# GBase 8s 数据库运维操作手册

南 大 通 用 数 据 技 术 股 份 有 限 公 司

**General Data Technologies Co., Ltd.**



**天津总公司：天津市高新区华苑产业园区工华道 2 号天百中心 3 层**

**电　　话：022-58815678**

**传　　真：022-58815679**

**北京分公司：北京市朝阳区安定路五号院 13 号北投投资大厦 A 座 606-608**

**电　　话：010-88866866**

**传　　真：010-88864556**

**http://www.gbase.cn**

**E-mail:info@gbase.cn**

**版权声明**



**免责声明**



**通讯方式**

南大通用数据技术股份有限公司

中国天津市高新区华苑产业园区工华道 2 号天百中心 3 层(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

**商标声明**

# 版本记录

| 日期 | 版本 | 变更说明 | 作者 | 审核 |
|---|---|---|---|---|
| 2022-09-17 | V1.00 | 新建 | 廖晋清 | |
| 2024-08-09 | V1.01 | 去除专用信息，改为通用信息<br>改为宽页版 | 廖晋清 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# 前言

## 概述

本文档描述了 GBase 8s 数据库运维人员日常操作手册。

## 读者对象

本文档主要适用于以下技术人员：

- 数据库管理员
- 数据库运维人员
- 数据库初学者

## 初学者注意事项

需要具备一定的 Linux 操作经验，了解常用的命令行含义。

# 目 录

# 1. 编写说明

本文档为南大通用 GBase 8s 数据库运维操作示例手册。

## 1.1. 文档说明

提示符说明：

[root@gbasehost01 ~]#

表示在节点 gbasehost01 上，使用 root 用户执行，当前目录是~（表示 HOME 目录）

[root@gbasehost01 software]#

表示在节点 gbasehost01 上，使用 root 用户执行，当前目录是 software

[gbasedbt@gbasehost01 ~]$

表示在节点 gbasehost01 上，使用 gbasedbt 用户执行，当前目录是~（表示 HOME 目录）

# 2. 实例检查

## 2.1. 基本信息

| 参数 | 值 | 补充说明 |
| --- | --- | --- |
| 实例 IP | 生产库：<br>集群 1：192.168.0.101 和 192.168.0.102 | |
| 实例名 | 生产库：<br>集群 1：gbase01 和 gbase02 | |
| 环境变量文件 | 生产库：<br>/home/gbase/.bash_profile | |
| 配置文件 | 生产库：<br>集群 1：onconfig.gbase01 和 onconfig.gbase02 | |
| SQLHOST 文件 | 生产库：<br>集群 1：$GBASEDBTDIR/etc/sqlhosts | |
| 审计日志目录 | 生产库：<br>/home/dbaao/aaodir | |
| 默认字符集 | zh_CN.utf8 | |
| 数据库个数 | N | |
| 数据库名 | 多个 | |
| 数据库日志模式 | unbuffered | |

## 2.2. 用户登录检查

使用 xshell 等 ssh 客户端工具，通过 ssh 远程登录到 192.168.0.X：

```
ssh 192.168.0.X
```

用户名：gbasedbt，密码：

## 2.3. 检查实例状态

使用 onstat -命令检查当前实例状态：

```
onstat -
```

如显示状态为 On-Line（主节点）或者 Read-Only（SSC 节点或者 HAC 节点），表示数据库实例运行正常，具体输出解释参考上文中 onstat -的解释。

## 2.4. 检查数据库连接数

使用 onstat -g ntt 命令检查数据库的网络连接情况

```
onstat -g ntt | grep sqlexec | wc -l
```

如当前连接数量过多，需要关注。

## 2.5. 检查集群状态

使用 onstat -g cluster 命令检查当前集群状态：

```
onstat -g cluster
```

如 Primary Server 的服务名称为本机，最下部分的服务名称为集群的其他节点的状态是 Connected,Active，表示整个集群正常。如 Primary Server 的服务名称为集群主节点，最下部分的服务名称为本机，且状态是 Connected,Active，表示本机与主节点间的集群状态状态。

