 5分钟

监控项: /data0

开

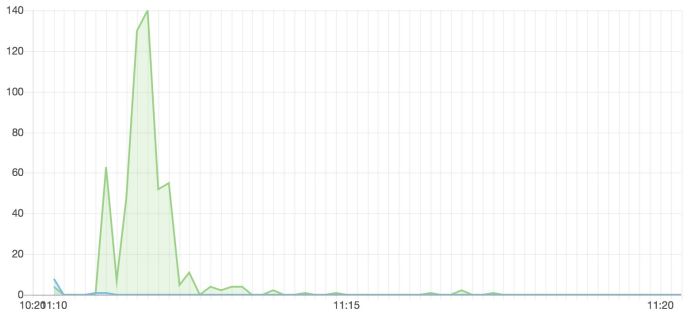


硬盘 IOPS

单位: 硬盘 IOPS　　间隔: 5分钟

监控项: 读 写

开

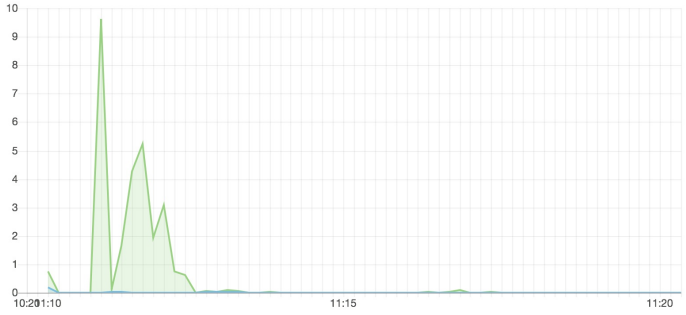


硬盘吞吐量

单位: KB/s　　间隔: 5分钟

监控项: 读 写

开



排查系统问题

如果您在使用 HashData 数据仓库过程中遇到任何问题，欢迎通过应用页中（下图所示）的“创建工单”联系我们。

同时，HashData 数据仓库作为一个自服务的 PaaS 产品，提供多种系统信息，让您能够自己排查系统问题，加快问题解决速度。

查看数据库活动情况

您可以从以下三方面检查数据库状态：

- 活跃的会话（工作负载）
- 锁（竞争）
- 系统资源使用

检查活跃的会话

系统视图 *pg_stat_activity* 提供活跃会话的信息，每行记录对应一个活跃的会话。这些信息包括数据库的 OID，数据库名字，服务器进程的 PID，用户的 OID，用户名字，当前的查询语句，当前查询开始时间，当前服务器进程启动时间，客户端地址，和端口号。为了获取关于当前工作负载最详细的信息，建议您以超级用户的角色去查询这个系统视图。例如：

```
=> SELECT * FROM pg_stat_activity;
```

> 注意：这些信息不是实时更新的。

检查锁

系统视图 *pg_locks* 允许您查看关于正在使用中的锁的信息。如果一个事务拿着一把对应某个对象的锁，那么其它所有需要访问那个对象的查询必须等待这把锁被释放后才能继续往前执行。从用户的角度看，这种现象就像查询被挂起了。

通过检查那些还没被授予的锁，我们能够对不同数据库客户端会话之间的竞争情况有所了解。pg_locks 提供了这个数据库系统全局的锁信息，而不仅仅局限于当前数据库。您可以将锁信息与表（仅限于当前数据库中的表）进行关联，从而了解哪些表正处于竞争状态；您也可以锁信息与 *pg_stat_activity* 进行关联，从而了解哪些客户端会话拿着锁或者正在等待锁。例如：

```
=> SELECT locktype, database, c.relname, l.relation,
   l.transactionid, l.transaction, l.pid, l.mode, l.granted,
   a.current_query
   FROM pg_locks l, pg_class c, pg_stat_activity a
   WHERE l.relation=c.oid AND l.pid=a.procpid
   ORDER BY c.relname;
```

检查系统资源使用

您可以通过上面提到的节点物理资源监控数据来检查系统资源使用情况，包括 CPU，内存，磁盘 I/O 和网络 I/O，从而判断当前系统的性能瓶颈，如当前查询是计算密集型的还是 I/O 密集型的。

取消或中断后台操作

有些情况下，您之前提交的查询运行了很长时间还没有结束，于是想把查询取消了。除了可以通过在客户端发取消信号（如 CTRL + C）外，您还可以通过内置信号函数给服务器后台进程发送取消或者中断信号，从而将查询取消。通过检查活跃的会话我们可以得到查询对应的后台进程 PID，然后通过 pg_cancel_backend(pid int) 和 pg_terminate_backend(pid int) 给 pid 对应的后台进程发送 SIGINT 或者 SIGTERM 信号，从而取消或者中断相应的查询。

系统目录参考

该参考文献描述了 HashData 数据库系统目录表和视图。以 `gp_` 为前缀的系统表与 HashData 数据库的并行特性有关。 以 `pg_` 为前缀的表是 HashData 数据库中支持的标准 PostgreSQL 系统目录表， 或与 HashData 数据库性能有关，以提升 PostgreSQL 的数据仓库工作负载。请注意，HashData 数据库的全局系统目录位于主实例上。

> 警告:不支持对 HashData 数据库系统目录表或视图的更改。如果更改目录表或视图，则必须重新初始化并还原集群

- 系统表
- 系统视图
- 系统目录定义

系统表

- gp_configuration（弃用。请见gp_segment_configuration）。
- gp_configuration_history
- gp_db_interfaces
- gp_distribution_policy
- gp_fastsequence
- gp_fault_strategy
- gp_global_sequence
- gp_id
- gp_interfaces
- gp_master_mirroring（弃用。请见 pg_stat_replication）。
- gp_persistent_database_node
- gp_persistent_filespace_node
- gp_persistent_relation_node
- gp_persistent_tablespace_node
- gp_relation_node
- gp_segment_configuration
- gp_version_at_initdb
- gpexpand.status
- gpexpand.status_detail
- pg_aggregate
- pg_am
- pg_amop
- pg_amproc
- pg_appendonly
- pg_appendonly_alter_column（不支持）
- pg_attrdef
- pg_attribute
- pg_auth_members
- pg_authid
- pg_autovacuum（不支持）
- pg_cast
- pg_class
- pg_constraint
- pg_conversion
- pg_database
- pg_depend
- pg_description
- pg_exttable
- pg_filespace
- pg_filespace_entry
- pg_foreign_data_wrapper（不支持）
- pg_foreign_server（不支持）
- pg_foreign_table（不支持）
- pg_index
- pg_inherits
- pg_language
- pg_largeobject
- pg_listener
- pg_namespace

- pg_opclass
- pg_operator
- pg_partition
- pg_partition_rule
- pg_pltemplate
- pg_proc
- pg_resourcetype
- pg_resqueue
- pg_resqueuecapability
- pg_rewrite
- pg_shdepend
- pg_shdescription
- pg_stat_last_operation
- pg_stat_last_shoperation
- pg_statistic
- pg_tablespace
- pg_trigger
- pg_type
- pg_user_mapping（不支持）
- pg_window

上级主题： 系统目录参考

系统视图

HashData 数据库提供以下系统视图，它们在PostgreSQL中不可用。

- gp_distributed_log
- gp_distributed_xacts
- gp_pgdatabase
- gp_resqueue_status
- gp_transaction_log
- gpexpand.expansion_progress
- pg_max_external_files
- pg_partition_columns
- pg_partition_templates
- pg_partitions
- pg_resqueue_attributes
- pg_resqueue_status（已弃用，请使用gp_toolkit.gp_resqueue_status）
- pg_stat_activity
- pg_stat_replication
- pg_stat_resqueues
- session_level_memory_consumption（见 HashData 数据库管理员指南中的“查看会话内存使用信息”）

有关PostgreSQL和 HashData 数据库支持的标准系统视图的更多信息，请参阅 PostgreSQL 文档的以下部分：

- 系统视图
- 统计收集器视图
- 信息方案

上级主题： 系统目录参考

系统目录定义

按字母顺序排列的系统目录表和视图定义。

- pg_authid
- pg_available_extension_versions
- pg_available_extensions
- pg_cast
- pg_class
- pg_compression
- pg_constraint
- pg_conversion
- pg_database
- pg_depend
- pg_description
- pg_enum
- pg_extension
- pg_exttable
- pg_filespace
- pg_filespace_entry
- pg_index
- pg_inherits
- pg_language
- pg_largeobject
- pg_listener
- pg_locks
- pg_max_external_files
- pg_namespace
- pg_opclass
- pg_operator
- pg_partition
- pg_partition_columns
- pg_partition_encoding
- pg_partition_rule
- pg_partition_templates
- pg_partitions
- pg_pltemplate
- pg_proc
- pg_resourcetype
- pg_resqueue

- pg_resqueue_attributes
- pg_resqueuecapability
- pg_rewrite
- pg_roles
- pg_shdepend
- pg_shdescription
- pg_stat_activity
- pg_stat_last_operation
- pg_stat_last_shoperation
- pg_stat_operations
- pg_stat_partition_operations
- pg_stat_replication
- pg_statistic
- pg_stat_resqueues
- pg_tablespace
- pg_trigger
- pg_type
- pg_type_encoding
- pg_user_mapping
- pg_window

gp_configuration_history

gp_configuration_history 表包含有关故障检测和恢复操作相关的系统更改信息。 fts_probe进程将数据记录到此表，和相关的gpcheck、gprecoverseg和gpinitsystem之类的管理工具一样。 例如，当用户向系统添加新的Segment和镜像Segment时，这些事件会被记录到 gp_configuration_history。

该表仅在Master上填充。该表在 pg_global表空间中定义, 这意味着它在系统中的所有数据库之间全局共享。

表 1. pg_catalog.gp_configuration_history

列	类型	引用	描述
time	timestamp with time zone		记录事件的时间戳。
dbid	smallint	gp_segment_configuration.dbid	系统分配的ID。Segment（或者Master）实例的唯一标识符。
desc	text		时间的文本描述

有关gpcheck、 gprecoverseg 和 gpinitsystem 的信息，请参阅 HashData 数据库工具指南。

上级主题： 系统目录定义

gp_db_interfaces

gp_db_interfaces 表包含有关Segment与网络接口的关系的信息。该信息和 gp_interfaces 的数据连接在一起被系统用于为多种目的网络接口的使用，包括故障检测。

表 1. pg_catalog.gp_db_interfaces

列	类型	引用	描述
dbid	smallint	gp_segment_ configuration.dbid	系统分配的ID。Segment（或Master）实例的唯一标识符。
interfaceid	smallint	gp_interfaces.interfaceid	系统分配的网络接口ID。
priority	smallint		该Segment的网络接口的优先级。

上级主题： 系统目录定义

gp_distributed_log

gp_distributed_log视图包含有关分布式事务及其关联的本地事务的状态信息。分布式事务是涉及修改Segment实例上数据的事务。 HashData 的分布式事务管理器确保了这些Segment保持同步。 此视图允许用户查看分布式事务的状态。

表 1. pg_catalog.gp_distributed_log

列	类型	引用	描述
segment_id	smallint	gp_segment_ configuration.content	如果是Segment，则是其内容ID。Master总是为-1（无内容）。
dbid	small_int	gp_segment_ configuration.dbid	Segment实例的唯一ID。
distributed_xid	xid		全局事务ID。
distributed_id	text		分布式事务的系统分配ID。
status	text		分布式事务的状态（提交或者中止）。
local_transaction	xid		本地事务ID。

上级主题： 系统目录定义

gp_distributed_xacts

gp_distributed_xacts 视图包含有关 HashData 数据库分布式事务的信息。分布式事务是涉及修改 Segment 实例上数据的事务。HashData 的分布式事务管理器确保了这些 Segment 保持同步。此视图允许用户查看当前活动的会话及其关联的分布式事务。

表 1. pg_catalog.gp_distributed_xacts

列	类型	引用	描述
distributed_xid	xid		分布式事务在HashData数据库阵列中使用的事务ID。
distributed_id	text		分布式事务标识符。它有2个部分 - 一个唯一的时间戳和分布式事务编号。
state	text		本次会话对于分布式事务的当前状态。
gp_session_id	int		与此事务关联的HashData数据库会话的ID编号。
xmin_distributed_snapshot	xid		在此事务开始时，所有打开事务中发现的最小分布式事务号。它用于MVCC分布式快照目的。

上级主题： 系统目录定义

gp_distribution_policy

gp_distribution_policy 表包含有关 HashData 数据库表及其分布表数据的策略的信息。 该表仅在 Master 上填充。该表不是全局共享的，这意味着每个数据库都有自己的副本。

表 1. pg_catalog.gp_distribution_policy

列	类型	引用	描述
localoid	oid	pg_class.oid	表对象标识符（OID）。
attrnums	smallint[]	pg_attribute.attnum	分布列的编号。

上级主题： 系统目录定义

gpexpand.expansion_progress

gpexpand.expansion_progress 视图包含系统扩展操作的状态信息。这个视图提供了表的重分布的估算速率和完成重分布操作的估计时间。

扩展中需要的表的详细状态信息在 gpexpand.status_detail 视图中.

列	类型	引用	描述
name	text		Name for the data field provided. Includes:Bytes LeftBytes DoneEstimated Expansion RateEstimated Time to CompletionTables ExpandedTables Left
value	text		The value for the progress data. For example:Estimated Expansion Rate - 9.75667095996092 MB/s

上级主题： 系统目录定义

gpexpand.status

gpexpand.status 表包含有关系统扩展操作状态的信息。扩展中涉及的特定表的状态存储在 gpexpand.status_detail 中。

在正常的扩展操作中，不需要修改存储在该表中的数据。

表 1. gpexpand.status

列	类型	引用	描述
status	text		跟踪扩展操作的状态。有效值为： SETUP SETUP DONE EXPANSION STARTED EXPANSION STOPPED COMPLETED
updated	timestamp with time zone		最后状态变化的时间戳。

上级主题： 系统目录定义

gpexpand.status_detail

gpexpand.status_detail 表包含了系统扩展操作设计到的表的状态信息。你可以通过查询这个表来检查正在扩展的表的状态，或者查看已经完成扩展的表的开始和结束时间。

这个表也包含 oid，磁盘大小，和常规的分布策略及分布键这些相关信息。扩展的总体状态信息存储在 gpexpand.status 中。

在一个常规的扩展操作中，不需要修改这个表中的数据。

列	类型	应用	描述
dbname	text		Name of the database to which the table belongs.
fq_name	text		Fully qualified name of the table.
schema_oid	oid		OID for the schema of the database to which the table belongs.
table_oid	oid		OID of the table.
distribution_policy	smallint()		Array of column IDs for the distribution key of the table.
distribution_policy_names	text		Column names for the hash distribution key.
distribution_policy_coloids	text		Column IDs for the distribution keys of the table.
storage_options	text		Not enabled in this release. Do not update this field.
rank	int		Rank determines the order in which tables are expanded. The expansion utility will sort on rank and expand the lowest-ranking tables first.
status	text		Status of expansion for this table. Valid values are:NOT STARTEDIN PROGRESSFINISHEDNO LONGER EXISTS
last updated	timestamp with time zone		Timestamp of the last change in status for this table.
expansion started	timestamp with time zone		Timestamp for the start of the expansion of this table. This field is only populated after a table is successfully expanded.
expansion finished	timestamp with time zone		Timestamp for the completion of expansion of this table.
source bytes			The size of disk space associated with the source table. Due to table bloat in heap tables and differing numbers of segments after expansion, it is not expected that the final number of bytes will equal the source number. This information is tracked to help provide progress measurement to aid in duration estimation for the end-to-end expansion operation.

上级主题： 系统目录定义

gp_fastsequence

gp_fastsequence 表包含有关追加优化表和列存表的信息。last_sequence 值表示该表当前使用的最大行号。

表 1. pg_catalog.gp_fastsequence

列	类型	引用	描述
objid	oid	pg_class.oid	用于跟踪追加优化文件段的pg_aoseg.pg_aocsseg_*表的对象ID。
objmod	bigint		对象修饰符。
last_sequence	bigint		对象使用的最后一个序列号。

上级主题： 系统目录定义

gp_fault_strategy

The gp_fault_strategy 表指定了故障动作。

表 1. pg_catalog.gp_fault_strategy

列	类型	引用	描述
fault_strategy	char		当Segment失效发生时要执行的镜像故障转移操作：n = nothing； f =基于文件的故障切换

上级主题： 系统目录定义

gp_global_sequence

gp_global_sequence 表包含事务日志中的日志序列号位置，文件复制进程用它来确定要从主 Segment 复制到镜像 Segment 的文件块。

表 1. pg_catalog.gp_global_sequence

列	类型	引用	描述
sequence_num	bigint		日志序列号在事务日志中的位置。

上级主题： 系统目录定义

gp_id

gp_id 系统目录表标识了 HashData 数据库系统名称和系统的 Segment 数量。它还具有表所在的特定数据库实例（Segment 或 Master）的本地值。 该表在pg_global表空间中定义，这意味着它在系统中的所有数据库之间全局共享。

表 1. pg_catalog.gp_id

列	类型	引用	描述
gpname	name		这个 HashData 数据库系统的名称。
numsegments	integer		HashData 数据库系统中的Segment数量。
dbid	integer		此Segment（或Master）实例的唯一标识符。
content	integer		该Segment实例上的这部份数据的ID。主Segment及其镜像Segment将具有相同的内容ID。对于Segment，值为0-*N*，其中*N* 是 HashData 数据库中的Segment数量。对于Master,其值为-1。

上级主题： 系统目录定义

gp_interfaces

gp_interfaces 表包含有关Segment主机上的网络接口的信息。 该信息与gp_db_interfaces的数据连接在一起被系统用来为多种目的优化可用网络接口的使用，包括故障检测。

表 1. gp_interfaces

列	类型	引用	描述
interfaceid	smallint		系统分配的ID。 网络接口的唯一标识符。
address	name		包含网络接口的Segment主机的主机名地址。可以是数字IP地址或主机名。
status	smallint		网络接口的状态。值为0表示接口不可用。

上级主题： 系统目录定义

gp_persistent_database_node

gp_persistent_database_node 表跟踪与数据库对象的事务状态有关的文件系统对象状态。此信息用于确保系统目录以及持久化到磁盘的文件系统文件的状态保持同步。该信息由从主机到镜像机的文件复制过程使用。

表 1. pg_catalog.gp_persistent_database_node

列	类型	引用	描述
tablespace_oid	oid	pg_tablespace.oid	表空间对象ID。
database_oid	oid	pg_database.oid	数据库对象ID。
persistent_state	smallint		0 - 空闲 ；1 - 待定创建 ；2 - 已创建 ；3 - 待定删除 ；4 - 中止创建 ；5 - “即时”待定创建 ；6 - 待定批量加载创建
mirror_existence_state	smallint		0 - 无镜像 ；1 - 未被镜像 ；2 - 镜像创建待定 ；3 - 镜像已创建 ；4 - 创建前镜像宕机 ；5 - 创建期间镜像宕机 ；6 - 镜像删除待定 ；7 - 仅剩镜像删除
parent_xid	integer		全局事务ID。
persistent_serial_num	bigint		文件块在事务日志中的日志序列号位置。
previous_free_tid	tid		被 HashData 数据库用于在内部管理文件系统对象的持久表示。

上级主题： 系统目录定义

gp_persistent_filespace_node

gp_persistent_filespace_node 表跟踪与文件空间对象的事务状态有关的文件系统对象状态。此信息用于确保系统目录以及持久化到磁盘的文件系统文件的状态保持同步。该信息由从主机到镜像机的文件复制过程使用。

表 1. pg_catalog.gp_persistent_filespace_node

列	类型	引用	描述
filespace_oid	oid	pg_filespace.oid	文件空间的对象ID
db_id_1	smallint		主Segment的ID
location_1	text		主文件系统位置
db_id_2	smallint		镜像Segment的ID
location_2	text		镜像文件系统位置
persistent_state	smallint		0 - 空闲 ；1 - 待定创建 ；2 - 已创建 ；3 - 待定删除 ；4 - 中止创建 ；5 - “即时”待定创建 ；6 - 待定批量加载创建
mirror_existence_state	smallint		0 - 无镜像 ；1 - 未被镜像 ；2 - 镜像创建待定 ；3 - 镜像已创建 ；4 - 创建前镜像宕机 ；5 - 创建期间镜像宕机 ；6 - 镜像删除待定 ；7 - 仅剩镜像删除
parent_xid	integer		全局事务ID。
persistent_serial_num	bigint		文件块在事务日志中的日志序列号位置。
previous_free_tid	tid		被 HashData 数据库用于在内部管理文件系统对象的持久表示。

上级主题： 系统目录定义

gp_persistent_relation_node

gp_persistent_relation_node 表跟踪与关系对象（表、视图、索引等）的事务状态有关的文件系统对象状态。此信息用于确保系统目录以及持久化到磁盘的文件系统文件的状态保持同步。该信息由从主机到镜像机的文件复制过程使用。

表 1. pg_catalog.gp_persistent_relation_node

列	类型	引用	描述
tablespace_oid	oid	pg_tablespace.oid	表空间的对象ID
database_oid	oid	pg_database.oid	数据库对象ID
relfilenode_oid	oid	pg_class.relfilenode	关系文件节点的对象ID。
segment_file_num	integer		对于追加优化的表，追加优化的Segment文件编号。
relation_storage_manager	smallint		关系是堆存储还是追加优化存储。
persistent_state	smallint		0 - 空闲 ；1 - 待定创建 ；2 - 已创建 ；3 - 待定删除 ；4 - 中止创建 ；5 - “即时”待定创建 ；6 - 待定批量加载创建
mirror_existence_state	smallint		0 - 无镜像 ；1 - 未被镜像 ；2 - 镜像创建待定 ；3 - 镜像已创建 ；4 - 创建前镜像宕机 ；5 - 创建期间镜像宕机 ；6 - 镜像删除待定 ；7 - 仅剩镜像删除
parent_xid	integer		全局事务ID。
persistent_serial_num	bigint		文件块在事务日志中的日志序列号位置。
previous_free_tid	tid		被 HashData 数据库用于在内部管理文件系统对象的持久表示。

上级主题： 系统目录定义

gp_persistent_tablespace_node

gp_persistent_tablespace_node 表跟踪与表空间对象的事务状态有关的文件系统对象状态。此信息用于确保系统目录以及持久化到磁盘的文件系统文件的状态保持同步。该信息由从主机到镜像机的文件复制过程使用。

表 1. pg_catalog.gp_persistent_tablespace_node

列	类型	引用	描述
filespace_oid	oid	pg_filespace.oid	文件空间的对象ID
tablespace_oid	oid	pg_tablespace.oid	表空间的对象ID
persistent_state	smallint		0 - 空闲 ；1 - 待定创建 ；2 - 已创建 ；3 - 待定删除 ；4 - 中止创建 ；5 - “即时”待定创建 ；6 - 待定批量加载创建
mirror_existence_state	smallint		0 - 无镜像 ；1 - 未被镜像 ；2 - 镜像创建待定 ；3 - 镜像已创建 ；4 - 创建前镜像宕机 ；5 - 创建期间镜像宕机 ；6 - 镜像删除待定 ；7 - 仅剩镜像删除
parent_xid	integer		全局事务ID。
persistent_serial_num	bigint		文件块在事务日志中的日志序列号位置。
previous_free_tid	tid		被 HashData 数据库用于在内部管理文件系统对象的持久表示。

上级主题： 系统目录定义

gp_pgdatabase

gp_pgdatabase 视图显示 HashData 中 Segment 实例的状态信息，以及它们是作为镜像 Segment 还是主 Segment。 HashData 的故障检测和恢复工具在内部使用此视图来确定失效的 Segment。

表 1. pg_catalog.gp_pgdatabase

列	类型	引用	描述
dbid	smallint	gp_segment_configuration.dbid	系统分配的ID。Segment（或者Master）实例的唯一标识符。
isprimary	boolean	gp_segment_configuration.role	该实例是否处于活动状态。它目前是否作为主Segment（还是镜像Segment）。
content	smallint	gp_segment_ configuration.content	实例上部分数据的ID。主Segment实例及其镜像将具有相同的内容ID。对于Segment，值为 0- *N*, 其中 *N* 是 HashData 数据库中的Segment数量。对于Master，值为-1。
definedprimary	boolean	gp_segment_ configuration.preferred_role	在初始化系统时，该实例是否被定义为主Segment（而不是镜像Segment）。

上级主题： 系统目录定义

gp_relation_node

gp_relation_node 表包含有关关系（表、视图、索引等）的文件系统对象的信息。

表 1. pg_catalog.gp_relation_node

列	类型	引用	描述
relfilenode_oid	oid	pg_class.relfilenode	关系文件节点的对象ID。
segment_file_num	integer		对于追加优化的表，追加优化的Segment文件编号。
persistent_tid	tid		被 HashData 数据库用于在内部管理文件系统对象的持久表示。
persistent_serial_num	bigint		文件块在事务日志中的日志序列号位置。

上级主题： 系统目录定义

gp_resqueue_status

gp_toolkit.gp_resqueue_status视图允许管理员查看负载管理资源队列的状态和活动。它显示系统中有多少来自特定资源队列的查询操作正在等待运行，以及有多少个活动查询来自于特定的资源队列。

表 1. gp_toolkit.gp_resqueue_status

列	类型	引用	描述
queueid	oid	gp_toolkit.gp_resqueue_ queueid	资源队列的ID。
rsqname	name	gp_toolkit.gp_resqueue_ rsqname	资源队列的名称。
rsqcountlimit	real	gp_toolkit.gp_resqueue_ rsqcountlimit	资源队列的活动查询阈值。值-1表示无限制。
rsqcountvalue	real	gp_toolkit.gp_resqueue_ rsqcountvalue	资源队列中当前正被使用的活动查询槽的数量。
rsqcostlimit	real	gp_toolkit.gp_resqueue_ rsqcostlimit	资源队列的查询代价阈值。值-1表示无限制。
rsqcostvalue	real	gp_toolkit.gp_resqueue_ rsqcostvalue	资源队列中当前所有语句的总代价。
rsqmemorylimit	real	gp_toolkit.gp_resqueue_ rsqmemorylimit	资源队列的内存限制。
rsqmemoryvalue	real	gp_toolkit.gp_resqueue_ rsqmemoryvalue	资源队列中当前所有语句使用的总内存。
rsqwaiters	integer	gp_toolkit.gp_resqueue_ rsqwaiter	资源队列中正在等待的语句数。
rsqholders	integer	gp_toolkit.gp_resqueue_ rsqholders	目前在系统中运行的来自该资源队列的语句数目。

