GBase 8s 运维手册

南 大 通 用 数 据 技 术 股 份 有 限 公 司

General Data Technologies Co., Ltd.

天津总公司：天津市高新区开华道 22 号普天创新产业园东塔 20-23 层

电　　话：022-58815678

传　　真：022-58815679

北京分公司：北京市朝阳区太阳宫中路 12 号太阳宫大厦 10 层 1008 室

电　　话：010-88866866

传　　真：010-88864556

http://www.gbase.cn

E-mail:info@gbase.cn

GBase 8s 运维手册，南大通用数据技术股份有限公司

通讯方式

南大通用数据技术股份有限公司

中国天津华苑产业区海泰发展六道 6 号海泰绿色产业基地 J 座 5 层 (300384)

电话：400-013-9696　　邮箱：info@gbase.cn

目 录

1. 说明

本文档为南大通用 GBase 8s 数据库运维手册。

2. 基本概念

2.1. 进程（Processes）

数据库服务器的所有进程名为 oninit，一个数据库实例根据配置的不同会有多个 oninit 进程。

2.2. 共享内存（share memory）

数据库共享内存由常驻段、缓冲池、虚拟段和消息段四部分组成。

2.2.1. 常驻段（resident segment）

共享内存常驻段包括以下部分：

共享内存头、逻辑日志缓冲区、物理日志缓冲区、高可用数据复制缓冲区、锁表

2.2.2. 缓冲池（bufferpool）

缓冲池是共享内存段中的最大组成部分，用于缓存磁盘数据，降低 I/O 开销，提高数据库运行效率。缓冲池的大小由数据库配置参数 BUFFERPOOL 指定。

2.2.3. 虚拟段（virtual segment）

数据库所有的会话都有一个或多个内存池。当数据库服务器需要内存时，它会先查看指定的池。如果池中可用的内存不足以满足请求，那么数据库服务器将从系统池添加内存。如果数据库服务器不能在系统池中找到足够的内存，那么它会动态地给虚拟部分分配更多的段。

共享内存的虚拟部分存储以下数据：

内部表、大缓冲区、会话数据、线程数据（堆栈和堆）、数据分布高速缓存、字典高速缓存、SPL 例程高速缓存、SQL 字典高速缓存、排序池、全局池。

2.2.4. 消息段（message segment）

共享内存消息段用于使用 IPC 共享内存连接的应用与数据库之间的通信。

共享内存的通信部分的大小约等于 12 千字节乘以共享内存通信所需的期望连接数（nettype ipcshm）。如果 nettype ipcshm 不存在，那么期望的连接数将缺省为 50。

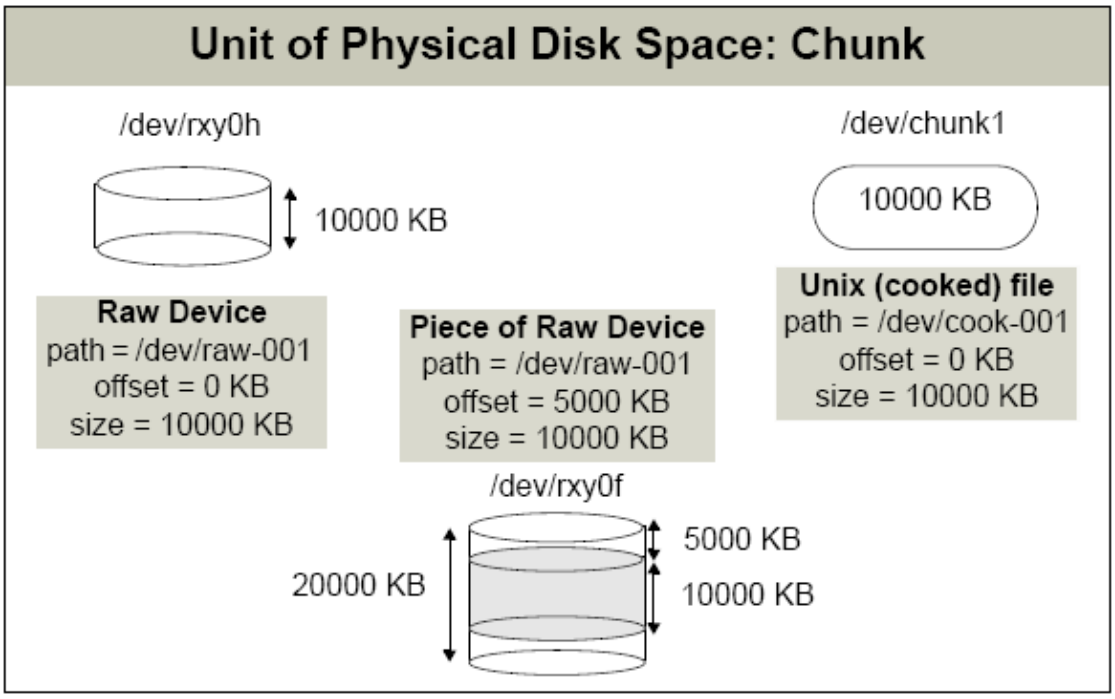
2.3. 磁盘(disk)

2.3.1. 块（chunk）

块是用于数据库服务器数据存储数据文件。可以是文件系统或裸设备。单个块的最大大小是 4TB。数据库最多可支持块数为 32,766 个。

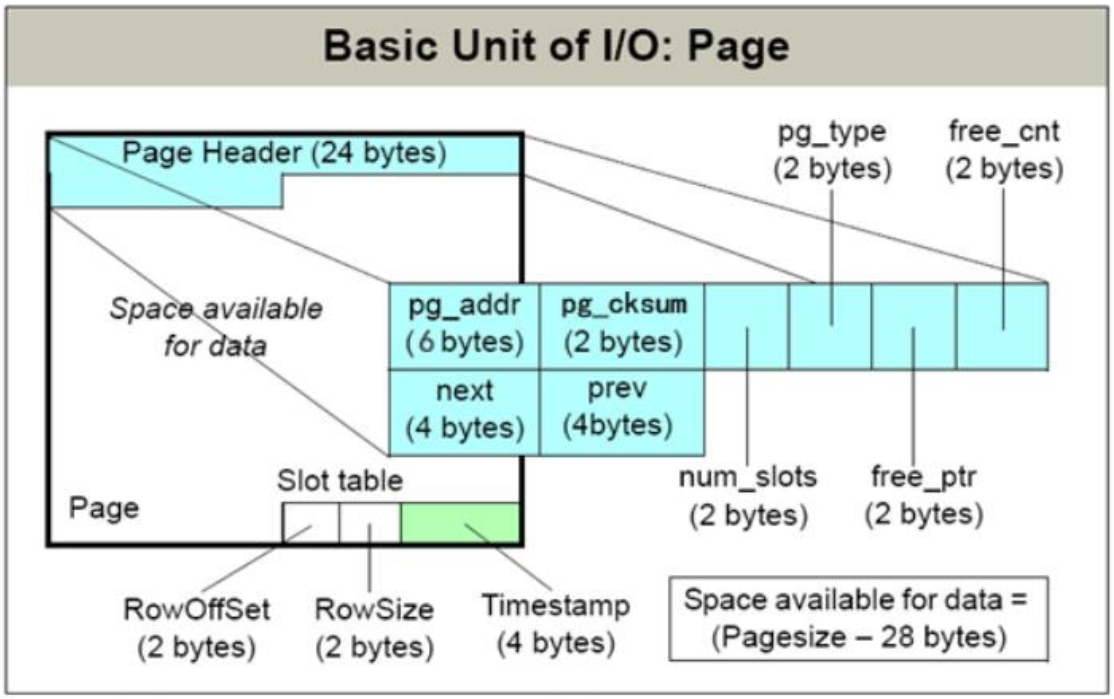
在 UNIX 或 LINUX 上，建议使用裸设备，在 Windows 上，建议使用 NTFS 文件系统。

下图对数据库块进行了图解说明。



2.3.2. 页（page）

页是数据库服务器读取和写入的最小单元。是磁盘 IO 的基本单位， 在 AIX、Windows 平台上，页大小默认为 4k，其他操作系统默认为 2k，可在创建数据库空间时指定页大小。下图图解说明了页的概念及内部结构。



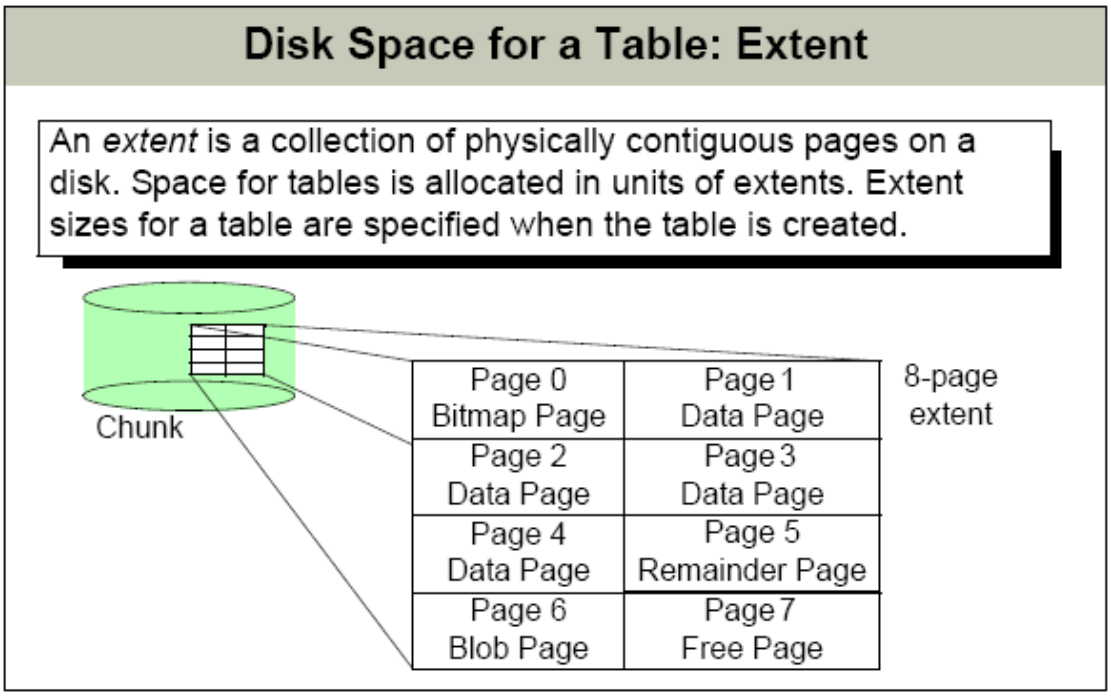
页结构由页头信息，数据部分和时间戳组成，数据部分每一行对应一个 slot，每行数据实际占用的页空间大小为行长加 4 字节，数据行从前往后写，slot 从后往前写。每页最多存储 255 行（由 num_solts 和 rowid 限定）

2.3.3. 智能大对象页（SBspace page）

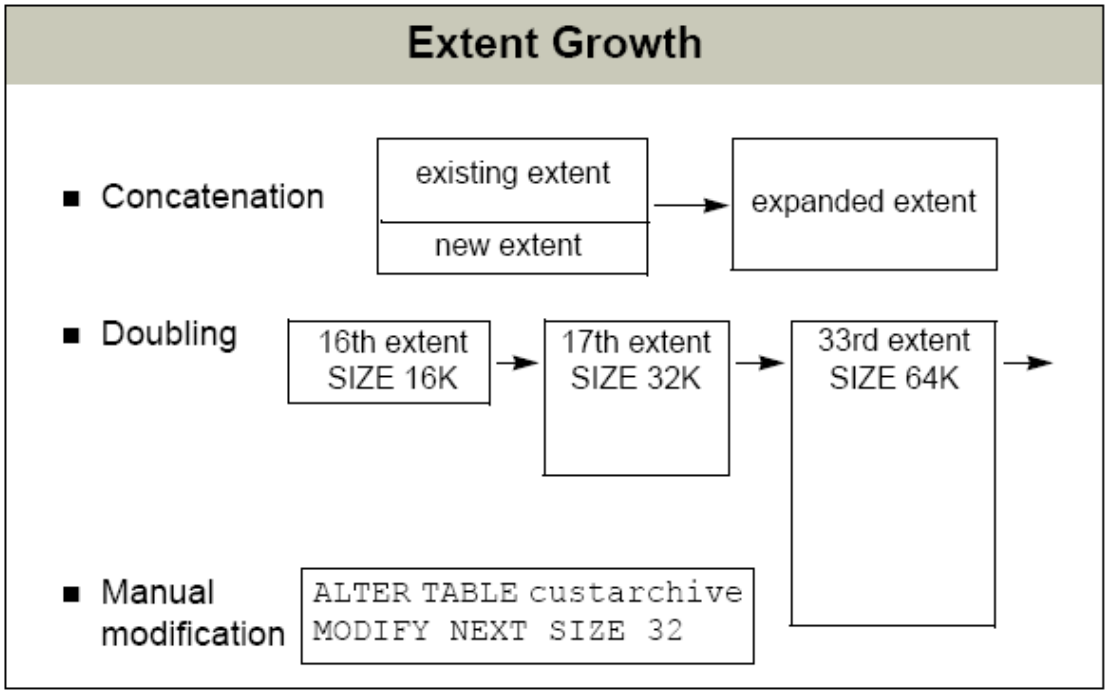
智能大对象页上数据库服务器用以在智能大对象空间内存储智能大对象的页类型。与 Blob 页不同，智能大对象页上不可配置的，智能大对象页的大小与数据库服务器页的相同。

2.3.4. 扩展数据库块（extent）

当表创建时，数据库服务器会分配固定数量的空间以存储表中的数据。当此空间填满时，数据库服务器必须分配额外的存储空间。数据库服务器用来分配初始和后续存储空间的物理存储单元称为扩展数据块。



chunk 中多个连续的物理页组成一个 extent，每个表由多个 extent 组成。可在建表时指定初始 extent 大小，默认大小为 8 页。表中现有 extent 填满后，自动扩展 extent，下图说明了 extent 的扩展规则。



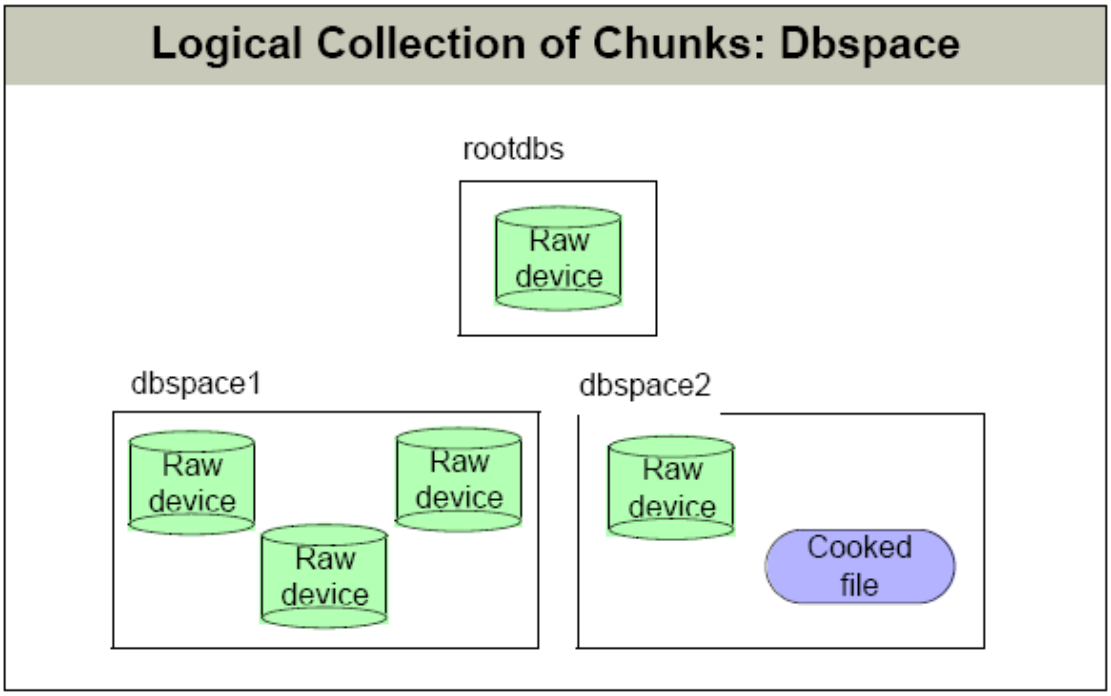
表中 extent 每扩展 16 个，下一个扩展的 extent 大小翻倍。

2.3.5. 数据库空间（dbspace）

数据库空间是一种可包含 1 到 32766 个块（chunk）的逻辑单元。放置数据库、表、逻辑日志文件以及数据库空间中的物理日志。

要指定数据库或表的存储位置，可使用 CREATE DATABASE 或 CREATE TABLE 语句的 IN *dbspace* 选项。

数据库空间可以包含一个或多个块，如下图所示。



2.3.6. 根数据库空间（root dbspace）

根数据库空间是数据库服务器创建的初始数据库空间。根数据库空间用于存储保留页和系统内部表。

根数据库空间是所有数据库默认的存储空间。

如系统未创建临时空间，根数据库空间也是默认的临时空间。

2.3.7. 保留页（reserved pages）

根数据库空间中的前 12 个数据页用于保存系统信息，称为保留页，第一页记录数据库版权信息，第二页记录数据库配置参数，从第三页开始，每两页为一对互为备份记录相关信息。每对中的两个页轮流更新来确保数据库的一致性。

page0：记录版权信息

page1：记录数据库最后一次启动时的配置参数

page2&page3：记录最后一次检查点的时间和地址，并记录每个逻辑日志的位置和当前状态

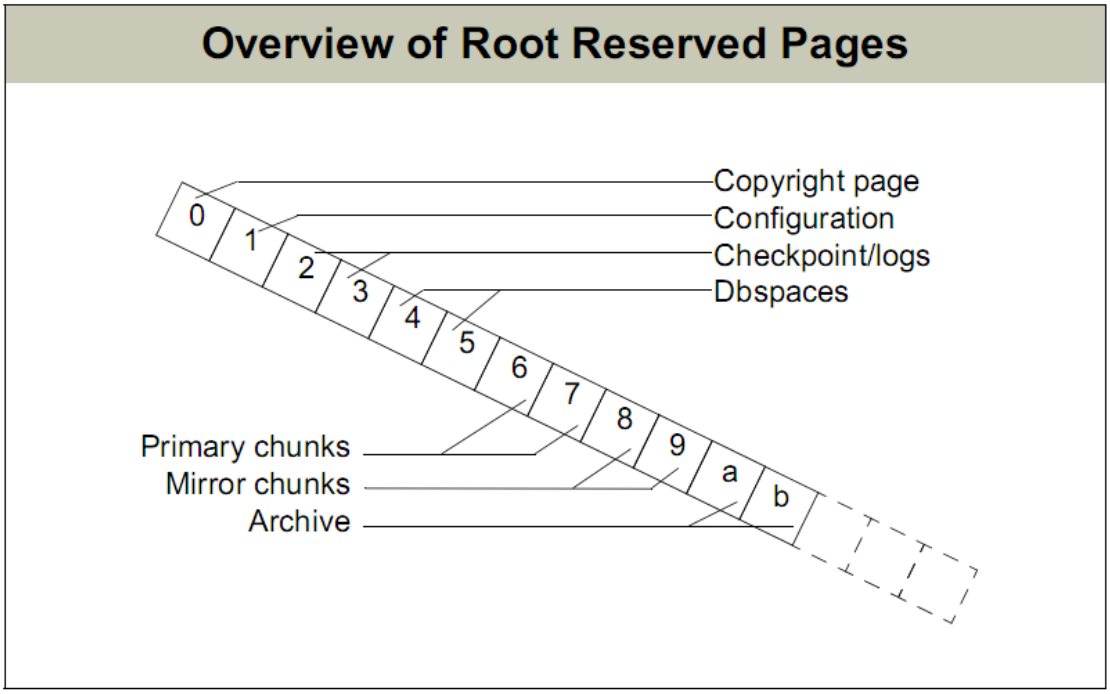
page4&page5：记录空间名称、状态、创建日期、备份情况等

page6&page7：记录数据块（chunk）信息，包括数据块状态、路径、大小、空余大小等

page8&page9：记录镜像数据库块信息

page10&page11：记录数据库实例备份信息

下图说明了根数据库空间 12 个保留页的作用，可使用 oncheck -pr 查看根数据库空间中保留页中保存的信息。



对于非根数据库空间，每个数据库空间 chunk 保留两个数据页。

2.3.8. 临时数据库空间（temp dbspace）

临时数据库空间用于存储用户会话的临时表，以及为排序、分组等计算提供临时空间。

2.3.9. 智能大对象空间（SBspace）

智能大对象空间是包含一个或多个存储智能大对象的块的逻辑存储单元。智能大对象由

CLOB（字符大对象）和 BLOB（二进制大对象）数据类型组成。用户定义的数据类型也可以使用智能大对象空间。

2.3.10. 临时智能大对象空间（temp SBspace）

临时智能大对象空间可在不用元数据日志记录和用户数据日志记录的情况下存储临时智能大对象。如果您将临时智能大对象存储在标准的智能大对象空间中，将记录元数据。临时智能大对象空间里斯临时数据库空间。