## 2.6. 检查连接管理器状态

使用 onstat -g cmsm 命令检查连接到本集群的连接管理器状态：

```
onstat -g cmsm
```

如果显示的连接管理器数量与配置的一致，表明正常。

使用 ps -aux | grep oninit | grep -v grep 命令检查连接管理器的内存占用情况

```
ps -aux | grep oninit | grep -v grep
```

第六列为进程的物理内存占用（单位 KB）。

## 2.7. 检查空间状态

使用 onstat -d 命令检查当前实例是否存在不可用空间：

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

无输出表示正常，有输出联系 GBASE 售后支持。

## 2.8. 检查空间使用情况

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：

```
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), 'p')  as DBS_SIZE ,
  format_units(sum(decode(mdsize,-1,nfree,udfree)), 'p')  as free_size,
  TRUNC(100-sum(decode(mdsize,-1,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;
```

plogdbs 及 llogdbs 的使用率会很高，可忽略，其他空间使用率如超过 90%，需要及时扩容，关于如何扩容，可参照空间管理章节内容。

## 2.9. 检查死锁情况

使用 onstat -p 命令检查存在死锁较多的情况：

```
onstat -p
```

检查 deadlks 输出是否较大，正常情况下，该数字应小于 onstat -输出的数据库运行天数。

## 2.10. 检查数据库日志输出

使用以下命令检查数据库日志近期是否存在错误：

```
onstat -c |grep '^MSG' |awk '{print "tail -10000 " $2}' |sh |egrep -i '(assert|err|fail)' | egrep -v '(Estimate|admin)'
```

如存在错误，使用 onstat -m 命令查看输出第二行，使用 more 或 vi 命令检查该文件中的具体错误信息提示，根据提示操作或联系 GBASE 支持。

## 2.11. 检查文件系统及数据库备份目录

使用 df -h 命令检查操作系统是否存在空间满的情况：

```
df -h
```

重点检查数据库安装目录/opt/gbase 文件系统是否已满，如已满，检查该目录下是否存在用户存放的可删除的文件并删除。

## 2.12. 检查数据库备份和如何进行备份

配置数据库的自动备份，备份的目录为共享目录，使用定时任务调用脚本执行备份。

查看数据库备份（仅需在主节点执行），检查 0 级（全量），1 级（增量）备份的时间

```
onstat -g arc
```

备份日志为

```
more /opt/gbase/script/backup_status.log
```

如需对数据库进行手工备份，参考以下步骤：

修改配置参数 TAPEDEV 为存储备份数据的目录，如备份数据到/data/backup 目录：

```
onmode -wf TAPEDEV=/data/backup
```