上级主题： 系统目录定义

gp_segment_configuration

gp_segment_configuration 表包含有关镜像 Segment 和 Segment 配置的信息。

表 1. pg_catalog.gp_segment_configuration

列	类型	引用	描述
dbid	smallint		Segment（或Master）实例的唯一标识符。
content	smallint		Segment实例的内容标识符。主Segment实例及其镜像将始终具有相同的内容标识符。对于Segment，值为0-*N*，其中 *N* 是系统中主Segment的数量。对于Master，该值始终为-1。
role	char		Segment当前的角色。值是p（主）或者m（镜像）。
preferred_role	char		在初始化时为Segment初始分配的角色。值为p（主）或者m（镜像）。
mode	char		Segment与其镜像副本的同步状态。值是s（已同步）、c（记录更改）或者r（重新同步中）。
status	char		Segment的故障状态。值是u（在线）或者d（离线）。
port	integer		数据库服务器监听器进程正在使用的TCP端口。
hostname	text		Segment主机的主机名。
address	text		用于访问Segment主机上的特定Segment的主机名。在从3.x升级而来的系统上或者没有为每个接口配置主机名的系统上，该值可能与hostname相同。
replication_port	integer		块复制进程用来保持主Segment和镜像Segment同步的TCP端口。

上级主题： 系统目录定义

gp_transaction_log

gp_transaction_log 视图包含特定段的本地事务的状态信息。此视图允许用户查看本地事务的状态。

表 1. pg_catalog.gp_transaction_log

列	类型	引用	描述
segment_id	smallint	gp_segment_ configuration.content	如果是Segment，则是内容ID。Master总为-1 (没有内容)。
dbid	smallint	gp_segment_ configuration.dbid	Segment实例的唯一ID。
transaction	xid		本地事务的ID。
status	text		本地事务的状态（提交或者中止）。

上级主题： 系统目录定义

gp_version_at_initdb

gp_version_at_initdb表在 HashData 数据库系统的Master和每一个Segment上被填充。它标识着系统初始化时使用的 HashData 数据库版本。 这个表被定义在pg_global 表空间中，意味着它在系统中的所有数据库中全局共享。

表 1. pg_catalog.gp_version

列	类型	引用	描述
schemaversion	integer		模式版本号。
productversion	text		产品版本号。

上级主题： 系统目录定义

pg_aggregate

pg_aggregate 存储关于聚集函数的信息。聚集函数是对一个值集合（通常是来自于匹配某个查询条件的每个行的一个列值）进行操作并且返回从这些值计算出的一个值的函数。 典型的聚集函数是 sum、count 和 max。 pg_aggregate 里的每个项都是一个 pg_proc 项的扩展。pg_proc的项记载该聚集的名字、输入输出的数据类型以及其他和普通函数类似的信息。

表 1. pg_catalog.pg_aggregate

列	类型	引用	描述
aggfnoid	regproc	pg_proc.oid	聚集函数的OID
aggtransfn	regproc	pg_proc.oid	转移函数的OID
aggprelimfn	regproc		预备函数的OID（如果没有就为0）
aggfinalfn	regproc	pg_proc.oid	最终函数的OID（如果没有就为0）
agginitval	text		转移状态的初始值。这是一个文本域，它包含初始值的外部字符串表现形式。 如果这个域为NULL，则转移状态从NULL开始
agginvtransfn	regproc	pg_proc.oid	*aggtransfn* 的反函数在 pg_proc 中的OID
agginvprelimfn	regproc	pg_proc.oid	*aggprelimfn* 的反函数在pg_proc中的OID
aggordered	Boolean		如果为true，则聚集定义为 ORDERED。
aggsortop	oid	pg_operator.oid	相关的排序操作符的OID（如果没有则为零）
aggtranstype	oid	pg_type.oid	聚集函数的内部转移（状态）数据的数据类型

上级主题： 系统目录定义

pg_am

pg_am 表存储有关索引访问方法的信息。系统所支持的每一种索引访问方法都有一行。

表 1. pg_catalog.pg_am

列	类型	引用	描述
amname	name		访问方法的名称
amstrategies	int2		此访问方法的操作符策略数量
amsupport	int2		此访问方法的支持例程数量
amorderstrategy	int2		如果索引不提供排序顺序，则为零。否则是描述排序顺序的策略操作符的策略编号
amcanunique	boolean		访问方法是否支持唯一索引？
amcanmulticol	boolean		访问方法是否支持多列索引？
amoptionalkey	boolean		访问方法是否支持对第一个索引列没有任何约束的扫描？
amindexnulls	boolean		访问方法是否支持空索引条目？
amstorage	boolean		索引存储数据类型能否与列数据类型不同？
amclusterable	boolean		能否以这种类型的索引作聚簇？
aminsert	regproc	pg_proc.oid	“插入此元组”函数
ambeginscan	regproc	pg_proc.oid	“开始新扫描”函数
amgettuple	regproc	pg_proc.oid	“下一个有效的元组”函数
amgetmulti	regproc	pg_proc.oid	“获取多个元组”函数
amrescan	regproc	pg_proc.oid	“重新启动扫描”函数
amendscan	regproc	pg_proc.oid	“结束扫描”函数
ammarkpos	regproc	pg_proc.oid	“标记当前扫描位置”函数
amrestrpos	regproc	pg_proc.oid	“恢复已标记扫描位置”函数
ambuild	regproc	pg_proc.oid	“建立新的索引”函数
ambulkdelete	regproc	pg_proc.oid	批量删除函数
amvacuumcleanup	regproc	pg_proc.oid	后-VACUUM清理函数
amcostestimate	regproc	pg_proc.oid	估计索引扫描代价的函数
amoptions	regproc	pg_proc.oid	为索引解析和验证reloptions的函数

上级主题： 系统目录定义

pg_amop

pg_amop 表存储关于与访问方法操作符类相关的操作符的信息。对于每一个操作符类中的成员即操作符都有一行。

表 1. pg_catalog.pg_amop

列	类型	引用	描述
amopclaid	oid	pg_opclass.oid	此项所对应的索引操作符类
amopsubtype	oid	pg_type.oid	用于区分一个策略的多个项的子类型，默认为零
amopstrategy	int2		操作符策略号
amopreqcheck	boolean		索引命中必须被复查
amopopr	oid	pg_operator.oid	操作符的OID

上级主题： 系统目录定义

pg_amproc

该 pg_amproc 表存储与索引访问操作符相关的支持过程的信息。对于属于一个操作符类的每个支持过程都有一行。

表 1. pg_catalog.pg_amproc

列	类型	引用	描述
amopclaid	oid	pg_opclass.oid	该条目对应的索引操作符类
amprocsubtype	oid	pg_type.oid	如果是跨类型例程，则为子类型，否则为0
amprocnum	int2		支持过程编号
amproc	regproc	pg_proc.oid	过程的OID

上级主题： 系统目录定义

pg_appendonly

pg_appendonly 表包含有关追加优化表的储存选项和其他特性的信息。

表 1. pg_catalog.pg_appendonly

列	类型	引用	描述
relid	oid		压缩表的表对象标识符（OID）。
blocksize	integer		用于追加优化表压缩的块尺寸。有效值为8K - 2M。默认值为32K。
safefswritesize	integer		在非成熟文件系统中追加优化表的安全写操作的最小尺寸。通常设置为文件系统扩展块尺寸的倍数，例如Linux ext3是4096字节，所以通常使用32768的值。
compresslevel	smallint		压缩级别，压缩比从1增加到9。
majorversion	smallint		pg_appendonly表的主要版本号。
minorversion	smallint		pg_appendonly表的次要版本号。
checksum	boolean		存储的校验和值，在压缩和扫描时用来比较数据块的状态，以确保数据完整性。
compresstype	text		用于压缩追加优化表的压缩类型。 有效值为： zlib（gzip压缩）
columnstore	boolean		1为列存，0为行存。
segrelid	oid		表在磁盘上的Segment文件ID。
segidxid	oid		索引在磁盘上的Segment文件ID。
blkdirrelid	oid		用于磁盘上列存表文件的块。
blkdiridxid	oid		用于磁盘上列存索引文件的块。
visimaprelid	oid		表的可见性映射。
visimapidxid	oid		可见性映射上的B-树索引。

上级主题： 系统目录定义

pg_attrdef

该 pg_attrdef 表储存列的默认值，关于列的主要信息储存在pg_attribute中。只有那些明确指定了默认值的列（在创建表或者添加列时）才会在该表中有一项。

表 1. pg_catalog.pg_attrdef

列	类型	引用	描述
adrelid	oid	pg_class.oid	该列所属的表
adnum	int2	pg_attribute.attnum	列编号
adbin	text		列默认值的内部表示
adsrc	text		默认值的人类可读的表示，这个字段是历史遗留下来的，最好别用。

上级主题： 系统目录定义

pg_attribute_encoding

该 pg_attribute_encoding 系统目录表包含列存储信息。

表 1. pg_catalog.pg_attribute_encoding

列	类型	修饰符	存储	描述
attrelid	oid	not null	plain	外键于pg_attribute.attrelid
attnum	smallint	not null	plain	外键于pg_attribute.attnum
attoptions	text [ ]		extended	选项（The options）

上级主题： 系统目录定义

pg_attribute

该 pg_attribute 存储关于表列的信息。数据库中每个表的每一列都正好对应 pg_attribute 表的一行（还有有索引的属性项，以及所有有 pg_class 项的对象的属性）。术语属性等效于列。

表 1. pg_catalog.pg_attribute

列	类型	引用	描述
attrelid	oid	pg_class.oid	该列所属的表
attname	name		列名
atttypid	oid	pg_type.oid	该列的数据类型
attstattarget	int4		控制由ANALYZE为此列积累的统计信息的详细程度。0值表示不应收集统计信息。负值表示使用系统默认的统计信息目标。正值的确切含义依赖于数据类型。对于标量数据类型，它既是要收集的“最常用值”的目标，也是要创建的柱状图的目标。
attlen	int2		该列类型的pg_type.typlen的副本。
attnum	int2		列编号，普通列从1开始编号。系统列（如OID），具有（任意）负编号。
attndims	int4		如果列是一个数组类型则是维度数；否则为 0（目前，数组的维数不是强制的，所以任何非0值都能有效地表示它为一个数组)。
attcacheoff	int4		在存储中始终为-1 ，但是当加载到内存中的行描述符时，这可能会被更新以缓存该属性在行中的偏移量。
atttypmod	int4		记录在创建时提供的特定类型的数据（例如，varchar列的最大长度）。它被传递到特定类型的输入函数和长度强制函数。对于不需要它的类型，该值通常为-1。
attbyval	boolean		该列类型的pg_type.typbval副本。
attstorage	char		通常是该列类型的pg_type.typstorage副本。对于可TOAST的数据类型来说，可以在列创建之后更改这些数据类型，以控制存储策略。
attalign	char		该列类型的pg_type.typalign副本
attnotnull	boolean		这表示一个非空约束。可以更改此列以启用或禁用该约束。
atthasdef	boolean		此列具有默认值，这种情况下， 将在pg_attrdef目录中存在相应的条目实际定义默认值。
attisdropped	boolean		该列已被删除，不再有效。已删除的列仍然物理存在于表中，但是会被解析器忽略，所以无法通过SQL访问。
attislocal	boolean		该列在关系中本地定义。 请注意，列可以同时在本地定义和继承。
attinhcount	int4		这列的直接祖先的数量。具有非0数量祖先的列不能被删除或重命名。

上级主题： 系统目录定义

pg_auth_members

pg_auth_members 系统目录表显示角色之间的成员关系。允许任何非循环关系。因为角色是系统范围的，所以pg_auth_members表在Greenplum数据库系统的所有数据库之间共享。

表 1. pg_catalog.pg_auth_members

列	类型	参考	描述
roleid	oid	pg_authid.oid	父级（组）角色的ID
member	oid	pg_authid.oid	成员角色的ID
grantor	oid	pg_authid.oid	授予此成员关系的角色的ID
admin_option	boolean		如果角色成员可以向其他人授予成员关系，则为true

上级主题： 系统目录定义

pg_authid

pg_authid 表包含了有关数据库认证标识符（角色）的信息。角色包含用户和组的概念。用户是设置了rolcanlogin标志的角色。任何角色（有或者没有 rolcanlogin）可能有其他角色作为其成员。请参阅 pg_auth_members。

由于此目录包含密码，因此不是公众可读的。pg_roles是pg_authid 上的一个公开可读的视图，其中模糊化了密码字段。

由于用户身份是全系统范围的，因此pg_authid在 HashData 数据库系统中的所有数据库之间共享：每个系统只有一个pg_authid副本，而不是每个数据库一个。

表 1. pg_catalog.pg_authid

列	类型	参考	描述
rolname	name		角色名称
rolsuper	boolean		角色具有超级用户特权
rolinherit	boolean		角色自动继承其所属角色的权限
rolcreaterole	boolean		角色可以创建其他更多角色
rolcreatedb	boolean		角色可以创建数据库
rolcatupdate	boolean		角色可以直接更新系统目录（即使超级用户也不能这样做，除非此列为真）
rolcanlogin	boolean		角色可以登录。也就是说，该角色可以作为初始会话授权标识符。
rolconnlimit	int4		对于那些可以登录的角色，这一列设置此角色可以创建的最大并发连接数。-1表示没有限制。
rolpassword	text		密码（可能加密）；如果没有则为NULL
rolvaliduntil	timestamptz		密码到期时间（仅用于密码验证）；不过期则为NULL
rolconfig	text[]		服务器配置参数的会话默认值
relresqueue	oid		*pg_resqueue* 中相关的资源队列ID的对象ID。
rolcreaterextgpfd	boolean		使用gpfdist或gpfdists协议创建可读外部表的特权。
rolcreaterexhttp	boolean		使用http协议创建可读外部表的特权。
rolcreatewextgpfd	boolean		使用gpfdist或gpfdists协议创建可写外部表的特权。
rolcreaterexthdfs	boolean		使用gphdfs协议协议创建可读外部表的特权。
rolcreatewexthdfs	boolean		使用gphdfs协议创建可写外部表的特权。

上级主题： 系统目录定义

pg_available_extension_versions

该 pg_available_extension_versions 视图列出了可用于安装的特定扩展版本。该 pg_extension 系统目录表 当前已经安装的扩展。

该视图只读。

表 1. pg_catalog.pg_available_extension_versions

列	类型	描述
name	name	扩展名字。
version	text	版本名字。
installed	boolean	如果此扩展程序的此版本已安装，则为 True ，否则为 False。
superuser	boolean	如果只允许超级用户安装此扩展，则为 True，否则为False。
relocatable	boolean	如果扩展可以重新定位到另一个模式，则为 True，否则为False。
schema	name	扩展必须装入的模式的名称，如果部分或完全可重定位，则为 NULL。
requires	name[]	必备扩展的名称，如果没有则为 NULL。
comment	text	扩展控制文件的注释字符串。

上级主题： 系统目录定义

pg_available_extensions

该 pg_available_extensions 视图列出了可用于安装的扩展。该 pg_extension 系统目录表显示当前安装的扩展。

该视图只读。

表 1. pg_catalog.pg_available_extensions

列	类型	描述
name	name	扩展名
default_version	text	默认版本名，如果没指定的话为NULL。
installed_version	text	当前安装的扩展的版本，如果未安装则为 NULL。
comment	text	扩展控制文件的注释字符串。

上级主题： 系统目录定义

pg_cast

pg_cast 表存储数据类型转换路径，包括内置路径和用 CREATE CAST 定义的路径。 pg_cast 中列出的造型函数必须始终以造型的源类型作为其第一个参数类型，并且返回造型的目标类型作为其结果类型。造型函数最多可以有三个参数。第二个参数如果存在，必须是整数类型，它接受与目标类型相关的类型修饰符；如果没有，则返回-1。第三个参数如果存在，必须是布尔类型；如果转换是显式的，则为true，否则为false。

如果关联函数不止一个参数，则创建一个源和目标类型相同的pg_cast项是合法的。这种项表示“长度强制函数”，它们能把该类型的值强制为对特定类型修饰符值合法的值。但是注意，目前不支持将非默认类型修饰符和用户创建的数据类型相关联，因此该功能仅适用于少量类型修饰符被内置在语法中的内建类型。

当pg_cast项具有不同的源和目标类型，并且具有多个参数的函数时，它表示从一种类型转换为另一种类型，并在一个步骤中应用长度强制。当没有这样的项可用时，使用类型修饰符强制到一种类型需要两个步骤，第一个用于在数据类型之间的转换，第二个应用修饰符。

表 1. pg_catalog.pg_cast

列	类型	参考	描述
castsource	oid	pg_type.oid	源数据类型的OID
casttarget	oid	pg_type.oid	目标数据类型的OID
castfunc	oid	pg_proc.oid	用于执行此造型的函数的OID，如果数据类型是二进制兼容的（即不需要运行时操作来执行造型）则存储0。
castcontext	char		指示应该在什么上下文中调用造型。e表示只能作为显式造型（使用CAST或者::语法）。a意味着隐式或者显式地为目标列赋值。i表示隐式地用在表达式以及其他情况中。

上级主题： 系统目录定义

pg_class

系统目录表 pg_class 记录表以及其他大部分具有列或者与表（也称为关系）相似的东西。这包括索引（另见 pg_index）、序列、视图、组合类型和 TOAST 表。并不是所有的列对所有的关系类型都有意义。

表 1. pg_catalog.pg_class

列	类型	参考	描述
relname	name		表、索引、视图等的名字。
relnamespace	oid	pg_namespace.oid	包含这个关系的命名空间（方案）的OID
reltype	oid	pg_type.oid	如果有的话（索引为0，没有pg_type项），对应与此表的行类型的数据类型的OID
relowner	oid	pg_authid.oid	关系的所有者
relam	oid	pg_am.oid	如果这是一个索引，则表示访问方法（B树、位图、哈希等。）
relfilenode	oid		此关系的磁盘文件的名称，如果没有则为0。
reltablespace	oid	pg_tablespace.oid	存储此关系的表空间。如果为0，则表示数据库的默认表空间（如果关系没有磁盘文件，则无意义）。
relpages	int4		该表的磁盘尺寸，以页面为单位（每页32k）。这只是规划器使用估计值。它由VACUUM、ANALYZE和一些DDL命令更新。
reltuples	float4		表中的行数。这只是规划器使用的估计值。它由VACUUM、ANALYZE和一些DDL命令更新。
reltoastrelid	oid	pg_class.oid	与这张表关联的TOAST表的OID，没有的就为0。TOAST在一张二级表中存储普通行不能存储的大属性。
reltoastidxid	oid	pg_class.oid	对于TOAST表，其上的索引的OID。如果不是TOAST表则为0。
relaosegidxid	oid		已弃用。
relaosegrelid	oid		已弃用。
relhasindex	boolean		如果这是一个表，它有（或最近有）任何索引，则为true。这由CREATE INDEX设置，但不会被DROP INDEX立刻清除。VACUUM 如果没有找到表的索引，则会清除这个属性。
relisshared	boolean		如果此表在系统中的所有数据库中共享，则为true。只有某些系统目录表被共享。
relkind	char		对象的类型,r = 堆或追加优化表， i = 索引，S = 序列，v = 视图，c = 组合类型，t = TOAST值， o = 内部的追加优化Segment文件和EOF，u = 未登记的临时堆表。
relstorage	char		表的存储模式,a= 追加优化，c= 列存，h = 堆， v = 虚拟，x= 外部表。
relnatts	int2		关系中用户列的数量（系统列不计入）。pg_attribute中必须有这么多个相应的项。
relchecks	int2		表中检查约束的个数。
reltriggers	int2		表中触发器的个数。
relukeys	int2		未使用
relfkeys	int2		未使用
relrefs	int2		未使用
relhasoids	boolean		如果为关系的每一行生成OID即为真。

relhaspkey	boolean		如果该表有（或曾经有）主键则为真。
relhasrules	boolean		如果表有规则则为真
relhassubclass	boolean		如果表有（或曾经有）任何继承的子代，则为真。
relfrozenxid	xid		表中在此之前的所有事务的ID已替换为的永久（冻结）事务ID。这用于跟踪该表是否需要清理以防止事务ID回卷或允许pg_clog收缩。如果关系不是表，则为0（InvalidTransactionId）。
relacl	aclitem[]		由GRANT和REVOKE分配的访问权限。
reloptions	text[]		访问方法相关的选项，为形如“键=值"的字符串。

上级主题： 系统目录定义

pg_compression

该 pg_compression 系统目录表描述了可用的压缩方法。

表 1. pg_catalog.pg_compression

列	类型	修饰符	存储	描述
compname	name	not null	plain	压缩的名字
compconstructor	regproc	not null	plain	压缩构造函数的名称
compdestructor	regproc	not null	plain	压缩销毁函数的名称
compcompressor	regproc	not null	plain	压缩机的名称
compdecompressor	regproc	not null	plain	解压缩器的名称
compvalidator	regproc	not null	plain	压缩验证器的名称
compowner	oid	not null	plain	来自pg_authid的Oid

上级主题： 系统目录定义

pg_constraint

pg_constraint 系统目录表储存表上的检查、主键、唯一和外键约束。列约束不被特别对待。每个列约束都等效于某种表约束。非空约束在pg_attribute目录表中表示。域上的检查约束也储存在这里。

表 1. pg_catalog.pg_constraint

列	类型	参考	描述
conname	name		约束名称（不一定唯一！）
connamespace	oid	pg_namespace.oid	包含约束的命名空间（方案）的OID
contype	char		c = 检查约束，f = 外键约束，p = 主键约束， u = 唯一约束。
condeferrable	boolean		约束是否可以延迟？
condeferred	boolean		约束是否默认延迟？
conrelid	oid	pg_class.oid	约束所在的表；如果不是一个表约束则为0。
contypid	oid	pg_type.oid	该约束所在的域；如果不是一个域约束则为0。
confrelid	oid	pg_class.oid	如果是一个外键，则表示所引用的表；否则为0。
confupdtype	char		外键更新动作代码。
confdeltype	char		外键删除动作代码。
confmatchtype	char		外键匹配类型。
conkey	int2[]	pg_attribute.attnum	如果是一个表约束，则表示所约束的列的列表。
confkey	int2[]	pg_attribute.attnum	如果是一个外键，则表示所引用列的列表。
conbin	text		如果是一个检查约束，则表示表达式的内部表示。
consrc	text		如果是一个检查约束，则表示表达式的人类可读的表示。当被引用对象改变时候，这个属性不会被更新；例如，它不会跟踪列的重命名。最好使用pg_get_constraintdef()来提取检查约束的定义，而不是依赖此字段。

上级主题： 系统目录定义

pg_conversion

pg_conversion 系统目录表描述了由 CREATE CONVERSION 定义的可用编码转换过程。

表 1. pg_catalog.pg_conversion

列	类型	参考	描述
conname	name		转换的名称（在命名空间中唯一）。
connamespace	oid	pg_namespace.oid	包含此转换的命名空间（方案）的OID。
conowner	oid	pg_authid.oid	转换的拥有者。
conforencoding	int4		源编码ID。
contoencoding	int4		目标编码ID。
conproc	regproc	pg_proc.oid	转换过程。
condefault	boolean		如果是默认转换，则为真。

上级主题： 系统目录定义

pg_database

pg_database 系统目录表储存了可用数据库的信息。数据库由 SQL 命令 CREATE DATABASE 创建。和大多数系统目录不同，pg_database 在系统中所有数据库之间共享。每个系统只有一个 pg_database 副本，而不是每个数据库一个。

表 1. pg_catalog.pg_database

列	类型	参考	描述
datname	name		数据库名称。
datdba	oid	pg_authid.oid	数据库的拥有者，通常是创建它的人。
encoding	int4		数据库的字符编码。 pg_encoding_to_char()可以将此编号转换为编码名称。
datistemplate	boolean		如果为真则这个数据库可以被用在 CREATE DATABASE 的 TEMPLATE 子句中来创建一个新的数据库作为这个数据库的克隆体。
datallowconn	boolean		如果为假，则该数据库不可连接。这用于保护数据库 template0 不被修改。
datconnlimit	int4		设置该数据库最大并发连接数。-1表示没有限制。
datlastsysoid	oid		数据库中的最后一个系统OID。
datfrozenxid	xid		这个数据库中在此之前的所有事务ID已被替换为的永久（冻结）事务ID。这用于追踪该数据库是否需要清理，以防止事务ID回卷或允许pg_clog收缩，它是每个表的 *pg_class.relfrozenxid* 的最小值。
dattablespace	oid	pg_tablespace.oid	数据库的默认表空间。在此数据库中，所有 *pg_class.reltablespace* 为 0 的表都将存储在此表空间中。所有非共享的系统目录也将放在那里。
datconfig	text[]		用户可设置的服务器配置参数的会话默认值。
datacl	aclitem[]		由 GRANT 和 REVOKE 所给予的数据库访问特权。

上级主题： 系统目录定义

pg_depend

pg_depend 系统目录表记录数据库对象之间依赖关系。此信息允许 DROP 命令查找哪些其他对象必须由 DROP CASCADE 删除或者必须防止在 DROP RESTRICT 情况中被删除。另请参见 pg_shdepend，它对涉及 HashData 系统共享对象的依赖执行类似的功能。

在所有的情况下，pg_depend 项表示在不删除依赖对象的前提下也不能删除被引用对象。不过，有几种由 deptype 区分的子类型：

- **DEPENDENCY_NORMAL (n)** — 单独创建的对象之间的正常关系。可以删除依赖对象而不影响被引用的对象。被引用对象只能通过指定CASCADE来删除，在这种情况下依赖对象也会被删除。例如：表列对其数据类型具有正常的依赖性。
- **DEPENDENCY_AUTO (a)** — 依赖对象可以独立于被引用对象而删除，并且如果被引用对象被删除，则依赖对象应该自动被删除（不管是RESTRICT还是CASCADE模式）。例如：表上的命名约束自动依赖表，因此如果表被删除，该约束也将消失。
- **DEPENDENCY_INTERNAL (i)** — 依赖对象作为被引用对象的一部分创建，实际只是其内部实现的一部分。依赖对象的DROP 操作将被完全禁止（我们会告诉用户针对被引用对象发出DROP命令来代替）。被引用对象的DROP命令会被传播以删除依赖对象，不管是否指定CASCADE（级联删除）。
- **DEPENDENCY_PIN (p)** — 没有依赖对象；这种类型的项是系统本身依赖被引用对象的信号，此类型的项仅通过系统初始化创建，依赖对象包含0。