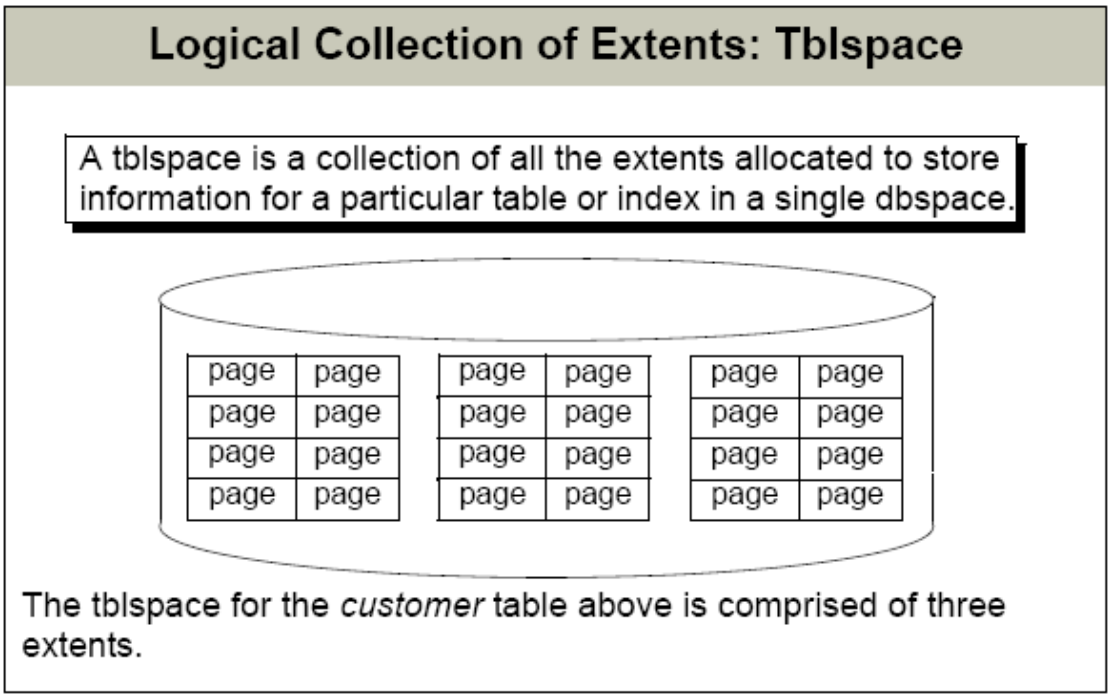
2.3.11. 数据库（database）

数据库是包含表和索引及其他数据库对象的逻辑存储单元。

数据库存储在 CREATE DATABASE 语句指定的数据库空间中。当未在 CREATE DATABASE 语句中显式地指定数据库空间时，数据库将存储在根数据库空间中。如在 CREATE DATABASE 语句中明确指定了数据库空间，此数据库空间就是该数据库中所有对象默认的存储位置。

2.3.12. 表空间（tblspace）

表空间包含分配到给定表或表分段（如果表已分段）的所有磁盘空间。是一个或多个扩展块（extent）的逻辑组合。下图描述了表空间与 extent 及 page 的关系。



2.3.13. 物理日志（physical log）

物理日志记录是在更改页数据之前存储将要更改的页的过程。在数据库服务器修改共享内存缓冲池中的某些页之前，它将页的前映像存储在共享内存中的物理日志缓冲区。

数据库服务器为这些页而将前映像页保留在共享内存的物理日志中，直至一个或多个页清除程序将页清仓到磁盘。未修改的页在数据库服务器发生故障或备份过程需要它们的情况下可用于提供数据库服务器数据的准确抽点转储。快速恢复和数据库服务器备份使用这些抽点转储。

在故障后，数据库服务器使用页的前映像将磁盘上这些页恢复至它们在最近检查点中的状态。然后数据库服务器使用逻辑日志记录使所有数据返回至最近完成的事务之时物理和逻辑上的一致状态。在快速恢复介绍中将更详细说明这一过程。

当您执行备份时，数据库服务器执行检查点并检查物理日志，以确定备份上所属的页的正确版本。在 0 级备份中，数据库服务器备份所有磁盘页。

2.3.14. 逻辑日志（logical log）

为了保留自上次存储空间备份以来的事务和数据库服务器更改的历史记录，数据库服务器生成日志记录。数据库服务器将日志记录存储在逻辑日志中，这是由三个或更多逻辑日志文件组成的循环文件。将该日志称为逻辑的事因为日志记录代表数据库服务器的逻辑操作（而不是

物理操作）。存储空间备份加上逻辑日志备份的组合在任何时候均包含数据库服务器数据的完整副本。

作为数据库服务器管理员，您必须配置并管理逻辑日志。例如：如果您不定期备份日志文件，那么逻辑日志会填满，而数据库服务器暂挂处理。

2.4. 检查点与快速恢复

2.4.1. 检查点(checkpoint)

数据库服务器会定期将缓冲池内的事务和数据清仓到磁盘。直到将事务和数据清仓到磁盘之前，数据和事务都处于流出的状态。除了在事务完成后立即强制将每个事务清仓到磁盘，数据库服务器还将事务写入到逻辑日志中。数据库服务器在事务发生时记录事务。如果系统出现故障，那么服务器执行以下操作：

- 重放日志以重做和恢复事务
- 将数据库返回至与发生故障时数据库系统的状态一致的状态

为了便于数据库系统的复原或逻辑恢复，数据库服务器生成一致性点，称为检查点。检查点是建立数据库系统的已知和一致状态时日志中的某个时间点。通常，检查点涉及到记录特定数量的信息，因此，如果发生故障，数据库服务器可在已建立的点上重新启动。

检查点的目的在与定期将逻辑日志中的重新启动点向前移动。如果检查点不存在而且发生故障，那么数据库服务器需要处理自系统重新启动以来逻辑日志中记录的所有事务。

检查点的执行过程：

1、阻止线程进入临界资源区

2、物理日志缓冲区写入磁盘

3、数据缓冲区的脏数据刷新到磁盘

4、检查点记录写入逻辑日志缓冲区

5、逻辑日志缓冲区刷新到磁盘

6、逻辑上清空物理日志

2.4.2. 快速恢复(fast recovery)

快速恢复是自动容错功能，数据库服务器每次从脱机方式转向静默、管理或联机方式时执

行该功能。无需为快速恢复采取任何管理操作；它是一种自动功能。

快速恢复过程检查数据库服务器上次脱机时是否在不受控条件下进行的。如果是，那么快速恢复将数据库服务器返回至物理和逻辑一致性状态。

如果快速恢复过程发现数据库服务器在受控方式下脱机，那么快速恢复过程终止，并且数据库服务器转向联机方式。

快速恢复在任何导致数据库服务器内存内容丢失的故障之后将数据库服务器恢复至物理和逻辑一致性。例如：操作系统发生故障，但没有任何警告。系统故障不损坏数据库，但却影响发生故障时正在进行的事务。

快速恢复按以下步骤发生：

1、数据库服务器使用物理日志中的数据将所有磁盘页返回至它们在最近检查点时的状态。该点称为物理一致性。

2、数据库服务器在逻辑日志文件中查找哦啊最新的检查点记录

3、数据库服务器前滚所有在最近检查点记录之后写入的逻辑日志记录

4、数据库服务器回滚所有未落实的事务。

2.5. 数据库日志模式

数据库服务器管理的每个数据库均有日志记录状态。日志记录状态指示数据库是否使用事务日志记录以及人（如果使用）数据库运用哪一种日志缓冲机制。数据库日志记录状态将指示下列日志记录类型中的任意一种：

- 未缓冲的事务日志记录
- 已缓冲的事务日志记录
- 符合 ANSI 的事务日志记录
- 不记录

所有逻辑日志记录在数据库服务器将其写入磁盘上的逻辑日志之前，均经过共享内存中的逻辑日志缓冲区。但是，数据库服务器清仓逻辑日志缓冲区的时刻对于已缓冲的事务日志记录和未缓冲的事务日志记录是不同的。

2.5.1. 未缓冲的事务日志记录

如果事务是对使用未缓冲日志记录的数据库进行的，那么逻辑日志缓冲区中的记录可保证在提交处理期间写入磁盘。当应用程序在 COMMIT 语句之后（对于分布式事务，在 PREPARE

语句之前）重获控制权时，逻辑日志记录在磁盘上。一旦提交缓冲区中的任何事务（即，将提交记录写入逻辑日志缓冲区），数据库服务器就立即清除记录。

当数据库服务器清仓缓冲区是，仅将以使用的页希尔磁盘。已使用的页包含哪些仅部分已满的页，但这样就浪费了一些空间。由于这个原因，相比在同一数据库服务器上的所有数据库均使用已缓冲的日志记录这种情况而言，磁盘上的逻辑日志文件会更快地填满。

未缓冲的日志记录对于大多数数据库而言是最好的选择，因为它保证所有已提交的事务可得以恢复。在发生故障的情况下，仅丢失发生故障时未提交的事务，但是，有了未缓冲的日志记录，数据库服务器会更频繁地将逻辑日志缓冲区清仓到磁盘，而缓冲区将包含更多部分已满的页，因此它比已缓冲的日志记录会更快地填充逻辑日志。

2.5.2. 已缓冲的事务日志记录

如果事务是对使用已缓冲日志记录的数据库进行的，那么记录尽可能久地保留（已缓冲的）在逻辑日志缓冲区中。直至发生以下情况之一，这些记录才会从共享内存中的逻辑日志缓冲区刷新到磁盘上的逻辑日志：

- 缓冲区已满
- 具有未缓冲日志记录的数据库上的提交清仓了缓冲区
- 出现了检查点
- 连接关闭

如果您使用已缓冲的日志记录并且发生了故障，那么不能期望数据库服务器恢复那些在发生故障时位于逻辑日志缓冲区中的事务。因此，会丢失一些已提交的事务。作为对该风险的补偿，变更期间的性能会稍有提高。只要您在发生故障的情况下可重新创建更新，已缓冲的日志记录对于频繁更新的数据库就是最佳的。您可调整逻辑日志缓冲区的大小以便为您的系统在性能和因系统故障而丢失事务的风险之间找到可接受的平衡。

2.5.3. 符合 ANSI 的事务日志记录

符合 ANSI 标准的日志模式，采用未缓冲的日志记录方式，使用 ANSI 日志方式记录的数据库，不能更改日志记录方式为缓冲日志记录或无日志记录。符合 ANSI 的数据库中发出的所有 SQL 语句都将自动包含在事务中，不能使用 BEGIN WORK 和 COMMIT WORK 来定义事务，所有符合 ANSI 标准日志模式的数据库缺省隔离级别为“可重复读”。支持事务日志记录的不符合 ANSI 数据库的缺省隔离级别是“已落实读”,不使用事务日志记录的不符合 ANSI 数据库

的缺省隔离级别是”未落实读“。

2.5.4. 不进行数据库日志记录

如果您关闭了数据库的日志记录，那么将不记录事务，但会记录其他操作。关闭数据库日志记录，能提供最好的数据库性能，但在数据库出现故障时无法提供全面的数据恢复。

3. 数据库实例监控

3.1. 使用 onstat 命令监控

3.1.1. 概述

onstat 命令读取共享内存结构，并提供有关数据库服务器在该命令执行时的统计信息。onstat 命令不在共享内存上放置任何锁，因此运行该命令不会影响数据库性能。

onstat 各参数的使用及功能：

onstat 选项	功能
-	输出打印头
--	显示所有 onstat 选项及其功能的列表
-a	解释为 onstat -cuskbtdlp. 以该顺序显示输出。
-b	显示有关当前正在使用的缓冲区的信息，包括缓冲池中常驻页的数量
-B	获得有关所有数据库服务器缓冲区的信息
-c	显示 ONCONFIG 文件
-C	打印 B 树扫描程序信息
-d	显示每个存储空间中的块信息
-D	显示每个数据库空间中前 50 个块页读取和写入信息
-f	列出当前受 DATASKIP 功能影响的数据库空间
-F	显示将页清仓到磁盘上的每种类型的写操作的计数
-g *option*	提供监视选项
-G	打印全局事务标识
-h	提供有关缓冲区头散列链的信息
-i	使 onstat 实用程序成为交互方式
-j	打印活动的 onpload 进程的交互式状态
-k	显示关于活动锁信息
-l	显示有关物理和逻辑日志信息，包括页地址
-m	显示数据库服务器消息日志中的最新 20 行
-o	将共享内存段副本保存至 outfile 文件中
-O	显示关于 Optical Subsystem 内存高速缓存和登台区域 Blob 空间的信息
-p	显示该要文件计数
-P	显示所有分区的分区号和属于该分区的缓冲池页的拆离
-pu	显示概要文件计数及用户活动的该要文件
-r *seconds*	在每次执行之间等待指定的 *seconds* 秒后重复 onstat 选项
-R	显示关于 LRU 队列、FLRU 队列和 MLRU 队列的详细信息
-s	显示一般锁存器信息
-t	显示活动表空间的表空间信息（包括驻留状态）
-T	显示所有表空间的表空间信息

-u	打印用户活动的该要文件
-V	显示软件版本号及序列号
-version	显示构建版本、主机、操作系统、编号、日期及 GLS 版本
-x	显示有关事务的信息
-X	获取关于正在共享和等待缓冲区的线程的确切信息
-z	将概要文件计数设置为 0
infile	为 onstat 工具指定源文件（onstat -o 输出文件）

onstat -g 选项	**功能**
-g act	打印活动的线程
-g afr *pool name* | *session id*	打印指定会话或共享内存吃的已分配内存分段
-g all	从所有 onstat -g 选项打印输出
-g arc	打印数据库备份信息
-g ath	打印所有线程
-g bug	为每个缓冲池打印概要文件信息
-g cat	打印 Enterprise Replication 全局目录中的信息
-g cac agg	打印当前处于高速缓存中的用户定义聚集的定义
-g cac stmt	打印 SQL 语句高速缓存的内容
-g cdr	打印 Enterprise Replication 配置参数和环境变量信息
-g ckp	打印检查点历史记录并显示配置建议
-g cmsm	打印连接管理器守护程序实例并显示每个守护程序已处理的连接数
-g con	打印带有等待程序的条件
-g dbc	打印数据库调度程序和 SQL 管理 API 的信息
-g ddr	打印 Enterprise Replication 数据库日志阅读器状态
-g dic *table*	打印共享内存字典中高速缓存信息
-g dis	打印数据库服务器实例信息
-g dll	打印已装入的动态库的列表
-g dmp	为若干给定的字节在给定的地址处打印原内存
-g dri	打印数据复制信息
-g dsc	打印数据分布高速缓存
-g dss	打印有关个别数据同步线程的活动和用户定义数据类型的详细统计信息
-g dtc	打印有关当行不再需要时从 delete 表中除去的 delete 表清除程序的统计信息
-g env	打印数据库服务器当前使用的环境变量值
-g ffr *pool name*|*session id*	打印共享内存池的可用分段
-g glo	打印全局多线程信息
-g grp	打印 Enterprise Replication 分组的统计信息
-g his	打印关于 SQLTrace 配置参数的信息
-g imc	打印已连接到数据库服务器的 Gbasedbt MaxConnect 实例信息
-g iob	打印大缓冲区使用摘要
-g iof	打印按块文件排列的异步 I/O 统计信息