执行 0 级备份：

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

注意：/data/backup 目录需要有足够的空间（无论是本地的文件系统还是远程的共享文件系统）

注意：执行数据库备份仅需要在集群中主节点上执行。

## 2.13. 如何执行恢复

要恢复数据库，先把数据关闭，之后执行 ontape -r 恢复数据库

关闭数据库：

```
onmode -ky
```

执行 ontape -r 恢复数据库：

```
ontape -r
```

关于 ontape -r 交互输入，可参照备份与恢复章节中的 ontape 恢复操作。

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

```
onmode -m
```

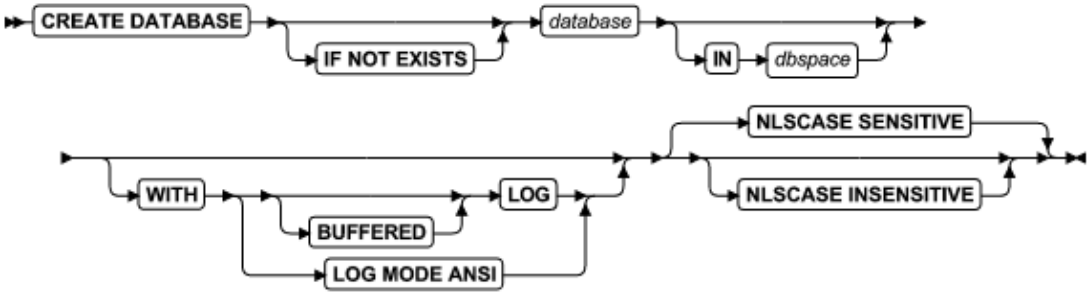
注意：如果需要恢复集群，需先恢复主节点，然后在重建集群。

# 3. 常用运维操作

## 3.1. 数据库管理

### 3.1.1. 创建默认字符集数据库

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 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;
!
```

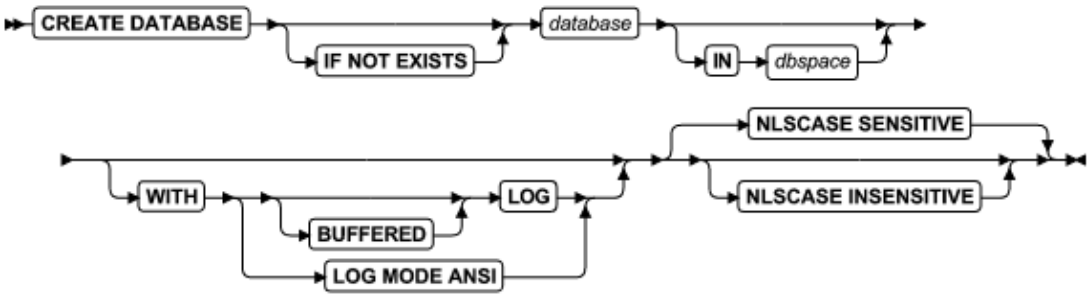
示例 2：创建名为 testdb 数据库，数据库采用 unbuffered 日志模式，数据存储于数据库空间 datadbs01，如未配置 DB_LOCALE 环境变量，使用默认字符集 ISO8859-1：