表 1. pg_catalog.pg_depend

列	类型	参考	描述
classid	oid	pg_class.oid	依赖对象所在的系统目录的OID。
objid	oid	any OID column	特定依赖对象的OID。
objsubid	int4		对于表列，这是列编号。对于其他对象类型，此列为0。
refclassid	oid	pg_class.oid	被引用对象所在的系统目录的OID。
refobjid	oid	any OID column	特定被引用对象的OID。
refobjsubid	int4		对于表列，这是被引用的列编号。对于其他对象类型，该列为0。
deptype	char		定义此依赖关系的特定语义的代码。

上级主题： 系统目录定义

pg_description

pg_description 系统目录表存储每个数据库对象的可选描述（注释）。可以使用 COMMENT 命令来操作描述，并使用 psql 的 \d 元命令来查看。 pg_description 的初始内容中提供了许多内建系统对象的描述。另请参见 pg_shdescription ，它对涉及 HashData 数据库系统共享对象的描述执行类似的功能。

表 1. pg_catalog.pg_description

列	类型	参考	描述
objoid	oid	any OID column	该描述所涉及对象的OID。
classoid	oid	pg_class.oid	该对象所出现的系统目录的OID。
objsubid	int4		对于表列的注释，这是列编号。对于其他对象类型，此列为0。
description	text		作为此对象描述的任意文本。

上级主题： 系统目录定义

pg_enum

该 pg_enum 表包含与其相关联的值和标签匹配的枚举类型条目。给定枚举值的内部表示其实是 pg_enum表中相关行的Oid。特定枚举类型的Oid保证按照应有类型排序方式进行排序，但是不保证不相关枚举类型Oid的排序。

表 1. pg_catalog.pg_enum

列	类型	参考	描述
enumtypid	oid	pgtype.oid	拥有此枚举类型 pg_type 条目的Oid。
enumlabel	name		此枚举值的文本标签。

上级主题： 系统目录定义

pg_extension

该系统目录表 pg_extension 存储了关于安装的扩展的信息。

表 1. pg_catalog.pg_extension

列	类型	参考	描述
extname	name		扩展名
extowner	oid	pg_authid.oid	扩展所有者
extnamespace	oid	pg_namespace.oid	包含扩展导出对象的模式
extrelocatable	boolean		如果该扩展能够重定位到其他模式，则为真。
extversion	text		扩展的版本名字
extconfig	oid[]	pg_class.oid	扩展配置表的 regclass OIDs 数组， 如果为空则为 NULL。
extcondition	text[]		扩展配置表的 WHERE-clause过滤条件数组，如果没有则为 NULL 。

与大多数命名空间列的目录不同，extnamespace 并不意味着该扩展属于这个模式。扩展名不符合模式限制。 extnamespace 模式指示其包含大部分或全部扩展对象的模式。如果 extrelocatable 为 真，则该模式必须包含所有属于扩展名的符合方案限定的对象。

上级主题： 系统目录定义

pg_exttable

pg_exttable 系统目录表用来追踪由 CREATE EXTERNAL TABLE 命令创建的外部表和 Web 表。

表 1. pg_catalog.pg_exttable

列	类型	参考	描述
reloid	oid	pg_class.oid	该外部表的 OID。
urilocation	text[]		外部表文件的 URI 地址。
execlocation	text[]		为外部表定义的 ON Segment 位置。
fmttype	char		外部表文件的格式：t是文本，c是csv。
fmtopts	text		外部表文件的格式化选项，如字段定界符、空字符串、转义字符等。
options	text[]		为外部表定义的选项。
command	text		访问外部表所执行的OS命令。
rejectlimit	integer		每个Segment拒绝错误行的限制，之后装载将失败。
rejectlimittype	char		拒绝限制的阀值类型：r 表示行数。
fmterrtbl	oid	pg_class.oid	记录格式错误的错误表的对象ID。注意：此列不再使用，将在以后的版本中被删除。
encoding	text		客户端编码。
writable	boolean		0表示可读外部表， 1表示可写外部表。

上级主题： 系统目录定义

pg_filespace_entry

表空间需要一个文件系统位置来储存其数据库文件。在 HashData 数据库中，Master 和每个 Segment（主 Segment 和镜像 Segment）需要自己独特的存储位置。 HashData 系统中所有组件的文件系统位置的集合被称为 文件空间 。pg_filespace_entry 表包含构成一个 HashData 数据库文件空间的 HashData 数据库系统中文件系统位置集合的信息。

表 1. pg_catalog.pg_filespace_entry

列	类型	参考	描述
fsefsoid	OID	pg_filespace.oid	文件空间的对象 ID
fsedbid	integer	gp_segment_ configuration.dbid	Segment 的 ID。
fselocation	text		这个 Segment ID 的文件系统位置。

上级主题： 系统目录定义

pg_filespace

pg_filespace 表包含在 HashData 数据库系统中创建的文件空间的信息。每个系统都包含一个默认的文件空间 pg_system，它是在系统初始化的时候创建的数据目录位置的集合。

表空间需要一个文件系统位置来储存其数据库文件。在 HashData 数据库中，Master 和每个 Segment（主 Segment 和镜像 Segment）都需要自己的储存位置。 HashData 系统中所有组件的文件系统位置的集合就被称为文件空间。

表 1. pg_catalog.pg_filespace

列	类型	参考	描述
fsname	name		文件空间的名字。
fsowner	oid	pg_roles.oid	创建文件空间角色的对象ID。

上级主题： 系统目录定义

pg_index

pg_index 系统目录表包含部分关于索引的信息。剩下的信息主要在 pg_class 表中。

表 1. pg_catalog.pg_index

列	类型	参考	描述
indexrelid	oid	pg_class.oid	该索引的 pg_class 项的 OID。
indrelid	oid	pg_class.oid	该索引所对应的表的pg_class项的OID。
indnatts	int2		索引中的列数（与其 pg_class.relnatts重复）。
indisunique	boolean		如果为真，则这是一个唯一索引。
indisprimary	boolean		如果为真，则这个索引表示表的主键（此属性为真时indisunique总是为真）。
indisclustered	boolean		如果为真，表最后一次是通过CLUSTER命令在此索引上聚簇。
indisvalid	boolean		如果为真，则该索引目前对查询有效。如果为假，意味着该索引目前可能不完整：它仍然必须被INSERT/UPDATE操作所修改，但是它不能被安全地用于查询。
indkey	int2vector	pg_attribute.attnum	这是一个长度为indnatts值的数组，它指出了这个索引索引哪些表列。例如，值为1 3意味着1号、3号列组成索引键。该数组中的0表示相应的索引属性是表列上的表达式，而不是简单的列引用。
indclass	oidvector	pg_opclass.oid	对于索引键中的每个列，它包含要使用的操作符类OID。
indexprs	text		非简单列引用的索引属性的表达式树（表示为 nodeToString()的形式）。这是一个列表，indkey中每一个为0的元素在其中都有一项。如果所有索引属性都是简单引用，则为NULL。
indpred	text		用于部分索引谓词的表达式树（表示为 nodeToString()的形式），如不是部分索引则为NULL。

上级主题： 系统目录定义

pg_inherits

pg_inherits 系统目录表记录有关继承层次结构的信息。数据库中的每个直接子表在其中都有一项（间接继承可以通过 pg_inherits 项构成的链来确定）。在 HashData 数据库中，继承关系由 INHERITS 子句（独立继承）和 CREATE TABLE 的 PARTITION BY 子句（分区子表继承）创建。

表 1. pg_catalog.pg_inherits

列	类型	参考	描述
inhrelid	oid	pg_class.oid	子表的OID。
inhparent	oid	pg_class.oid	父表的OID。
inhseqno	int4		如果对于一个子表有多个直接父表（多继承），则这个数字将指出被继承列的安排顺序。计数从1开始。

上级主题： 系统目录定义

pg_language

pg_language 系统目录表注册用户可以用来编写函数和存储程序的语言。它由CREATE LANGUAGE填充。

表 1. pg_catalog.pg_language

列	类型	参考	描述
lanname	name		语言的名称
lanowner	oid	pg_authid.oid	语言的拥有者
lanispl	boolean		对内部语言（如SQL）而言，值为假。而对于用户自定义的语言为真。目前，pg_dump 仍然使用它来确定哪些语言需要被转存，但是，在将来它可能会被不同的机制所代替。
lanpltrusted	boolean		如果这是一种可信的语言，则为真，表示它不会为正常SQL执行环境之外的任何东西授予访问。只有超级用户才能用不可信语言创建函数。
lanplcallfoid	oid	pg_proc.oid	对于非内部的语言，该属性引用了一个语言处理程序，该程序是一个特殊的函数，负责执行所有以特定语言编写的函数。
laninline	oid	pg_proc.oid	这个属性引用一个函数负责执行内联匿名代码块的函数。如果不支持匿名块，则为0。
lanvalidator	oid	pg_proc.oid	这个属性引用一个语言验证器函数，负责在创建新函数时检查新函数的语法和合法性。如果没有提供验证器，则为0。
lanacl	aclitem[]		语言的访问特权。

上级主题： 系统目录定义

pg_largeobject

pg_largeobject 系统目录表保存构成“大对象”的数据。大对象由创建时分配的 OID 来识别。每个大对象被分解成便于在 pg_largeobject 表中储存为行的足够小的段或者“页”。每页的数据量被定义为LOBLKSIZE（当前是 BLCKSZ/4，或者通常是 8K）。

pg_largeobject 的每一行都保存一个大对象的一个页，从对象内的字节偏移量（ *pageno* * LOBLKSIZE）开始。该实现允许稀疏存储：页面可以丢失，并且即使它们不是对象的最后一页也可能比 LOBLKSIZE 字节更短。大对象中缺少的区域被读作0。

表 1. pg_catalog.pg_largeobject

列	类型	参考	描述
loid	oid		包含这一页的大对象的标识符。
pageno	int4		大对象中该页的页号（从0开始计数）。
data	bytea		存储在大对象中的实际数据。该数据不会超过LOBLKSIZE字节数而且可能会更小。

上级主题： 系统目录定义

pg_listener

pg_listener系统目录表支持LISTEN和NOTIFY命令。监听器在pg_listener中为每个正在监听的通知名称创建一项。通知者扫描和更新每个匹配的项以显示通知已经发生。通知者也会发送一个信号（使用表中记录的PID）将监听器从睡眠中唤醒。

此表目前不在 HashData 数据库中使用。

表 1. pg_catalog.pg_listener

列	类型	参考	描述
relname	name		通知条件名称（该名称不需要匹配数据库中任何实际关系）。
listenerpid	int4		创建这个项的服务器进程的PID。
notification	int4		如果这个监听器没有事件待处理，则为0。如果有待处理事件，则为发送通知的服务器进程的PID。

上级主题： 系统目录定义

pg_locks

该 pg_locks 视图提供了有关在 HashData 数据库中由开放事务持有的锁的信息的访问。

pg_locks 包含一行关于每个积极可锁对象，请求的锁模式和相关事务。 因此，如果多个事务正在持有或等待其上的锁，同样的可锁对象可能会出现多次。 但是，目前没有锁的对象根本就不会出现。

有几种不同类型的可锁对象：整个关系（如表），关系的个别页，关系的个别元组，事务Id和通用数据库对象。另外，扩展关系的权利表示为单独的可锁对象。

表 1. pg_catalog.pg_locks

列	类型	参考	描述
locktype	text		可锁对象的类型：relation, extend, page, tuple, transactionid, object, userlock, resource queue, 或 advisory
database	oid	pg_database.oid	该对象存在的数据库的Oid， 如果该对象是共享对象，则为0。如果对象是事务ID，则为空。
relation	oid	pg_class.oid	关系的Oid，如果对象不是关系或者关系的一部分，则为NULL。
page	integer		关系中的页码，如果对象不是元组或者关系页则为NULL
tuple	smallint		页中的元组号，如果该对象不是个元组则为NULL。
transactionid	xid		事务的Id，如果该对象不是一个事务Id，则为NULL。
classid	oid	pg_class.oid	包含对象的系统目录的Oid，如果对象不是一般数据库对象，则为NULL。
objid	oid	any OID column	其系统目录中对象的Oid，如果对象不是一般数据库对象，则为NULL。
objsubid	smallint		对一个表列来说， 这是列号（ classid和objid引用表本身）。 对于所有其他的对象类型，此列为0。如果对象不是数据库对象，则为NULL。
transaction	xid		等待或持有该锁的事务的Id。
pid	integer		持有或等待该锁的事务进程的进程Id，如果锁由准备（prepared）的事务持有，则为NULL。
mode	text		该进程所持有或期望的锁模式的名称。
granted	boolean		锁被持有为真，锁为等待为假。
mppsessionid	integer		与锁相关的客户端会话的id。
mppiswriter	boolean		指明该锁是否由一个写进程所持有。
gp_segment_id	integer		该 HashData 持有该锁的段的id（dbid）

上级主题： 系统目录定义

pg_max_external_files

pg_max_external_files 视图显示使用外部表文件协议时每台Segment主机允许的外部表文件最大数量。

表 1. pg_catalog.pg_max_external_files

列	类型	参考	描述
hostname	name		在Segment主机上访问特定Segment实例所使用的主机名。
maxfiles	bigint		该主机上的主Segment实例数。

上级主题： 系统目录定义

pg_namespace

pg_namespace 系统目录表存储命名空间。命名空间是 SQL 模式的底层结构：每个命名空间都可以具有单独的没有命名冲突的关系、类型等集合。

表 1. pg_catalog.pg_namespace

列	类型	参考	描述
nspname	name		命名空间的名字
nspowner	oid	pg_authid.oid	命名空间的所有者
nspacl	aclitem[]		由GRANT和REVOKE给予的访问特权。

上级主题： 系统目录定义

pg_opclass

系统目录表 pg_opclass 定义索引访问方法的操作符类。每一个操作符类定义了一种特定数据类型和一种特定索引访问方法的索引列的语义。注意对于一个给定的类型/访问方法的组合可以有多个操作符类，因此可以支持多种行为。定义一个操作符类的主要信息其实现在并不在其 pg_opclass 行中，而是在 pg_amop 和 pg_amproc 中的相关行中。这些行被看认为是操作符类定义的一部分 - 这不同于通过单个 pg_class 行外加 pg_attribute 和其他表中的相关行定义一个关系的情况。

表 1. pg_catalog.pg_opclass

列	类型	参考	描述
opcamid	oid	pg_am.oid	操作符类所属的索引访问方法。
opcname	name		操作符类的名称
opcnamespace	oid	pg_namespace.oid	操作符类所属的名字空间
opcowner	oid	pg_authid.oid	操作符类的拥有者
opcintype	oid	pg_type.oid	操作符类索引的数据类型。
opcdefault	boolean		如果此操作符类为数据类型的opcintype默认值则为真。
opckeytype	oid	pg_type.oid	存储在索引中的数据类型，如果值为0则与opcintype相同。

上级主题： 系统目录定义

pg_operator

系统目录表pg_operator存储关于操作符的信息，包括内建的和通过CREATE OPERATOR语句定义的操作符。未用的列包含零值。例如，一个前缀操作符的oprleft为0。

表 1. pg_catalog.pg_operator

列	类型	参考	描述
oprname	name		操作符的名称。
oprnamespace	oid	pg_namespace.oid	操作符所属的名字空间的OID。
oprowner	oid	pg_authid.oid	操作符的拥有者。
oprkind	char		b = 中缀 (前缀和后缀)，l = 前缀 ("左")，r = 后缀 ("右")
oprcanhash	boolean		该操作符是否支持哈希连接。
oprleft	oid	pg_type.oid	左操作数类型。
oprright	oid	pg_type.oid	右操作数类型。
oprresult	oid	pg_type.oid	结果类型。
oprcom	oid	pg_operator.oid	该操作符的交换子（如果存在）。
oprnegate		pg_operator.oid	该操作符的求反器（如果存在）。
oprlsortop	oid	pg_operator.oid	如果该操作符支持归并连接，这个属性记录排序左操作数类型的操作符（L<L）。
oprrsortop	oid	pg_operator.oid	如果该操作符支持归并连接，这个属性记录排序右操作数类型的操作符（R<R）。
oprltcmpop	oid	pg_operator.oid	如果该操作符支持归并连接，这个属性记录比较左右操作数的小于操作符（L<R）。
oprgtcmpop	oid	pg_operator.oid	如果该操作符支持归并连接，这个属性记录比较左右操作数的小于操作符（L>R）。
oprcode	regproc	pg_proc.oid	实现该操作符的函数。
oprrest	regproc	pg_proc.oid	该操作符的限制选择度估算函数。
oprjoin	regproc	pg_proc.oid	该操作符的连接选择度估算函数。

上级主题： 系统目录定义

pg_partition_columns

pg_partition_columns系统视图被用来显示一个分区表的分区键列。

表 1. pg_catalog.pg_partition_columns

列	类型	参考	描述
schemaname	name		该分区表所在方案的名称。
tablename	name		顶层父表的表名。
columnname	name		分区键列的名称。
partitionlevel	smallint		该子分区在层次中的级别。
position_in_partition_key	integer		对于列表分区可以有组合（多列）分区键。此处显示了列在组合键中的位置。

上级主题： 系统目录定义

pg_partition_encoding

pg_partition_encoding 系统目录表描述一个分区模板的可用的列压缩选项。

表 1. pg_catalog.pg_attribute_encoding

名称	类型	修改	存储	描述
parencoid	oid	not null	plain	
parencattnum	snallint	not null	plain	
parencattoptions	text [ ]		extended	

上级主题： 系统目录定义

pg_partition_rule

pg_partition_rule系统目录表被用来跟踪分区表、它们的检查约束以及数据包含规则。pg_partition_rule表中的每一行要么代表了一个叶子分区（最底层包含数据的分区），要么是一个分支分区（用于定义分区层次的顶层或者中间层分区，但不包含数据）。

表 1. pg_catalog.pg_partition_rule

列	类型	参考	描述
paroid	oid	pg_partition.oid	这个分区所属的分区级别的行标识符（来自 pg_partition ）。对于分支分区，相应的表（由pg_partition_rule标识）是一个空的容器表。对于叶子分区，这个表含有分区包含规则的行。
parchildrelid	oid	pg_class.oid	分区（子表）的表标识符。
parparentrule	oid	pg_partition_rule.paroid	与该分区的父表相关的规则的行标识符。
parname	name		该分区的给定名称。
parisdefault	boolean		该分区是否为默认分区。
parruleord	smallint		对于范围分区表，该分区在分区层次的这个级别上的排名。
parrangestartincl	boolean		对于范围分区表，开始值是否被包括。
parrangeendincl	boolean		对于范围分区表，结束值是否被包括。
parrangestart	text		对于范围分区表，范围的开始值。
parrangeend	text		对于范围分区表，范围的结束值。
parrangeevery	text		对于范围分区表，EVERY子句的间隔值。
parlistvalues	text		对于列表分区表，指派给该分区的值列表。
parreloptions	text		一个描述特定分区存储特性的数组。

上级主题： 系统目录定义

pg_partition_templates

pg_partition_templates 系统视图用来显示通过子分区模板创建的子分区。

表 1. pg_catalog.pg_partition_templates

列	类型	参考	描述
schemaname	name		该分区表所在的方案的名称。
tablename	name		顶层父表的表名。
partitionname	name		子分区的名称（如果在ALTER TABLE命令中引用分区，用的就是这个名称）。如果该分区在创建时或者通过EVERY子句产生时没有给定一个名称则为NULL。
partitiontype	text		子分区的类型（范围或者列表）。
partitionlevel	smallint		该分区在层次中的级别。
partitionrank	bigint		对于范围分区，该分区相对于同级中其他分区的排名。
partitionposition	smallint		该子分区的规则顺序位置。
partitionlistvalues	text		对于列表分区，该子分区相关的列表值。
partitionrangestart	text		对于范围分区，该子分区的开始值。
partitionstartinclusive	boolean		如果该子分区包含了开始值则为T，否则为F。
partitionrangeend	text		对于范围分区，该子分区的结束值。
partitionendinclusive	boolean		如果该子分区包含了结束值则为T，否则为F。
partitioneveryclause	text		该子分区的 EVERY 子句（间隔）。
partitionisdefault	boolean		如果这是一个默认子分区则为T，否则为F。
partitionboundary	text		该子分区的整个分区说明。

上级主题： 系统目录定义

pg_partition

pg_partition 系统目录表被用来跟踪分区表以及它们的继承层级关系。pg_partition 中的每一行要么代表了一个分区表在分区层级关系中的等级，要么是一个子分区模板的描述。 paristemplate 的属性值决定了一个特定行代表的含义。

表 1. pg_catalog.pg_partition

列	类型	参考	描述
parrelid	oid	pg_class.oid	表的对象标识符。
parkind	char		分区类型 - R（范围）或者 L（列表）。
parlevel	smallint		该行的分区级别：0代表最顶层的父表，1代表父表下的第一个级别，2代表第二个级别，以此类推。
paristemplate	boolean		该行是否代表一个子分区模板定义（true）或者实际分区级别（false）。
parnatts	smallint		定义该级别的属性数目。
paratts	smallint()		参与定义该级别的属性编号（正如在 pg_attribute.attnum 中的）数组。
parclass	oidvector	pg_opclass.oid	分区列上的操作符类的标识符。

上级主题： 系统目录定义

pg_partitions

pg_partitions 系统视图被用于显示分区表的结构。

表 1. pg_catalog.pg_partitions

列	类型	参考	描述
schemaname	name		分区表所属方案的名称。
tablename	name		顶层父表的名称。
partitiontablename	name		分区表的关系名（直接访问分区时使用的表名）。
partitionname	name		分区的名称（在ALTER TABLE命令引用分区时，使用该名称）。如果在分区创建时或者由EVERY子句产生时没有给定名称则为NULL。
parentpartitiontablename	name		该分区上一层父表的关系名。
parentpartitionname	name		该分区上一层父表给定的名称。
partitiontype	text		分区的类型（范围或者列表）。
partitionlevel	smallint		该分区在层次中的级别。
partitionrank	bigint		对于范围分区，该分区相对于同级其他分区的排名。
partitionposition	smallint		该分区的规则顺序位置。
partitionlistvalues	text		对于列表分区，与该分区相关的列表值。
partitionrangestart	text		对于范围分区，该分区的开始值。
partitionstartinclusive	boolean		如果该分区包含了开始值则为 T ，否则为F。
partitionrangeend	text		对于范围分区，该分区的结束值。
partitionendinclusive	boolean		如果该分区包含了结束值则为 T ，否则为F。
partitioneveryclause	text		该分区的EVERY子句（间隔）。
partitionisdefault	boolean		如果该分区为默认分区则为 T否则为 F.
partitionboundary	text		该分区的整个分区描述。

上级主题： 系统目录定义

pg_pltemplate

pg_pltemplate 系统目录表存储了过程语言的“模板”信息。一种语言的模板允许我们在一个特定数据库中以简单的 CREATE LANGUAGE 命令创建该语言，而不需要指定实现细节。和大部分系统目录不同，pg_pltemplate 在 HashData 系统的所有数据库之间共享：在一个系统中只有一份 pg_pltemplate 拷贝，而不是每个数据库一份。这使得在每个需要的数据库中都可以访问该信息。

目前任何命令都不能操纵过程语言模板。要改变内建信息，超级用户必须使用普通的 INSERT、DELETE或UPDATE 命令修改该表。

表 1. pg_catalog.pg_pltemplate

列	类型	参考	描述
tmplname	name		该模板适用的语言名称
tmpltrusted	boolean		如果该语言被认为是可信的则为真
tmplhandler	text		调用处理器函数的名字
tmplvalidator	text		验证器函数的名字，如果没有则为空
tmpllibrary	text		实现语言的共享库的路径
tmplacl	aclitem[]		模板的访问特权（并未实现）。

上级主题： 系统目录定义

pg_proc

pg_proc 系统目录表存储关于函数（或过程）的信息，包括所有的内建函数以及由 CREATE FUNCTION 定义的函数。该表也包含了聚集和窗口函数以及普通函数的数据。如果 proisagg 为真，在 pg_aggregate 中应该有一个相匹配的行。如果 proiswin 为真，在 pg_window 应该有一个相匹配的行。

对于编译好的函数（包括内建的和动态载入的），prosrc 包含了函数的 C 语言名字（链接符号）。对于所有其他已知的语言类型，prosrc 包含函数的源码文本。除了对于动态载入的 C 函数之外，probin 是不被使用的。对于动态载入的 C 函数，它给定了包含该函数的共享库文件的名称。