-g iog	打印 AIO 全局信息
-g ioq *queue name*	打印 queue name 的暂挂 I/O 操作
-g iov	打印按虚拟处理器排列的异步 I/O 统计信息
-g ipl	打印索引页的日志记录状态
-g lap	以细体附加的状态打印信息
-g lmx	打印所有已锁定的互斥
-g lsc	显示有关轻度扫描的信息
-g mem *pool name*|*session id*	打印内存池统计信息
-g mgm	打印内存分配管理器资源信息
-g nbm	打印非常驻段的块位图
-g nif	打印有关网络接口的统计信息
-g nsc *client id*	按 *client id* 打印共享内存状态
-g nsd	打印轮询线程的网络共享内存数据
-g nss *session id*	按 *session id* 打印网络共享内存状态
-g nta	打印-g ntd、-g ntm、-g ntu 的组合网络统计信息
-g ntd	按服务打印网络统计信息
-g ntm	打印网络邮件统计信息
-g ntt	打印网络用户时间
-g ntu	打印网络用户统计信息
-g opn	打印打开的分区
-g pos	打印$GBASEDBTDIR/etc/ .infos.DBSERVERNAME
-g ppf *partition number*|0	打印分区该要文件
-g prc	打印有关 SPL 例程高速缓存的信息
-g proxy	打印代理分发器信息
-g qst	打印队列等待统计信息
-g que	打印高级别队列接口的统计信息
-g rbm	打印常驻段的块位图
-g rcv [serverid]	打印有关接收管理器的统计信息
-g rea	打印就绪队列
-g rep [replname]	打印调度管理器队列中的事务
-g rqm	打印由可靠队列管理器（RQM）管理的低级别队列的统计信息
-g rss	打印远程单机辅助（RHAC）服务器信息
-g rwm	打印读/写互斥
-g sch	打印每个虚拟处理器的信号量操作、自旋和忙等待的数量
-g sds	打印共享磁盘辅助（SSC）服务器信息
-g seg	打印共享内存段的统计信息
-g ses *sessionid*	按 *sessionid* 打印会话信息，如果缺少 sessionid，那么打印每个会话的一行摘要
-g sle	打印所有睡眠线程
-g smb *option*	打印有关智能大对象空间的详细信息
-g smx	显示服务器多路复用器组的连接信息
-g spi	打印带有长自旋的自旋锁
-g sql *session id*	按 *session id* 打印 SQL 信息，如果省略 session id，那么打印每个会话的

	单行摘要
-g ssc	监视数据库服务器读取高速缓存中的 SQL 语句的次数
-g ssc all	报告仅键高速缓存条目以及完全高速缓存的语句
-g ssc pool	报告 SQL 语句高速缓存的所有内存池的使用情况
-g stk *tid*|all	转储由线程标志所指定线程的堆栈或所有线程的堆栈
-g stm [*session id*]	显示每个准备好的 SQL 语句所使用的内存
-g stq	打印队列流信息
-g sts	打印每个线程的最大和当前堆栈使用
-g sync	显示哪个同步是活动的
-g tpf *tid*	打印特定线程标识的线程该要
-g ufr *pool name*|*session id*	按使用情况打印已分配分段
-g vpcache	返回关于 CPU VP 内存块高速缓存统计值的信息
-g wai	打印等待的线程
-g wmx	打印所有带有等待程序的互斥
-g wst	为线程打印等待统计信息

3.1.2. 监控实例状态（onstat -）

onstat -命令显示数据库实例的当前状态，正常情况数据库应处于 On-Line 模式，如以下输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:11:54 -- 833360 Kbytes
```

如出现以下输出，则该数据库实例未启动或已关闭，需使用 oninit -v 启动

```
shared memory not initialized for GBASEDBTSERVER 'gbase01'
```

具体输出格式解释如下：

```
Version--Mode (Type)--(Checkpnt)--Up Uptime--Sh_mem Kbytes
```

输出说明：

参数名	说明	格式
Version	产品版本	字符
Mode	当前数据库运行模式，可能包括以下几种模式，数据库正常对外提供服务应为 On-Line 模式 Initialization 实例启动过程 Shutting Down 实例关闭过程 Quiescent 静默模式，数据库不可连接 On-Line 多用户模式，数据库正常运行状态模式 Fast Recovery 数据库启动或故障恢复过程 Single-User 管理员单用户模式	字符
(Type)	如果数据库为高可用集群，该处显示为主节点（Prim）或辅节点（Sec），如果数据库不是高可用集群，此处不显示	字符

参数名	说明	格式
(Checkpnt)	检查点标志，可能会显示以下两个值 (CKPT REQ)　数据库服务器需要执行检查点 (CKPT INP)　数据库服务器正在执行检查点	字符
Uptime	数据库启动以来运行的时间	字符
Sh_mem	数据库服务器使用共享内存的大小	十进制数字

3.1.3. 监控空间（onstat -d）

onstat -d 命令用于检查数据库空间的使用情况，重点需要关注是否存在空间 chunk down 的情况：

检查是否存在空间 down 不可用的情况：

```
onstat -d |grep PD
```

检查空间使用情况：

```
onstat -d
```

要具体统计各空间的详细使用情况，建议使用 SMI 监控中的相关 sql 监控。

onstat -d 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:08:31 -- 833360 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
45c9c028         1        0x40001    1        1        2048     N  BA    gbasedbt rootdbs
45ddb308         2        0x40001    2        1        2048     N  BA    gbasedbt plogdbs
45ddb538         3        0x40001    3        1        2048     N  BA    gbasedbt llogdbs
45ddb768         4        0x42001    4        1        16384    N TBA    gbasedbt tempdbs01
45ddb998         5        0x48001    5        1        2048     N SBA    gbasedbt sbspace01
45ddbbc8         6        0x40001    6        1        16384    N  BA    gbasedbt datadbs01
 6 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
45c9c258         1      1      0          512000     498315               PO-B-- /data/gbase/rootchk
56ffe028         2      2      0          512000     11947                PO-B-- /data/gbase/plogchk
56fff028         3      3      0          512000     11947                PO-B-- /data/gbase/llogchk
57000028         4      4      0          64000      63947                PO-B-- /data/gbase/tempchk01
57047028         5      5      0          512000     477465     477465    POSB-- /data/gbase/sbspace01
                                 Metadata 34482      25659      34482
57048028         6      6      0          64000      63947                PO-B-- /data/gbase/datachk01
```

```
 6 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

以下为 onstat -d 输出解释：

第一部分输出为空间信息：

列名	说明	格式
address	共享内存中存储的空间地址	十进制数字
number	空间唯一标志号	十进制数字
flags	0x0001 不允许镜像且数据库空间上未镜像的 0x0002 允许镜像且数据库空间上未镜像的 0x0004 允许镜像且数据库空间上镜像的 0x0008 新镜像的 0x0010 Blob 空间 0x0200 正在恢复空间 0x0400 空间已完全恢复 0x0800 正在恢复逻辑日志 0x2000 临时空间 0x4000 正在备份的 Blob 空间 0x8000 智能大对象空间 0x10000 物理或逻辑日志已更改. 0x20000 数据库空间或块表已更改 0x040000 包含大块的 Blob 空间 0x080000 有重命名 Chunk 的空间 0x00100000 SSC 备机上 SDS_TEMPDBS 参数指定的 SSC 专用的临时空间， 0x00200000 SSC 备机上参数 DBSPACETEMP 指定的临时空间 0x00400000 执行了外部备份的数据库空间 0x00800000 空间正在进行碎片整理 0x01000000 物理日志空间	十六进制数字
fchunk	数据库空间中的主 chunk	十进制数字
nchunks	数据库空间的 chunk 数量	十进制数字
pagesize	数据库空间的页大小	十进制数字
flags	Position 1:M 镜像 N 未镜像 Position 2:X 新镜像 D Down，不可用 chunk	字符

列名	说明	格式
	P 物理恢复完成，等待逻辑恢复 L 正在逻辑恢复 R 正在恢复 Position 3:B BLOB 空间 P 物理日志空间 S 智能大对象空间 T 临时空间 U 临时智能大对象空间 W SSC 主节点的临时空间，只在 SSC 备节点显示 Position 4:B 空间可包含大于 2G 的 chunk Position 5:A 空间自动扩展	
owner	空间属主	字符
name	空间名	字符

active 为当前活动的空间数量，maximum 为最大个数。

第二部分输出为 chunk 信息：

列名	说明	格式
address	内存中记录的 chunk 地址	十六进制数字
chk	chunk 唯一 id 号	十进制数字
dbs	chunk 所属空间号	十进制数字
offset	chunk 偏移量	页数
size	chunk 大小	页数
free	chunk 可用大小	页数
bpages	如果是 BLOB 空间的 chunk，显示空间 chunk 大页数量	十进制数字
flags	Position 1:P 主 chunk M 镜像 chunk Position 2:O 可用 D 不可用 I 不一致状态，高可用环境下备机显示该状态 R 正在恢复 X 新镜像 chunk N 重命名 chunk Position 3:- 标准数据空间 B BLOB 空间 S 智能大对象空间 Position 4:B 大于 2G 的 chunk	字符

列名	说明	格式
	Position 5:E 可扩展 chunk - 不可扩展 chunk Position 6:- 文件 chunk 未开启直接 I/O 或同步 I/O C AIX 上的文件 chunk 开启同步 I/O D 文件 chunk 开启直接 I/O	
pathname	chunk 的绝对路径	字符

3.1.4. 监控逻辑日志（onstat -l）

onstat -l 命令用于监控物理日志及逻辑日志的使用情况，需关注已使用的逻辑日志是否及时备份。

如以下命令检查是否存在未备份的逻辑日志：

```
onstat -l |grep "U--" |grep -v C
```

如没有输出，则所有逻辑日志已正常备份，如有输出，表示存在未及时备份的逻辑日志，需要及时备份，避免因逻辑日志未及时备份导致数据库挂起。

以下为 onstat -l 输出的 flags 列标志说明：

标志	说明	格式
A------	日志文件已添加且可用，但尚未使用	字符
D------	如果您删除具有状态 U-B 的日志文件，那么它会标记为已删除。该日志文件被删除，其空间得以释放，可在您为所有存储空间进行 0 级备份时重新使用。	字符
F------	日志文件是空闲的且可供使用。逻辑日志文件在备份后得以释放，逻辑日志文件中的所有事务均关闭，存储在该文件中的最旧更新将会清仓到磁盘	字符
U	日志文件已使用但尚未备份。	字符
U-B----	日志文件已备份但仍是恢复所需。（当恢复不再需要该日志文件时就释放）	字符
U-B---L	日志文件已备份但仍是恢复所需。包含最就检查点记录。	字符
U---C	数据库服务器当前正在填充日志文件。	字符
U---C-L	当前日志文件包含最就检查点记录。	字符
A------	日志文件已添加且可用，但尚未使用	字符

3.1.5. 监控运行信息（onstat -p）

onstat -p 命令用于监控数据库运行以来数据库各指标的统计情况。

```
onstat -p
```

如以下示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 1 days 15:40:15 -- 857260 Kbytes

Profile
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
4991       9808       1011086    99.52   14709      18053      93931      84.34

isamtot    open      start      read       write      rewrite    delete     commit     rollbk
1082785    116872    123376     369966     17752      14036      2693       10794      0

gp_read    gp_write   gp_rewrt   gp_del     gp_alloc   gp_free    gp_curs
0          0          0          0          0          0          0

ovlock     ovuserthread ovbuff     usercpu  syscpu   numckpts   flushes
0          0            0          11.78    18.31    26         52

bufwaits   lokwaits   lockreqs   deadlks    dltouts    ckpwaits   compress   seqscans
114        0          770920     0          0          0          2074       2911

ixda-RA    idx-RA     da-RA      logrec-RA  RA-pgsused lchwaits
0          1534       1388       2          1383       2
```

3.1.6. 监控日志（onstat -m）

onstat -m 输出数据库日志文件路径及最后 20 行日志，需要重点关注日志中是否出现 assert、err、fail 关键字。

如以下命令检查日志文件最后一万行是否存在错误：

```
onstat -c MSGPATH |awk '{print "tail -10000 " $1}' |sh |egrep -i 'assert|err|fail'
```

3.1.7. 监控配置参数（onstat -c）

onstat -c 输出数据库参数配置文件$GBASEDBTDIR/etc/$ONCONFIG。

如以下命令检查数据库配置参数 LTAPEDEV：

```
onstat -c LTAPEDEV
```

3.1.8. 监控实例监听（onstat -g ntt）

onstat -g ntt 输出当前实例监听 IP 端口及 session 连接信息。

onstat -g ntt 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:51:27 -- 961360 Kbytes

global network information:
  #netscb connects         read        write    q-free  q-limits  q-exceed alloc/max
   3/   4       30        18387         8265   11/  11  240/  10    0/1422   12/  12

Individual thread network information (times):
          netscb thread name    sid     open     read    write address
        57458c78 sqlexec         55 10:58:04 10:58:35 10:58:35
        4c8216f0 soctcplst        3 10:07:16 10:58:04          192.168.0.4|9088|soctcp
        4c01e6f0 soctcppoll       2 10:07:16
```

输出说明：

以 soctcp 结尾的行，显示当前实例的监听 IP 及端口，以上输出表示监听 IP 为 192.168.0.4，端口为 9088，如无此行输出，则数据库实例未正常启动监听。

3.1.9. 监控备份情况（onstat -g arc）

onstat -g arc 检查数据库备份情况，根据输出的备份信息检查数据库是否及时备份。

onstat -g arc 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:17:38 -- 833360 Kbytes


num   DBSpace            Q Size Q Len  Buffer partnum    size     scanner

Dbspaces - Archive Status
name               number level date             log        log-position
rootdbs            1      0     07/28/2020.10:04 7          0x25ed018
                          1     07/27/2020.18:00 7          0x7c1018
                          2     07/28/2020.06:00 7          0x180e018
plogdbs            2      0     07/28/2020.10:04 7          0x25ed018
                          1     07/27/2020.18:00 7          0x7c1018
                          2     07/28/2020.06:00 7          0x180e018
llogdbs            3      0     07/28/2020.10:04 7          0x25ed018
                          1     07/27/2020.18:00 7          0x7c1018
                          2     07/28/2020.06:00 7          0x180e018
```

```
sbspace01          5       0       07/28/2020.10:04 7          0x25ed018
datadbs01          6       0       07/28/2020.10:04 7          0x25ed018
```

onstat -g arc 输出解释：

列名	说明	格式
name	空间名称	字符
number	空间编号	十进制数字
level	备份的级别，可能值为 0、1、2	十进制数字
date	备份时间	字符
log	记录备份的逻辑日志	十进制数字
log-position	记录备份的逻辑日志位置	十六进制数字

3.1.10. 监控正在运行的 sql（onstat -g sql）

onstat -g sql 检查数据库正在运行的 sql，要检查某个 session 正在运行的 sql，可使用 onstat -g sql <sid>检查某个会话正在运行的 sql 语句。

onstat -g sql 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:26:18 -- 833360 Kbytes


Sess       SQL             Current            Iso Lock       SQL  ISAM F.E.
Id         Stmt type       Database           Lvl Mode       ERR  ERR  Vers  Explain
26                         sysadmin           DR  Wait 5     0    0    -     Off
25                         sysadmin           DR  Wait 5     0    0    -     Off
24                         sysadmin           DR  Wait 5     0    0    -     Off
23                         sysadmin           CR  Not Wait   0    0    -     Off
```

输出解释：

列名	说明	格式
Sess ID	session ID.	十进制数字
SQL Stmt type	当前执行的 sql 类型，可能为 select、update、delete 等	字符
Current Database	当前连接的数据库	字符
Iso Lvl	会话使用的隔离级别 DR Dirty Read CR Committed Read CS Cursor Stability RR Repeatable Read LC Committed Read Last Committed	字符
Lock Mode	锁等待时间	字符
SQL ERR	出错的错误号	十进制数字

列名	说明	格式
ISAM ERR	出错的 ISAM 号	十进制数字
F.E. Vers	sqli 协议版本	字符
Explain	是否打开执行计划，可能为 on、off	字符

3.1.11. 监控所有实例配置（onstat -g dis）

onstat -g dis 检查本台服务器上配置的所有数据库实例，显示实例的配置文件及运行状况。

onstat -g dis 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:14:25 -- 833360 Kbytes
There are 1 servers found
Server         : gbase01
Server Number : 0
Server Type    : IDS
Server Status : Up
Server Version: GBase Database Server Version 12.10.FC4G1AEE
Shared Memory : 0x44000000
GBASEDBTDIR    : /opt/gbase
ONCONFIG       : /opt/gbase/etc/onconfig.gbase01
SQLHOSTS       : /opt/gbase/etc/sqlhosts
Host           : gbasehost01
```

3.1.12. 监控服务器配置（onstat -g osi）

onstat -g osi 检查本台服务器的操作系统资源配置情况。

onstat -g osi 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:26:38 -- 833360 Kbytes



Machine Configuration....
OS Name                                 Linux
OS Release                              3.10.0-1062.18.1.el7.x86_64
OS Node Name                            gbasehost01
OS Version                              #1 SMP Tue Mar 17 23:49:17 UTC 2020
OS Machine                              x86_64
Number of processors                    1
Number of online processors             1
System memory page size                 4096 bytes
System memory                           990 MB
```

```
System free memory                      851 MB
Number of open files per process        65535
shmmax                                  9223372036854775807
shmmin                                  1
shmids                                  4096
shmNumSegs                              9223372036854775807
semmap                                  <<  UnSupported  >>
semids                                  128
semnum                                  32000
semundo                                 <<  UnSupported  >>
semNumPerID                             250
semops                                  32
semUndoPerProc                          <<  UnSupported  >>
semUndoSize                             20
semMaxValue                             32767
```

3.1.13. 监控实例进程（onstat -g glo）

数据库的一个 VP 在操作系统中是一个 oninit 进程，onstat -g glo 输出当前实例的所有 VP 信息，可通过 pid 列与操作系统进程号对应，在多实例环境下，可检查哪些进程为当前实例的进程，如果需要 kill 进程，可有效避免误杀了别的实例的 oninit 进程。

onstat -g glo 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:26:56 -- 833360 Kbytes

MT global info:
sessions threads  vps      lngspins time
0        40       19       1        1616

          sched calls     thread switches yield 0   yield n   yield forever
total:    93435           62497           31867     24025     14324
per sec:  48              46              2         43        0

Virtual processor summary:
 class       vps       usercpu   syscpu    total
 cpu         1         2.08      1.02      3.10
 aio         12        0.09      3.12      3.21
 lio         1         0.01      0.10      0.11
 pio         1         0.01      0.03      0.04
 adm         1         0.04      0.09      0.13
```

```
 soc          1          0.43       0.55       0.98
 msc          1          0.00       0.00       0.00
 fifo         1          0.00       0.01       0.01
 total        19         2.66       4.92       7.58

Individual virtual processors:
 vp     pid        class        usercpu    syscpu     total      Thread     Eff
 1      19858      cpu          2.08       1.02       3.10       14.60      21%
 2      19861      adm          0.04       0.09       0.13       0.00       0%
 3      19862      lio          0.01       0.10       0.11       0.85       12%
 4      19864      pio          0.01       0.03       0.04       0.08       50%
 5      19866      aio          0.06       1.60       1.66       17.54      9%
 6      19869      msc          0.00       0.00       0.00       0.01       0%
 7      19872      fifo         0.00       0.01       0.01       0.01       100%
 8      19874      soc          0.43       0.55       0.98       NA         NA
 9      19875      aio          0.02       1.30       1.32       15.48      8%
 10     19876      aio          0.00       0.05       0.05       0.54       9%
 11     19878      aio          0.00       0.01       0.01       0.03       34%
 12     19879      aio          0.00       0.02       0.02       0.03       70%
 13     19880      aio          0.01       0.03       0.04       0.04       100%
 14     19881      aio          0.00       0.01       0.01       0.01       100%
 15     19882      aio          0.00       0.01       0.01       0.01       100%
 16     19884      aio          0.00       0.02       0.02       0.02       100%
 17     19886      aio          0.00       0.02       0.02       0.02       100%
 18     19887      aio          0.00       0.02       0.02       0.02       100%
 19     19889      aio          0.00       0.03       0.03       0.03       100%
                   tot          2.66       4.92       7.58
```