```
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。

### 3.1.2. 创建 GB18030 字符集数据库

要创建数据库，请参考以下语法：

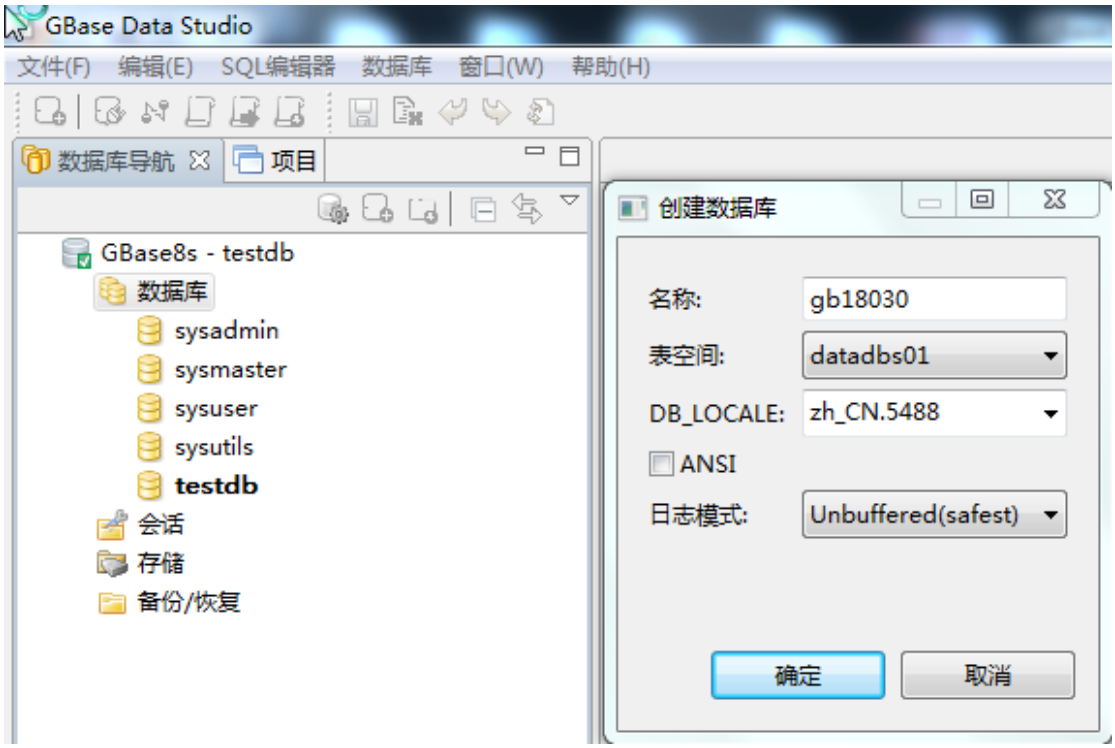



| 参数选项 | 参数说明 |
|---|---|
| database | 需要创建的数据库名称 |
| dbspace | 指定数据库数据存放的数据库空间，缺省为根数据库空间 rootdbs |