表 1. pg_catalog.pg_proc

列	类型	参考	描述
proname	name		函数的名字。
pronamespace	oid	pg_namespace.oid	函数所属的名字空间的OID。
proowner	oid	pg_authid.oid	函数的拥有者。
prolang	oid	pg_language.oid	该函数的实现语言或调用接口。
procost	float4		估计的执行代价（以cpu_operator_cost为单位），如果proretset为true，这是返回每行的代价。
provariadic	oid	pg_type.oid	可变数组参数的元素的数据类型，如果函数没有可变参数则为0。
proisagg	boolean		函数是否为一个聚集函数。
prosecdef	boolean		函数是一个安全性定义器（例如，一个"setuid"函数）
proisstrict	boolean		当任意调用参数为空时，函数是否会返回空值。在那种情况下函数实际上根本不会被调用。非“strict”函数必须准备好处理空值输入。
proretset	boolean		函数是否返回一个集合（即，指定数据类型的多个值）。
provolatile	char		说明函数的结果是仅依赖于它的输入参数，还是会被外部因素影响。值i表示“不变的”函数，它对于相同的输入总是输出相同的结果。值s表示“稳定的”函数，它的结果（对于固定输入）在一次扫描内不会变化。值v表示“不稳定的”函数，它的结果在任何时候都可能变化（使用具有副作用，结果可能在任何时刻发生改变）。
pronargs	int2		输入参数的个数。
pronargdefaults	int2		具有默认值的参数个数。
prorettype	oid	pg_type.oid	返回值的数据类型。
proiswin	boolean		既不是聚集函数也不是标量函数，而是一个窗口函数。
proargtypes	oidvector	pg_type.oid	函数参数的数据类型的数组。这只包括输入参数（包括INOUT和VARIADIC参数），因此也表现了函数的调用特征。
proallargtypes	oid[]	pg_type.oid	函数参数的数据类型的数组。这包括所有参数（含OUT和INOUT参数）。不过，如果所有参数都是IN参数，这个属性将为空。注意下标是从1开始，然而由于历史原因proargtypes的下标是从0开始。
proargmodes	char[]		函数参数的模式的数组：i表示IN参数，o表示OUT参数，b表示INOUT参数，v表示VARIADIC参数。如果所有的参数都是IN参数，这个属性为空。注意这里的下标对应着proallargtypes而不是proargtypes中的位置。
proargnames	text[]		函数参数的名字的数组。没有名字的参数在数组中设置为空字符串。如果没有一个参数有名字，这个属性为空。注意这里的下标对应着proallargtypes而不是proargtypes中的位置。
proargdefaults	text		默认值的表达式树。这是一个pronargdefaults元素的列表，对应于最后N个输入参数（即最后N个N位置）。如果没有一个参数具有默认值，这个属性为空。
prosrc	text		这个域告诉函数处理者如何调用该函数。它可能是针对解释型语言的实际源码、一个符号链接、一个文件名或任何其他东西，这取决于实现语言/调用规范。
probin	bytea		关于如何调用函数的附加信息。其解释是与语言相关的。
proacl	aclitem[]		由GRANT/REVOKE给予的函数访问特权。

上级主题： 系统目录定义

pg_resourcetype

pg_resourcetype系统目录表包含那些被指派给 HashData 资源队列的扩展属性的信息。每行详细描述了一个属性及其固有特性，例如它的默认设置、是否必须以及禁用它的值（如果允许）。

该表只在Master上被填充。该表定义在pg_global表空间中，意味着它会在系统中所有的数据库间共享。

表 1. pg_catalog.pg_resourcetype

列	类型	参考	描述
restypid	smallint		资源类型ID。
resname	name		资源类型名。
resrequired	boolean		该资源类型对于一个有效的资源队列是否必需。
reshasdefault	boolean		资源类型是否有默认值。如果为真，默认值在reshasdefaultsetting中指定。
rescandisable	boolean		类型是否能够被移除或者禁用。如果为真，默认值在resdisabledsetting中指定。
resdefaultsetting	text		可用时，资源类型的默认设置。
resdisabledsetting	text		禁用资源类型的值（当允许时）。

上级主题： 系统目录定义

pg_resqueue_attributes

pg_resqueue_attributes 视图允许管理员查看资源队列的属性，例如活动语句限制、查询代价限制以及优先级。

表 1. pg_catalog.pg_resqueue_attributes

列	类型	参考	描述
rsqname	name	pg_resqueue.rsqname	资源队列的名称。
resname	text		资源队列属性的名称。
ressetting	text		资源队列属性的当前值。
restypid	integer		系统分配的资源类型ID。

上级主题： 系统目录定义

pg_resqueue

pg_resqueue系统目录包含了用于负载管理特性的 HashData 数据库资源队列的信息。该表只在Master上被填充。该表定义在pg_global表空间中，意味着它被系统中所有数据库共享。

表 1. pg_catalog.pg_resqueue

列	类型	参考	描述
rsqname	name		资源队列名。
rsqcountlimit	real		资源队列的活动查询阈值。
rsqcostlimit	real		资源队列的查询代价阈值。
rsqovercommit	boolean		当系统是空闲时，允许超过代价阈值的查询运行。
rsqignorecostlimit	real		查询被认为是一个“小查询”的查询代价限制。代价低于该限制的查询将不会被入队列而是立即被执行。

上级主题： 系统目录定义

pg_resqueuecapability

pg_resqueuecapability 系统目录表包含现有 HashData 数据库资源队列的扩展属性或者能力的信息。只有已经被指定了扩展能力（例如，优先级设置）的资源队列会被记录在该表中。该表通过资源队列对象 ID 与 pg_resqueue 表连接，通过资源类型 ID（restypid）与 pg_resourcetype 表连接。

该表只在 Master 上被填充。该表定义在 pg_global 表空间中，意味着它在系统中所有的数据库间共享。

表 1. pg_catalog.pg_resqueuecapability

列	类型	参考	描述
rsqueueid	oid	pg_resqueue.oid	相关资源队列的对象ID。
restypid	smallint	pg_resourcetype.restypid	资源类型，该值从 *pg_resqueuecapability* 系统表中得来。
ressetting	opaque type		为这个记录所引用的能力设置的特定值。根据实际的资源类型，该值可能有不同的数据类型。

上级主题： 系统目录定义

pg_rewrite

pg_rewrite 系统目录表存储表和视图的重写规则。如果一个表在这个目录中有任何规则，其 pg_class.relhasrules 必须为真。

表 1. pg_catalog.pg_rewrite

列	类型	参考	描述
rulename	name		规则名称。
ev_class	oid	pg_class.oid	使用该规则的表。
ev_attr	int2		使用该规则的列（当前总是0，表示在整个表上使用）。
ev_type	char		使用该规则的事件类型：1 = SELECT，2 = UPDATE，3 = INSERT，4 = DELETE
is_instead	boolean		如果规则是一个INSTEAD规则，则为真。
ev_qual	text		规则条件的表达式树（按照nodeToString()的表现形式）。
ev_action	text		规则动作的查询树（按照nodeToString()的表现形式）。

上级主题： 系统目录定义

pg_roles

pg_roles 视图提供关提供了关于数据库角色的信息。这是 pg_authid 的一个公共可读视图，它隐去了口令域。此视图显示了低层表的OID列，因为需要它来和其他目录做连接。

表 1. pg_catalog.pg_roles

列	类型	参考	描述
rolname	name		角色名。
rolsuper	bool		角色是否具有超级用户权限？
rolinherit	bool		如果此角色是另一个角色的成员，角色是否能自动继承另一个角色的权限？
rolcreaterole	bool		角色能否创建更多角色？
rolcreatedb	bool		角色能否创建数据库？
rolcatupdate	bool		角色能够更新直接更新系统目录（除非该列设置为真，否则超级用户也不能执行该操作）？
rolcanlogin	bool		角色是否能登录？即此角色能否被作为初始会话授权标识符？
rolconnlimit	int4		对于一个可登录的角色，这里设置角色可以发起的最大并发连接数。-1表示无限制。
rolpassword	text		不是口令（看起来是**）。
rolvaliduntil	timestamptz		口令失效时间（只用于口令认证），如果永不失效则为空。
rolconfig	text[]		运行时配置变量的角色特定默认值。
rolresqueue	oid	pg_resqueue.oid	该角色指定的资源队列的对象ID。
oid	oid	pg_authid.oid	角色的ID。
rolcreaterextgpfd	bool		角色能够创建使用gpfdist协议的可读的外部表？
rolcreaterexthttp	bool		角色能够创建使用http协议的可读的外部表？
rolcreatewextgpfd	bool		角色能否创建使用了gpfdist协议的可写外部表？

上级主题： 系统目录定义

pg_shdepend

pg_shdepend 系统目录表记录数据库对象和共享对象（例如角色）之间的依赖关系。这些信息使得 HashData 数据库可以确保对象在被删除时没有被其他对象引用。另见 pg_depend ，它对单个数据库中对象之间的依赖提供了相似的功能。 与大部分其他系统目录不同，pg_shdepend 在 HashData 系统的所有数据库之间共享：在每一个系统中只有一份 pg_shdepend 拷贝，而不是每个数据库一份。

在所有情况下，一个 pg_shdepend 项表明被引用对象不能在没有删除其依赖对象的情况下被删除。但是，其中也有多种依赖类型，由 deptype 标识：

- **SHARED_DEPENDENCY_OWNER (o)** — 被引用对象（必须是一个角色）是依赖对象的拥有者。
- **SHARED_DEPENDENCY_ACL (a)** — 被引用对象（必须是一个角色）在依赖对象的ACL（访问控制列表）中被提到。
- **SHARED_DEPENDENCY_PIN (p)** — 没有依赖对象；这种类型的项是系统本身依赖被引用对象的信号，因此对象绝不能被删除。此类型的项仅通过系统初始化创建，依赖对象列包含0。

表 1. pg_catalog.pg_shdepend

列	类型	参考	描述
dbid	oid	pg_database.oid	依赖对象所在的数据库OID，如果是一个共享对象则值为0。
classid	oid	pg_class.oid	依赖对象所在的系统目录的OID。
objid	oid		任意OID列。依赖对象的OID。
objsubid	int4		对于一个表列，这将是列编号。对于所有其他对象类型，该列值为0。
refclassid	oid	pg_class.oid	被引用对象所在的系统目录的OID（必须是一个共享的目录）。
refobjid	oid		任意OID列。被引用对象的OID。
refobjsubid	int4		对于一个表列，这将是被引用列的列编号。对于所有其他对象类型，该列值为0。
deptype	char		一个定义该依赖关系的特定语义的代码。

上级主题： 系统目录定义

pg_shdescription

pg_shdescription 系统目录表存储共享数据库对象的可选描述（注释）。描述可以通过 COMMENT 命令操作，并且可以使用 psql 的 \d 元命令来查看。另见 pg_description ，它对单个数据库中对象的描述提供了相似的功能。与大部分其他系统目录不同， pg_shdescription 在 HashData 系统的所有数据库之间共享：在每一个系统中只有一份 pg_shdescription 拷贝，而不是每个数据库一份。

表 1. pg_catalog.pg_shdescription

列	类型	参考	描述
objoid	oid		任意OID列,该描述所属的对象的OID。
classoid	oid	pg_class.oid	该对象所在系统目录的OID。
description	text		作为该对象描述的任意文本。

上级主题： 系统目录定义

pg_stat_activity

视图 pg_stat_activity 每行显示一个服务器进程同时详细描述与之关联的用户会话和查询。这些列报告当前查询上可用的数据，除非参数 stats_command_string 被关闭。此外，只有在检查视图的用户是超级用户或者是正在报告的进程的拥有者时，这些列才可见。

列 current_query 中存储的查询文本字符串的最大长度可以通过服务器配置参数 pgstat_track_activity_query_size 来控制。

表 1. pg_catalog.pg_stat_activity

列	类型	参考	描述
datid	oid	pg_database.oid	数据库OID
datname	name		数据库名称
procpid	integer		服务进程的进程ID
sess_id	integer		会话ID
usesysid	oid	pg_authid.oid	角色ID
usename	name		角色名
current_query	text		进程正在执行的当前查询
waiting	boolean		如果正等待一个锁则为真，否则为假
query_start	timestamptz		查询开始执行的时间
backend_start	timestamptz		后台进程开始的时间
client_addr	inet		客户端地址
client_port	integer		客户端端口
application_name	text		客户端应用名
xact_start	timestamptz		事务开始时间
waiting_reason	text		服务进程正在等待的原因。值可以是：lock replication

上级主题： 系统目录定义

pg_stat_last_operation

pg_stat_last_operation表包含关于数据库对象（表、视图等）的元数据跟踪信息。

表 1. pg_catalog.pg_stat_last_operation

列	类型	参考	描述
classid	oid	pg_class.oid	包含对象的系统目录的OID。
objid	oid	any OID column	对象在其系统目录内的对象OID。
staactionname	name		在一个对象上采取的动作。
stasysid	oid	pg_authid.oid	pg_authid.oid的外键。
stausename	name		在该对象上执行操作的角色的名称。
stasubtype	text		被执行操作的对象的类型或者被执行操作的子类。
statime	timestamp with timezone		操作的时间戳。这和写到 HashData 数据库服务器日志文件的时间戳是相同的，以便在日志中查询更多关于操作细节的信息。

上级主题： 系统目录定义

pg_stat_last_shoperation

pg_stat_last_shoperation 表包含全局对象（角色，表空间等）的元数据跟踪信息。

表 1. pg_catalog.pg_stat_last_shoperation

列	类型	参考	描述
classid	oid	pg_class.oid	包含对象的系统目录的OID。
objid	oid	any OID column	对象在其系统目录内的对象OID。
staactionname	name		在对象上采取的动作。
stasysid	oid		
stausename	name		在该对象上执行操作的角色的名称。
stasubtype	text		被执行操作的对象的类型或者被执行操作的子类。
statime	timestamp with timezone		操作的时间戳。这和写到 HashData 数据库服务器日志文件的时间戳是相同的，以便在日志中查询更多关于操作细节的信息。

上级主题： 系统目录定义

pg_stat_operations

视图 pg_stat_operations 显示了关于最后一个执行在数据库对象（例如，表、索引、视图或者数据库）上或者全局对象（例如角色）的操作的细节信息。

表 1. pg_catalog.pg_stat_operations

列	类型	参考	描述
classname	text		在 pg_catalog 模式中的系统表的名称。(pg_class=relations, pg_database=databases,pg_namespace=schemas,pg_authid=roles)
objname	name		对象名。
objid	oid		对象OID。
schemaname	name		对象所属的模式的模式名。
usestatus	text		上一个在对象上执行操作的角色的状态（CURRENT=当前系统中活跃的角色，DROPPED=已经不再系统中的角色， CHANGED=仍然有同名的角色在系统中，但是从上次操作执行后已经发生了改变）。
usename	name		在对象上执行操作的角色的角色名。
actionname	name		执行在对象上的操作。
subtype	text		被执行操作的对象的类型或者 The type of object operated on or the subclass of operation performed.
statime	timestamptz		操作的时间戳。这和写到 HashData 数据库系统日志文件的时间戳是相同的，以防需要在日志中查询更多关于操作细节的信息。

上级主题： 系统目录定义

pg_stat_partition_operations

pg_stat_partition_operations 视图显示了执行在一个分区表上的上一个操作的细节信息。

表 1. pg_catalog.pg_stat_partition_operations

列	类型	参考	描述
classname	text		The name of the system table in the pg_catalog schema where the record about this object is stored (always pg_class for tables and partitions).
objname	name		对象名。
objid	oid		对象OID。
schemaname	name		对象所属的模式的模式名。
usestatus	text		上一个在对象上执行操作的角色的状态（CURRENT=当前系统中活跃的角色，DROPPED=已经不再系统中的角色， CHANGED=仍然有同名的角色在系统中，但是从上次操作执行后已经发生了改变）。
usename	name		在对象上执行操作的角色的角色名。
actionname	name		执行在对象上的操作。
subtype	text		被执行操作的对象的类型或者The type of object operated on or the subclass of operation performed.
statime	timestamptz		操作的时间戳。这和写到 HashData 数据库系统日志文件的时间戳是相同的，以防需要在日志中查询更多关于操作细节的信息。
partitionlevel	smallint		该分区在分区层次中的级别。
parenttablename	name		该分区上一个层次的父表的关系名。
parentschemaname	name		父表所属模式的名称。
parent_relid	oid		该分区上一个层次的父表的OID。

上级主题： 系统目录定义

pg_stat_replication

pg_stat_replication 视图包含用来镜像 HashData 数据库的 Master 的 walsender 进程的元数据。

表 1. pg_catalog.pg_stat_replication

列	类型	参考	描述
procpid	integer		WAL发送器后台进程的进程ID。
usesysid	integer		运行WAL发送器后台进程的用户系统ID。
usename	name		运行WAL发送器后台进程的用户名。
application_name	oid		客户端应用名。
client_addr	name		客户端IP地址。
client_port	integer		客户端端口号。
backend_start	timestamp		操作开始的时间戳。
state	text		WAL发送状态。可取值有：startup,backup,catchup,streaming
sent_location	text		WAL发送器已发送的xlog记录位置。
write_location	text		WAL接收器的xlog记录写位置。
flush_location	text		WAL接收器的xlog记录刷入位置。
replay_location	text		后备机xlog记录的重放位置。
sync_priority	text		优先级，值为1。
sync_state	text		WAL发送器的同步状态。该值为sync。

上级主题： 系统目录定义

pg_stat_resqueues

pg_stat_resqueues 视图允许管理员查看资源队列一段时间的负载度量。用户必须在 HashData 数据库的 Master 示例上启用 stats_queue_level 服务器配置参数。启用对这些度量的收集确实会招致小小的性能惩罚，因为每个通过资源队列提交的语句必须被记录在系统目录表中。

表 1. pg_catalog.pg_stat_resqueues

列	类型	参考	描述
queueoid	oid		资源队列的OID。
queuename	name		资源队列的名称。
n_queries_exec	bigint		已提交的要从该资源队列中执行的查询数目。
n_queries_wait	bigint		已提交到该资源队列中，在执行之前必须进行等待的查询数目。
elapsed_exec	bigint		通过该资源队列提交的语句的总执行时间。
elapsed_wait	bigint		通过该资源队列提交的语句，在执行之前必须进行等待的总时间。

上级主题： 系统目录定义

pg_statistic

pg_statistic 系统目录表存储有关数据库内容的统计数据。其中的项由 ANALYZE 创建，查询规划器会使用这些数据来进行查询规划。每个已经被分析的表列都有一项。注意所有的统计数据天然就是近似的，即使它刚刚被更新。

pg_statistic 也存储关于索引表达式值的统计数据，就好像它们是真正的数据列，但在这种情况中 starelid 引用索引。不过，对一个普通非表达式索引列不会创建项，因为它将是底层表列的项的冗余。

因为不同类型的统计信息适用于不同类型的数据，pg_statistic 被设计成不对自己存储的统计数据类型做太多假设。 只有极为常用的统计信息（比如空值）才在 pg_statistic 里给予专用的字段。其它所有东西都存储在“槽位”中，而槽位是一组相关的列， 它们的内容用槽位中的一个列里的代码表示。

pg_statistic 不应该是公共可读的，因为即使是一个表内容的统计性信息也可能被认为是敏感的（例子：一个薪水列的最大和最小值）。pg_stats 是 pg_statistic 上的一个公共可读的视图，它只会显示出当前用户可读的表的信息。

表 1. pg_catalog.pg_statistic

列	类型	参考	描述
starelid	oid	pg_class.oid	被描述列所属的表或索引。
staattnum	int2	pg_attribute.attnum	被描述列的编号。
stanullfrac	float4		列项为空的比例。
stawidth	int4		非空项的平均存储宽度，以字节计。
stadistinct	float4		列中非空唯一值的数目。一个大于零的值是唯一值的真正数目。一个小于零的值是表中行数的乘数的负值（例如，对于一个 每个值平均出现两次的列，可以表示为stadistinct = -0.5）。一个0值表示唯一值的数目未知。
stakind N	int2		一个代码，它表示存储在该pg_statistic行中第N个“槽位”的统计类型。
staop N	oid	pg_operator.oid	一个用于生成这些存储在第N个“槽位”的统计信息的操作符。 比如，一个柱状图槽位会用<操作符，该操作符定义了该数据的排序顺序。
stanumbers N	float4[]		第N个“槽位”的类型的数值类型统计，如果该槽位不涉及数值类型则为NULL。
stavalues N	anyarray		第N个“槽位”的类型的列值，如果该槽位类型不存储任何数据值则为NULL。 每个数组的元素值实际上都是指定列的数据类型，因此，除了把这些列的类型定义成 anyarray 之外别无他法。

上级主题： 系统目录定义

pg_tablespace

pg_tablespace 系统目录表存储关于可用表空间的信息。 表可以被放置在特定表空间中以实现磁盘布局的管理。与大部分其他系统目录不同，pg_tablespace 在 HashData 系统的所有数据库之间共享：在每一个系统中只有一份 pg_tablespace 的拷贝，而不是每个数据库一份。

表 1. pg_catalog.pg_tablespace

列	类型	参考	描述
spcname	name		表空间名。
spcowner	oid	pg_authid.oid	表空间的拥有者，通常是创建它的用户。
spclocation	text[]		已弃用。
spcacl	aclitem[]		访问权限。
spcprilocations	text[]		不赞成。
spcmrilocations	text[]		不赞成。
spcfsoid	oid	pg_filespace.oid	被该表空间使用的文件空间的对象ID。一个文件空间定义在Master、主Segment和镜像Segment上的目录位置。

上级主题： 系统目录定义

pg_trigger

pg_trigger 系统目录表存储表上的触发器。

> 注意： HashData 数据库不支持触发器。

表 1. pg_catalog.pg_trigger

名称	类型	引用	描述
tgrelid	oid	pg_class.oid	注意 HashData 数据库不强制参照完整性。触发器所在的表。
tgname	name		触发器名（同一个表的触发器名必须唯一）。
tgfoid	oid	pg_proc.oid	注意 HashData 数据库不强制参照完整性。要被触发器调用的函数
tgtype	int2		触发器触发条件的位掩码。
tgenabled	boolean		为真是，启用触发器。
tgisconstraint	boolean		如果触发器实现一个参照完整性约束，则为真。
tgconstrname	name		参照完整性的名称。
tgconstrrelid	oid	pg_class.oid	注意 HashData 数据库不强制参照完整性。被一个参照完整性约束引用的表。
tgdeferrable	boolean		如果可延迟则为真。
tginitdeferred	boolean		如果初始可延迟则为真。
tgnargs	int2		传递给触发器函数的参数字符串个数。
tgattr	int2vector		当前没有使用。
tgargs	bytea		传递给触发器的参数字符串，每一个都以NULL结尾。

上级主题： 系统目录定义

pg_type_encoding

pg_type_encoding 系统目录表包含了列存储类型信息。

表 1. pg_catalog.pg_type_encoding

名称	类型	修改	存储	描述
typeid	oid	not null	plain	pg_attribute 的外键。
typoptions	text [ ]		extended	实际的选项。

上级主题： 系统目录定义

pg_type

pg_type 系统目录表存储有关数据类型的信息。基类（标量类型）由CREATE TYPE创建。而域由CREATE DOMAIN创建。数据库中的每一个表都会有一个自动创建的组合类型，用于表示表的行结构。也可以使用CREATE TYPE AS 创建组合类型。

表 1. pg_catalog.pg_type

名称	类型	引用	描述
typname	name		数据类型的名字。
typnamespace	oid	pg_namespace.oid	包含此类型的名字空间的OID。
typowner	oid	pg_authid.oid	类型的拥有者。
typlen	int2		对于一个固定尺寸的类型，typlen是该类型内部表示的字节数。对于一个变长类型，typlen为负值。-1表示一个“varlena”类型（具有长度字），-2表示一个以空值结尾的C字符串。
typbyval	boolean		typbyval 决定内部例程传递这个类型的数值时是通过传值还是传引用方式。如果typlen不是1、2或4（或者在Datum为8字节的机器上为8），typbyval最好是假。变长类型总是传引用。注意即使长度允许传值，typbyval也可以为假；例如，当前对于类型float4这个属性为真。
typtype	char		b表示基类，c表示组合类型（例如一个表的行类型），d表示域，e表示枚举类型，p表示伪类型。
typisdefined	boolean		如果此类型已被定义则为真，如果此类型只是一个表示还未定义类型的占位符则为假。当typisdefined为假，除了类型名字、名字空间和OID之外什么都不能被依赖。
typdelim	char		在分析数组输入时，分隔两个此类型值的字符。注意该分隔符是与数组元素数据类型相关联的， 而不是和数组的数据类型关联。
typrelid	oid	pg_class.oid	如果这是一个组合类型， 那么这个列指向pg_class中定义对应表的项（对于自由存在的组合类型，pg_class项并不表示一个表，但不管怎样该类型的pg_attribute项需要链接到它）。对非组合类型此列为零。
typelem	oid	pg_type.oid	如果typelem不为0，则它标识pg_type里面的另外一行。 当前类型可以被加上下标得到一个值为类型typelem的数组来描述。 一个"真的"数组类型是变长的（typlen = -1），但是一些定长的（tylpen > 0）类型也拥有非零的typelem，比如name和point。 如果一个定长类型拥有一个typelem， 则它的内部形式必须是某个typelem数据类型的值，不能有其它数据。变长数组类型有一个由该数组子例程定义的头。
typinput	regproc	pg_proc.oid	输入转换函数（文本格式）。
typoutput	regproc	pg_proc.oid	输出转换函数（文本格式）。
typreceive	regproc	pg_proc.oid	输入转换函数（二进制格式），如果没有则为0。
typsend	regproc	pg_proc.oid	输出转换函数（二进制格式），如果没有则为0。
typmodin	regproc	pg_proc.oid	类型修改器输入函数，如果类型没有提供修改器则为0
typmodout	regproc	pg_proc.oid	类型修改器输出函数，如果类型没有提供修改器则为0。
typanalyze	regproc	pg_proc.oid	自定义 ANALYZE 函数，0表示使用标准函数。
typalign	char		typalign 是当存储此类型值时要求的对齐性质。它应用于磁盘存储以及该值在 Greenplum内部的大多数表现形式。 如果多个值是连续存放的，比如在磁盘上的一个完整行，在这种类型的数据前会插入填充，这样它就可以按照指定边界存储。 对齐引用是该序列中第一个数据的开头。对齐引用是序列中第一个数据的开始。可能的值有：c = char对齐，即不需要对齐。s = short对齐（在大部分机器上为2字节）。i = int对齐（在大部分机器上为4字节）。d = double对齐（在很多机器上为8字节，但绝不是全部）。

typstorage	char		如果一个变长类型（typlen = -1）可被TOAST，typstorage说明这种类型的列应采取的默认策略。可能的值是：p: 值必须平面存储。e: 值可以被存储在一个"二级"关系 如果有，见pg_class.reltoastrelid)。m:值可以被压缩线内存储。x: 值可以被压缩线内存储或存储在"二级"存储。注意： m 列也可以被移动到二级存储，但只能是作为最后一种方案 (e 和 x列会先被移动）。
typnotnull	boolean		表示类型上的一个非空约束。只用于域。
typbasetype	oid	pg_type.oid	标识这个域基于的类型。如果此类不是一个域则为0。
typtypmod	int4		域使用typtypmod来记录被应用于它们基类型的typmod（如果基类型不使用typmod，则为-1）。如果此类型不是一个域则为-1。
typndims	int4		对于一个数组上的域，typndims是数组维度数（如果，typbasetype是一个数组类型；域的typelem会匹配基类型的typelem）。除数组类型上的域之外的类型的此列为0。
typdefaultbin	text		如果typdefaultbin为非空，那么它是 该类型默认表达式的nodeToString()表现形式。这个列只用于域。
typdefault	text		如果某类型没有相关默认值，那么typdefault为空。如果typdefaultbin不为空， 那么typdefault必须包含一个typdefaultbin所指的默认表达式的人类可读的版本。 如果typdefaultbin为空但typdefault不为空，则typdefault是该类型默认值的外部表现形式， 它可以被交给该类型的输入转换器来产生一个常量。