3.1.14. 监控实例内存（onstat -g seg）

onstat -g seg 输出当前实例的所有共享内存段的信息。

onstat -g glo 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:27:13 -- 833360 Kbytes

Segment Summary:
id         key        addr               size               ovhd     class blkused  blkfree
589824     52564801   44000000           14401536           601928   R     3516     0
622593     52564802   44dbc000           393216000          4609704  V     80338    15662
655362     52564803   5c4bc000           113287168          1        B     27658    0
688131     52564804   630c6000           332455936          1        B     81166    0
Total:     -          -                  853360640          -        -     192678   15662

   (* segment locked in memory)
```

```
No reserve memory is allocated
```

输出说明：

列名	说明	格式
id	共享内存段在系统中的 id 号，与操作系统命令 ipcs -m 的输出的 shmid 对应	十进制
key	通过 SERVERNUM 计算出来的内存段键值	十进制
addr	内存段在内存中的地址	十六进制
size	内存段大小，单位为字节	十进制
ovhd	跟踪该内存段的页头大小	十进制
class	内存段的类型，包括以下类型： R 常驻段 V 虚拟段 B 缓冲池 M 消息段	字符
blkused	已用 blk 数，blk 在 linux 上为 4k	Dec
blkfree	未使用 blk 数，blk 在 linux 上为 4k	Dec

3.1.15. 监控队列情况（onstat -g rea）

onstat -g rea 检查数据库队列情况，正常情况如数据库服务器配置能满足当前业务压力需求，不会出现队列。

onstat -g rea 示例输出：

```
GBase Database Server Version 12.10.FC4G1AEE -- On-Line -- Up 00:25:42 -- 833360 Kbytes

Ready threads:
 tid      tcb              rstcb            prty status               vp-class       name
```

类似以上输出表示数据库没有等待队列，数据库运行无压力。

3.2. 使用 SMI 表监控

数据库服务器创建并维护 sysmaster 数据库。它类似于数据库的系统目录，每个数据库服务器的 sysmaster 数据库对有关数据库服务器的信息进行跟踪。sysmaster 数据库包含系统监视接口（SMI）表。SMI 表提供有关数据库服务器状态信息。可以查询这些表以识别处理瓶颈、确定资源的使用、跟踪会话或数据库服务器活动等等。

3.2.1. 监控空间使用

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查数据库空间使用情况，需要重

点关注 dbsstatus 列的输出是否存在 Disabled 的情况，如出现该情况，参照紧急故障处理中的 chunk down 流程处理。

```
SELECT
A.dbsnum as No,
trim(B.name) as name,
CASE
WHEN (bitval(B.flags,'0x10')>0 AND bitval(B.flags,'0x2')>0) THEN 'MirroredBlobspace'
WHEN bitval(B.flags,'0x10')>0 THEN 'Blobspace'
WHEN bitval(B.flags,'0x2000')>0 AND bitval(B.flags,'0x8000')>0 THEN 'TempSbspace'
WHEN bitval(B.flags,'0x2000')>0 THEN 'TempDbspace'
WHEN (bitval(B.flags,'0x8000')>0 AND bitval(B.flags,'0x2')>0) THEN 'MirroredSbspace'
WHEN bitval(B.flags,'0x8000')>0 THEN 'SmartBlobspace'
WHEN bitval(B.flags,'0x2')>0 THEN 'MirroredDbspace'
ELSE 'Dbspace'
END as dbstype,
CASE
WHEN bitval(B.flags,'0x4')>0 THEN 'Disabled'
WHEN bitand(B.flags,3584)>0 THEN 'Recovering'
ELSE 'Operational'
END as dbsstatus,
format_units(sum(chksize),2048) as DBS_SIZE ,
format_units(sum(decode(mdsize,-1,decode(nfree,-1,chksize-overhead,nfree),udfree)),2048) as
free_size,
TRUNC(100-sum(decode(mdsize,-1,decode(nfree,-1,chksize-overhead,nfree),udfree))*100/sum(chksize),
2)||'%' as used,
TRUNC(MAX(A.pagesize/1024)) as pgsize,
MAX(B.nchunks) as nchunks
FROM syschktab A, sysdbstab B
WHERE A.dbsnum = B.dbsnum
GROUP BY A.dbsnum,name, 3, 4
ORDER BY A.dbsnum;
```

3.2.2. 监控数据库基本信息

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查数据库的基本信息，包括数据库存储的空间，数据库创建日期，数据库日志模式等：

```
SELECT trim(name) dbname,trim(owner) owner,created,
TRIM(DBINFO('dbspace',partnum)) AS dbspace,
CASE WHEN is_logging+is_buff_log=1 THEN "Unbuffered logging"
     WHEN is_logging+is_buff_log=2 THEN "Buffered logging"
     WHEN is_logging+is_buff_log=0 THEN "No logging"
ELSE "" END Logging_mode
```

```
FROM sysdatabases;
```

3.2.3. 监控实例中数据库占用空间情况

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查实例中每个数据库分配空间的总大小及使用大小。

```
select t1.dbsname,
format_units(sum(ti_nptotal),max(ti_pagesize)) allocated_size,
format_units(sum(ti_npused),max(ti_pagesize))  used_size
from systabnames t1, systabinfo t2,sysdatabases t3
where t1.partnum = t2.ti_partnum
and trim(t3.name)=trim(t1.dbsname)
group by dbsname
order by sum(ti_nptotal) desc;
```

3.2.4. 监控逻辑日志使用

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查数据库逻辑日志使用情况：

```
SELECT  A.number as num,  A.uniqid as uid,  format_units(A.size,'P') as size,
 TRIM( TRUNC(A.used*100/A.size,0)||'%') as used,
 d.name as spacename,
 TRIM( A.chunk||'_'||A.offset ) as location,
 decode(A.filltime,0,'NotFull',
 dbinfo('UTC_TO_DATETIME', A.filltime)::varchar(50)) as filltime,
 CASE  WHEN bitval(A.flags,'0x1') > 0 AND bitval(A.flags,'0x4')>0
   THEN 'UsedBackedUp'
   WHEN bitval(A.flags,'0x1') > 0 AND bitval(A.flags,'0x2')>0
   THEN 'UsedCurrent'
   WHEN bitval(A.flags,'0x1') > 0   THEN 'Used'
   ELSE   hex(A.flags)::varchar(50)
 END as flags,
 CASE  WHEN A.filltime-B.filltime > 0 THEN
  format_units(CAST(TRUNC(A.size/(A.filltime-B.filltime),4)
      as varchar(20)) ,'P')||'/S'
   ELSE    ' N/A '   END as pps
FROM syslogfil A, syslogfil B,syschktab c, sysdbstab d
WHERE  A.uniqid-1 = B.uniqid
and c.dbsnum = d.dbsnum
and a.chunk=c.chknum
```

```
UNION
SELECT  A.number as num,  A.uniqid as uid, format_units(A.size,'P') as size,
 TRIM( TRUNC(A.used*100/A.size,0)||'%') as used,
 d.name as spacename,
 TRIM( A.chunk||'_'||A.offset ) as location,
 decode(A.filltime,0,'NotFull',
 dbinfo('UTC_TO_DATETIME', A.filltime)::varchar(50)) as filltime,
 CASE   WHEN bitval(A.flags,'0x1') > 0 AND bitval(A.flags,'0x4')>0
   THEN 'UsedBackedUp'
   WHEN bitval(A.flags,'0x1') > 0 AND bitval(A.flags,'0x2')>0
   THEN 'UsedCurrent'
   WHEN bitval(A.flags,'0x1') > 0  THEN 'Used'
   WHEN bitval(A.flags,'0x8') > 0  THEN 'NewAdd'
   ELSE hex(A.flags)::varchar(50)  END as flags,
   'N/A' as pps
FROM syslogfil A ,syschktab c, sysdbstab d
WHERE ( A.uniqid = (SELECT min(uniqid) FROM syslogfil WHERE uniqid > 0)
   OR A.uniqid = 0  )
and c.dbsnum = d.dbsnum
and a.chunk=c.chknum
ORDER BY A.uniqid ;
```

3.2.5. 监控表锁的持有者和等待者

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查表锁的持有者和等待者，红色字体分别为库名、表名，可根据 owner 输出的 session 检查当前 session 执行的 sql（onstat -g sql <sid>），如有必要，可执行 onmode -z 杀掉 owner session(onmode -z <sid>)：

```
select
dbsname,b.tabname,rowidr,keynum,
e.txt type,d.sid owner,g.username ownername,f.sid waiter,
h.username waitname
from syslcktab a,systabnames b,systxptab c,sysrstcb d,
sysscblst g,flags_text e, outer ( sysrstcb f , sysscblst h )
where a.partnum = b.partnum
and a.owner = c.address
and c.owner = d.address
and a.wtlist = f.address
and d.sid = g.sid
and e.tabname = 'syslcktab'
and e.flags = a.type
and f.sid = h.sid
and dbsname='testdb'
```

```
and b.tabname ='t';
```

3.2.6. 监控锁等待及死锁

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查死锁较多的表：

```
select dbsname,tabname,partnum, lockreqs , lockwts ,deadlks , seqscans
from sysmaster:sysptprof
where dbsname='testdb'
and tabname[1,3] not in('sys','TBL')
order by lockwts desc;
```

3.2.7. 监控数据库配置参数

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查数据库配置参数：

```
select cf_name,cf_effective,cf_original,cf_default from sysconfig;
```

3.2.8. 监控表的创建时间及锁模式

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查表的创建时间及锁粒度，红色字体为库名：

```
select st.owner,st.dbsname,t.tabname,
dbinfo('UTC_TO_DATETIME',ti.ti_created) tab_createtime,
t.locklevel
from testdb:systables t,  sysmaster:systabinfo ti,sysmaster:systabnames st
where  t.tabid>99
and st.partnum = ti.ti_partnum
and t.tabname = st.tabname
-- and t.tabid=102
-- and t.tabname='tabname'
-- and dbinfo('UTC_TO_DATETIME',ti.ti_created)>='2010-11-03 08:00:00'
and st.dbsname='testdb'
order by  t.tabname;
```

3.2.9. 监控 session 的空闲时间

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查 session 的连接的空闲时间：

```
SELECT s.sid, s.username, s.hostname, q.odb_dbname database,
dbinfo('UTC_TO_DATETIME',s.connected) conection_time,
dbinfo('UTC_TO_DATETIME',t.last_run_time) last_run_time,
```

```
current - dbinfo('UTC_TO_DATETIME',t.last_run_time) idle_time
FROM syssessions s, systcblst t, sysrstcb r, sysopendb q
WHERE t.tid = r.tid AND s.sid = r.sid AND s.sid = q.odb_sessionid
ORDER BY 7 DESC;
```

3.2.10. 监控当前运行慢的 sql

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查当前数据库实例中运行最慢的 25 条 sql：

```
select first 25 sqx_estcost,
sqx_estrows, sqx_sessionid,
sqx_sqlstatement
from sysmaster:syssqexplain
order by sqx_estcost desc;
```

3.2.11. 监控没有数据的 chunk

使用 dbaccess 或工具连接 sysmaster 库，执行以下 sql 检查没有数据的 chunk：

```
select chknum,offset,chksize,nfree,fname,b.name as dbsname
from sysmaster:syschunks a, sysmaster:sysdbspaces b
where a.dbsnum=b.dbsnum
and a.chksize-a.nfree=3
and b.is_temp=0;
```

4. 单机运维管理

4.1. 实例管理

4.1.1. 多实例环境切换

多实例环境下，使用 env 命令检查当前实例名，如：

```
env |grep GBASEDBTSERVER
```

在已知有实例环境文件的情况下，执行实例环境文件切换实例，如：

```
. .bash_profile
```

在未知实例环境文件的情况下，可使用 onstat -g dis 显示所有实例的配置信息，根据 Server、ONCONFIG 最后文件名、SQLHOSTS 值手动 export 环境变量，如：

```
export GBASEDBTSERVER=gbase01
export ONCONFIG=onconfig.gbase01
export GBASEDBTSQLHOSTS=/opt/gbase/etc/sqlhosts.gbase01
```

4.1.2. 启动数据库实例

使用 oninit -v 启动数据库实例：

```
oninit -v
```

最后为以下输出表示数据库已正常启动：

```
Verbose output complete: mode = 5
```

如数据库未能正常启动，参考紧急故障处理章节中的数据库无法启动章节处理。

4.1.3. 关闭数据库实例

使用 onmode -ky 命令关闭数据库实例：

```
onmode -ky
```

使用 onstat -m 检查数据库日志出现以下输出，表示数据库已正常关闭：

```
00:03:39  GBase 8s Database Server Stopped.
```

如 onmode -ky 长时间无法关闭数据库，参考紧急故障处理的数据库无法关闭章节处理。

4.1.4. 切换实例运行模式

可使用 onmode 命令切换数据库实例状态，关于实例状态说明，参照 onstat -命令的输出解释：

```
>>-onmode--+- -k-+-------------------------------------------><
           +- -m-+
           +- -s-+
           +- -u-+
           '- -j-'
```

参数选项	参数说明
-k	切换到脱机模式并清理内存(关闭数据库实例)
-m	将数据库实例从静默状态或管理状态更改为联机模式
-s	以宽限方式关闭实例。该方式允许当前连接到数据库的会话完成所有事务，但不允许新的连接接入。
-u	立即关闭数据库实例。该方式直接终止所有已经连接到数据库的会话，未提交的事务将回滚。
-j	切换数据库服务器到管理模式。进入管理模式后，只有 DBSA 用户(gbasedbt)和 ADMIN_MODE_USERS 参数指定的用户可连接数据库。

4.1.5. 修改实例配置参数

大部分参数可使用 onmode -wf 命令动态修改，如：

```
onmode -wf LTAPEDEV=/dev/null
```

修改完成后，onstat -m 可查看到日志有类似以下记录：

```
20:45:06  Value of LTAPEDEV has been changed to /dev/null.
```

对于不支持动态修改的参数，需要修改数据库配置文件$GBASEDBTDIR/etc/$ONCONFIG 进行修改，并重启数据库后生效。

4.1.6. 查看/修改实例名

查看当前实例名，如：

```
env |grep GBASEDBTSERVER
```

关闭实例，如：

```
onmode -ky
```

编辑实例环境变量文件并生效，修改 GBASEDBTSERVER 环境变量为新的实例名：

```
vi .bash_profile
export GBASEDBTSERVER=gbase01_new
```

```
source .bash_profile
```

修改 ONCONFIG 配置文件中的实例名 DBSERVERNAME 为新的实例名：

```
vi $GBASEDBTDIR/etc/$ONCONFIG
DBSERVERNAME gbase01_new
```

修改 sqlhosts 配置文件中的实例名为新的实例名：

```
vi $GBASEDBTSQLHOSTS
gbase01_new  onsoctcp   *  9088
```

启动数据库实例：

```
oninit -v
```

4.1.7. 查看/修改实例监听 IP/端口

查看当前实例监听端口，如：

```
onstat -g ntt |grep soctcp
```

示例输出：

```
    45a18710 soctcplst        3 11:34:35 11:39:01          127.0.0.1|9088|soctcp
       45a14be0 soctcppoll       2 11:34:35
```

以上输出表示监听 IP 为 127.0.0.1，端口为 9088

修改 sqlhosts 配置文件中的 IP/端口为新的 IP/监听端口：

vi $GBASEDBTSQLHOSTS，以下示例监听数据库服务器所有 IP，端口为 9088：

```
gbase01  onsoctcp   *  9088
```

重启数据库实例：

```
onmode -ky
oninit -v
```

4.2. 数据库管理

4.2.1. 创建默认字符集数据库

要创建数据库，请参考以下语法：

```
>>-CREATE DATABASE--+---------------+--database--+-------------+-->
                    '-IF NOT EXISTS-'            '-IN--dbspace-'

                                    .-NLSCASE SENSITIVE-------.
>--+-------------------------------+--+-+--------------------+-+-><
   '-WITH--+-+----------+--LOG-+-'    '-NLSCASE INSENSITIVE-'
           | '-BUFFERED-'       |
           '-LOG MODE ANSI-----'
```

参数选项	参数说明
database	需要创建的数据库名称
dbspace	指定数据库数据存放的数据库空间，缺省为根数据库空间 rootdbs

示例 1：创建名为 testdb 的数据库（缺省为无日志模式），数据存储于根数据库空间，如未配置 DB_LOCALE 环境变量，使用默认字符集 ISO8859-1：

```
dbaccess - -<<!
create database testdb;
!
```

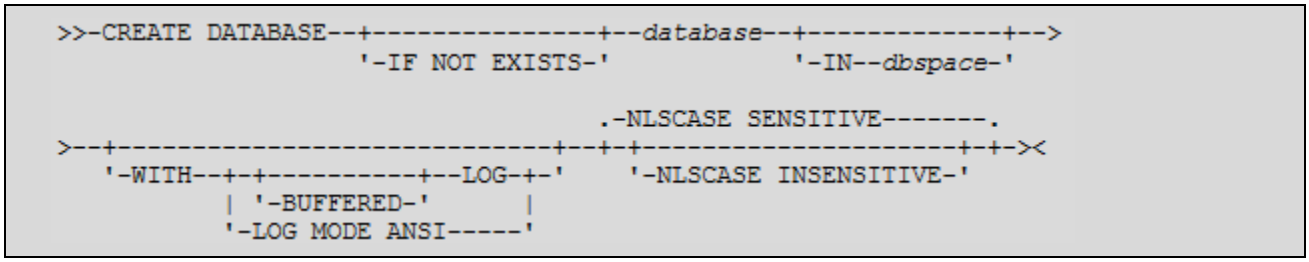
示例 2：创建名为 testdb 数据库，数据库采用 unbuffered 日志模式，数据存储于数据库空间 datadbs01，如未配置 DB_LOCALE 环境变量，使用默认字符集 ISO8859-1：