上级主题： 系统目录定义

pg_user_mapping

pg_user_mapping 存储从本地用户到远程用户的映射。 必须有管理员权限才能访问该目录。

表 1. pg_catalog.pg_user_mapping

名称	类型	引用	描述
umuser	oid	pg_authid.oid	将要被映射的本地角色的OID，如果用户映射是公共的则为0。
umserver	oid	pg_foreign_server.oid	包含此映射的外部服务器的OID。
umoptions	text [ ]		用户映射相关选项，以"keyword=value"字符串形式。

上级主题： 系统目录定义

pg_window

pg_window 表存储关于窗口函数的信息。窗口函数通常用于创建复杂的 OLAP（在线分析处理）查询。窗口函数被应用到单个查询表达式的范围分区结果集。一个窗口分区是一个查询返回行的子集，由一个特定的OVER()子句定义。典型的窗口函数有 rank、dense_rank 和 row_number。在pg_window中每个条目是 pg_proc 中一个条目的扩展。 pg_proc 中的一个条目包含了窗含函数的名称、输入以及输出数据的类型以及其它同普通函数类似的信息。

表 1. pg_catalog.pg_window

名称	类型	引用	描述
winfnoid	regproc	pg_proc.oid	窗口函数在 pg_proc 中的OID。
winrequireorder	boolean		窗口函数要求它的窗口说明有一个 ORDER BY 子句。
winallowframe	boolean		窗口函数允许它的窗口说明有一个 ROWS 或 RANGE 帧子句。
winpeercount	boolean		计算这个窗口函数要求平级分组行计数，因此 Window 节点实现必须在需要时“向前看”来保证在其中间状态中该计数可用。
wincount	boolean		计算该窗口函数要求分区行计数。
winfunc	regproc	pg_proc.oid	一个计算中间类型窗口函数值的函数在 pg_proc 中的 OID。
winprefunc	regproc	pg_proc.oid	一个计算延迟型窗口函数的部分值的预备窗口函数在 pg_proc 中的 OID。
winpretype	oid	pg_type.oid	预备窗函数的结果类型在 pg_type 中OID。
winfinfunc	regproc	pg_proc.oid	一个从分区行计数以及 winprefunc 结果计算延迟型窗口函数最终值的函数在 pg_proc 中的OID。
winkind	char		一个显示该窗口函数在一类相关函数中成员关系的字符：w - 普通的窗函数,n - NTILE函数,f - FIRST_VALUE函数,l - LAST_VALUE函数,g - LAG函数,d - LEAD函数

上级主题： 系统目录定义

gp_toolkit 管理方案

HashData 提供了一个称为 gp_toolkit 的管理方案，它能用来查询系统表目录、日志文件以及操作系统环境来获取系统状态信息。gp_toolkit 方案包含了一些能够通过 SQL 命令来访问的视图。gp_toolkit 能够被所有的数据库用户访问，但是一些对象需要超级用户权限。为了方便，用户或许想要添加 gp_toolkit 方案到其方案搜索路径中，例如：

```
=> ALTER ROLE myrole SET search_path TO myschema,gp_toolkit;
```

这份文档描述了 gp_toolkit 中大部分的有用视图。用户可能注意到了 gp_toolkit 方案中的其他对象（视图、函数以及外部表）并没有在本文档中进行描述（这些是本节中描述的视图的支持对象）。

> 警告：不要修改在 gp_toolkit 方案中的数据库对象。不要在其中创建数据库对象。改变里面的对象可能会影响到从方案对象返回的管理信息的准确性。在数据库进行备份然后通过 gpcrondump 和 gpdbrestore 工具进行恢复时，任何在 gp_toolkit 方案中的修改将会丢失。

检查需要日常维护的表

下面的视图能够帮助识别需要日常维护（VACUUM 和 / 或者 ANALYZE）的表。

- gp_bloat_diag
- gp_stats_missing

VACUUM 或者 VACUUM FULL 命令回收已经删除的或者废弃的行占据的空间。由于在 HashData 中使用了多版本并发控制（MVCC）事务并发模型 ，被删除或者更新的数据行仍然在磁盘中占据物理空间即使它们在新的事务中是不可见的。过期的行增加了在磁盘上表的大小，同时降低了表扫描的速度。

ANALYZE 命令收集了查询优化器所需的列级别的统计信息。 HashData 使用了基于代价依赖于数据库统计信息的查询优化器。准确的统计允许查询优化器能够更好的评估一个查询操作的选择率以及检索的行数，这样能选择一个更加有效的查询计划。

gp_bloat_diag

该视图显示了那些膨胀的（在磁盘上实际的页数超过了根据表统计信息得到预期的页数）正规的堆存储的表。膨胀的表需要执行 VACUUM 或者 VACUUM FULL 来回收被已经删除或者废弃的行占据的磁盘空间。该视图能够被所有用户访问，但是非超级用户看到他们有权限访问到的表。

> 注意：对于返回追加优化表信息的诊断函数，见检查追加优化表部分。

表 1. gp_bloat_diag 视图

列名	描述
bdirelid	表对象id。
bdinspname	方案名。
bdirelname	表名。
bdirelpages	在磁盘上的实际页数。
bdiexppages	根据表数据得到的期望的页数。
bdidiag	膨胀诊断信息。

gp_stats_missing

该视图显示那些没有统计信息的表，因此可能需要在表上执行 ANALYZE 命令。

表 2. gp_stats_missing 视图

列名	描述
smischema	方案名。
smitable	表名。
smisize	这些表有统计信息吗？如果这些表没有行数量统计以及行大小统计记录在系统表中，取值为false，这也表明该表需要被分析。如果表没有包含任何的函数时，值也为false。例如，分区表的父表总是空的，同时也总是返回一个false。
smicols	表中的列数。
smirecs	表中的行数。

检查锁

当一个事务访问一个关系（如，一张表），它需要获取一个锁。取决于获取锁类型，后续事务可能在它们能访问相同表之前进行等待。更多更与锁类型的信息，见 HashData 管理员指南中“管理数据“部分。 HashData 资源队列（用来负载管理）也需要使用锁来控制查询进入系统的许可。

gp_locks_* 这一类的视图能够帮助诊断那些由于一个锁而正在发生等待的查询和会话。

- gp_locks_on_relation
- gp_locks_on_resqueue

gp_locks_on_relation

该视图显示了当前所有表上持有锁，以及查询关联的锁的相关联的会话信息。更多关于锁类型的信息，见 HashData 管理员指南中“管理数据“部分。该视图能够被所有用户访问，但是非超级用户只能够看到他们有权限访问的关系上持有的锁。

表 3. gp_locks_on_relation 视图

列名	描述
lorlocktype	能够加锁对象的类型：relation、 extend、page、tuple、transactionid、object、userlock、resource queue以及advisory
lordatabase	对象存在的数据库对象ID，如果对象为一个共享对象则该值为0。
lorrelname	关系名。
lorrelation	关系对象ID。
lortransaction	锁所影响的事务ID 。
lorpid	持有或者等待该锁的服务器端进程的进程ID 。如果该锁被一个预备事务持有则为NULL。
lormode	由该进程持有或者要求的锁模式名。
lorgranted	显示是否该锁被授予（true）或者未被授予（false）。
lorcurrentquery	会话中的当前查询。

gp_locks_on_resqueue

该视图显示当前被一个资源队列持有的所有的锁，以及查询关联的锁的相关联的会话信息。该视图能够被所有用户访问，但是非超级用户只能够看到关联他们自己会话的锁。

表 4. gp_locks_on_resqueue 视图

列名	描述
lorusename	执行该会话用户的用户名。
lorrsqname	资源队列名。
lorlocktype	能够加锁的对象类型： resource queue
lorobjid	加锁事务的ID 。
lortransaction	受到锁影响的事务ID。
lorpid	受到锁影响的事务的进程ID 。
lormode	该进程持有的或者要求的锁模式的名称。
lorgranted	显示是否该锁被授予（true）或者未被授予（false）。
lorwaiting	显示是否会话正在等待。

检查追加优化表

gp_toolkit 方案包括了一系列能够用来研究追加优化表状态的诊断函数。

当一张追加优化表（或者面向列的追加优化表）被创建，另外一张表被隐式地创建，包含该表当前状态的元数据。元数据包含了诸如每个表的段的记录数的信息。

追加优化表可能有一些不可见的行，这些行已经被更新或者删除，但是仍旧保留在存储空间中直到通过使用 VACUUM 进行压缩处理。隐藏的行通过使用辅助的可见性映射表或者 visimap 来记录。

下面的函数能够让用户访问追加优化表和基于列存的表的元数据以及查看那些不可见的行。一些函数有两个版本：一个版本使用表 oid，另外一个版本使用表名。后一种版本有 "_name" 添加到函数名之后。

__gp_aovisimap_compaction_info(oid)

该函数显示一张追加优化表的压缩信息。这些信息属于 HashData 数据库 Segment 上存储该表数据的磁盘数据文件。用户可以用这些信息决定那些将要通过追加优化表上的 VACUUM 操作进行压缩的数据文件。

> 注意：直到一个 VACUUM 命令从数据文件中删除了行，已经被删除或者被更新的元组仍旧在磁盘上占有物理空间，即使它们对新的事务是不可见的。配置参数 gp_appendonly_compaction 控制 VACUUM 的功能。

该表描述了 __gp_aovisimap_compaction_info 函数输出表。

表 5. __gp_aovisimap_compaction_info 输出表

列名	描述
conten	HashData 段ID。
datafile	在段中的数据文件的ID。
compaction_possible	该值的取值要么为t要么为f。当取值为t是表明数据文件中的数据在一个 VACUUM命令执行时被压缩。 服务器端配置参数 gp_appendonly_compaction_threshold 影响该值。
hidden_tupcount	在数据文件中，被隐藏（已经删除的或者被更新的）的行数。
total_tupcount	在数据文件中，总的行数。
percent_hidden	在数据文件中，隐藏行（已经被删除或被更新的）占总的行数的比率（百分比）。

__gp_aoseg_name('table_name')

该函数返回包含在追加优化表的磁盘上的段文件中的元数据信息。

表 6. __gp_aoseg_name 输出表

列名	描述
segno	文件段号。
eof	该文件段有效的结尾。
tupcount	该段中总的元组数（包括不可见的元组）。
varblockcount	段文件中总的 varlbock 数。
eof_uncompressed	没有压缩的文件的文件结尾。
modcount	数据修改操作的数量。
state	文件段的状态。指示段是否是活跃的或者在压缩后可以被删除。

__gp_aoseg_history(oid)

该函数返回包含在追加优化表的磁盘上的段文件中的元数据信息。它显示 aoseg 元信息的所有不同版本（堆元组）。该数据较为复杂，但是对于那些对系统有深入了解的用户能够用这些信息来调试。

输入参数为追加优化表的 OID。

调用以表名作为参数的函数 __gp_aoseg_history_name('table_name') 可以获取相同的结果。

表 7. __gp_aoseg_history 输出表

列名	描述
gp_tid	元组的id。
gp_xmin	最早事务的id。
gp_xmin_status	gp_xmin 事务的状态。
gp_xmin_commit_	gp_xmin 事务的提交分布式 ID。
gp_xmax	最晚的事务ID。
gp_xmax_status	最晚事务的状态。
gp_xmax_commit_	gp_xmax 事务的提交分布式ID。
gp_command_id	查询命令的ID。
gp_infomask	包含状态信息的位图。
gp_update_tid	如果行被更新，该字段指明新元组的ID 。
gp_visibility	元组的可见状态。
segno	在段文件中段的数量。
tupcount	总的元组数目（包括隐藏元组）。
eof	段的有效文件结尾。
eof_uncompressed	数据没有被压缩的文件的文件结尾。
modcount	数据修改的计数。
state	段的状态。

__gp_aocsseg(oid)

该函数返回包含在列存式的追加优化表的磁盘段文件中的元数据信息，但不包括不可见行。每行描述表中一个列的一个段。

输入参数是列存追加优化表的 OID。调用以表名作为参数的函数 __gp_aocsseg_name('table_name')能够获取相同的结果。

表 8. __gp_aocsseg(oid) 输出表

列名	描述
gp_tid	表的id。
segno	段号。
column_num	列号。
physical_segno	在段文件中的段编号。
tupcount	段中行的数量，不考虑隐藏的元组。
eof	段的有效文件结尾。
eof_uncompressed	数据没有被压缩的段的有效文件结尾。
modcount	段中数据修改操作的计数。
state	段的状态。

__gp_aocsseg_history(oid)

该函数返回包含在列存式的追加优化表的磁盘段文件中的元数据信息。每行描述了一列的一个段。该数据较为复杂，但是对于那些对系统有深入了解用户能够可用这些信息来调试。

输入参数是列存式的追加优化表的OID。调用以表名作为参数的函数__gp_aocsseg_history_name('table_name') 能够得到相同的结果。

表 9. __gp_aocsseg_history 输出表

列名	描述
gp_tid	元组的OID。
gp_xmin	最早的事务。
gp_xmin_status	gp_xmin事务的状态。
gp_xmin_	gp_xmin的文本表示。
gp_xmax	最晚的事务。
gp_xmax_status	gp_xmax事务的状态。
gp_xmax_	gp_xmax的文本表示。
gp_command_id	在元组上的操作的命令ID。
gp_infomask	包含状态信息的位图。
gp_update_tid	如果行被更新，这里是更新的元组的ID。
gp_visibility	元组的可见性状态。
segno	段文件中的段号。
column_num	列号。
physical_segno	包含列上数据的段。
tupcount	段中总的元组数目。
eof	段的有效文件结尾。
eof_uncompressed	数据没有被压缩的段的文件结尾。
modcount	数据修改操作的计数。
state	段的状态。

__gp_aovisimap(oid)

该函数返回元组id、段文件、根据可见性映射得到的每个不可见元素的行号。

输入参数为一个追加优化表的id。

通过使用以表名作为参数的函数 __gp_aovisimap_name('table_name') 能够获取相同的结果。

列名	描述
tid	元组id。
segno	段文件的编号。
row_num	一个已经被删除或者更新的行的行号。

__gp_aovisimap_hidden_info(oid)

该函数返回一个追加优化表在段文件中隐藏行和可见元组的数目。

输入参数为追加优化表的 oid。

调用以表名为输入参数的函数 __gp_aovisimap_hidden_info_name('table_name')能够获取相同的结果。

列名	描述
segno	段文件的段号。
hidden_tupcount	段中隐藏元组的数目。
total_tupcount	段文件中总的元组数目。

__gp_aovisimap_entry(oid)

该函数返回表的每个可见映射的入口信息。

输入参数为一个追加优化表的 oid。

调用以表名为参数的函数 __gp_aovisimap_entry_name('table_name') 能够获取相同的结果。

表 10. __gp_aovisimap_entry 输出表

列名	描述
segno	可见性映射项的段号。
first_row_num	项的第一行行号。
hidden_tupcount	在项中的隐藏元组的总数。
bitmap	可见性位图的文本表示。

查看 HashData 服务器日志文件

每个 HashData 的组件（Master、后备Master、主Segment、镜像Segment）都会保存它们主机的服务器日志文件。gp_log_* 视图家族允许用户发出针对服务器日志文件的SQL查询来查找感兴趣的特定项。使用这些视图需要有超级用户权限。

- gp_log_command_timings
- gp_log_database
- gp_log_master_concise
- gp_log_system

gp_log_command_timings

该视图用一个外部表来读取在主机上的日志文件同时报告在数据库会话中 SQL 命令的执行时间。使用该视图需要拥有超级用户权限。

表 11. gp_log_command_timings 视图

列名	描述
logsession	会话标识符（带有“con”前缀）。
logcmdcount	一个会话中的命令数（带有“cmd”前缀）。
logdatabase	数据库名。
loguser	数据库用户名。
logpid	进程id（带有前缀“p”）
logtimemin	该命令的第一条日志信息的时间。
logtimemax	该命令最后一条信息日志信息的时间。
logduration	语句从开始到结束的持续时间。

gp_log_database

该视图使用一个外部表来读取整个 HashData 系统（主机，段，镜像）的服务器日志文件和列出与当前数据库关联的日志的入口。相关联的日志的入口能够使用会话 id（logssesion）和命令 id（logcmdcount）来标识。使用这个视图需要超级用户的权限。

表 12. gp_log_database 视图

列名	描述
logtime	日志信息的时间戳。
loguser	数据库用户名。
logdatabase	数据库名。
logpid	相关联的进程id（以“p”为前缀）。
logthread	相关联的线程计数（以“th”为前缀）。
loghost	Segment或者Master主机名。
logport	Segment或者Master端口。
logsessiontime	会话连接打开的时间。
logtransaction	全局事务ID。
logsession	会话标识符（以“con”为前缀）。
logcmdcount	一个会话中的命令编号（以“cmd”为前缀）。
logsegment	Segment内容的标识符（基础以“seg”为前缀，镜像以“mir”为前缀）。Master总是有一个-1的内容ID。
logslice	切片id（正在被执行的查询计划一部分）。
logdistxact	分布式式事务ID。
loglocalxact	本地事务ID。
logsubxact	子事务ID。
logseverity	LOG、ERROR、FATAL、 PANIC、DEBUG1 或者 DEBUG2。
logstate	与日志消息相关的 SQL 状态编码。
logmessage	日志或者错误消息文本。
logdetail	与错误消息相关的详细消息文本。
loghint	与错误消息相关的提示消息文本。
logquery	内部产生的查询文本。
logquerypos	内部产生的查询文本中的光标索引。
logcontext	该消息产生的上下文。
logdebug	带有全部细节的查询字符串，方便调试。
logcursorpos	查询字符串中的光标索引。
logfunction	该消息产生的函数。
logfile	该消息产生的日志文件。
logline	指明在日志文件产生该消息的行。
logstack	与该消息相关的栈追踪的完整文本信息。

gp_log_master_concise

该视图使用一个外部表读取来自 Master 日志文件中日志域的一个子集。使用该视图需要超级用户权限。

表 13. gp_log_master_concise视图

列名	描述
logtime	日志消息的时间戳。
logdatabase	数据库的名称。
logsession	会话标识符（以“con”为前缀）。
logcmdcount	会话中的命令编号（以“cmd”为前缀）。
logmessage	日志或者出错消息文本。

gp_log_system

该视图使用一个外部表来读取来自整个 HashData（Master、Segment、镜像）的服务器日志文件并且列出所有的日志项。相关的日志项可以通过会话 ID（logsession）和命令 ID（logcmdcount）来标识。该视图的使用需要有超级用户权限。

表 14. gp_log_system视图

列名	描述
logtime	日志消息的时间戳。
loguser	数据库用户名。
logdatabase	数据库名。
logpid	相关联的进程id（以“p”为前缀）。
logthread	相关联的线程计数（以“th”为前缀）。
loghost	Segment或者Master的主机名。
logport	Segment或者Master的端口。
logsessiontime	会话连接打开的时间。
logtransaction	全局事务ID。
logsession	会话标识符（以“con”为前缀）。
logcmdcount	会话中的命令编号（以“cmd“为前缀）。
logsegment	Segment内容标识符（基础以”seg“为标识符，镜像以“mir”为标识符，Master的内容ID总是-1）。
logslice	切片ID（正在被执行的查询计划的部分）。
logdistxact	分布式事务ID。
loglocalxact	本地事务ID。
logsubxact	子事务ID。
logseverity	LOG、ERROR、FATAL、 PANIC、DEBUG1或者DEBUG2。
logstate	和日志消息关联的SQL状态代码。
logmessage	日志或者错误消息文本。
logdetail	与错误消息关联的详细消息文本。
loghint	与错误消息关联的提示消息文本。
logquery	内部产生的查询文本。
logquerypos	内部产生的查询文本中的光标索引。
logcontext	该消息产生的上下文。
logdebug	查询字符串，带有完整的细节用于调试。
logcursorpos	查询字符串中的光标索引。
logfunction	该消息产生的函数。
logfile	该消息产生的日志文件。
logline	指明在日志文件产生该消息的行。
logstack	与该消息关联的跟踪栈的完整文本。

检查服务器端配置文件

每个 HashData 数据库组件（Master、后备 Master、主 Segment 及镜像 Segment）都有它们自己的服务器配置文件（postgresql.conf）。下面的 gp_toolkit 对象能够用来检查系统中所有的主postgresql.conf 文件中的参数设置：

- gp_param_setting('parameter_name')
- gp_param_settings_seg_value_diffs

gp_param_setting('parameter_name')

该函数接受服务器配置参数的名称作为参数，并且返回 Master 和每个活动 Segment 的 postgresql.conf 值。该函数能够被所有用户访问。

表 15. gp_param_setting('parameter_name') 函数

列名	描述
paramsegment	Segment内容ID（只显示活动Segment）。Master的内容ID总是为-1。
paramname	参数的名字。
paramvalue	参数的值。

示例：

```
SELECT * FROM gp_param_setting('max_connections');
```

gp_param_settings_seg_value_diffs

那些被分类为本地（local）（表示每个 Segment 从其自己的 postgresql.conf 文件中获取参数值）的服务器配置参数，应该在所有 Segment 上做相同的设置。该视图显示了那些不一致的本地参数的设置。那些被认为应该有不同值的参数（例如 port）不包括在内。该视图能够被所有的用户访问。

表 16. gp_param_settings_seg_value_diffs 视图

列名	描述
psdname	参数名。
psdvalue	参数值。
psdcount	拥有该值的 Segment 的数目。

检查失效的 Segment

gp_pgdatabase_invalid 视图能够被用来检查状态为 “down” 的 Segment。

gp_pgdatabase_invalid

该视图显示系统目录中被标记为 down 的 Segment 的信息。该视图能够被所有的用户访问。

表 17. gp_pgdatabase_invalid 视图

列名	描述
pgdbidbid	Segment的dbid。每个Segment有唯一的dbid。
pgdbiisprimary	Segment当前是否为主（活动）Segment？（t或者f）
pgdbicontent	该Segment的内容ID。一个主Segment和镜像Segment将有相同的内容ID。
pgdbivalid	该Segment是否为up和有效？（t或者f）
pgdbidefinedprimary	在系统初始化时，该Segment是否已经被指定为主Segment角色？（t或者f）

检查资源队列的活动和状态

使用资源队列的目的是限制任何给定时刻系统中的活跃查询的数量，从而避免使得一些系统资源（例如，内存、CPU、磁盘I/O）耗尽。所有的数据库用户都被指定了一个资源队列，每个由用户提交的语句在它们能够运行前，首先在资源队列限制上进行评估。gp_resq_* 视图族可以被用来通过当前已被提交给系统的语句的相应资源队列检查其状态。注意那些由超级用户发出的语句会从资源队列中豁免。

gp_resq_activity

对于那些有活动负载的资源队列，该视图为每一个通过资源队列提交的活动语句显示一行。该视图能够被所有用户访问。

表 18. gp_resq_activity 视图

列名	描述
resqprocpid	指定给该语句的进程ID（在Master上）。
resqrole	用户名。
resqoid	资源队列对象ID。
resqname	资源队列名。
resqstart	语句被发送到系统的时间。
resqstatus	语句的状态：正在执行、等待或者取消。

gp_resq_activity_by_queue

对于有活动负载的资源队列，该视图显示了队列活动的总览。该视图能够被所有用户访问。

表 19. gp_resq_activity_by_queue 列

列名	描述
resqoid	资源队列对象ID。
resqname	资源队列名。
resqlast	最后一个语句发送到队列的时间。
resqstatus	最后一个语句的状态：正在执行、等待或者取消。
resqtotal	该队列总的语句数。

gp_resq_priority_statement

该视图为当前运行在 HashData 数据库系统上的所有语句显示资源队列优先级、会话 ID 以及其他信息。该视图能够被所有用户访问。

表 20. gp_resq_priority_statement 视图

列名	描述
rqpdatname	该会话连接到的数据库的名称。
rqpusename	发出该语句的用户。
rqpsession	会话ID。
rqpcommand	该会话中的语句编号（命令ID和会话ID唯一标识一条语句）。
rqppriority	语句的资源队列优先级（MAX、HIGH、MEDIUM、LOW）。
rqpweight	一个与该语句的优先级相关的整数值。
rqpquery	语句的查询文本。

gp_resq_role

该视图显示与角色相关的资源队列。该视图能够被所有用户访问。

表 21. gp_resq_role 视图

列名	描述
rrrolname	角色（用户）名。
rrrsqname	指定给角色的资源队列名。如果一个角色没有被明确地指定一个资源队列，那么它将会在默认资源队列 pg_default 中。

gp_resqueue_status

该视图允许管理员查看到一个负载管理资源队列的状态和活动。它显示在系统中一个特定的资源队列有多少查询正在等待执行以及有多少查询当前是活动的。

表 22. gp_resqueue_status 视图

列名	描述
queueid	资源队列的ID。
rsqname	资源队列名。
rsqcountlimit	一个资源队列的活动查询数的阈值。如果值为-1则意味着没有限制。
rsqcountvalue	资源队列中当前正在被使用的活动查询槽的数量。
rsqcostlimit	资源队列的查询代价阈值。如果值为-1则意味着没有限制。
rsqcostvalue	当前在资源队列中所有语句的总代价。
rsqmemorylimit	资源队列的内存限制。
rsqmemoryvalue	当前资源队列中所有语句使用的总内存。
rsqwaiters	当前在资源队列中处于等待状态的语句数目。
rsqholders	资源队列中当前正在系统上运行的语句数目。

检查查询磁盘溢出空间使用

gp_workfile_* 视图显示当前所有使用磁盘溢出空间的查询的信息。如果没有充足的内存来执行查询， HashData 会在磁盘上创建工作文件。该信息能够别用来排查问题以及调优查询。在这些视图中的信息也能够被用来指定 HashData 数据库配置参数 gp_workfile_limit_per_query 和 gp_workfile_limit_per_segment 的值。

- gp_workfile_entries

- gp_workfile_usage_per_query
- gp_workfile_usage_per_segment

gp_workfile_entries

该视图为当前在Segment上使用磁盘空间作为工作文件的操作符包含一行。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的数据库的信息。

表 23. gp_workfile_entries

列名	类型	参考	描述
command_cnt	integer		查询的命令ID。
content	smallint		一个Segment实例的内容标识符。
current_query	text		当前进程正在执行的查询。
datname	name		HashData 数据库名。
directory	text		工作文件的路径。
optype	text		创建工作文件的查询操作符类型。
procpid	integer		服务器进程的进程ID 。
sess_id	integer		会话ID 。
size	bigint		以字节为单位的工作文件的大小。
numfiles	bigint		被创建文件的数目。
slice	smallint		查询计划切片。查询计划正在被执行的部分。
state	text		创建工作文件的查询的状态。
usename	name		角色名。
workmem	integer		分配给操作符的内存量（以KB为单位）。

gp_workfile_usage_per_query

该视图对当前时间每个段上的每个使用磁盘空间作为工作文件的查询包含一行。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的数据库的信息。

表 24. gp_workfile_usage_per_query

列名	类型	参考	描述
command_cnt	integer		查询的命令ID。
content	smallint		一个Segment实例的内容标识符。
current_query	text		当前进程正在执行的查询。
datname	name		HashData 数据库名。
procpid	integer		服务器进程的进程ID 。
sess_id	integer		会话ID 。
size	bigint		以字节为单位的工作文件的大小。
numfiles	bigint		被创建文件的数目。
state	text		创建工作文件的查询的状态。
usename	name		角色名。

gp_workfile_usage_per_segment

这个视图为每个Segment包含一行。每一行显示当前在该Segment被用于工作文件的磁盘空间总量。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的数据库的信息。

表 25. gp_workfile_usage_per_segment

列名	类型	参考	描述
content	smallint		一个Segment实例的内容标识符。
size	bigint		一个Segment上工作文件的总大小。
numfiles	bigint		已经创建的文件数目。

浏览用户和组（角色）

将用户（角色）分组在一起来简化对象特权的管理常常会很方便：这样，特权能够对一个组进行整体的分配和回收。在 HashData 数据库中，这可以通过创建一个代表组的角色来实现，然后授予组角色的成员关系给每个单独的用户角色。

gp_roles_assigned 视图能够用来查看系统中所有的角色以及指派给它们的成员（如果该角色同时也是一个组角色）。

gp_roles_assigned

该视图显示系统中所有的角色以及指派给它们的成员（如果该角色同时也是一个组角色）。该视图能够被所有用户访问。

表 26. gp_roles_assigned 视图

列名	描述
raroleid	角色对象ID。如果该角色有成员（用户），它将被认为是一个组角色。
rarolename	角色（用户或者组）名。
ramemberid	该角色的一个成员角色的角色对象ID 。
ramembername	该角色的一个成员角色的名称。

检查数据库对象的大小和磁盘空间

gp_size_* 视图族能被用来确定分布式 HashData 数据库、方案、表或者索引的磁盘空间使用。下面的视图计算一个对象在所有主 Segment（镜像没有包括在尺寸计算中）上的总尺寸。

- gp_size_of_all_table_indexes
- gp_size_of_database
- gp_size_of_index
- gp_size_of_partition_and_indexes_disk
- gp_size_of_schema_disk
- gp_size_of_table_and_indexes_disk
- gp_size_of_table_and_indexes_licensing
- gp_size_of_table_disk
- gp_size_of_table_uncompressed
- gp_disk_free

表和索引的尺寸视图通过对象 ID（不是名字）列出关系。为了通过名字来检查一个表或者索引的尺寸，用户必须在 pg_class 中查看查看关系名（relname）。例如：

```
SELECT relname as name, sotdsize as size, sotdtoastsize as
toast, sotdadditionalsize as other
FROM gp_size_of_table_disk as sotd, pg_class
WHERE sotd.sotdoid=pg_class.oid ORDER BY relname;
```

gp_size_of_all_table_indexes

该视图显示了一个表上所有索引的总尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的关系。

表 27. gp_size_of_all_table_indexes 视图

列名	描述
soatioid	表的对象ID 。
soatisize	所有表索引的总尺寸（以字节为单位）。
soatischemaname	模式名。
soatitablename	表名。

gp_size_of_database

该视图显示数据库的总尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的数据库。

表 28. gp_size_of_database 视图

列名	描述
sodddatname	数据库名。
sodddatsize	数据库的尺寸（以字节为单位）。

gp_size_of_index

该视图显示一个索引的总尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的关系。

表 29. gp_size_of_index 视图

列名	描述
soioid	索引的对象ID 。
soitableoid	索引所属表的对象ID 。
soisize	索引的尺寸（以字节为单位）。
soiindexschemaname	索引的方案名。
soiindexname	索引名。
soitableschemaname	表的方案名。
soitablename	表名。

gp_size_of_partition_and_indexes_disk

该视图显示分区子表及其索引在磁盘上的尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的关系。

表 30. gp_size_of_partition_and_indexes_disk 视图

列名	描述
sopaidparentoid	父表的对象ID 。
sopaidpartitionoid	分区表的对象ID 。
sopaidpartitiontablesize	分区表的尺寸（以字节为单位）。
sopaidpartitionindexessize	在该分区上，所有索引的总尺寸。
Sopaidparentschemaname	父表所在的方案名。
Sopaidparenttablename	父表名。
Sopaidpartitionschemaname	分区所在的方案名。
sopaidpartitiontablename	分区表名。

gp_size_of_schema_disk

该视图显示当前数据库中 public 方案和用户创建方案的方案尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的方案。

表 31. gp_size_of_schema_disk 视图

列名	描述
sosdnsp	方案名。
sosdschematablesize	方案中所有表的总尺寸（以字节为单位）。/td>
sosdschemaidxsize	方案中所有索引的总尺寸（以字节为单位）。

gp_size_of_table_and_indexes_disk

该视图显示表及其索引在磁盘上的大小。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的关系。

表 32. gp_size_of_table_and_indexes_disk 视图

列名	描述
sotaidoid	父表的对象ID 。
sotaidtablesize	表在磁盘上的尺寸。
sotaididxsize	表上所有索引的总尺寸。
sotaidschemaname	方案名。
sotaidtablename	表名。

gp_size_of_table_and_indexes_licensing

该视图显示表及其索引的总尺寸，用于特许目的。使用该视图需要超级用户权限。

表 33. gp_size_of_table_and_indexes_licensing 视图

列名	描述
sotailoid	表的对象ID。
sotailtablesizedisk	表的总磁盘尺寸。
sotailtablesizeuncompressed	如果表是一个压缩的追加优化表，显示未压缩时的尺寸（以字节为单位）。
sotailindexessize	表中所有索引的总尺寸。
sotailschemaname	方案名。
sotailtablename	表名。

gp_size_of_table_disk

该视图显示一个表在磁盘上的尺寸。该视图能够被所有的用户访问，但是非超级用户只能看到他们有权访问的表。

表 34. gp_size_of_table_disk 视图

列名	描述
sotdoid	表的对象ID 。
sotdsize	表的尺寸（以字节为单位）。只考虑主表尺寸。其中并未包括辅助对象，例如超尺寸（toast）属性或者AO表的额外存储对象。
sotdtoastsize	TOAST表（超尺寸属性存储）的尺寸，如果存在TOAST。
sotdadditionalsize	反映追加优化（AO）表的段和块目录表尺寸。
sotdschemaname	方案名。
sotdtablename	表名。

gp_size_of_table_uncompressed

该视图显示追加优化（AO）表没有压缩时的尺寸。否则，显示表在磁盘上的尺寸。使用该视图需要超级用户权限。

表 35. gp_size_of_table_uncompressed 视图

列名	描述
sotuoid	表对象ID。
sotusize	如果是一个压缩的AO表则显示它没有被压缩时的尺寸（以字节为单位）。否则显示该表在磁盘上的尺寸。
sotuschemaname	方案名。
sotutablename	表名。

gp_disk_free

该外部表在活动Segment主机上运行df（磁盘空闲）并且报告返回的结果。计算中不包括不活动的镜像Segment。这个外部表的使用要求超级用户权限。

表 36. gp_disk_free external 表

列名	描述
dfsegment	Segment的内容ID（只显示活动的Segment）。
dfhostname	Segment主机的主机名。
dfdevice	设备名。
dfspace	Segment文件系统中的空闲磁盘空间（以KB为单位）。

检查不均匀的数据分布

在 HashData 数据库中，所有的表都是分布式的，意味着它们的数据被分布在了系统中所有的Segment中。如果数据没有被均匀地分布，那么查询处理性能可能会受到影响。如果一个表有不均匀的数据分布，下面的视图能够帮助诊断：

- gp_skew_coefficients
- gp_skew_idle_fractions

gp_skew_coefficients

该视图通过计算存储在每个Segment上的数据的变异系数（CV）来显示数据分布倾斜。该视图能够被所有用户访问，但是非超级用户只能看到他们有权访问的表。

表 37. gp_skew_coefficients 视图

列名	描述
skcoid	表的对象id。
skcnamespace	表定义在其中的命名空间。
skcrelname	表名。
skccoeff	变异系数（CV）计算为标准差除以平均值。它同时考虑了一个数据序列的平均值以及平均值的变化性。值越低，情况越好。值越大表明数据倾斜越严重。

gp_skew_idle_fractions

该视图通过计算在表扫描过程中系统空闲的百分比来显示数据分布倾斜，这是一种数据处理倾斜的指示器。该视图能够被所有用户访问，但是非超级用户只能看到他们有权访问的表。

表 38. gp_skew_idle_fractions 视图

列名	描述
sifoid	表的对象ID。
sifnamespace	表被定义在其中的命名空间。
sifrelname	表名。
siffraction	在一次表扫描过程中系统空闲的百分比，该值是不均匀数据分布或者查询处理倾斜的指示器。例如，一个0.1的值表示有10%的倾斜，0.5的值表示有50%的倾斜等等。如果一个表的倾斜超过10%，就应该评估它的分布策略。

gpperfmon 数据库

gpperfmon 数据库是一个专门用于数据收集代理在 HashData 的 Segment 主机上保存统计信息的指定数据库。可选的 HashData 命令中心管理工具需要该数据库。gpperfmon 数据库通过使用命令行实用程序 gpperfmon_install 创建。该程序创建 gpperfmon 数据库以及 gpmon 数据库上的角色，同时，开启在 Segment 主机上的监视代理。更多关于使用该工具和配置数据收集代理的信息请见 HashData 数据库工具指南中的 gpperfmon_install 参考。

gpperfmon 数据库包含下面三种表集合。

- now 这类表存储当前系统度量数据，例如活跃的查询。
- history 这类表存储历史度量数据。
- tail 这类表存储临时数据。Tail 这类表只在内部使用不应该被用户查询。now 和 tail 这两类表在 Master 主机文件系统中被存储为文本文件，能够被 gpperfmon 数据库以外部表的形式访问。history 这类表是存储在 gpperfmon 数据库中的普通数据库表。

gpperfmon 数据库中包含下面类别的表：

- database_* 表存储一个 HashData 数据库实例的查询负载信息。
- diskspace_* 表存储磁盘空间度量。
- filerep_* 表存储文件复制过程的健康和状态度量。该过程是 HashData 数据库实例实现高可用/镜像 的方式。为每一对主-镜像都会维护统计信息。
- interface_stats_* 表存储每个 HashData 数据库实例的每个活跃接口的统计度量。注意：这些表已经就位以备将来所用，现在还没有进行填充。
- log_alert_* 表存储关于 pg_log 错误和警告的信息。
- queries_* 表存储高层次的查询状态信息。
- segment_* 表存储 HashData 数据库 Segment 实例的内存分配统计。
- socket_stats_*表存储一个 HashData 数据库实例的套接字使用的统计信息。注意：这些表已经就位以备将来使用，现在还没有进行填充。
- system_* 表存储系统利用的度量。

gpperfmon 数据库还包含以下的视图：

- dynamic_memory _info 视图显示 每台主机上所有Segment的聚合以及每台主机上已经使用的动态内存量。
- memory_info 视图显示从 system_history 和 segment_history 表获取的关于每个主机的内存信息。

database_*

database_* 表存储一个 HashData 数据库实例的查询负载信息。有三个数据库表，它们都有相同的列：

- database_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 database_history 表之间的时段，当前查询负载数据存储在 database_now 中。
- database_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 database_now 中清除但是还没有提交到 database_history 的查询负载数据。它通常只包含了几分钟的数据。
- database_history 是一个常规表，它存储数据库范围的历史查询负载数据。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
ctime	timestamp(0)	此行创建的时间
queries_total	int	数据收集时刻，数据库中总的查询数量。
queries_running	int	数据收集时刻，数据库中活跃的查询数量。
queries_queued	int	数据收集时刻，在资源队列中等待的查询数量。

diskspace_*

diskspace_* 表存储磁盘空间度量。

- diskspace_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 diskspace_history 表之间的时段，当前磁盘空间度量存储在 diskspace_now 中。
- diskspace_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 diskspace_now 中清除但是还没有提交到 diskspace_history 的磁盘空间度量。它通常只包含了几分钟的数据。
- diskspace_history 是一个常规表，它存储数据库范围的历史磁盘空间度量。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
ctime	timestamp(0) without time zone	磁盘空间测试时间。
hostname	varchar(64)	磁盘空间测试的主机名称。
Filesystem	text	磁盘空间测试的文件系统名称。
total_bytes	bigint	文件系统中的总的字节数。
bytes_used	bigint	在文件系统中使用的总的字节数。
bytes_available	bigint	在文件系统中空的字节数

dynamic_memory_info

dynamic_memory_info 视图显示在一个 Segment 主机上所有 Segment 实例已经使用以及可用动态内存的总额。动态内存是指 HashData 数据库实例开始取消进程之前允许单个 Segment 实例的查询进程消耗的最大内存量。该限制通过 gp_vmem_protect_limit 服务器配置参数来进行设置，并且会为每个 Segment 分别计算。

Column	Type	Description
ctime	timestamp without time zone	在segment_history表中创建该行的时间。
hostname	varchar(64)	与这些系统内存度量相关的Segment或者Master主机名。
dynamic_memory_used_mb	numeric	分配给在该Segment上运行的查询进程的动态内存总量（以MB为单位）。
dynamic_memory_available_mb	numeric	在这台Segment主机上运行的查询进程可用的额外动态内存量（以MB为单位）。注意这个值是一台主机上对所有Segment可用的内存总量。即便这个值报告了可用内存，仍有可能该主机上的一个或者更多Segment已经超过了它们的内存限制（通过gp_vmem_protect_limit参数设置）。

filerep_*

filerep* 表为 HashData 数据库实例存储高可用文件复制进程信息。有三个 filerep 表，它们都有相同的列：

- filerep_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 filerep_history 表之间的时段，当前的文件复制数据存储在 filerep_now 中。
- filerep_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 filerep_now 中清除但是还没有提交到 filerep_history 的文件复制数据。它通常只包含了几分钟的数据。
- filerep_history 是一个常规表，它存储数据库范围的历史文件复制数据。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
ctime	timestamp	此行创建的时间
primary_measurement_microsec	bigint	说明主度量（包含在 UDP 消息中）是在多长的时间段上被收集的。
mirror_measurement_microsec	bigint	说明镜像度量（包含在UDP消息中）是在多长的时间段上被收集的。
primary_hostname	varchar(64)	primary 主机的名称
primary_port	int	Primary 主机的端口号
mirror_hostname	varchar(64)	Mirror 主机的名称
mirror_port	int	Mirror 主机的端口号
primary_write_syscall_bytes_avg	bigint	每个间隔中，主服务器上通过写系统调用写到磁盘上的平均数据量
primary_write_syscall_byte_max	bigint	每个间隔中，主服务器上通过写系统调用写到磁盘上的最大数据量
primary_write_syscall_microsecs_avg	bigint	每个间隔中，主服务器上一次写系统调用写数据到磁盘上所要求的平均时间
primary_write_syscall_microsecs_max	bigint	每个间隔中，主服务器上一次写系统调用写数据到磁盘上所要求的最大时间
primary_write_syscall_per_sec	double precision	主服务器上每秒写系统调用的数量。它只反映将写入磁盘放入内存队列中的时间
primary_fsync_syscall_microsec_avg	bigint	每个间隔中，主服务器上一次文件同步系统调用写数据到磁盘上所要求的平均时间
primary_fsync_syscall_microsec_max	bigint	每个间隔中，主服务器上一次文件同步系统调用写数据到磁盘上所要求的最大时间
primary_fsync_syscall_per_sec	double precision	主服务器上每秒的文件同步系统调用数量。和在数据被递送或者加入队列后就立刻返回的写系统调用不同，文件同步系统调用会等待所有未解决的写入被写到磁盘上。这一列中的文件同步系统调用值反映了大量数据的实际磁盘访问时间
primary_write_shmem_bytes_avg	bigint	每个间隔中，主服务器上被写到共享内存中的平均数据量
primary_write_shmem_bytes_max	bigint	每个间隔中，主服务器上被写到共享内存中的最大数据量
primary_write_shmem_microsec_avg	bigint	每个间隔中，主服务器上写入数据到共享内存中所需要的平均时间
primary_write_shmem_microsec_max	bigint	每个间隔中，主服务器上写入数据到共享内存中所需要

		的最大时间
primary_write_shmem_per_sec	double precision	主服务器上每秒写到共享内存中的次数
primary_fsync_shmem_microsec_avg	bigint	每个间隔中，主服务器上文件同步系统调用写入数据到共享内存中所需要的平均时间
primary_fsync_shmem_microsec_max	bigint	每个间隔中，主服务器上文件同步系统调用写入数据到共享内存中所需要的最大时间
primary_fsync_shmem_per_sec	double precision	主服务器每秒写到共享内存的文件同步调用次数。这一列中的文件同步系统调用值反映了大数据量时的实际磁盘访问次数
primary_roundtrip_fsync_msg_microsec_avg	bigint	每个间隔中，主服务器和镜像服务器之间一次往返文件同步所需要的平均时间。包括：从主服务器到镜像服务器的一个文件同步消息的排队。消息在网络中的传播。在镜像服务器上文件同步的执行。文件同步确认信息通过网络传回主服务器。
primary_roundtrip_fsync_msg_microsec_max	bigint	每个间隔中，主服务器和镜像服务器之间一次往返文件同步所需要的最大时间。包括：从主服务器到镜像服务器的一个文件同步消息的排队。消息在网络中的传播。在镜像服务器上文件同步的执行。文件同步确认信息通过网络传回主服务器。
primary_roundtrip_fsync_msg_per_sec	double precision	每秒往返文件同步的数量。
primary_roundtrip_test_msg_microsec_avg	bigint	每个间隔，主服务器和镜像服务器之间一次往返测试消息完成所需的平均时间。这与primary_roundtrip_fsync_msg_microsec_avg类似，不过它不包括磁盘访问部分。因此，这是一个显示文件复制过程中平均网络延迟的有用的度量。
primary_roundtrip_test_msg_microsec_max	bigint	每个间隔，主服务器和镜像服务器之间一次往返测试消息完成所需的最大时间。这与primary_roundtrip_fsync_msg_microsec_max类似，不过它不包括磁盘访问部分。因此，这是一个显示文件复制过程中最大网络延迟的有用的度量。
primary_roundtrip_test_msg_per_sec	double precision	每秒的往返文件同步数量。类似于primary_roundtrip_fsync_msg_per_sec，但它不包括磁盘访问部分。因此，这是一个显示文件复制过程中网络延迟的有用的度量。注意：测试消息通常每分钟发生一次，所以通常在不包含测试消息的时间段会看到值为“0”。
mirror_write_syscall_size_avg	bigint	每个间隔中，镜像服务器上写系统调用写入到磁盘的平均数据量。
mirror_write_syscall_size_max	bigint	每个间隔中，镜像服务器上写系统调用写入到磁盘的最大数据量。
mirror_write_syscall_microsec_avg	bigint	每个间隔中，镜像服务器上一次写系统表用写数据到磁盘所需的平均时间。
mirror_write_syscall_microsec_max	bigint	每个间隔中，镜像服务器上一次写系统表用写数据到磁盘所需的最大时间。

interface_stats_*

interface_stats_* 表存储 HashData 数据库实例上每个活动接口上通信的统计度量。

这些表已经就位以备将来的使用，当前还没有进行填充。

这里有三个 interface_stats 表，它们都有相同的列：

- interface_stats_now 是一个外部表，其数据文件存储在$MASTER_DATA_DIRECTORY/gpperfmon/data中。
- interface_stats_tail是一个外部表，其数据文件存储在$MASTER_DATA_DIRECTORY/gpperfmon/data中。这是一个过渡表，其中存放着已经从interface_stats_now中清除但是还没有提交到interface_stats_history的统计性接口度量。它通常只包含了几分钟的数据。
- interface_stats_history是一个常规表，它存储数据库范围的历史统计性接口度量。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

列名	类型	描述
interface_name	string	接口名。例如，eth0、eth1、lo等。
bytes_received	bigint	收到的数据量（以字节为单位）。
packets_received	bigint	收到的数据包数量。
receive_errors	bigint	在数据接收过程中遇到的错误数量。
receive_drops	bigint	在数据接收过程中发生丢包的次数。
receive_fifo_errors	bigint	在数据接收过程中遇到FIFO（先进先出）错误的次数。
receive_frame_errors	bigint	在数据接收过程中帧错误的次数。
receive_compressed_packets	int	收到的压缩格式的包的数目。
receive_multicast_packets	int	接收的多播数据包的数量。
bytes_transmitted	bigint	数据的传输量（以字节为单位）。
packets_transmitted	bigint	被传输的包的数量。
transmit_errors	bigint	在数据传输过程中遇到的错误数量。
transmit_drops	bigint	在数据传输过程中丢包的次数。
transmit_fifo_errors	bigint	在数据传输过程中遇到的FIFO错误次数。
transmit_collision_errors	bigint	在数据传输过程中遇到冲突错误的次数。
transmit_carrier_errors	bigint	在数据传输过程中，遇到载体错误的次数。
transmit_compressed_packets	int	以压缩格式进行传输的包的数目。

log_alert_*

log_alert_* 表存储 pg_log 的错误和警告。

这里有三种 log_alert 表，所有的表都有相同的列：

- log_alert_now 是一个外部表，其数据文件存在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 log_alert_history 表之间的时段，当前的 pg_log 错误和警告存储在 log_alert_now 中。
- log_alert_tail 是一个外部表，其数据文件存在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 log_alert_now 中清除但是还没有提交到 log_alert_history 的 pg_log 错误和警告。它通常只包含了几分钟的数据。
- log_alert_history 是一个常规表，它存储数据库范围的历史错误和警告数据。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
logtime	timestamp with time zone	此日志文件的时间戳
loguser	text	查询的用户
logdatabase	text	访问的数据库
logpid	text	进程 ID
logthread	text	线程数
loghost	text	主机名或 IP 地址
logport	text	端口号
logsessiontime	timestamp with time zone	会话时间戳
logtransaction	integer	事务IDTransaction id
logsession	text	会话IDSession id
logcmdcount	text	指令统计
logsegment	text	Segment 数量
logslice	text	片数量
logdistxact	text	分布事务
loglocalxact	text	本地事务
logsubxact	text	子事务
logseverity	text	日志严重性
logstate	text	状态
logmessage	text	日志信息
logdetail	text	详细信息
loghint	text	提示信息
logquery	text	执行查询
logquerypos	text	查询位置
logcontext	text	内容信息
logdebug	text	调试
logcursorpos	text	游标位置
logfunction	text	函数信息
logfile	text	源代码文件
logline	text	源代码行
logstack	text	栈追踪

日志处理和切换

HashData 数据系统日志记录器把警告日志写到 $MASTER_DATA_DIRECTORY/gpperfmon/logs 目录中。

代理进程（gpmmon）执行下面的步骤来合并日志文件然后将它们加载到 gpperfmon 数据库：

1. 收集日志目录（除了最新的日志，它已被 syslogger 打开并且正被写入）下的所有 gpdb-alert-* 文件到单个文件 alert_log_stage 中。

2. 装载 alert_log_stage 文件到 gpperfmon 数据库中的 log_alert_history 表。
3. 清空 alert_log_stage 文件。
4. 移除所有的 gp-alert-* 文件，除了最新的那个。

syslogger 每24小时或者当当前的日志文件大小超过 1M 时切换一次警告日志。如果有单个错误消息包含一个大型 SQL 语句或者大型的栈追踪，被轮转的日志文件可能会超过 1MB。此外，syslogger 以块的方式处理错误消息，每个日志进程都会有一个单独的块。块的大小取决于操作系统；比如，在 Red Hat Enterprise Linux 上，块的大小为 4096 字节。如果许多 HashData 数据库会话同时产生错误消息，那么在它大小被检查前以及日志选择被触发前，日志文件会显著地增大。

memory_info

memory_info 视图显示在 system_history 和 segment_history 中每台主机的内存信息。这允许管理员去比较在一台 Segment 主机上总的可用内存、一台 Segment 主机上总的已经用内存以及查询处理使用的动态内存。