```
dbaccess - -<<!
create database testdb in datadbs01 with log;
!
```

注意：生产环境建议数据库采用 unbuffered 日志模式。

4.2.2. 创建 GB18030 字符集数据库

要创建数据库，请参考以下语法：



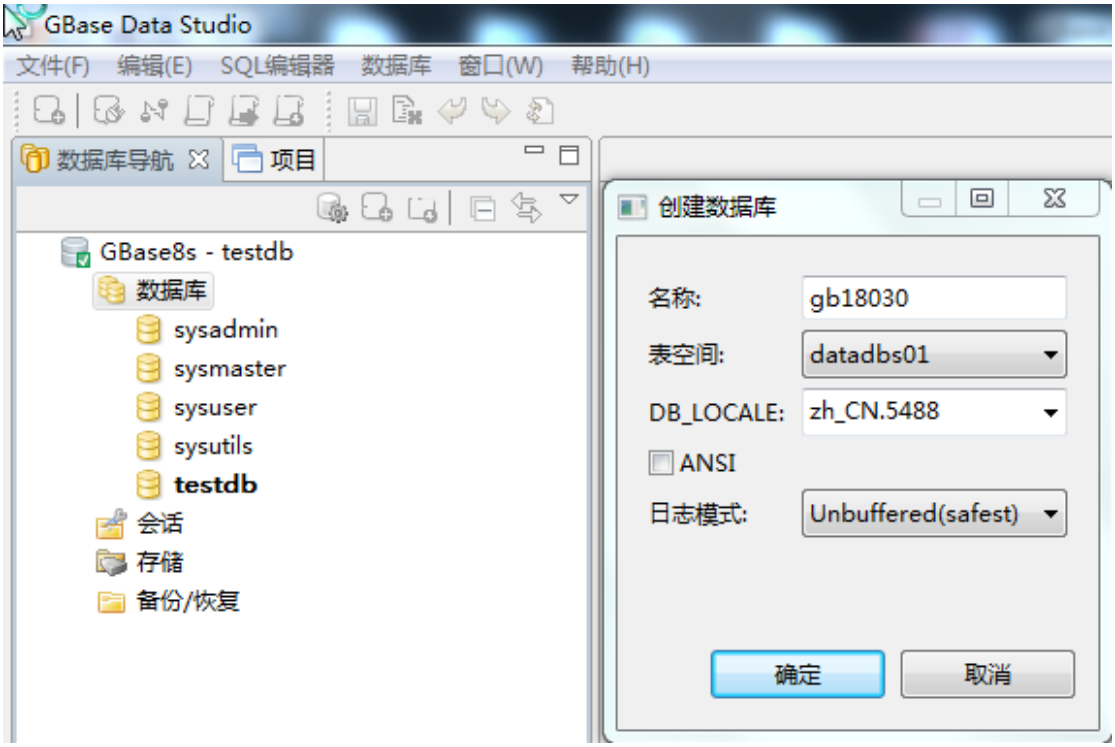
参数选项	参数说明
database	需要创建的数据库名称
dbspace	指定数据库数据存放的数据库空间，缺省为根数据库空间 rootdbs

示例 1：创建名为 testdb 数据库，数据库采用 unbuffered 日志模式，数据存储于数据库空间 datadbs01，字符集为 GB18030-2000：

```
export DB_LOCALE=zh_CN.gb18030-2000
export CLIENT_LOCALE=zh_CN.gb18030-2000
dbaccess - -<<!
create database testdb in datadbs01 with log;
!
```

或者在 GBase DataStudio 中在已经连接的会话上，在 数据库 右键 新建 数据库，指定 名称、数据库空间（表空间）、字符集（DB_LOCALE）为 zh_CN.5488 或者 zh_CN.GB18030-2000，

日志模式为 Unbuffered(safest)。



注意：生产环境建议数据库采用 unbuffered 日志模式。

4.2.3. 创建 UTF8 字符集数据库

要创建数据库，请参考以下语法：

```
>>-CREATE DATABASE--+---------------+--database--+-------------+-->
                    '-IF NOT EXISTS-'            '-IN--dbspace-'

                                    .-NLSCASE SENSITIVE-------.
>--+-------------------------------+--+-+--------------------+-+-><
   '-WITH--+-+----------+--LOG-+-'    '-NLSCASE INSENSITIVE-'
           | '-BUFFERED-'      |
           '-LOG MODE ANSI-----'
```

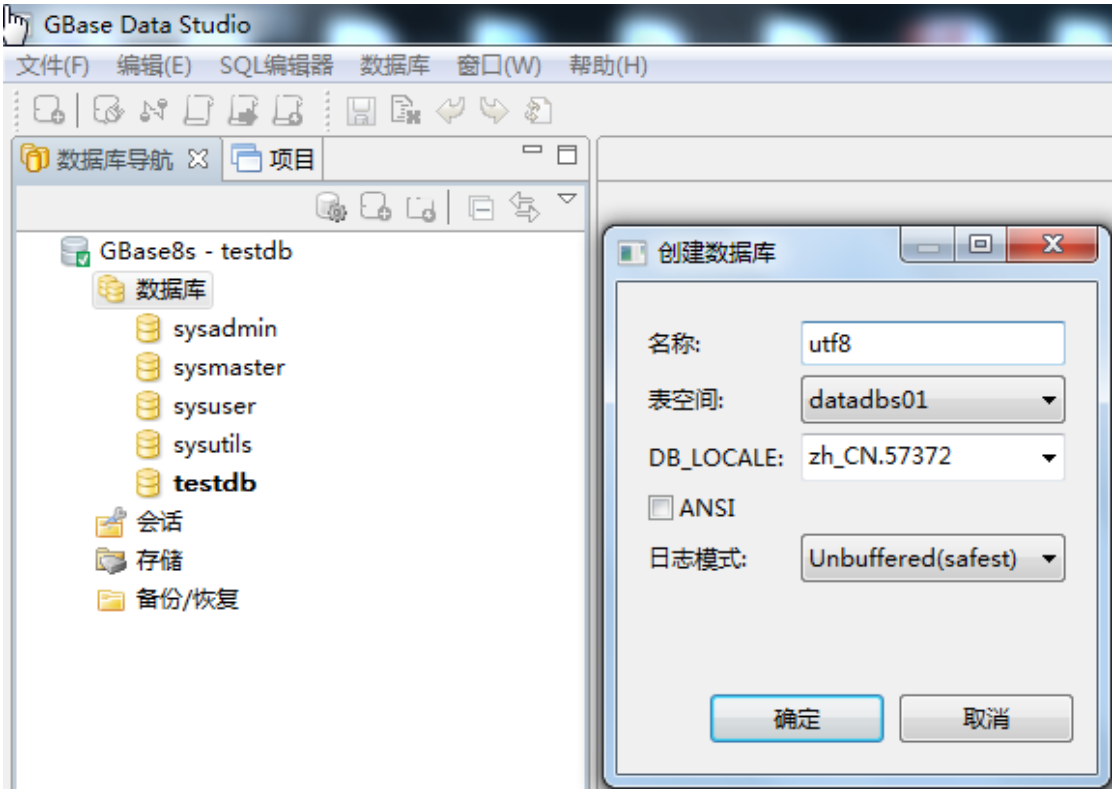
参数选项	参数说明
database	需要创建的数据库名称
dbspace	指定数据库数据存放的数据库空间，缺省为根数据库空间 rootdbs

示例 1：创建名为 testdb 数据库，数据库采用 unbuffered 日志模式，数据存储于数据库空间 datadbs01，字符集为 utf8：

```
export DB_LOCALE=zh_CN.utf8
export CLIENT_LOCALE=zh_CN.utf8
dbaccess - -<<!
```

```
create database testdb in datadbs01 with log;
!
```

或者在 GBase DataStudio 中在已经连接的会话上，在 数据库 右键 新建 数据库，指定 名称、数据库空间（表空间）、字符集（DB_LOCALE）为 zh_CN.57372 或者 zh_CN.utf8，日志模式为 Unbuffered(safest)。



注意：生产环境建议数据库采用 unbuffered 日志模式。

4.2.4. 删除数据库

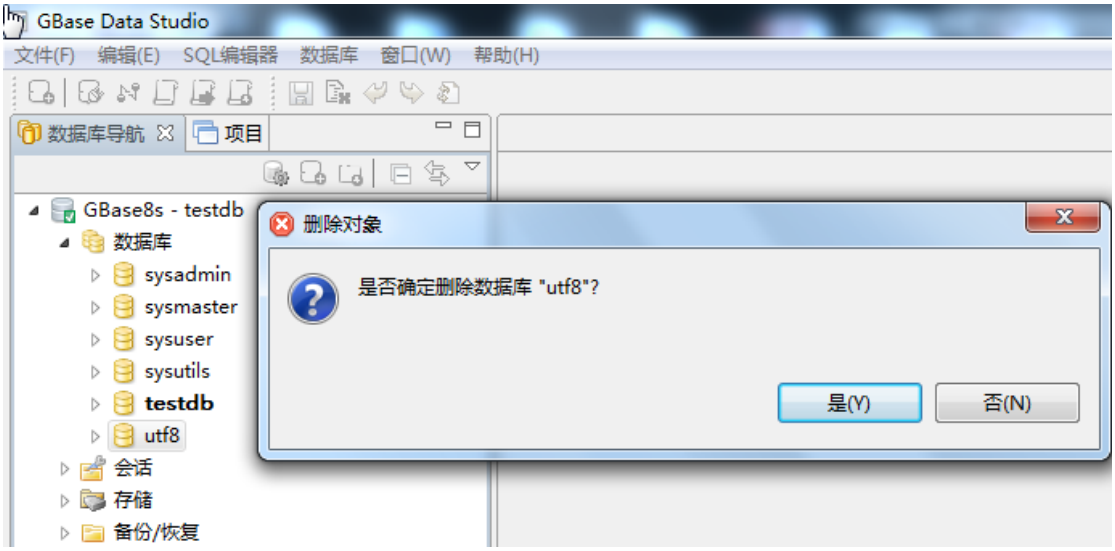
要删除数据库，请参考以下语法：

```
                                                    (1)
>>-DROP DATABASE--+------------+--| Database Name |------------><
                  '-IF EXISTS-'
```

示例 1：删除名为 testdb 的数据库

```
echo "drop database testdb;" | dbaccess - -
```

或者在 GBase DataStudio 中在已经连接的会话上，在 数据库 树下 需要删除的库名上 右键 删除，然后确认 删除对象。



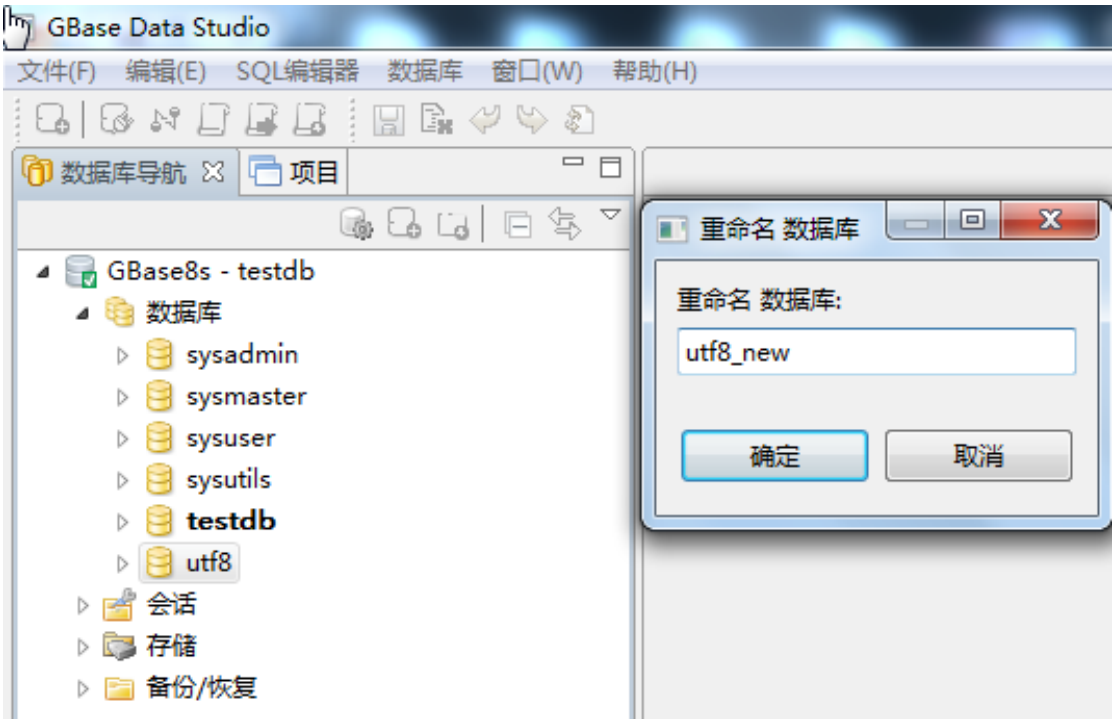
注意：无法删除当前有连接的活动数据库。

4.2.5. 重命名数据库

示例 1：重命名 testdb 数据库为 testdb1

```
echo "rename database testdb to testdb1;" | dbaccess - -
```

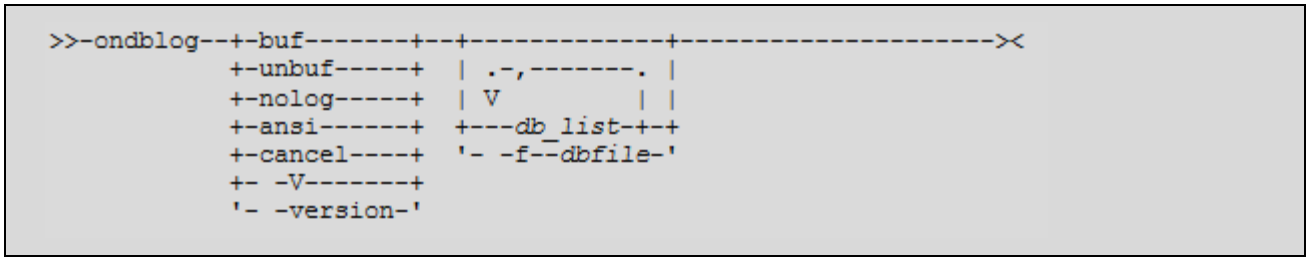
或者在 GBase DataStudio 中在已经连接的会话上，在 数据库 树下 需要修改的库名上 右键 重命名，输入新 数据库名 确认。



注意：无法重命名当前有连接的活动数据库。

4.2.6. 修改数据库日志模式

可使用 ondblog 工具修改数据库日志模式，关于如何检查数据库日志模式，可参照 SMI 监控中数据库基本信息的相关 sql。



参数选项	参数说明
-buf	修改数据库日志模式为已缓冲日志记录
-unbuf	修改数据库日志模式为未缓冲日志记录
-nolog	关闭数据库日志记录
-ansi	修改数据库日志记录方式为 ansi
-cancel	在下一次 0 级备份之前取消日志记录方式更改请求
-f dbfile	更改文本中列出的（每个一行）数据库日志记录状态，该文本文件的路径由 dbfile 给出
dblist	给出要更改其日志记录状态的数据库名称列表

数据库日志模式的切换，可能会导致数据库处于 pending 状态不可用，需要对数据库进行备份，下表说明了各种数据库日志模式之间的切换途径：

转换源	不记录	未缓冲的日志记录	已缓冲的日志记录	符合 ANSI
不记录	不适用	0 级备份	0 级备份	0 级备份
未缓冲的日志记录	是	不适用	是	是
已缓冲的日志记录	是	是	不适用	是
符合 ANSI	非法	非法	非法	不适用

示例 1：如修改 nolog 的 testdb 库日志模式为 unbuffed:

```
ondblog unbuf testdb
```

如使用 onbar 执行伪备份（只更新备份记录，不执行备份操作）

```
onbar -b -F
```

4.2.7. dbaccess 执行 sql

dbaccess 提供了用于输入、执行和调试结构化查询语言（SQL）语句与存储过程语言（SPL）例程的用户界面。

以下示例使用 dbaccess 连接 testdb 数据库：

```
dbaccess testdb -
```

SQL 语句以“；”结束，要退出 dbaccess，使用“Ctrl+c”或“Ctrl+\”

要使用 dbaccess 执行 sql 文本，请参考以下语法：

```
dbaccess testdb yoursql.sql
```

4.2.8. 配置 DBLINK 访问远程库

要访问远程数据库实例中的数据，需要通过配置用户信任及 SQLHOSTS 文件来实现，以下是两种配置方法示例。

如在 node1（192.168.17.101）实例 gbase01 访问 node2（192.168.17.102）上数据库实例 gbase02 的数据。

方法一、配置信任

1、在 node1 数据库实例 gbase01 的 sqlhosts 文件中增加 gbase02 的实例信息：

```
echo "gbase02          onsoctcp        192.168.17.102      9088">>$GBASEDBTSQLHOSTS
```

2、在 node2 服务器配置 node1 服务器信息：

```
echo "192.168.17.101 node1" >>/etc/hosts
```

3、在 node2 服务器配置对 node1 的信任，需使用哪个用户访问数据库即配置哪个用户：

```
echo "node1 gbasedbt" >>/etc/hosts.equiv
```

如 node1 上有多个 IP，可能导致以上配置无效，考虑使用以下配置信任所有 IP：

```
echo "+ gbasedbt" >>/etc/hosts.equiv
```

4、node1 的 gbase01 实例中使用 gbasedbt 用户访问 gbase02 实例

```
dbaccess testdb -<<!
select * from testdb@gbase02:systables;
!
```

方法二、配置.netrc 文件

1、在 node1 数据库实例 gbase01 的 sqlhosts 文件中增加 gbase02 的实例信息：

```
echo "gbase02          onsoctcp        192.168.17.102      9088">>$GBASEDBTSQLHOSTS
```

2、如 node1 需要使用 test 用户以 gbasedbt 用户连接到 gbase02，在 node1 的 test 用户主目录下创建.netrc 文件，test 用户执行：

```
echo "machine 192.168.17.102 login gbasedbt password GBase123" >.netrc
```

3、node1 的 gbase01 实例中使用 test 用户访问 gbase02 实例：

```
dbaccess testdb -<<!
select * from testdb@gbase02:systables;
!
```

4.3. 用户和权限管理

4.3.1. 创建用户

数据库的用户使用操作系统中的系统用户或者数据库内部用户，gbasedbt 用户为超级用户，具有数据库所有权限，gbasedbt 组用户为 DBSA 用户，具有管理数据库实例权限。系统用户中的非 gbasedbt 组用户，经数据库授权后，可连接数据库，并具有被授予的相关权限。

方式一，创建操作系统用户

要创建数据库用户，需在操作系统创建该用户，并设定初始密码。经数据库授权后，该用户可连接数据库。如以下示例创建名为 user1 的数据库用户：

使用 root 用户创建系统用户：