列名	类型	描述
ctime	timestamp(0) without time zone	这一行在segment_history表中被创建的时间。
hostname	varchar(64)	与这些系统内存度量相关的Segment或者Master主机名。
mem_total_mb	numeric	这台Segment主机的总系统内存（以MB为单位）。
mem_used_mb	numeric	这台Segment主机已用的总系统内存（以MB为单位）。
mem_actual_used_mb	numeric	这台Segment主机实际使用的系统内存（以MB为单位）。
mem_actual_free_mb	numeric	这台Segment主机实际的空闲系统内存（以MB为单位）。
swap_total_mb	numeric	这台Segment主机的总交换空间（以MB为单位）。
swap_used_mb	numeric	这台Segment主机已用的总交换空间（以MB为单位）。
dynamic_memory_used_mb	numeric	执行在这个Segment上的查询处理分配的动态内存大小（以MB为单位）。
dynamic_memory_available_mb	numeric	执行在这个Segment上的查询处理可用的额外动态内存的大小（以MB为单位）。注意：该值是在一台主机上所有Segment的可用内存的总和。尽管该值报告了可用内存，但是也有可能在一个或者更多Segment已经超过了通过参数gp_vmem_protect_limit为它们设置的内存限制。

queries_*

queries_* 表存储高级的查询状态信息。

tmid、ssid 以及 ccnt 列是唯一标识一个特定查询的组合键。

有三种查询表，所有表都有相同的列：

- queries_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到queries_history表之间的时段，当前查询状态存储在queries_now中。
- queries_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 queries_now 中清除但是还没有提交到 queries_history 的查询状态数据。它通常只包含了几分钟的数据。
- queries_history 是一个常规表，它存储数据库范围的历史查询状态数据。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
ctime	timestamp	该行被创建的时间。
tmid	int	一个特定查询的时间标识符。与一个查询关联的所有记录将有同样的tmid。
ssid	int	gp_session_id所显示的会话id。与一个查询关联的所有记录将有同样的ssid。
ccnt	int	gp_command_count所显示的当前会话的命令数。与一个查询关联的所有记录将有同样的ccnt。
username	varchar(64)	发出该查询的 HashData 角色名。
db	varchar(64)	被查询的数据库名。
cost	int	在这个版本中没有实现。
tsubmit	timestamp	查询被提交的时间。
tstart	timestamp	查询开始的时间。
tfinish	timestamp	查询结束的时间。
status	varchar(64)	查询的状态 -- start、done或者 abort。
rows_out	bigint	该查询输出的行。
cpu_elapsed	bigint	所有Segment上执行该查询的所有进程使用的CPU总量（以秒计）。它是在数据库系统中所有活动主Segment的CPU使用值的综合。注意：如果查询的执行时间短于定额的值，那么该值被记录为0。这种情况即使在查询执行时间大于min_query_time的值以及这个值低于定额的值时也会发生。
cpu_currpct	float	执行此查询的所有进程的当前CPU 平均百分比。运行在每个Segment上的所有进程的百分比都会被用于计算平均，然后所有那些值的平均值被算出来提供这个度量。在历史和尾部数据中，当前CPU的百分比平均数总为零。
skew_cpu	float	显示此查询在系统中的处理倾斜量。当一个Segment为一个查询执行不成比例的处理量时，就发生了处理/CPU倾斜。这个值是所有Segment上这一查询的CPU%度量的变异系数乘以100。例如, .95的值显示为95。
skew_rows	float	显示在系统中行倾斜的量。当一个Segment为一个查询产生不成比例的行数时，就发生了行倾斜。这个值是所有Segment上这一查询的rows_in度量的变异系数乘以100。例如, .95的值显示为95。
query_hash	bigint	在这个版本中没有实现。
query_text	text	查询的SQL文本。
query_plan	text	查询计划的文本。在这个版本中没有实现。
application_name	varchar(64)	应用名。
rsqname	varchar(64)	资源队列名。
rqppriority	varchar(64)	查询的优先级--max、high、med、low、或 min。

segment_*

segment_* 表包含 HashData 数据库 Segment 实例的内存分配统计。这会跟踪一个特定 Segment 实例上所有 postgres 进程消耗的内存量，以及根据 postgresql.conf 配置参数 gp_vmem_protect_limit 留给 Segment 的可用内存量。一个导致 Segment 超过该限制的查询将被取消从而防止系统级别的内存不足错误。

有三个 Segment 表，它们都有相同的列：

- segment_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 segment_history 表之间的时段，当前内存分配数据存储在 segment_now 中。
- segment_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 segment_now 中清除但是还没有提交到 segment_history 的内存分配数据。它通常只包含了几分钟的数据。
- segment_history 是一个常规表，它存储数据库范围的历史内存分配数据。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

一个特定的 Segment 实例通过它的 hostname 和 dbid（根据 gp_segment_configuration 的唯一 Segment 标识符）来标识。

Column	Type	Description
ctime	timestamp(0) (without time zone)	此行创建的时间
dbid	int	Segment 的 ID（dbid 来自于 gp_segment_configuration）。
hostname	varchar(64)	Segment 的主机名。
dynamic_memory_used	bigint	在该 Segment 上执行的查询处理分配的动态内存量（以字节为单位）。
dynamic_memory_available	bigint	在达到通过参数 gp_vmem_protect_limit 设置的值前该 Segment 还能请求的额外的动态内存量（以字节为单位）。

主机的聚合内存分配与利用另见视图 memory_info 和 dynamic_memory_info。

system_*

system_* 表存储了系统利用度量。有三张 system 表，它们有相同的列：

- system_now 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。在从数据收集代理收集数据和自动提交到 system_history 表之间的时段，当前系统利用度量存储在 system_now 中。
- system_tail 是一个外部表，其数据文件存储在 $MASTER_DATA_DIRECTORY/gpperfmon/data 中。这是一个过渡表，其中存放着已经从 system_now 中清除但是还没有提交到 system_history 的系统利用数据。它通常只包含了几分钟的数据。
- system_history 是一个常规表，它存储数据库范围的历史系统利用度量。它被预分区为每月的分区表。分区会根据需要进行两个月的增量添加。管理员必须要删除那些不再需要的月份的旧分区。

Column	Type	Description
ctime	timestamp	此行创建的时间
hostname	varchar(64)	与这些系统度量关联的Segment或Master的主机名。
mem_total	bigint	该主机上总的系统内存量（以字节为单位）。
mem_used	bigint	该主机上已经使用了的系统内存量（以字节为单位）。
mem_actual_used	bigint	该主机上实际使用的内存量（以字节为单位，不考虑留给cache以及缓冲区的内存）。
mem_actual_free	bigint	该主机上实际空闲的内存量（以字节为单位，不考虑留给cache以及缓冲区的内存）。
swap_total	bigint	该主机上总的交换空间（以字节为单位）。
swap_used	bigint	该主机上已经使用的交换空间（以字节为单位）。
swap_page_in	bigint	换进的页数。
swap_page_out	bigint	换出的页数。
cpu_user	float	HashData 系统用户的CPU使用。
cpu_sys	float	该主机上CPU使用。
cpu_idle	float	在度量收集期间的空闲CPU容量。
load0	float	前一分钟内的CPU负载平均值。
load1	float	前五分钟内的CPU负载平均值。
load2	float	前十五分钟内的CPU负载平均值。
quantum	int	这个度量项的度量收集间隔。
disk_ro_rate	bigint	每秒磁盘的读操作。
disk_wo_rate	bigint	每秒磁盘的写操作。
disk_rb_rate	bigint	磁盘写操作每秒的字节数。
net_rp_rate	bigint	读取操作每秒在系统网络上的包。
net_wp_rate	bigint	写入操作每秒在系统网络上的包。
net_rb_rate	bigint	读取操作每秒在系统网络上的字节数。
net_wb_rate	bigint	写入操作每秒在系统网络上的字节数。

使用资源队列进行工作负载管理

使用 HashData 数据库工作负载管理根据业务需求为查询分配优先级并分配资源，并防止资源不可用时启动查询。

本节介绍 HashData 数据库工作负载管理，并说明如何使用资源队列来管理资源。使用资源队列，可用内存和 CPU 资源可以分配给在您的 HashData 数据库系统上执行的不同类型的查询。您可以限制并发查询的数量，用于执行查询的内存量以及用于处理查询的相对 CPU 数量。

主要的资源管理问题是可以同时执行的查询的数量以及分配给每个查询的内存量。在不限制并发和内存使用的情况下，不可能保证可接受的性能。内存是最有可能限制系统处理能力的资源。因此，我们从 HashData 数据库内存使用概述开始。

HashData 数据库内存使用概述

内存是 HashData 数据库系统的关键资源，高效使用内存可确保高性能和高吞吐量。本主题介绍如何在 segment 之间分配 segment 主机内存以及管理员可用于配置内存的选项。

HashData 数据库 segment 主机运行多个 PostgreSQL 实例，全部共享主机的内存。 每个 Segment 具有相同的配置，并行处理查询时同时消耗相似数量的内存，CPU 和磁盘 IO。

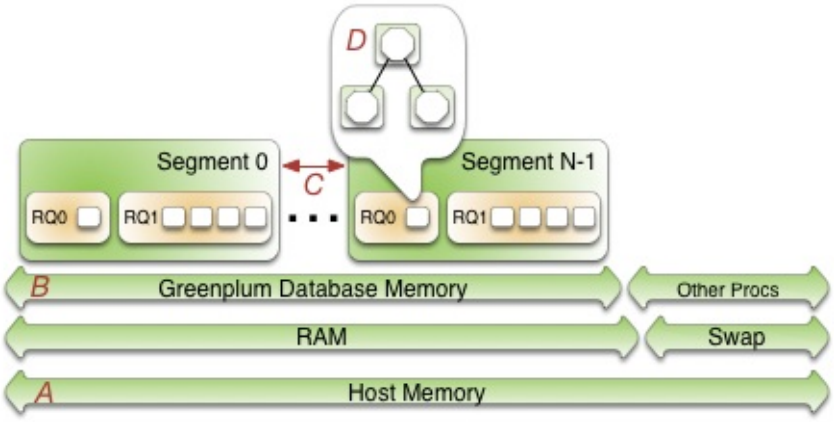
为了获得最佳查询吞吐量，应该仔细管理内存配置。HashData 数据库中的每个级别都有内存配置选项，从操作系统参数到管理具有资源队列的工作负载，以及设置分配给单个查询的内存量。

segment 主机内存

在 HashData Segment 主机上，可用主机内存在计算机上执行的所有进程之间共享，包括操作系统， HashData 数据库 Segment 实例和其他应用程序进程。管理员必须确定 HashData 数据库和非 HashData 数据库进程共享主机的内存，并配置系统高效地使用内存。监控内存使用情况同样重要，以检测 HashData 数据库或其他进程占用主机内存的任何变化。

下图说明了如何在 HashData 数据库 Segment 主机上使用内存。

图1. HashData 数据库 Segment 主机内存



从图中的底部开始，标有 A 的线表示总的主机内存。在 A 行之上的行显示了总的主机内存包括物理 RAM 和交换空间。

标记为 B 的行显示可用的总内存必须由 HashData 数据库和主机上的所有其他进程共享。非 HashData 数据库进程包括操作系统和任何其他应用程序，例如系统监视代理程序。某些应用程序可能会占用相当大一部分内存，因此可能需要调整每个 HashData 数据库主机的 Segment 数量或每个 Segment 的内存量。

Segment（C）分别获得 HashData 数据库内存（B）的相同份额。

在一个 Segment 中，资源队列控制着如何分配内存来执行 SQL 语句。资源队列允许您将业务需求转换为您的 HashData 数据库系统中的执行策略，并防范可能降低性能的查询。

非管理用户向 HashData 数据库系统提交的每个语句都与一个资源队列相关联。队列确定是否允许语句执行，并且在资源可用时允许执行。该语句可能被拒绝，立即执行，或在资源可用时排队执行。

您可以为不同类型的查询创建资源队列，并为每个队列保留一段固定的 Segment 内存。或者，您可以设置一个服务器配置参数，以指定为每个查询分配多少内存，并且不对资源队列设置最大内存限制。

查询优化器产生一个查询执行计划，由一系列称为运算符的任务（图中标为 D ）组成。操作员执行诸如表扫描或连接之类的任务，并且通常通过处理一组或多组输入行来生成中间查询结果。操作者收到资源队列分配给查询的内存份额。如果操作员不能在分配给它的内存中执行所有的工作，它会将数据缓存在磁盘上的溢出文件中。

配置 segment 主机内存的选项

主机内存是 Segment 主机上所有应用程序共享的总内存。主机内存的数量可以使用以下任何一种方法进行配置：

- 向节点添加更多的 RAM 以增加物理内存。
- 分配交换空间来增加虚拟内存的大小。
- 设置内核参数 vm.overcommit_memory 和 vm.overcommit_ratio 来配置操作系统如何处理大内存分配请求。

物理 RAM 和 OS 配置通常由平台团队和系统管理员进行管理。

要为操作系统和其他进程保留的内存量与工作负载有关。操作系统内存的最低建议是 32 GB ，但是如果 HashData 数据库中有很多并发，则可能需要增加到 64 GB 的预留内存。操作系统内存最大的用户是 SLAB，随着 HashData 数据库并发性和使用的套接字数量的增加，SLAB 也随之增加。

vm.overcommit_memory 内核参数应始终设置为 2， HashData 数据库的唯一安全值。

vm.overcommit_ratio 内核参数设置用于应用程序进程的 RAM 的百分比，其余部分为操作系统保留。红帽的默认值是 50（50%）。将此参数设置得太高可能导致操作系统保留的内存不足，从而导致 Segment 主机故障或数据库故障。将设置保留为默认值 50 通常是安全的，但保守。将该值设置得过低会减少可以同时运行的并发性和查询复杂性，这是通过减少可用于 HashData 数据库的内存量来实现的。增加 vm.overcommit_ratio 时记住总是为操作系统活动保留一些内存是非常重要的。

计算 vm.overcommit_ratio 的安全值，首先确定可用于 HashData 数据库进程的总内存 gp_vmem ，用这个公式：

```
gp_vmem = ((SWAP + RAM) – (7.5GB + 0.05 * RAM)) / 1.7
```

SWAP 是主机上的交换空间以 GB 为单位，RAM 是安装在主机上的 RAM 的数量以 GB 为单位。

使用下面的公式计算 vm.overcommit_ratio 的值：

```
vm.overcommit_ratio = (RAM - 0.026 * gp_vmem) / RAM
```

配置 HashData 数据库内存

HashData 数据库内存是所有 HashData 数据库 segments 实例可用的内存量。

设置 HashData 数据库集群时，需要确定每个主机要运行的 primary Segment 的数量以及要为每个 Segment 分配的内存量。根据 CPU 内核，物理内存量和工作负载特征，Segment 数通常是介于 4 和 8 之间的值。启用 Segment Mirror 后，设置为按照在失败期间主机上执行的 Primary Segment 的最大数量分配内存非常重要。例如，如果使用默认分组镜像配置，则 Segment 主机故障会使发生故障的主机的镜像所在的主机上的 Primary Segment 数量加倍。将每个主机的镜像分布在多个其他主机上的镜像配置可以降低最大值，允许为每个 Segment 分配更多的内存。例如，如果使用块镜像配置（每个块有 4 个主机，每个主机有 8 个 Primary Segment ），则单个主机故障将导致块中的其他主机最多有 11 个活动的 primaries，而默认的分组镜像则为 16 个。

gp_vmem_protect_limit 值是分配给每个 Segment 的总的内存。通过计算可用于所有 HashData 数据库进程的内存并除以故障期间主要分区的最大数量来估计。如果 gp_vmem_protect_limit 设置得太高，查询可能会失败。使用以下公式来计算 gp_vmem_protect_limit 的安全值。

计算 gp_vmem ，可用于所有 HashData 数据库进程的内存，使用以下公式：

```
gp_vmem =( ( SWAP + RAM ) - ( 7.5GB + 0.05 * RAM ) )/1.7
```

SWAP 是主机上的交换空间以 GB 为单位，RAM 是安装在主机上的 RAM 的数量以 GB 为单位。

使用下面的公式计算 gp_vmem_protect_limit 的值：

```
gp_vmem_protect_limit = gp_vmem / max_acting_primary_segments
```

max_acting_primary_segments 是由于主机或 Segment 故障而激活 Segment 镜像时，主机上可能运行的 Primary segments 的最大数量。

另一个重要的 HashData 数据库服务器配置参数是 statement_mem。此参数设置分配给执行查询的最大内存量。要确定此参数的值，请将每个 Segment 的内存量（ gp_vmem_protect_limit ）减去 10% 安全余量乘以您希望同时执行的最大查询数。默认的 HashData 数据库资源队列允许最多 20 个并发查询。以下是计算 statement_mem 的公式：

```
(gp_vmem_protect_limit * .9 ) / max_expected_concurrent_queries
```

资源队列允许更大程度地控制为查询分配的内存量。详细信息请参阅 配置工作负载管理。

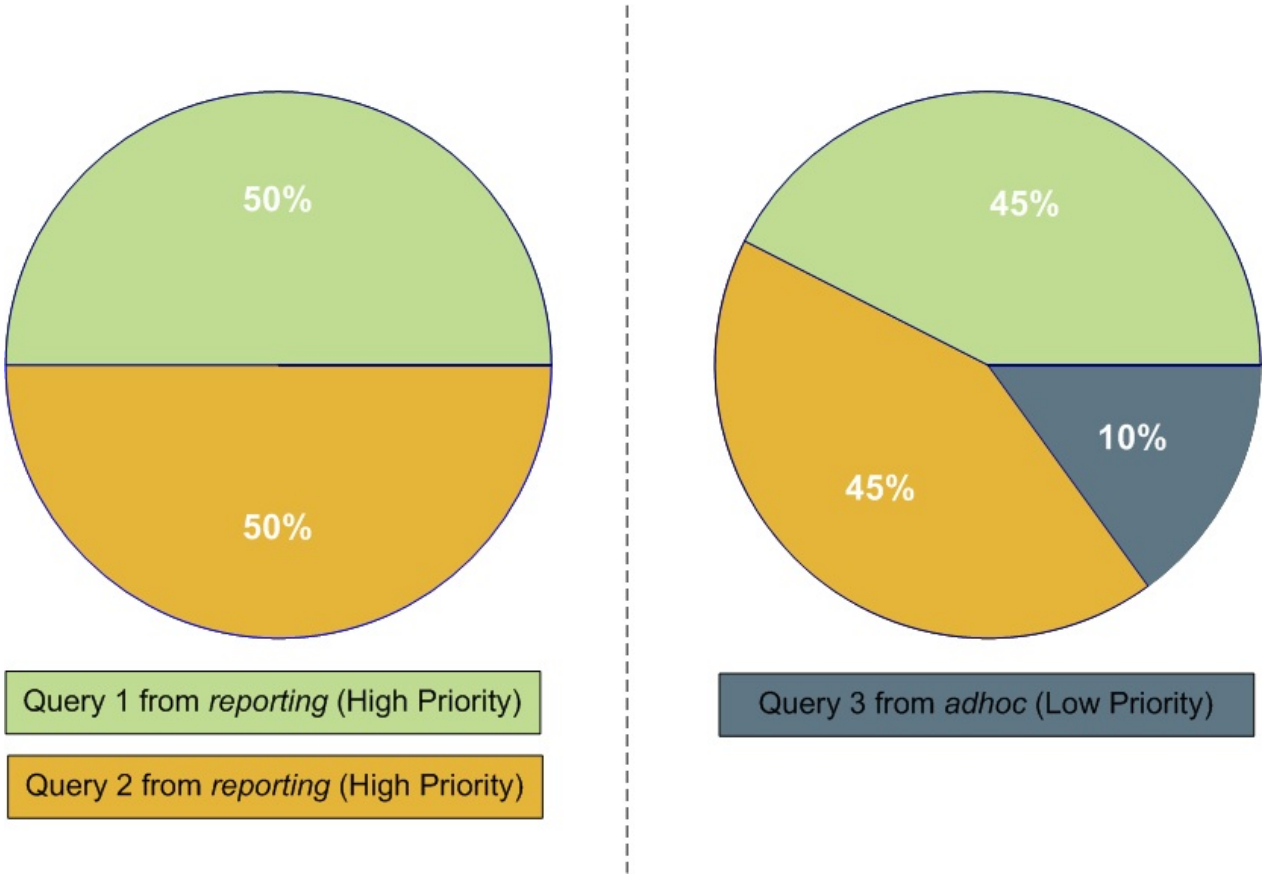
使用资源队列管理工作负载的概述

资源队列是管理 HashData 数据库系统中并发程度的主要工具。资源队列是用你 CREATE RESOURCE QUEUE SQL 语句创建的数据库对象。您可以使用它们来管理可能同时执行的活动查询的数量，每种查询分配的内存量以及查询的相对优先级。资源队列还可以防范会消耗太多资源并降低整体系统性能的查询。

每个数据库角色都与一个资源队列相关联;多个角色可以共享相同的资源队列。角色可以使用 CREATE ROLE 或 ALTER ROLE 的 RESOURCE QUEUE 表达式标注。如果未指定资源队列，则角色与默认资源队列关联， pg_default。

当用户提交执行查询时，将根据资源队列的限制来评估查询。如果查询不会导致队列超过其资源限制，那么该查询将立即运行。如果查询导致队列超出其限制（例如，如果当前正在使用活动语句插槽的最大数目），则查询必须等到队列资源空闲后才能运行。查询按照先进先出的原则进行评估。如果启用了查询优先级，则会定期评估系统上的活动工作负载，并根据查询优先级重新分配处理资源。拥有超级用户属性的用户免受资源队列限制。超级用户查询总是立即运行，而不管其分配的资源队列施加的限制。

图2.资源队列进程



资源队列定义具有相似资源需求的查询的类。管理员应该为组织中的各种类型的工作负载创建资源队列。例如，您可以为以下类查询创建资源队列，对应于不同的服务级别协议：

- ETL 查询
- 报告查询
- 执行查询

资源队列具有以下特征：

MEMORY_LIMIT：在队列（每个 Segment）中的所有查询使用的总的内存。例如，在 ETL 队列上设置 MEMORY_LIMIT 为 2GB，允许 ETL 查询在每个 segment 上面使用最大为 2GB 的内存。

ACTIVE_STATEMENTS：一个队列的插槽数量；一个队列的最大并发级别。当所有的插槽都被使用，新的查询必须等待。默认情况下每个查询使用相同大小的内存。例如，pg_default 资源队列有如下配置 ACTIVE_STATEMENTS = 20。

PRIORITY：查询的相对 CPU 使用率。可能是一下的几个级别之一：LOW, MEDIUM, HIGH, MAX。默认级别为 MEDIUM。查询优先级机制监视系统中运行的所有查询的 CPU 使用情况，并调整 CPU 的使用情况以符合其优先级。例如，你可以设置 MAX 优先级给执行资源队列， MEDIUM 优先级给其它的队列，以确保执行查询可以获取更多的 CPU 资源。

MAX_COST：查询计划成本限制。HashData 数据库优化器为每个查询分配一个数字成本。如果成本超过了为资源队列设置的 MAX_COST 值，查询会因为太昂贵被拒绝。

> 注意：使用 MEMORY_LIMIT 和 ACTIVE_STATEMENTS 为资源队列设置限制比 MAX_COST 更好。

HashData 数据库系统的默认配置有一个名为 pg_default 的单个默认资源队列。pg_default 资源队列有一个 ACTIVE_STATEMENTS 设置 20，没有 memory_limit 的，中等优先，没有设置 MAX_COST。这意味着所有的查询都被立即接受和运行，并且具有相同的优先级并且没有内存限制。但是，只有 20 个查询可以同时执行。

资源队列允许的并发查询数取决于 memory_limit 的参数设置：

- 如果资源队列没有设置 MEMORY_LIMIT，每个查询分配的内存量就是 statement_mem 这个服务器配置参数的值。资源队列可以使用的最大内存是 statement_mem 与 ACTIVE_STATEMENTS 的乘积。
- 当在资源队列上设置了 MEMORY_LIMIT ，队列可以同时执行的查询数受限于队列的可用内存。

被分配给系统的查询被分配一定数量的内存，并为其生成查询计划树。树的每个节点都是一个运算符，如排序或散列连接。每个运算符都是一个单独的执行线程，并被分配到总体语句内存的一小部分，至少为 100KB。如果计划中有大量的操作符，则操作员所需的最小内存可能会超过可用内存，并且查询将被拒绝，报内存不足的错误。操作员确定他们是否可以在分配的内存中完成任务，或者是否必须将数据泄漏到磁盘上的工作文件中。分配和控制每个操作员使用的内存量的机制称为内存配额。

并非所有通过资源队列提交的 SQL 语句都根据队列限制进行评估。默认情况下，只有 SELECT, SELECT INTO, CREATE TABLE AS SELECT, DECLARE CURSOR 语句被评估。如果服务器配置参数 resource_select_only 被设置为关闭，则 INSERT, UPDATE, DELETE 也将被评估。

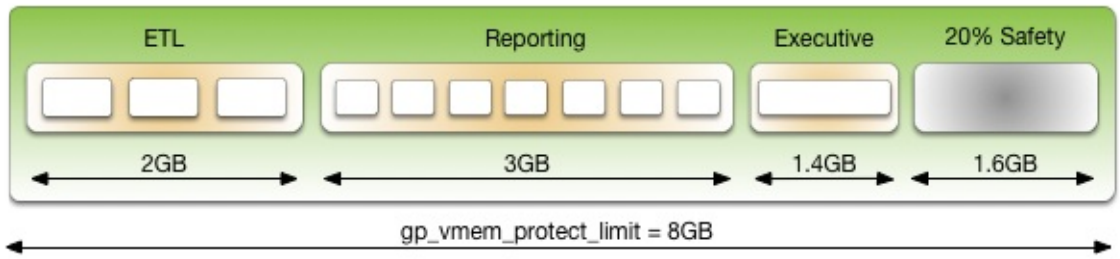
另外，一个在执行期间运行的 SQL 语句 EXPLAIN ANALYZE 命令被排除在资源队列之外。

资源队列示例

默认的资源队列，pg_default，最多允许 20 个活动查询并为每个分配相同数量的内存。对于生产系统来说，这通常不是适当的资源控制。为确保系统满足性能预期，可以定义查询类并将其分配给资源队列，这些资源队列被配置为使用最适合于该类查询的并发性，内存和 CPU 资源执行它们。

下图显示了 HashData 数据库系统的示例资源队列配置， gp_vmem_protect_limit 被设置为 8GB：

图3.资源队列配置示例



本示例具有三类具有不同特征和服务级别协议（ SLA ）的查询。为它们配置了三个资源队列。segment 存储器的一部分被保留为安全余量。

Resource Queue	Active Statements	Memory Limit	Memory per Query
ETL	3	2GB	667MB
Reporting	7	3GB	429MB
Executive	1	1.4GB	1.4GB

分配给队列的总内存是 6.4GB，或者是由内存定义的总分区内存的 80% gp_vmem_protect_limit 服务器配置参数。允许 20% 的安全余量容纳一些已知使用比由资源队列分配的更多内存的运算符和查询。