```
useradd user1
passwd user1
```

使用 gbasedbt 用户为该用户授权（数据库 testdb 的 connect 权限）：

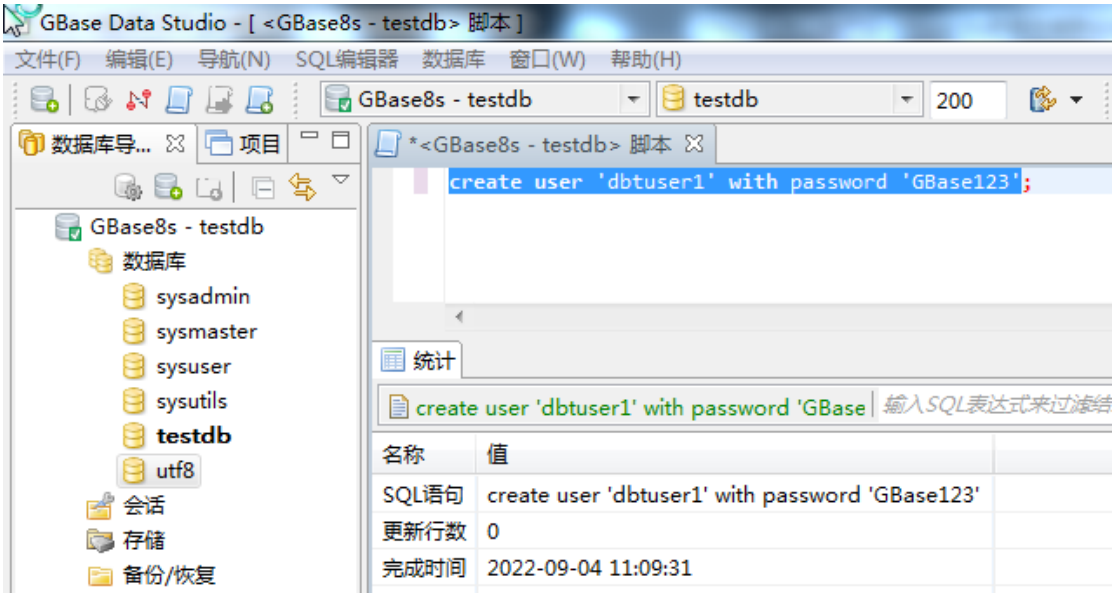
```
echo "grant connect to user1" | dbaccess testdb -
```

方式二，创建数据库内部用户

要创建数据库内部用户，需要在数据库已经启动内部用户，创建内部用户并设置初始密码。经数据库授权后，该用户可连接数据库。如以下示例创建名为 dbtuser1 的数据库用户：

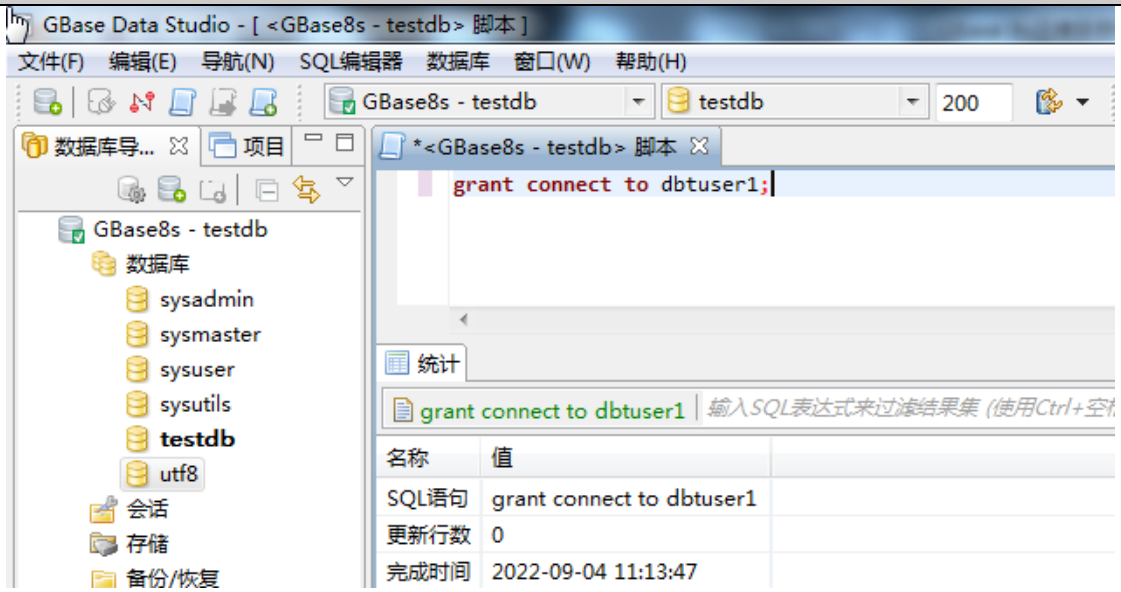
在 GBase DataStudio 中，gbasedbt 用户连接，执行 SQL 语句：

```
create user 'dbtuser1' with password 'GBase123';
```



在指定的库内，为该用户授权（数据库 testdb 的 connect 权限）：

```
grant connect to dbtuser1;
```



注意：创建操作系统用户需要 root 用户执行，创建数据库内部用户需要数据库预先配置开启内部用户，且具有 DBSA 权限的用户才能创建内部用户。如果操作系统用户与数据库内部存在同名，将使用操作系统用户验证。

4.3.2. 修改用户

数据库用户使用操作系统用户时，受操作系统的管理。修改用户（比如：修改用户密码，修改用户属性或者用户删除等）由操作系统管理。数据库用户为数据库库内部用户时，由数据

库系统管理员控制。

方式一，修改操作系统用户

修改用户密码

```
passwd user1
```

锁定用户

```
usermode -L user1
```

删除用户

```
userdel user1
```

方式二，修改数据库内部用户

修改用户密码

```
alter user 'user1' modify password 'GBase123';
```

锁定用户

```
alter user 'user1' ACCOUNT lock;
```

删除用户

```
drop user 'user1';
```

注意：更多的用户修改操作，请参阅操作系统的用户管理相关操作，或者 SQL 语句中的 alter user，drop user 和 rename user 语法。

4.3.3. 用户授权

数据库级授权

数据库级别的访问特权影响对数据库的访问。仅有个别用户（而非角色）可具有数据库特权。数据库访问级别是（从最低到最高）CONNECT、RESOURCE 和 DBA。使用相应的关键字来授予访问特权级别。

特权	作用
CONNECT	查询和修改数据 如果您拥有想要修改的数据库对象，就可以修改数据库模式。具有 CONNECT 特权的任何用户都可以执行以下操作： • 使用 connect 语句或另外的语句连接至数据库 • 执行 SELECT、INSERT、UPDATE 和 DELETE 语句，如果用户具有必要的表级特权 • 创建视图，如果用户对底层表有 selelct 特权 • 创建同义词

	• 创建临时表以及创建临时表的索引 • 改变或删除表或索引，如果用户拥有表或索引（或对表有 Alter、Index 或 References 特权） • 授予对表或视图的特权，前提是用户拥有该表（或已通过 WITH GRANT OPTION 关键字被授予对表的特权）
RESOURCE	用来扩展数据库的结构。除了 Connect 特权的能力外，Resource 特权的拥有者还可以执行以下功能： • 创建新表 • 创建新索引 • 创建新 UDR • 创建新数据类型
DBA	拥有 Resource 特权的所有能力，并且能够执行以下附加操作： • 将任何数据库特权（包括 DBA 特权）授予其他用户 • 将任何表级特权授予其他用户或角色 • 将角色授予用户或其他角色 • 取消某项特权，它的授予者被您在 REVOKE 语句的 AS 字句中指定为 revoke。 • 当注册 UDR 时，限制对 DBA 的 Execute 特权 • 执行 SET SESSION AUTHORIZATION 语句 • 创建任何数据库对象 • 创建表、视图和索引，指定其他用户作为这些对象的所有者 • 改变、删除或重命名数据库对象，不管谁拥有它们 • 执行 UPDATE STATISTICS 语句的 DROP DISTRIBUTIONS 选项 • 执行 DROP DATABASE 和 RENAME DATABASE 语句

示例 1 使用 PUBLIC 关键字将对当前活动数据库的 CONNECT 特权授予所有用户：

```
grant connect to public;
```

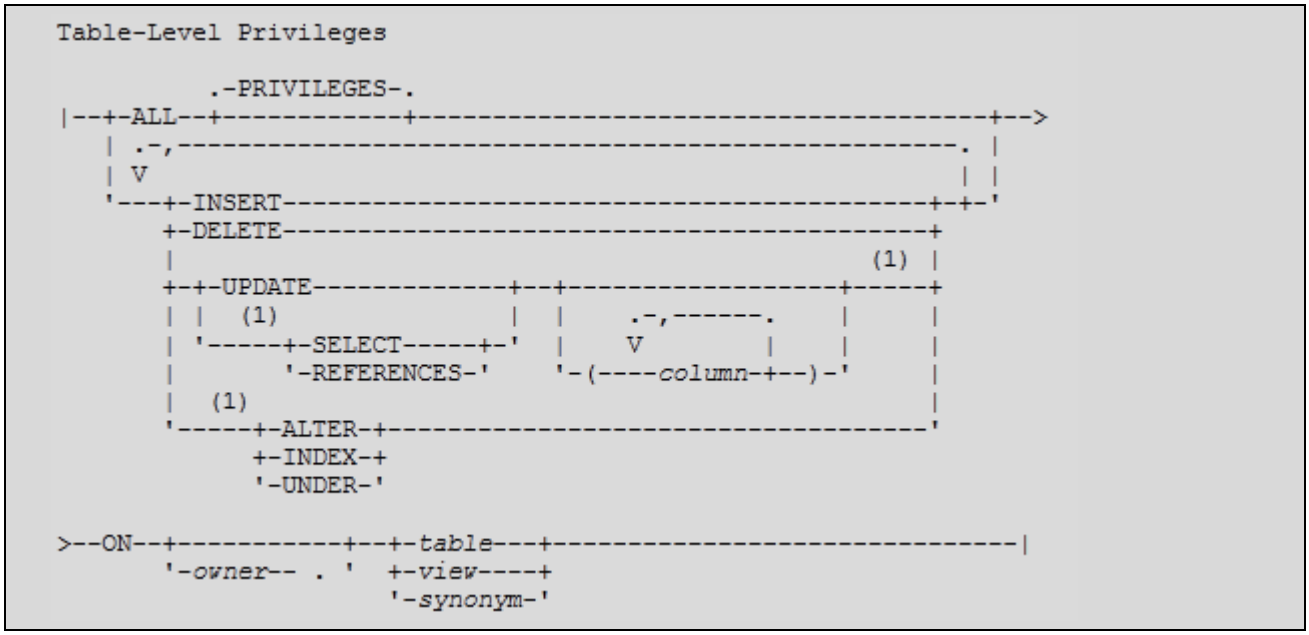
不能将数据库级特权授予角色。

表级授权

当使用 CREATE TABLE 语句创建表是，您就是表所有者并将自动获得所有表级别的特权。不能将所有权转移给其他用户，但可以将表级特权授予其他用户或角色。

具有库级别 DBA 特权的用户将自动获得对该数据库中每个表的所有表级特权。

授予表级特权语法：



特权	作用
INSERT	让您插入行
DELETE	让您删除行
SELECT	用来访问在 SELECT 语句中的任何列
UPDATE	用来访问在 UPDATE 语句中的任何列
REFERENCES	用来定义对列的引用约束。必须拥有 Resource 特权，才能利用 References 特权。
INDEX	让您创建永久索引。要使用 Index 特权，必须有 Resource 特权。
ALTER	用来添加或删除列、修改列数据类型、添加或删除约束、将表的锁定方式从 PAGE 更改为 ROW，或者为表添加或删除对应的 ROW 数据类型。还可以用于启用或禁用索引、约束和触发器。
UNDER	用来在类型表下创建子表
ALL	提供以上所列的所有特权。

如以下示例 2 所示，将表 customer 列 customer_num,fname,lname 的 DELETE、SELECT、UPDATE 权限授予所有用户。

示例 2：

```
grant delete,select,update(customer_num,fname,lname) on customer to public;
```

4.4. 空间管理

4.4.1. 创建数据库空间

要创建数据库空间，使用 onspace -c 命令，语法如下：

```
onspaces  { -c { -d <DBspace> [-k <pagesize>] [-t]
                  -p <path> -o <offset> -s <size> [-m <path> <offset>] } |
              { -d <DBspace> [-k <pagesize>]
                  -p <path> -o <offset> -s <size> [-m <path> <offset>]
                  [-ef <first_extent_size>] [-en <next_extent_size>] } |
              { -P <PLOGspace>
                  -p <path> -o <offset> -s <size> [-m <path> <offset>] } |
              { -b <BLOBspace> -g <pagesize>
                  -p <path> -o <offset> -s <size> [-m <path> <offset>] } |
              { -S <SBLOBspace> [-t]
                  -p <path> -o <offset> -s <size> [-m <path> <offset>]
                  [-Mo <mdoffset>] [-Ms <mdsize>] [-Df <default-list>] }
              { -x <Extspace> -l <Location> } }
```

参数	说明
-d	创建标准数据库空间
-p	创建物理日志空间
-b	创建简单大对象空间
-S	创建智能大对象空间
-x	创建外部空间
-p	数据块（chunk）路径
-o	数据块（chunk）偏移量
-s	空间大小（KB）
-k	空间页大小
-t	创建临时空间

注意：在创建数据库空间之前，需要确认 chunk 路径的设备存在，要求属主为 gbasedbt:gbasedbt，且权限为 660。

如果使用的是文件系统，可使用 gbasedbt 用户参考以下命令创建数据文件：

```
touch /data/gbase/datachk01
chmod 660 /data/gbase/datachk01
```

如果是在 SSC 集群，或者是使用了共享磁盘或者磁盘分区的系统中，则需要做分区/磁盘绑定（原因是在 linux 系统中，重启操作系统后，磁盘的顺序可能会不一致）。分区/磁盘绑定需要在集群的所有节点上都执行。需要完成以下操作：

操作系统上通过 fdisk -l 确认新加的硬盘盘符，比如： /dev/sdX

```
fdisk -l
```

获取磁盘或者磁盘分区的 UUID

```
/usr/lib/udev/scsi_id -g -u -d /dev/sdX
```

假定返回的值为：36000c29e30e2fa02d6b6baa102b29900

在/usr/lib/udev/rules.d/99-dm-gbase.rules 中增加磁盘绑定到指定链接文件

```
-- 对于使用磁盘时使用以下
KERNEL=="sd*", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d
/dev/$name",RESULT=="36000c29e30e2fa02d6b6baa102b29900", SYMLINK+="datachk01", OWNER="gbasedbt",
GROUP="gbasedbt",MODE="0660"

-- 如果是使用磁盘分区，则使用类似以下
KERNEL=="sd*8", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d
/dev/$parent",RESULT=="36000c29e30e2fa02d6b6baa102b29900", SYMLINK+="datachk01", OWNER="gbasedbt",
GROUP="gbasedbt",MODE="0660"
```

重新加载 udev 规则

```
udevadm control --reload-rules
```

重新加载磁盘

```
partprobe /dev/sdX
```

确认磁盘或者磁盘分区正常加载

```
ls -al /dev/sdX                    # 看权限有没有改变;
ls -al /dev/datachk01              # 对应的链接文件是否生成;
```

在数据库空间目录下创建连接到 udev 生成的链接文件;

```
cd /data/gbase
ln -s /dev/datachk01 datachk01     # 使用 gbasedbt 用户操作
```

示例 1：创建页大小为 16k 的标准数据库空间（空间名 datadbds01，chunk 路径 /data/gbase/datachk01，偏移量为 0，大小 1024000KB，页大小 16k）

```
onspaces -c -d datadbs01 -p /data/gbase/datachk01 -o 0 -s 1024000 -k 16
```

或使用 sysadmin 库的接口函数创建数据库空间：

```
echo "EXECUTE FUNCTION task('create dbspace','datadbs01', '/data/gbase/datachk01', '1024000',
'0','16')"|dbaccess sysadmin
```

示例 2：创建页大小为 16k 的临时数据库空间（空间名 tempdbs01，chunk 路径 /data/gbase/tempchk01，偏移量为 0，大小 1024000KB）

```
onspaces -c -d tempdbs01 -p /data/gbase/tempchk01 -o 0 -s 1024000 -k 16 -t
```

或使用 sysadmin 库的接口函数创建临时数据库空间：

```
echo "EXECUTE FUNCTION task('create tempdbspace','tempdbs01', '/data/gbase/tempchk01', '1024000',
'0','16')"|dbaccess sysadmin
```

示例 3：创建智能大对象空间（空间名 sbspace01，chunk 路径/data/gbase/sbspace01，偏移

量为 0，大小 1024000KB）

```
onspaces -c -S sbspace01 -p /data/gbase/sbspace01 -o 0 -s 1024000
```

或使用 sysadmin 库的接口函数创建智能大对象空间：

```
echo "EXECUTE FUNCTION task('create sbspace','sbspace01', '/data/gbase/sbspace01', '1024000',
'0')"|dbaccess sysadmin
```

注意：集群中的磁盘需要在所有节点都绑定或者创建，且权限正确。创建数据库空间操作仅需要在集群的主节点上执行。

4.4.2. 扩容数据库空间

使用 onspaces -a 命令为数据库空间增加数据块（chunk），可参考以下语法：

```
onspaces { -a <spacename> -p <path> -o <offset> -s <size> [-m <path> <offset>]
                { { [-Mo <mdoffset>] [-Ms <mdsize>] } | -U }
                }
```

参数	说明
-a	指定需要添加 chunk 的数据库空间
-p	数据块（chunk）路径
-o	数据块（chunk）偏移量
-s	空间大小（KB）

注意：在扩容数据库空间之前，需要确认 chunk 路径的设备存在，要求属主为 gbasedbt:gbasedbt，且权限为 660。

如果使用的是文件系统，可使用 gbasedbt 用户参考以下命令创建数据文件：

```
touch /data/gbase/datachk02
chmod 660 /data/gbase/datachk02
```

如果是在 SSC 集群，或者是使用了共享磁盘或者磁盘分区的系统中，则需要做分区/磁盘绑定（原因是在 linux 系统中，重启操作系统后，磁盘的顺序可能会不一致）。分区/磁盘绑定需要在集群的所有节点上都执行。需要完成以下操作：

操作系统上通过 fdisk -l 确认新加的硬盘盘符，比如： /dev/sdY

```
fdisk -l
```

获取磁盘或者磁盘分区的 UUID

```
/usr/lib/udev/scsi_id -g -u -d /dev/sdY
```

假定返回的值为：36000c29e30e2fa02d6b6baa102b29900

在/usr/lib/udev/rules.d/99-dm-gbase.rules 中增加磁盘绑定到指定链接文件

```
-- 对于使用磁盘时使用以下
KERNEL=="sd*",      SUBSYSTEM=="block",      PROGRAM=="/usr/lib/udev/scsi_id      -g      -u      -d
/dev/$name",RESULT=="36000c29e30e2fa02d6b6baa102b29900",  SYMLINK+="datachk02",  OWNER="gbasedbt",
GROUP="gbasedbt",MODE="0660"

-- 如果是使用磁盘分区，则使用类似以下
KERNEL=="sd*9",      SUBSYSTEM=="block",      PROGRAM=="/usr/lib/udev/scsi_id      -g      -u      -d
/dev/$parent",RESULT=="36000c29e30e2fa02d6b6baa102b29900", SYMLINK+="datachk02", OWNER="gbasedbt",
GROUP="gbasedbt",MODE="0660"
```

重新加载 udev 规则

```
udevadm control --reload-rules
```

重新加载磁盘

```
partprobe /dev/sdY
```

确认磁盘或者磁盘分区正常加载

```
ls -al /dev/sdY                    # 看权限有没有改变;
ls -al /dev/datachk02              # 对应的链接文件是否生成;
```

在数据库空间目录下创建连接到 udev 生成的链接文件;

```
cd /data/gbase
ln -s /dev/datachk01 datachk01        # 使用 gbasedbt 用户操作
```

示例 1：为数据库空间 datadbs01 增加一个 1024000KB 的 chunk

```
onspaces -a datadbs01 -p /data/gbase/datachk02 -o 0 -s 1024000
```

或使用 sysadmin 库的接口函数增加 chunk：

```
echo  "EXECUTE  FUNCTION  task('add  chunk','datadbs01',  '/data/gbase/datachk02',  '1024000',
'0')"|dbaccess sysadmin
```

其他类型数据库空间扩容与标准数据库空间扩容语法一致。

示例 2：将 chunk 号为 6 的 chunk 文件设置为自动扩展（默认不自动扩展）：

```
echo "execute function task('modify chunk extendable', '6')"|dbaccess sysadmin
```

示例 3：将 chunk 号为 6 的 chunk 文件设置为不自动扩展：

```
echo "EXECUTE FUNCTION task('modify chunk extendable off', '6')"|dbaccess sysadmin
```

注意：集群中的磁盘需要在所有节点都绑定或者创建，且权限正确。增加数据库空间操作仅需要在集群的主节点上执行。

4.4.3. 删除数据库空间或块

要删除数据库空间或空间中的数据库块，可使用 onspaces -d 命令，参考以下语法：

```
onspaces { -d <spacename> [-p <path> -o <offset>] [-f] [-y] }
```

参数	说明
-d	指定需要删除 chunk 的数据库空间
-p	数据块（chunk）路径
-o	数据块（chunk）偏移量
-f	删除不包含元数据的智能大对象数据块
-y	数据库服务器对所有提示响应“是”

示例 1：删除数据库空间 datadbs01 中的数据块/data/gbase/datachk02

```
onspaces -d datadbs01 -p /data/gbase/datachk02 -o 0 -y
```

或使用 sysadmin 库的接口函数删除 chunk：

```
echo "EXECUTE FUNCTION task('drop chunk', 'datadbs01', '/data/gbase/datachk02', '0')" | dbaccess sysadmin
```

示例 2：删除数据库空间 datadbs01

```
onspaces -d datadbs01 -y
```

或使用 sysadmin 库的接口函数删除空间：

```
echo "EXECUTE FUNCTION task('drop dbspace','datadbs01')"|dbaccess sysadmin
```

注意：要删除数据库空间或数据库空间中的数据块，前提是该空间或数据块上无用户数据，否则无法删除。关于如何确认数据块是否存在用户数据，可参考 SMI 中检查没有数据的 chunk 一节。空间删除后，数据文件不会自动删除，可使用 rm 命令删除对应数据文件。

4.5. 数据迁移

4.5.1. dbexport/dbimport 迁移数据库

使用 dbexport 导出数据库

dbexport 工具将数据库卸载到文本文件，并创建数据库的模式文件。可通过 dbimport 使用该模式文件在其他服务器中重新创建数据库。

示例：dbexport 导出 testdb 数据库

```
dbexport testdb -l;
```

注意：在使用 dbexport 导出数据库之前，必须禁用 select 触发器。dbexport 程序在导出期

间运行 select 语句。Select 语句触发器可能会修改数据库内容。

dbexport 导出数据库，在当前目录生成以数据库名开头的文件夹，名字为 *dbname*.exp, *dbname* 为您导出的数据库名，如 testdb，该文件夹包含数据库的所有数据，用于 dbimport 导入。

使用 dbimport 导入数据库

dbimport 工具从文本文件创建数据库并将数据导入数据库。

将 dbexport 导出的 *dbname*.exp 传输到目标服务器，在 *dbname*.exp 所在当前目录执行 dbimport 导入数据库。

示例：dbimport 导入 testdb 数据库到 datadbs01：

```
dbimport testdb -d datadbs01
```

dbimport 导入完成后，数据库无事务日志记录，通过 ondblog 命令修改 testdb 数据库的日志模式。关于数据库日志模式，请参考数据库管理中相关章节。

示例：将 testdb 数据库日志模式修改为 unbuf 日志模式：

```
ondblog unbuf testdb
onbar -b -F
```

示例 2：dbimport 从当前目录导入 testdb 数据库到集群的 datadbs01，并使用 unbuf 日志模式：

```
dbimport testdb -i ./ -d datadbs01 –l
```

注意：导入数据库时，请使用与创建该数据库时使用的相同的环境变量，否则导入可能失败。

特别注意：集群中不能导入 nolog 方式的库，原因是不能同步。

如需导入为其他名称数据库，将 *dbname*.exp 重命名为 *dbname_new*.exp 及该文件夹下相关 sql 文件重命名为 *dbname_new*.sql

示例：将 testdb 数据库导入为 testdb1，存放于 datadbs01 空间：

```
mv testdb.exp testdb1.exp
mv testdb1.exp/testdb.sql testdb1.exp/testdb1.sql
mv testdb1.exp/testdb_ora.sql testdb1.exp/testdb1_ora.sql
dbimport testdb -d datadbs01;
ondblog unbuf testdb1;
onbar -b -F;
```

4.5.2. dbload 分批提交导入文本数据

dbload 工具将数据从一个或多个文本文件传送到一个或多个现有表中。dbload 工具使用灵活，但没有其他方法快，并且必须准备一个命令文件来控制输入。dbload 的优势在于可分批提交需要导入的数据，从而避免长事务的发生。

以下示例将文本数据 test.unl 导入 test 表中，将 test1.unl 导入到 test1 表中，test 表列数为 2，test1 列数为 5。数据库名为 testdb，每次提交 10000 行，错误输出到 err.log

示例：

```
dbload -d testdb -c f.cmd -l err.log -n 10000
```

f.cmd 文件：

```
file test.unl delimiter '|' 2;
insert into test;
file test1.unl delimiter '|' 5;
insert into test1;
```

test.unl 文件：

```
1|good|
2|good|
......
```

test1.unl 文件：

```
1|good|2|gg|dd|
2|good|3|gg|dd|
......
```

4.5.3. dbschema 查看表/对象结构

dbschema 工具输出创建表、视图或数据库所需的 SQL 语句。

示例 1：导出库 testdb 的建库 SQL 语句到文本 testdb.sql

```
dbschema -d testdb -ss testdb.sql
```

示例 2：导出 testdb 库中表 test 的建表语句

```
dbschema –d testdb -t test -ss
```

4.5.4. load/unload 导入导出单表文本数据

load 语句速度较快且较易于使用，但它只接受指定的数据格式。通常可将使用 unload 语句准备好的数据用于 load。

load/unload 用于单表文本数据的装载和卸载，unload 导出表数据到指定文件，load 则将指定的文本文件数据插入到表。

示例 1：导出表 test 的所有数据到 test.unl 文件

```
unload to test.unl select * from test;
```

示例 2：将文本文件 test.unl 的数据导入到表 test

```
load from test.unl insert into test;
```

4.5.5. onunload/onload 导出导入二进制数据

onunload 和 onload 工具提供在相同平台上使用相同数据库服务器的计算机之间移动数据的最快方法。

onunload 导出库或库中单表到指定设备，默认导出到 TAPEDEV 指定的设备，导出文件为二进制文件，不可跨版本或跨平台。

示例 1：使用 onunload 导出表 test 到/data/test.dat

```
onunload -t /data/test.dat testdb:test
```

示例 2：使用 onload 导入表到 testdb，数据驻留于 datadbs01

```
onload -t /data/test.dat testdb:test -d datadbs01
```

注意：集群中不能使用 onload 方式。

4.6. 备份与恢复

4.6.1. 备份恢复工具

GBase 8s 提供了 ontape 和 onbar 两种备份与恢复工具，ontape 适用中小系统简单快速的备份，不依赖存储管理软件，onbar 适用大型系统，依赖存储管理软件进行备份，如 NBU、TSM 等。

4.6.2. ontape

要使用 ontape 进行备份和恢复，需要配置的有以下几个参数：

参数	说明
TAPEDEV	用于存储空间备份的目录文件系统或磁带设备的绝对路径名称，如果 TAPEDEV 配置为文件系统目录，可忽略 TAPEDEVBLK、TAPESIZE
TAPEDEVBLK	用于存储空间备份的磁带的块大小（以 KB 为单位）
TAPESIZE	用于存储空间备份的磁带的大小（以 KB 为单位），0 为不限制
LTAPEDEV	逻辑日志磁带设备或文件系统目录，如果 LTAPEDEV 配置为文件系统目录，可忽略 LTAPEBLK 及 LTAPESIZE，如果 LTAPEDEV 配置为/dev/null，则不备份逻辑日志，逻辑日志填满后数据库自动标记为已备份
LTAPEBLK	用于逻辑日志备份的磁带的块大小（以 KB 为单位）
LTAPESIZE	用于逻辑日志备份的磁带的大小（以 KB 为单位）

示例 1：对数据库所有空间执行 0 级备份

```
ontape -s -L 0
```

备份文件存储于 TAPEDEV 参数指定的目录，命名规则为“主机名_实例编号_L0”，如：gbasehost01_0_L0

示例 2：恢复数据库：

使用 onmode -ky 关闭数据库：

```
onmode -ky
```

使用 ontape -r 恢复数据库：

```
ontape -r
```

可使用以下命令检查恢复是否在正常执行：

```
onstat -D -r 1
```

对比两次输出“page Wr”列是否变化，可确认数据库是否正在恢复写入数据。

恢复完成后，数据库处于 Quiescent 模式，使用 onmode -m 切换到 online

```
onmode -m
```

示例 3：恢复数据库过程

```
[gbasedbt@gbasehost01 ~]$ ontape -r
Restore will use level 0 archive file /opt/gbase/backup/gbasehost01_0_L0. Press Return to continue ...


Archive Tape Information

Tape type:      Archive Backup Tape
Online version: GBase Database Server Version 12.10.FC4G1AEE
Archive date:   Tue Jul 28 10:04:40 2020
```

```
User id:        gbasedbt
Terminal id:    /dev/pts/1
Archive level:  0
Tape device:    /opt/gbase/backup/
Tape blocksize (in k): 32
Tape size (in k): system defined for directory
Tape number in series: 1

Spaces to restore:1 [rootdbs                              ]
2 [plogdbs                 ]
3 [llogdbs                  ]
4 [datadbs01               ]
5 [sbspace01               ]

Archive Information

GBase Database Server Copyright 2001, 2018  General Data Corporation
Initialization Time       07/27/2020 14:57:35
System Page Size          2048
Version                   30
Index Page Logging        OFF
Archive CheckPoint Time   07/28/2020 10:04:40

Dbspaces
number   flags    fchunk   nchunks  flags    owner                          name
1        60001    1        1        N  BA    gbasedbt                       rootdbs
2        40001    2        1        N  BA    gbasedbt                       plogdbs
3        40001    3        1        N  BA    gbasedbt                       llogdbs
4        42001    4        1        N TBA    gbasedbt                       tempdbs01
5        68001    5        1        N SBA    gbasedbt                       sbspace01
6        60001    6        1        N  BA    gbasedbt                       datadbs01


Chunks
chk/dbs offset   size     free     bpages   flags pathname
1   1   0        512000   498315            PO-B- /data/gbase/rootchk
2   2   0        512000   11947             PO-B- /data/gbase/plogchk
3   3   0        512000   11947             PO-B- /data/gbase/llogchk
4   4   0        512000   511512            PO-B- /data/gbase/tempchk01
5   5   0        512000   25659             POSB- /data/gbase/sbspace01
6   6   0        512000   511576            PO-B- /data/gbase/datachk01

Continue restore? (y/n)y
Do you want to back up the logs? (y/n)y
```

```
Would you like to back up log 7? (y/n) y
Read/Write End Of Medium enabled: blocks = 2

Please label this tape as number 1 in the log tape sequence.

This tape contains the following logical logs:
    7
Log salvage is complete, continuing restore of archive.
Restore a level 1 archive (y/n) n
Do you want to restore log tapes? (y/n)n
/opt/gbase/bin/onmode -sy

Program over.
[gbasedbt@node2 ~]$ onmode -m
```

4.6.3. onbar

要使用 onbar 进行数据库备份，需要配置存储管理软件，数据库默认使用自带的 PSM 存储管理，默认备份目录为$GBASEDBTDIR/backups，使用 onpsm 命令检查备份目录：

```
onpsm -D list
```

DBSPOOL 为数据备份目录，LOGPOOL 为日志备份目录。

以下示例将例举如何使用 onbar 对数据库进行备份与恢复。

示例 1：对数据库实例执行 0 级备份，备份所有数据库空间及逻辑日志

```
onbar -b -w -L 0
```

示例 2：恢复数据库实例，包括物理恢复及逻辑恢复

```
onbar -r
```

使用 onbar -r 恢复过程中无需交互，等待恢复完成即可，可使用以下命令检查恢复是否在正常执行：

```
onstat -D -r 1
```

对比两次输出“page Wr”列是否变化，可确认数据库是否正在恢复写入数据。

onbar 如出现异常或报错，使用 onbar –m 检查日志是否有异常，默认日志位置为$GBASEDBTDIR/tmp/bar_act.log。

恢复完成后，使用 onstat -检查数据库状态，数据库处于 Quiescent 模式，使用 onmode -m 切换到 online

```
onmode -m
```

4.6.4. 逻辑日志自动备份

对于生产系统，建议配置逻辑日志的自动备份，避免逻辑日志未及时备份导致数据库暂挂。可通过对 ALARMPROGRAM 进行配置来自动备份逻辑日志，修改该配置参数值为:

参数	值
ALARMPROGRAM	$GBASEDBTDIR/etc/log_full.sh

如系统使用的是 onbar 备份，无需修改$GBASEDBTDIR/etc/log_full.sh 文件，如使用 ontape 备份，需要修改$GBASEDBTDIR/etc/log_full.sh 文件的以下行：

onbar 默认使用行	ontape 修改为
BACKUP_CMD="onbar -b -l"	BACKUP_CMD="ontape -a -d"

4.7. 紧急故障处理

4.7.1. 数据库无法启动

使用 oninit -v 启动数据库后，正常输出显示为：

```
Verbose output complete: mode = 5
```

如与以上输出不一致，参照以下步骤处理

1、确认当前实例是否正确：

```
env |grep GBASEDBTSERVER
```

2、实例确认无误后，检查使用 onstat -检查数据库实例状态，是否已处于运行状态：

```
onstat -
```

实例关闭状态下，onstat -正常输出显示为：

```
shared memory not initialized for GBASEDBTSERVER '$GBASEDBTSERVER'
```

3、确认实例正确且处于关闭状态后，使用 onstat -m 查看数据库日志错误：

```
onstat -m
```

确认日志中是否出现以下错误：

```
00:24:48  shmget: [EEXIST][17]: key 52564801: shared memory already exists
```

数据库无法启动最常见的原因是数据库未正常关闭，导致共享内存段未正常释放，即以上错误，出现此种情况，参照以下流程处理：

使用 ps 命令检查是否还存在 oninit 进程：

```
ps -ef |grep oninit
```

单实例环境下，使用 kill 杀掉 oninit 进程：

```
kill -9 <pid>
```

多实例环境下，可能还会有别的实例的进程在运行，可切换到其他实例执行 onstat -g glo 命令检查其他实例的进程号，排除这些进程后是否还存在 oninit 进程，如存在，使用 kill 命令杀掉当前实例的 oninit 进程。之后再启动数据库。如仍出现上面同样的错误，使用系统命令 ipcs -m 检查共享内存段：

```
ipcs -m
```

如存在共享内存段，单实例环境下使用 ipcrm 删除该内存段（root 执行）：

```
ipcrm -m <shmid>
```

多实例环境下，ipcs -m 的输出可能会包含其他实例的内存段，切换到其他实例使用 onstat -g seg 检查其他实例的内存段 id，与 ipcs -m 输出对比排除后，检查是否还多出内存段，如存在，使用 ipcrm 删除。

如 ipcrm 无法删除内存段，检查是否存在 dbaccess 进程，如存在 kill，如不存在，重启主机后启动数据库。

4、如未出现 3 中描述的错误，检查 online.log 中的错误信息，根据提示处理或联系 GBASE 支持。

4.7.2. 数据库无法关闭

如使用 onmode -ky 数据库长时间无响应，参照以下流程处理：

1、判断是否在执行检查点

```
onstat -R |grep dirty |grep -v start
```

多次执行以上命令，查看 dirty 数据是否在减少，如果是，表示数据库在执行检查点，等待完成即可。

2、如以上输出的 dirty 数据无变化，使用以下命令强制关闭数据库：

```
onclean -ky
```

3、如以上命令仍无响应，kill 掉 oninit 进程

```
kill -9 <pid>
```

数据库实例有多个 oninit 进程，任意 kill 一个即可。如果是多实例环境，切换到其他实例

执行 onstat -g glo 命令，查看 pid 输出，kill 这些 pid 之外的进程，避免误杀。

4.7.3. 数据库异常宕机

操作系统故障，存储故障或服务器掉电等因素都可能导致数据库异常宕机，出现此类紧急情况，建议按以下步骤处理：

1、确认当前实例：

```
env |grep GBASEDBTSERVER
```

2、确认实例无误的情况下，使用 onstat -命令检查当前实例状态：

```
onstat -
```

如出现以下输出，则表示数据库实例已宕机或已关闭：

```
shared memory not initialized for GBASEDBTSERVER '$GBASEDBTSERVER'
```

3、使用 oninit -v 命令启动数据库：