内存限制如何工作

在资源队列上设置 MEMORY_LIMIT ，即为此资源队列提交的所有活动的查询在一个 Segment 实例上可以占用的最大的内存量。分配给每个查询的内存量是队列内存限制除以活动语句限制。（将内存限制与基于活动语句的队列结合使用，而不是基于成本的队列）。例如，如果队列的内存限制为 2000MB，活动语句限制为 10，则通过队列提交的每个查询默认情况下将分配 200MB 的内存。默认的内存分配可以在每个查询上使用 statement_mem 服务器配置参数覆盖（最大为队列内存限制）。一旦查询已经开始执行，它就会将分配的内存保存在队列中，直到完成为止，即使在执行期间它实际上消耗的内存量少于分配的内存量。

你可以使用 statement_mem 服务器配置参数来覆盖当前资源队列设置的内存限制。在会话级别，您可以增加 statement_mem 直到资源队列的 memory_limit 值。这将允许单个查询使用为整个队列分配的所有内存，而不会影响其他资源队列。

statement_mem 的值的使用上限是 max_statement_mem 配置参数（超级用户参数）。对于在设置了 memory_limit 的资源队列中的查询，statement_mem 的最大值为 min(MEMORY_LIMIT，max_statement_mem) MEMORY_LIMIT，max_statement_mem 两个参数的最小值。当查询被允许时，从 memory_limit 中减去分配给它的内存。如果 memory_limit 耗尽，即使

ACTIVE_STATEMENTS 没有达到最大值，同一资源队列中的新查询也必须等待。请注意，这只能发生在 statement_mem 覆盖资源队列分配的内存。

例如，考虑一个名为 adhoc 的资源队列特别指定使用以下设置：

- memory_limit 是 1.5GB
- ACTIVE_STATEMENTS 是 3

默认情况下每个提交到队列中的语句都会被分配 500 MB 的内存。现在考虑下面一系列的时间：

1. ADHOC_1 提交了一个 Q1 查询，覆盖 STATEMENT_MEM 到 800MB。Q1 语句被允许进去系统。
2. ADHOC_2 提交了一个 Q2 查询，使用默认的 500MB。
3. Q1 和 Q2 在运行的同时，ADHOC3 提交 Q3 查询，使用默认的 500 MB。

Q1 和 Q2 查询使用了队列的 1500MB 中的 1300MB。因此 Q3 在运行之前必须等待 Q1 或 Q2 完成。

如果资源队列没有设置 MEMORY_LIMIT，直到 ACTIVE_STATEMENTS 限定的所有插槽都被使用之前查询都被允许进入系统，并且每个查询都可以被设置为任意高的 statement_mem。这可能会导致资源队列使用超量的内存。

Priorities 如何工作

该优先资源队列的设置不同于 memory_limit 和 ACTIVE_STATEMENTS 这些设置决定了一个查询是否会被允许进入队列并最终执行。该优先设置在查询生效后适用于查询。活动查询共享可用 CPU 资源，由其资源队列的优先级设置确定。当来自高优先级队列的语句进入正在运行的语句组时，它可能会声明可用 CPU 的更大份额，从而减少分配给具有较低优先级设置的队列中已经运行的语句的份额。

查询的相对大小或复杂度不会影响 CPU 的分配。如果一个简单，低成本的查询与一个庞大而复杂的查询同时运行，并且它们的优先级设置是相同的，那么它们将被分配相同的可用 CPU 资源份额。当新的查询变为活动状态时，CPU 份额将被重新计算，但同等优先级的查询将仍然具有相同数量的 CPU。

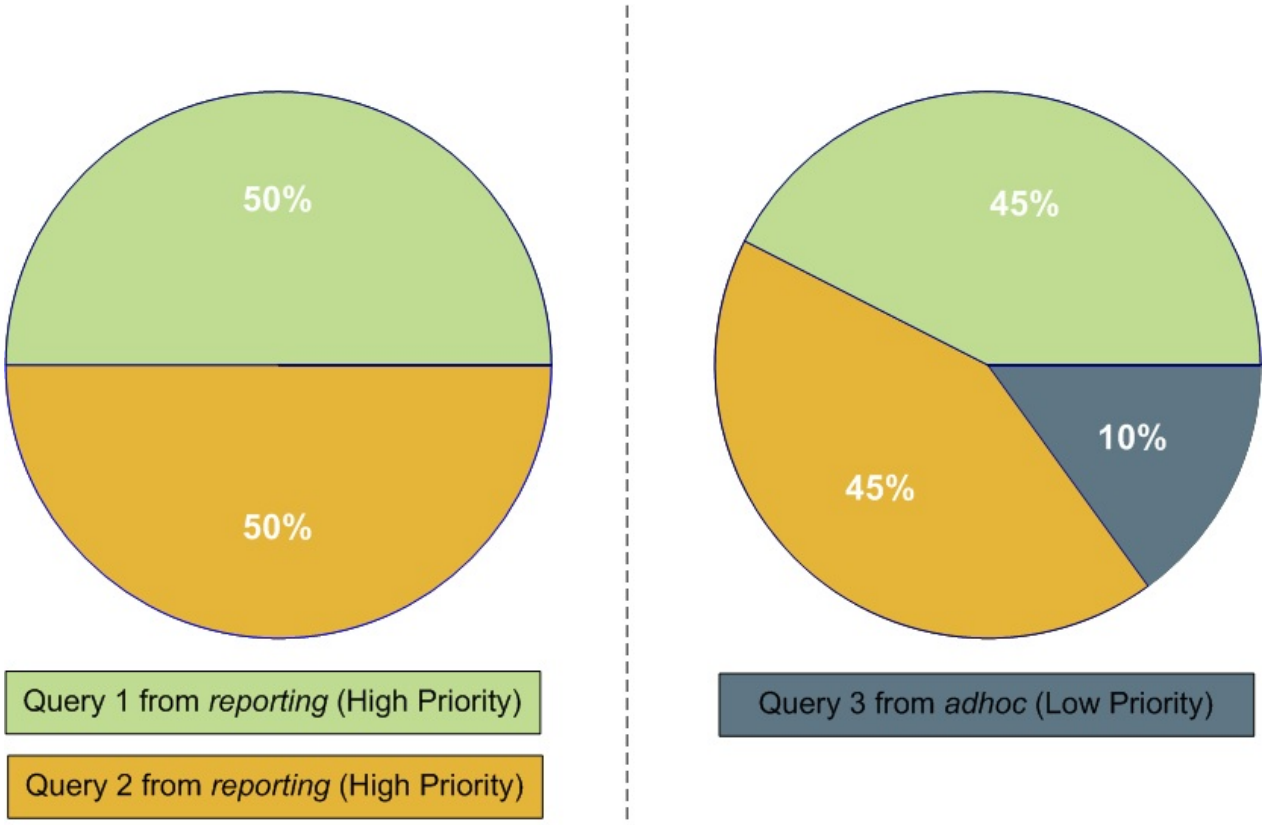
例如，管理员创建三个资源队列： adhoc 提交由业务分析正在进行的查询，reporting 定期的汇报工作， executive 对执行用户角色提交的查询。管理员希望确保定期的报告作业不受特别分析师查询的不可预测资源需求的严重影响。此外，管理员希望确保由执行角色提交的查询分配了 CPU 的重要份额。

相应地，资源队列优先级设置如下所示：

- *adhoc*
  - 低优先级
- *reporting*
  - 高度优先
- *executive*
  - 最高优先级

在运行时，活动语句的 CPU 份额由这些优先级设置确定。如果来自报告队列的查询 1 和 2 同时运行，则它们具有相同的 CPU 份额。当一个 adhoc 查询变为活动状态时，它宣称 CPU 的份额较小。报告查询使用的确切份额是经过调整的，但由于其优先级设置相同而保持不变：

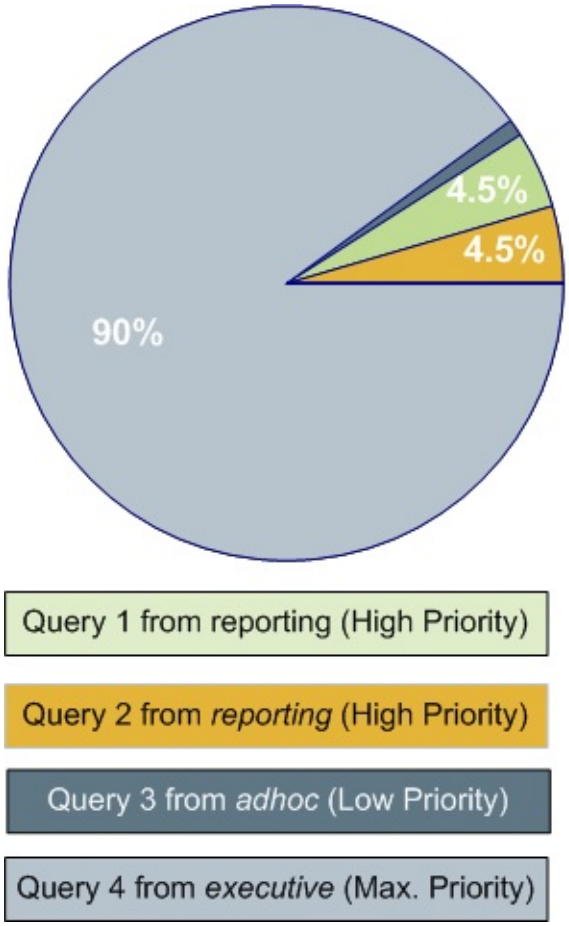
图4.根据优先级重新调整 CPU 份额



> 注意：这些插图中显示的百分比是近似值。高，低和最高优先级队列之间的 CPU 使用率并不总是精确地按照这些比例进行计算。

当执行查询进入运行语句组时，会调整 CPU 使用情况以考虑其最大优先级设置。与分析和报告查询相比，这可能是一个简单的查询，但是在完成之前，它将占用最大的 CPU 份额。

图5.为最大优先级查询重新调整的 CPU 份额



启用工作负载管理的步骤

在 HashData 数据库中启用和使用工作负载管理涉及以下高级任务：

1. 配置工作负载管理。
2. 创建资源队列并对其设置限制。
3. 为一个或多个用户角色分配一个队列。
4. 使用工作负载管理系统视图来监视和管理资源队列。

配置工作负载管理

安装 HashData 数据库时默认启用资源调度，并且对于所有角色都是必需的。默认的资源队列，pg_default，具有 20 的有效语句限制，没有内存限制和中等优先级设置。为各种类型的工作负载创建资源队列。

配置工作负载管理

1. 以下参数用于资源队列的一般配置：
    - max_resource_queues
        - 设置资源队列的最大数量。
    - max_resource_portals_per_transaction
        - 设置每个事务允许同时打开的游标的最大数量。请注意，打开的游标将在资源队列中保存一个活动的查询槽。
    - resource_select_only
        - 如果设置为 on，则 SELECT, SELECT INTO, CREATE TABLE AS SELECT, DECLARE CURSOR 命令会被评估。如果设置为 off ，则 INSERT, UPDATE, DELETE 命令也会被评估。
    - resource_cleanup_gangs_on_wait
        - 在资源队列中占用一个插槽之前清除空闲的 Segment 工作进程。

- stats_queue_level
  - 启用对资源队列使用情况的统计信息收集，然后通过查询 pg_stat_resqueues 系统视图来查看。

2. 以下参数与内存利用率有关：

   - gp_resqueue_memory_policy
     - 启用 HashData 数据库内存管理功能。在 HashData 中，分配算法 eager_free 利用了并非所有操作员同时执行的事实。查询计划分为多个阶段， HashData 数据库在该阶段执行结束时热切地释放分配给前一阶段的内存，然后将急切释放的内存分配给新阶段。当设置为 auto 时，查询内存的使用被 statement_mem 和资源队列的内存限制控制。
   - statement_mem 和 max_statement_mem
     - 用于在运行时为特定查询分配内存（覆盖由资源队列分配的默认分配）。max_statement_mem 由数据库超级用户设置，以防止常规数据库用户过度分配。
   - gp_vmem_protect_limit
     - 设置所有查询进程可以使用的上限，不应超过 Segment 主机的物理内存量。当一个 Segment 主机在查询执行过程中达到此限制时，导致限制超出的查询将被取消。
   - gp_vmem_idle_resource_timeout 和 gp_vmem_protect_segworker_cache_limit
     - 用于释放空闲数据库进程所在的 Segment 主机上的内存。管理员可能希望在具有大量并发性的系统上调整这些设置。
   - shared_buffers
     - 设置 HashData 服务器实例用于共享内存缓冲区的内存量。该设置必须至少为 128 kb，至少为 MAX_CONNECTIONS 的 16kb 倍。该值不得超过操作系统共享内存最大分配请求大小。

3. 以下参数与查询优先级有关。
   请注意，以下参数都是本地参数，这意味着它们必须设置在 master 和所有 segments 的 postgresql.conf 文件中：

   - gp_resqueue_priority
     - 默认情况下启用查询优先级排序功能。
   - gp_resqueue_priority_sweeper_interval
     - 设置所有活动语句的 CPU 使用率重新计算的时间间隔。对于典型的数据库操作，此参数的缺省值应该足够了。
   - gp_resqueue_priority_cpucores_per_segment
     - 指定每个 Segment 实例分配的 CPU 内核数量。master 和 segment 的默认值是 4。 每个主机检查自己的 postgresql.conf 中该参数的值。应该将其设置为反映 CPU 核心比率较高的值。例如，在每个主机有 10 个 CPU 内核，每个主机有 4 个 segments 的集群上，您可以指定 master 和 Standby master 上面 gp_resqueue_priority_cpucores_per_segment 的值为 10。通常，只有 master 实例在 Master 主机上。segment 主机上面参数值为 2.5。如果参数值设置不正确，则 CPU 可能没有被充分利用，或者查询优先级可能无法正常工作。例如，如果 HashData 数据库集群在您的 Segment 主机上每个 CPU 核心的 Segment 实例少于一个，请确保相应地调整此值。实际的 CPU 内核利用率基于 HashData 数据库并行查询的能力和执行查询所需的资源。

     > 注意：任何可用于操作系统的 CPU 内核都包含在 CPU 内核数量中。例如，虚拟 CPU 核心包含在 CPU 核心的数量中。

4. 如果您希望查看或更改任何工作负载管理参数值，则可以使用 gpconfig 实用程序。

5. 例如，要查看特定参数的设置：

   ```
   $ gpconfig --show gp_vmem_protect_limit
   ```

6. 例如，要在所有的 segments 实例上设置一个值，并在 master 设备上设置一个不同的值：

   ```
   $ gpconfig -c gp_resqueue_priority_cpucores_per_segment -v 2 -m 8
   ```

7. 重新启动 HashData 数据库以使配置更改生效：

   ```
   $ gpstop -r
   ```

创建资源队列

创建资源队列包含给它一个名字，设置一个活动的查询限制，并且可选地在资源队列上设置一个查询优先级。使用 CREATE RESOURCE QUEUE 命令来创建新的资源队列。

创建具有活动查询限制的队列

资源队列与一个 ACTIVE_STATEMENTS 设置限制可以由分配给该队列的角色执行的查询的数量。例如，要创建一个名为 adhoc 的资源队列，其活动查询限制为 3：

```
=# CREATE RESOURCE QUEUE adhoc WITH (ACTIVE_STATEMENTS=3);
```

这意味着，对于分配给 adhoc 资源队列的所有角色，在任何给定的时间，系统上只能运行三个活动查询。如果此队列有三个查询正在运行，并且第四个查询由该队列中的某个角色提交，那么该查询必须等待，直到某个插槽可以运行。

创建具有内存限制的队列

资源队列与一个 memory_limit 的设置控制通过队列提交的所有查询的内存量。总内存不应该超过每个 Segment 可用的物理内存。设置 memory_limit 到每个 Segment 可用内存的 90%。例如，如果主机具有 48GB 的物理内存和 6 个 Segments 实例，则每个 Segment 实例的可用内存为 8GB 。你可以计算推荐的 memory_limit 的对于单个队列为 0.90 * 8 = 7.2 GB。如果系统上创建了多个队列，则其总内存限制也必须合计为 7.2 GB。

与 ACTIVE_STATEMENTS 一起使用时，每个查询分配的默认内存量是： memory_limit/ACTIVE_STATEMENTS 。与 MAX_COST 一起使用时，每个查询分配的默认内存量是：MEMORY_LIMIT *（query_cost/MAX_COST）。使用 memory_limit 的和 ACTIVE_STATEMENTS 结合，而不是与 MAX_COST。

例如，要创建一个具有 10 的活动查询限制和 2000MB 总内存限制的资源队列（每个查询将在执行时分配 200MB 的 Segment 主机内存）：

```
=# CREATE RESOURCE QUEUE myqueue WITH (ACTIVE_STATEMENTS=20, MEMORY_LIMIT='2000MB');
```

默认的内存分配可以在每个查询上使用 statement_mem 服务器配置参数覆盖，前提是 memory_limit 或 max_statement_mem 没有超过。例如，要为特定的查询分配更多的内存：

```
=> SET statement_mem='2GB';
=> SELECT * FROM my_big_table WHERE column='value' ORDER BY id;
=> RESET statement_mem;
```

作为一般指导，对于所有资源队列的 MEMORY_LIMIT 的总值，不应超过 Segment 主机的物理内存量。如果工作负载在多个队列上错开，那么超额订阅内存分配可能是可以的。

设置优先级

为了控制资源队列对可用 CPU 资源的消耗，管理员可以分配适当的优先级。当高并发性导致 CPU 资源争用时，与高优先级资源队列相关联的查询和语句将声明比低优先级查询和语句更大的可用 CPU 份额。

优先级设置是使用 CREATE RESOURCE QUEUE WITH 和 ALTER RESOURCE QUEUE WITH 命令的 WITH 参数创建和修改的。例如，要为 adhoc 和 reporting 队列指定优先级设置，管理员将使用以下命令：

```
=# ALTER RESOURCE QUEUE adhoc WITH (PRIORITY=LOW);
=# ALTER RESOURCE QUEUE reporting WITH (PRIORITY=HIGH);
```

要创建具有最高优先级的 executive 队列，管理员将使用以下命令：

```
=# CREATE RESOURCE QUEUE executive WITH (ACTIVE_STATEMENTS=3, PRIORITY=MAX);
```

当启用查询优先级功能时，如果没有明确分配则资源队列被赋予一个 MEDIUM 优先级。

> 重要提示：为了在活动的查询工作负载上强制执行资源队列优先级，您必须通过设置关联的服务器配置参数来启用查询优先级排序功能。

将角色（用户）分配给资源队列

一旦创建资源队列，您必须将角色（用户）分配到其适当的资源队列。如果角色没有明确分配给资源队列，他们将转到默认资源队列， pg_default 。默认资源队列具有 20 的活动语句限制，没有成本限制和中等优先级设置。

使用 ALTER ROLE 或者 CREATE ROLE 命令将角色分配给资源队列。例如：

```
=# ALTER ROLE name RESOURCE QUEUE queue_name;
=# CREATE ROLE name WITH LOGIN RESOURCE QUEUE queue_name;
```

角色只能在任何给定时间分配给一个资源队列，因此您可以使用 ALTER ROLE 命令来初始分配或更改角色的资源队列。

资源队列必须按用户分配。如果您有角色层次结构（例如，组级角色），则将资源队列分配给该组不会传递给该组中的用户。

超级用户总是免除资源队列限制。超级用户查询将始终运行，而不管在其分配的队列上设置的限制。

从资源队列中删除角色

所有用户必须分配给一个资源队列。如果未明确分配给特定队列，则用户将进入默认资源队列， pg_default 。如果您希望从资源队列中删除角色并将其放入默认队列，请将角色的队列分配更改为 none。例如：

```
=# ALTER ROLE role_name RESOURCE QUEUE none;
```

修改资源队列

资源队列创建完成后，您可以使用 ALTER RESOURCE QUEUE 命令更改或者重置队列限制。您可以使用 DROP RESOURCE QUEUE 命令删除资源队列。要更改分配给资源队列的角色（用户），请将角色（用户）分配给资源队列。

改变资源队列

ALTER RESOURCE QUEUE 命令更改资源队列的限制。要更改资源队列的限制，请为队列指定新值。例如：

```
=# ALTER RESOURCE QUEUE adhoc WITH (ACTIVE_STATEMENTS=5);
=# ALTER RESOURCE QUEUE exec WITH (PRIORITY=MAX);
```

要将活动语句或内存限制重置为无限制，请输入值 -1。要将最大查询成本重置为无限制，请输入值 -1.0。例如：

```
=# ALTER RESOURCE QUEUE adhoc WITH (MAX_COST=-1.0, MEMORY_LIMIT='2GB');
```

你可以使用 ALTER RESOURCE QUEUE 命令来更改与资源队列关联的查询的优先级。例如，要将队列设置为最低优先级：

```
ALTER RESOURCE QUEUE webuser WITH (PRIORITY=MIN);
```

删除资源队列

DROP RESOURCE QUEUE 命令删除资源队列。要删除一个资源队列，队列不能分配任何角色，也不能在队列中存在任何等待语句。删除资源队列：

```
=# DROP RESOURCE QUEUE name;
```

检查资源队列状态

检查资源队列状态涉及以下任务：

- 查看排队的语句和资源队列状态
- 查看资源队列统计信息
- 查看分配给资源队列的角色
- 查看资源队列的等待查询
- 从资源队列中清除等待语句
- 查看主动语句的优先级
- 重置活动语句的优先级

查看排队的语句和资源队列状态

gp_toolkit.gp_resqueue_status 视图允许管理员查看工作负载管理资源队列的状态和活动。它显示了有多少查询正在等待运行，以及在特定的资源队列中当前有多少查询在系统中处于活动状态。要查看在系统中创建的资源队列，其限制属性及其当前状态：

```
=# SELECT * FROM gp_toolkit.gp_resqueue_status;
```

查看资源队列统计信息

如果要跟踪资源队列的统计信息和性能，可以启用资源队列的统计信息收集。这是通过设置在 master 服务器中的 postgresql.conf 文件中的以下服务器配置参数完成的：

```
stats_queue_level = on
```

一旦启用，您可以使用 pg_stat_resqueues 系统视图查看关于资源队列使用情况的统计信息。请注意，启用此功能会造成轻微的性能开销，因为必须跟踪通过资源队列提交的每个查询。启用资源队列上的统计信息以进行初始诊断和管理规划可能会很有用，然后禁用该功能以继续使用。

查看分配给资源队列的角色

要查看分配给资源队列的角色，请在 pg_roles 和 gp_toolkit.gp_resqueue_status 系统目录表上执行以下查询：

```
=# SELECT rolname, rsqname FROM pg_roles,
          gp_toolkit.gp_resqueue_status
   WHERE pg_roles.rolresqueue=gp_toolkit.gp_resqueue_status.queueid;
```

您可能希望创建此查询的视图以简化将来的查询。例如：

```
=# CREATE VIEW role2queue AS
   SELECT rolname, rsqname FROM pg_roles, pg_resqueue
   WHERE pg_roles.rolresqueue=gp_toolkit.gp_resqueue_status.queueid;
```

那么你可以查询视图：

```
=# SELECT * FROM role2queue;
```

查看资源队列的等待查询

当一个插槽用于资源队列时，它被记录在 pg_locks 系统目录表中。这是您可以查看所有资源队列的所有当前活动和等待查询的位置。要检查正在队列中的语句（即使是没有等待的语句），也可以使用 gp_toolkit.gp_locks_on_resqueue 视图。例如：

```
=# SELECT * FROM gp_toolkit.gp_locks_on_resqueue WHERE lorwaiting='true';
```

如果这个查询没有返回任何结果，那就意味着在资源队列中没有等待的语句。

从资源队列中清除等待语句

在某些情况下，您可能需要清除资源队列中的等待语句。例如，您可能希望删除正在队列中等待但尚未执行的查询。如果执行时间太长，或者在事务中处于空闲状态并占用其他用户所需的资源队列时，您可能还想停止已经启动的查询。要做到这一点，您必须首先确定要清除的语句，确定其进程 ID（pid），然后使用 pg_cancel_backend 和进程 ID 结束进程，如下所示。

例如，要查看有关当前在所有资源队列中处于活动状态或正在等待状态的所有语句的处理信息，请运行以下查询：

```
=# SELECT rolname, rsqname, pid, granted,
          current_query, datname
   FROM pg_roles, gp_toolkit.gp_resqueue_status, pg_locks,
        pg_stat_activity
   WHERE pg_roles.rolresqueue=pg_locks.objid
   AND pg_locks.objid=gp_toolkit.gp_resqueue_status.queueid
   AND pg_stat_activity.procpid=pg_locks.pid
   AND pg_stat_activity.usename=pg_roles.rolname;
```

如果这个查询没有返回任何结果，那就意味着资源队列中没有语句。其中包含两条语句的资源队列示例如下所示：

```
rolname | rsqname |  pid  | granted |     current_query      | datname
-------------------------------------------------------------------------
  sammy | webuser | 31861 | t       | <IDLE> in transaction  | namesdb
  daria | webuser | 31905 | f       | SELECT * FROM topten;  | namesdb
```

使用此输出来标识要从资源队列中清除的语句的进程标识（pid）。要清除语句，您需要打开一个终端窗口（使用数据库超级用户 gpadmin 或者系统用户 root），并取消相应的进程。例如：

```
=# pg_cancel_backend(31905)
```

> 注意：请勿使用 KILL 命令。

查看活动语句的优先级

gp_toolkit 行政模式有一个叫做 gp_resq_priority_statement 的视图，其中列出了目前正在执行的所有语句，并提供优先级，会话 ID 和其他信息。

重置活动语句的优先级

超级用户可以使用内置函数 gp_adjust_priority(session_id, statement_count, priority) 调整当前正在执行的语句的优先级。使用此函数，超级用户可以提高或降低任何查询的优先级。例如：

```
=# SELECT gp_adjust_priority(752, 24905, 'HIGH')
```

要获取此函数所需的会话 ID 和语句计数参数，超级用户可以使用 gp_toolkit 管理模式视图， gp_resq_priority_statement 。从视图中，使用这些值作为函数参数。

- rqpsession 列的值为 session_id 的值。
- rqpcommand 列的值为 statement_count 的值。
- rqppriority 列的值为当前的优先级。你可以指定一个字符串的值 MAX, HIGH, MEDIUM,或 LOW 作为优先级。

> 注：gp_adjust_priority() 函数只影响指定的语句。后面的语句在同一个资源队列中使用队列的通常分配的优先级来执行。