```
oninit -v
```

如数据库无法正常启动，参考数据库无法启动章节处理。

4.7.4. 数据库暂挂无响应

逻辑日志满未备份、文件系统满等可能会导致数据库处于暂挂状态，用户 sql 或其他操作无响应，出现此类情况，建议按以下步骤处理：

1、确认当前实例：

```
env |grep GBASEDBTSERVER
```

2、确认实例无误的情况下，使用 onstat -命令检查当前实例状态：

```
onstat -
```

检查输出是否在第二行出现 Blocked 字样，并根据输出提示 Blocked 的原因做下一步操作。

Blocked 输出	动作
Blocked: LAST_LOG_RESERVED4BACKUP	如系统使用 onbar 进行备份，使用 onbar -b -l 命令备份逻辑日志，如系统使用 ontape 备份，使用 ontape -a 备份逻辑日志
Blocked: LONGTX	等待长事务回滚完成，无需干预，不要做重启操作
Blocked: CKPT	等待检查点执行完成
Blocked: OVERRIDE_DOWN_SPACE	参照 5.3 章节 chunk down 处理

如第二行出现 Blocked 字样，根据以下步骤操作：

3、检查数据库安装目录所挂载的磁盘空间是否已满：

```
df -h
```

如是磁盘空间已满或操作系统其他原因导致数据库挂起，修复系统错误后重启数据库：

```
onmode -ky
oninit -v
```

4、如以上情况均未出现，再次确认数据库是否已暂挂或是误判，执行 onmode -c 检查点，看是否能正常完成，如能正常完成，则表示数据库工作正常未暂挂。

```
onmode -c
```

4.7.5. chunk down 故障

使用 onstat -命令出现以下输出，表示该空间已处于 offline 状态，已无法正常工作：

```
Blocked: OVERRIDE_DOWN_SPACE
```

出现此类情况，参考以下步骤处理：

1、确认 offline 的空间 chunk：

```
onstat -d |grep PD
```

2、如该 chunk 属于临时空间，将临时空间删除重建，如不是临时空间，使用 dd 读处于 offline 状态的 chunk：

```
dd if= /data/gbase/datachk01 of=/dev/null count=100
```

3、如 dd 不能正常读取该 chunk，表示该数据文件已丢失，关闭数据库使用 ontape 或 onbar 对全库恢复，如 dd 能正常读取该 chunk，参照以下语法将该 chunk 拉起：

```
onspaces -s datadbs01 -p /data/gbase/datachk01 -o 0 -0
```

4、如 onspaces 命令无法将该 chunk 切换到 online，关闭数据库使用 ontape 或 onbar 对全库恢复。

5. 集群运维管理

5.1. SSC 共享磁盘集群

5.1.1. 配置 SSC 共享磁盘集群（无独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	/	/	
3	服务实例名	gbase01	gbase02	
4	心跳实例名	/	/	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	
6	共享磁盘	存储映射到两台服务器，直接使用盘或分区作为数据库数据设备，或绑定裸设备后作为数据库数据设备		

在满足以上前提条件下，参考以下步骤配置 SSC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_TEMPDBS.*#SDS_TEMPDBS sdstmpdbs1,/opt/gbase/dbs/sdstmpdbs1,16,0,20480000#g"	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_ENABLE.*#SDS_ENABLE 1#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i	ssc 配置参数与主节点保持一致，只修改实例名相关参数，SDS_TEMPDBS，SDS_PAG

		$GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_PAGING.*#SDS_PAGING /opt/gbase/dbs/page1,/opt/gbase/dbs/page2#g" $GBASEDBTDIR/etc/$ONCONFIG	"s#^SDS_TEMPDBS.*#SDS_TEMPDBS sdstmpdbs2,/opt/gbase/dbs/sdstmpdbs2,16,0,20480000#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_PAGING.*#SDS_PAGING /opt/gbase/dbs/page3,/opt/gbase/dbs/page4#g" $GBASEDBTDIR/etc/$ONCONFIG	ING 指定的路径需要在磁盘上有足够的空间
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		重启主节点使配置生效
7	gbasedbt	onmode -d set SDS primary gbase01		设置主节点
8	gbasedbt		oninit -v	设置备节点
9	gbasedbt	onstat -g cluster		检查集群状态，显示 SDS connected active 则正常

5.1.2. 配置 SSC 共享磁盘集群（独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	192.168.1.1 hosthapri	192.168.1.2 hosthasec	
3	服务实例名	gbase01	gbase02	

4	心跳实例名	ha_pri	ha_ssc	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	
6	共享磁盘	存储映射到两台服务器，直接使用盘或分区作为数据库数据设备，或绑定裸设备后作为数据库数据设备		

在满足以上前提条件下，参考以下步骤配置 SSC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt hosthasec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt hosthapri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_TEMPDBS.*#SDS_TEMPDBS sdstmpdbs1,/opt/gbase/dbs/sdstmpdbs1,16,0,20480000#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_PAGING.*#SDS_PAGING /opt/gbase/dbs/page1,/opt/gbase/dbs/page2#g" $GBASEDBTDIR/etc/$ONCONFIG	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_ssc#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_ssc#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_ENABLE.*#SDS_ENABLE 1#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^SDS_TEMPDBS.*#SDS_TEMPDBS sdstmpdbs2,/opt/gbase/dbs/sdstmpdbs2,16,0,20480000#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i	ssc 配置参数与主节点保持一致，只修改实例名相关参数，SDS_TEMPDBS，SDS_PAGING 指定的路径需要在磁盘上有足够的空间

			"s#^SDS_PAGING.*#SDS_PAGING /opt/gbase/dbs/page3,/opt/gbase/dbs/page4#g" $GBASEDBTDIR/etc/$ONCONFIG	
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_ssc onsoctcp 192.168.1.2 9099 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_ssc onsoctcp 192.168.1.2 9099 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		重启主节点使配置生效
7	gbasedbt	onmode -d set SDS primary ha_pri		设置主节点
8	gbasedbt		oninit -v	设置备节点
9	gbasedbt	onstat -g cluster		检查集群状态，显示 SDS connected active 则正常

5.1.3. SSC 集群切换（手动）

按以下顺序执行 SSC 集群主备机之间的切换：

序号	切换流程	主节点(gbase01)状态或操作	SSC（gbase02）状态或操作	备注
1	切换前状态	On-Line	Read-Only (SDS)	onstat -输出
2	**切换操作**	**onmode -ky**		**gbasedbt 用户执行，关闭数据库**
3	**切换操作**		**onmode -d standard**	**gbasedbt 用户执**

				行，备节点切换为标准
4	**切换操作**		**onmode -d set SDS primary ha_ssc**	**gbasedbt 用户执行，备节点设置为主节点，如无心跳实例，ha_ssc 替换为备机服务实例名 gbase02**
5	**切换操作**	**oninit -v**		**gbasedbt 用户执行，启动原主节点加入为备节点**
6	集群状态		onstat -g cluster	gbasedbt 用户执行，显示 SDS connected active 则集群正常
7	**回切操作**		**onmode -ky**	**gbasedbt 用户执行，关闭数据库**
8	**回切操作**	**onmode -d standard**		**gbasedbt 用户执行，备节点切换为标准**
9	**回切操作**	**onmode -d set SDS primary ha_pri**		**gbasedbt 用户执行，备节点设置为主节点，如无心跳实例，ha_pri 替换为备机服务实例名 gbase01**
10	**回切操作**		**oninit -v**	**gbasedbt 用户执行，启动原主节点加入为备节点**
11	回切操作	onstat -g cluster		gbasedbt 用户执行，显示 SDS connected active

5.2. HAC 同城主备集群

5.2.1. 配置 HAC 主备集群（无独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	/	/	
3	服务实例名	gbase01	gbase02	
4	心跳实例名	/	/	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	

在满足以上前提条件下，参考以下步骤配置 HAC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g"	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g"	hac 配置参数与主节点保持一致，只修改实例名相关参数
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		如相关配置有修改，重启主节点使配置生效

6	gbasedbt	ontape -s -L 0 -t STDIO -F |ssh hostsec ". /home/gbase/.bash_profile;ontape -p -t STDIO"		恢复数据到备机
7	gbasedbt	onmode -d primary gbase02		设置主节点
8	gbasedbt		onmode -d secondary gbase01	设置备节点
9	gbasedbt	onstat -g dri		检查集群及备机状态，State 显示 on 则正常

5.2.2. 配置 HAC 主备集群（独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	192.168.1.1 hosthapri	192.168.1.2 hosthasec	
3	服务实例名	gbase01	gbase02	
4	心跳实例名	ha_pri	ha_hac	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	

在满足以上前提条件下，参考以下步骤配置 HAC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt hosthasec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt hosthapri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g"	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g"	hac 配置参数与主节点保持一

		$GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG	$GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_hac#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_hac#g" $GBASEDBTDIR/etc/$ONCONFIG	致，只修改实例名相关参数
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_hac onsoctcp 192.168.1.2 9099 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_hac onsoctcp 192.168.1.2 9099 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		重启主节点使配置生效
6	gbasedbt	ontape -s -L 0 -t STDIO -F |ssh hostsec ". /home/gbase/.bash_profile;ontape -p -t STDIO"		恢复数据到备机
7	gbasedbt	onmode -d primary ha_hac		设置主节点
8	gbasedbt		onmode -d secondary ha_pri	设置备节点
9	gbasedbt	onstat -g dri		检查集群及备机状态，State 显示 on 则正常

5.2.3. HAC 主备集群切换（手动）

按以下顺序执行 HAC 集群主备机之间的切换：

序号	切换流程	主节点状态或操作	RHAC 状态或操作	备注

1	切换前状态	On-Line (Prim)	Read-Only (Sec)	onstat -输出
2	**切换操作**	**onmode -ky**		**gbasedbt 用户执行，关闭数据库**
3	**切换操作**		**onmode -d standard**	**gbasedbt 用户执行**
4	**切换操作**	**oninit -PHY -v**		**gbasedbt 用户执行**
5	**切换操作**		**onmode -d primary ha_pri**	**gbasedbt 用户执行，如无心跳网络，ha_pri 替换为主节点实例名**
6	**切换操作**	**onmode -d secondary ha_hac**		**gbasedbt 用户执行，如无心跳网络，ha_hac 替换为备节点实例名**
7	切换后状态	Read-Only (Sec)	On-Line(Prim)	onstat -输出
8	**回切操作**		**onmode -ky**	**gbasedbt 用户执行**
9	**回切操作**	**onmode -d standard**		**gbasedbt 用户执行**
10	**回切操作**	**onmode -d primary ha_hac**		**gbasedbt 用户执行，如无心跳网络，ha_hac 替换为备节点实例名**
11	**回切操作**		**oninit -PHY -v**	**gbasedbt 用户执行**
12	**回切操作**		**onmode -d secondary ha_pri**	**gbasedbt 用户执行，如无心跳网络，ha_pri 替换为主节点实例名**
13	切换后状态	On-Line(Prim)	Read-Only (Sec)	onstat -输出
14	**集群状态**	**onstat -g dri**		**State 显示 on 则正常**

5.3. RHAC 远程主备集群

5.3.1. 配置 RHAC 主备集群（无独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	/	/	
3	服务实例名	gbase01	gbase02	
4	心跳实例名	/	/	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	

在满足以上前提条件下，参考以下步骤配置 RHAC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g"	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g"	rhac 配置参数与主节点保持一致，只修改实例名相关参数
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		如相关配置有修改，重启主节点使配置生效
6	gbasedbt	ontape -s -L 0 -t STDIO -F |ssh hostsec ". /home/gbase/.bash_profile;ontape -p -t STDIO"		恢复数据到备机
7	gbasedbt	onmode -d add RSS gbase02		设置主节点

8	gbasedbt		onmode -d RSS gbase01	设置备节点
9	gbasedbt	onstat -g cluster		显示 RSS connected active 则集群正常

5.3.2. 配置 RHAC 主备集群（有独立心跳网络）

前置条件配置：

序号	参数	主节点	备节点	备注
1	服务 IP 主机名	10.10.10.1 hostpri	10.10.10.2 hostsec	
2	心跳 IP 主机名	192.168.1.1 hosthapri	192.168.1.2 hosthasec	
3	服务实例名	gbase01	gbase02	
4	心跳实例名	ha_pri	ha_rhac	
5	安装状态	已完成软件安装并初始化实例	已完成软件安装及实例配置，无需初始化，数据库空间使用数据文件已创建，与主节点保持一致	

在满足以上前提条件下，参考以下步骤配置 RHAC 集群：

执行步骤	执行用户	主节点（实例名 gbase01）	备节点（实例名 gbase02）	备注
1	root	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	vi /etc/hosts，增加两行： 10.10.10.1 hostpri 10.10.10.2 hostsec 192.168.1.1 hosthapri 192.168.1.2 hosthasec	配置机器名
2	root	vi /etc/hosts.equiv，增加一行： hostsec gbasedbt hosthasec gbasedbt	vi /etc/hosts.equiv，增加一行： hostpri gbasedbt hosthapri gbasedbt	配置信任
3	gbasedbt	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase01#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG	sed -i "s#^DBSERVERNAME.*#DBSERVERNAME gbase02#g" $GBASEDBTDIR/etc/$ONCONFIG sed -i "s#^DBSERVERALIASES.*#DBSERVERALIASES ha_rhac#g" $GBASEDBTDIR/etc/$ONCONFIG	rhac 配置参数与主节点保持一致，只修改实例名相关参数

		sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_pri#g" $GBASEDBTDIR/etc/$ONCONFIG	sed -i "s#^HA_ALIAS.*#HA_ALIAS ha_rhac#g" $GBASEDBTDIR/etc/$ONCONFIG	
4	gbasedbt	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_rhac onsoctcp 192.168.1.2 9099 EOF	cat >$GBASEDBTSQLHOSTS <<EOF gbase01 onsoctcp 10.10.10.1 9088 gbase02 onsoctcp 10.10.10.2 9088 ha_pri onsoctcp 192.168.1.1 9099 ha_rhac onsoctcp 192.168.1.2 9099 EOF	配置 sqlhosts 文件
5	gbasedbt	onmode -ky oninit -v		重启主节点使配置生效
6	gbasedbt	ontape -s -L 0 -t STDIO -F |ssh hostsec ". /home/gbase/.bash_profile;ontape -p -t STDIO"		恢复数据到备机
7	gbasedbt	onmode -d add RSS ha_rhac		设置主节点
8	gbasedbt		onmode -d RSS ha_pri	设置备节点
9	gbasedbt	onstat -g cluster		显示 RSS connected active 则集群正常

5.3.3. RHAC 主备机集群切换（手动）

按以下顺序执行 RHAC 集群主备机之间的切换：

序号	切换流程	主节点状态或操作	RHAC 状态或操作	备注
1	切换前状态	On-Line	Read-Only (RSS)	onstat -输出
2	**切换操作**	**onmode -ky**		**gbasedbt 用户执行，关闭数据库**
3	**切换操作**		**onmode -d standard**	**gbasedbt 用户执行**

4	**切换操作**	**oninit -PHY -v**		**gbasedbt 用户执行**
5	**切换操作**		**onmode -d add RSS ha_pri**	**gbasedbt 用户执行，如无心跳网络，ha_pri 替换为主节点实例名**
6	**切换操作**	**onmode -d RSS ha_rhac**		**gbasedbt 用户执行，如无心跳网络，ha_rhac 替换为备节点实例名**
7	切换后状态	Read-Only (RSS)	On-Line	onstat -输出
8	**回切操作**		**onmode -ky**	**gbasedbt 用户执行**
9	**回切操作**	**onmode -d standard**		**gbasedbt 用户执行**
10	**回切操作**	**onmode -d add RSS ha_rhac**		**gbasedbt 用户执行，如无心跳网络，ha_rhac 替换为备节点实例名**
11	**回切操作**		**oninit -PHY -v**	**gbasedbt 用户执行**
12	**回切操作**		**onmode -d RSS ha_pri**	**gbasedbt 用户执行，如无心跳网络，ha_pri 替换为主节点实例名**
13	切换后状态	On-Line	Read-Only (RSS)	onstat -输出
14	**集群状态**	**onstat -g cluster**		**显示 RSS connected active 则集群正常**

5.4. 连接管理器管理

一个用于管理客户端对 GBase 8s 进行连接访问的组件，连接管理器的设置决定客户端的连接顺序与方向；决定 GBase 8s 高可用集群中，主节点到 HAC 节点的故障切换(Failover)、主节点角色的转移到辅节点服务器、SSC 高可用集群中巨量客户端的负载均衡的功能；连接管理器设置的流向契约(SLA)，决定客户端在 GBase 8s 高可用集群中的连接顺序。

在集群的任意在线的节点上，均能查看连接到集群的连接管理器状态

```
onstat -g cmsm
```

输出的结果类似以下内容，有多个连接管理器，均显示。

```
Unified Connection Manager: CM1                    Hostname: localhost.localdomain
CLUSTER         grp_app LOCAL
        GBasedbt Servers: grp_app
```

```
        SLA  Connections    Service/Protocol    Rule
        oltp1        0      9098/onsoctcp    DBSERVERS=primary        WORKERS=16,USEALIASES=OFF,MODE=proxy
        olap1        0      9099/onsoctcp    DBSERVERS=SDS,PRI        WORKERS=16,USEALIASES=OFF,MODE=proxy

        Failover Arbitrator: Failover is disabled
        ORDER=SDS,HDR,RSS PRIORITY=100 TIMEOUT=1
```

查看连接管理器的进程 PID 和内存占用情况

```
more ${GBASEDBTDIR}/tmp/cmsm.pid.CM1
```

输出的结果类似：

```
60274
```

通过操作系统命令 ps 查看 oncmsm 进程的 PID 和内存占用信息

```
ps -aux | grep oncmsm | grep -v grep
```

输出的结果类似以下内容（第六列为内存占用，单位 KB）：

```
gbasedbt    60274    0.0    0.5 227160 22584 ?              Ssl    17:25      0:00 oncmsm
```

关闭连接管理器

```
oncmsm -k CM1
```

输出信息类似如下：

```
Shut down Connection Manager CM1
```

注意：仅能关闭本服务器上的连接管理器。

启动连接管理器，将默认读取$GBASEDBTDIR/etc/cmsm.cfg 中的配置信息

```
oncmsm
```

输出信息类似如下：

```
Connection Manager started successfully
Please check GBase Connection Manager log file: /opt/gbase/tmp/cm1.log
```

如果启动非默认配置的连接管理器配置文件，使用类似以下的方式。

```
oncmsm -c /opt/gbase/etc/cm1.cfg
```

6. 其他

6.1. finderr 查看错误信息

数据库返回错误号后，可使用 finderr 命令查看错误信息，根据提示解决错误问题：

```
finderr 951
```

6.2. 售后电话

```
400-013-9696
```

6.3. 技术支持社区

```
https://www.gbase.cn/community/section/10
```