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;
!
```

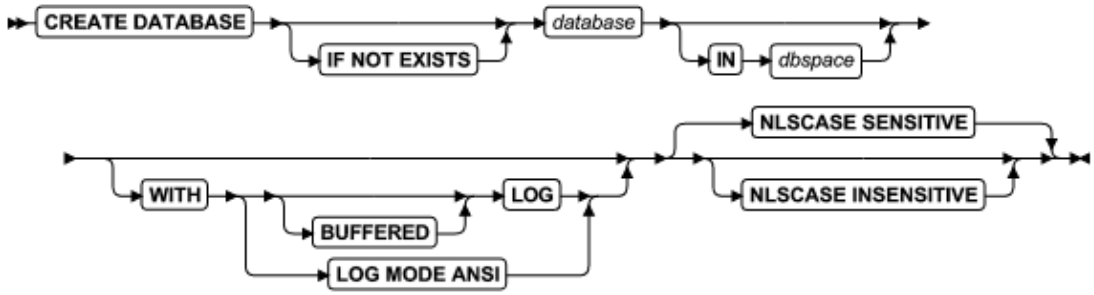
或者在 GBase DataStudio 中在已经连接的会话上，在 数据库 右键 新建 数据库，指定 名称、数据库空间（表空间）、字符集（DB_LOCALE）为 zh_CN.5488 或者 zh_CN.GB18030-2000，日志模式为 Unbuffered(safest)。




注意：生产环境建议数据库采用 unbuffered 日志模式。

### 3.1.3. 创建 UTF8 字符集数据库

要创建数据库，请参考以下语法：

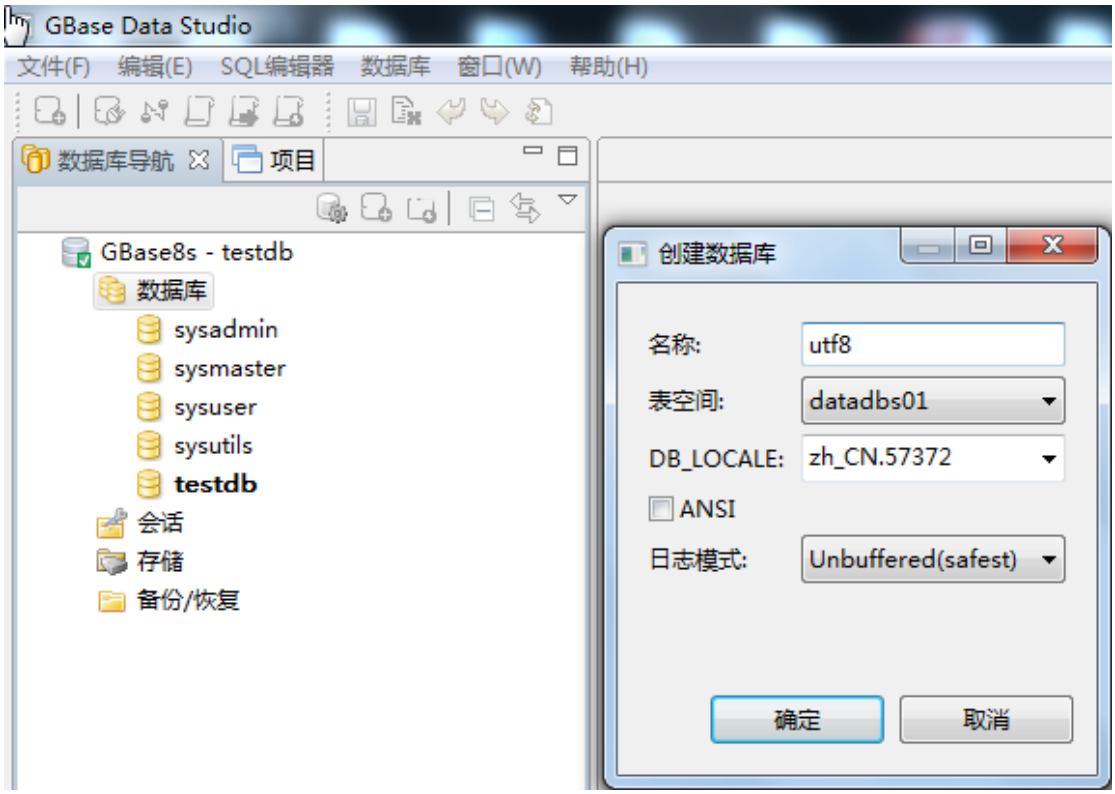



| 参数选项 | 参数说明 |
| --- | --- |
| database | 需要创建的数据库名称 |
| dbspace | 指定数据库数据存放的数据库空间，缺省为根数据库空间 rootdbs |

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 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。

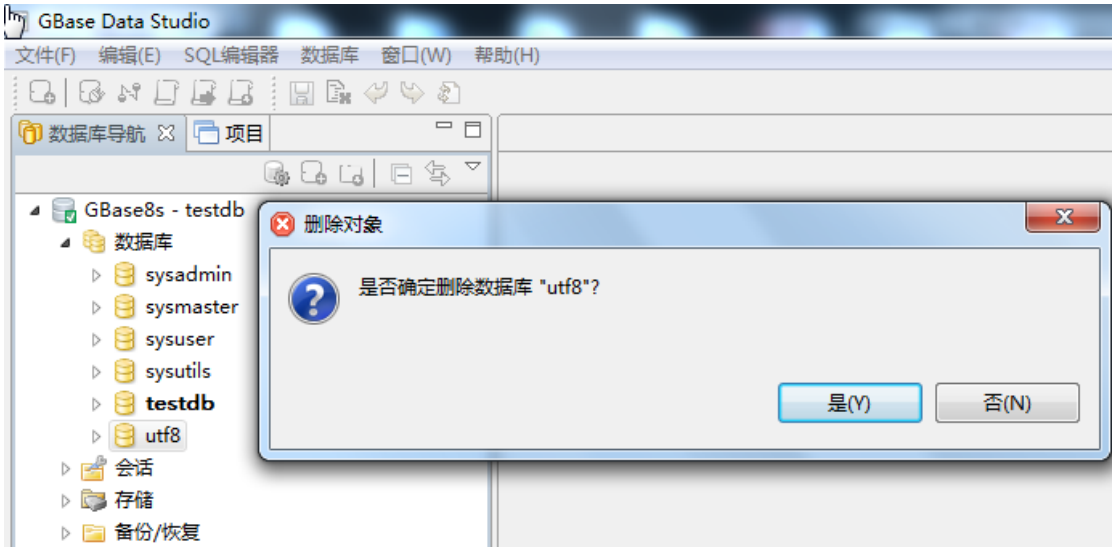
### 3.1.4. 删除数据库

要删除数据库，请参考以下语法：


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。

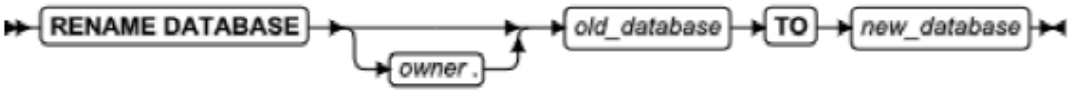



注意：无法删除当前有连接的活动数据库。

注意：需要 DBSA 权限的用户，或者库的所有者才有权删除库。

## 3.1.5. 重命名数据库

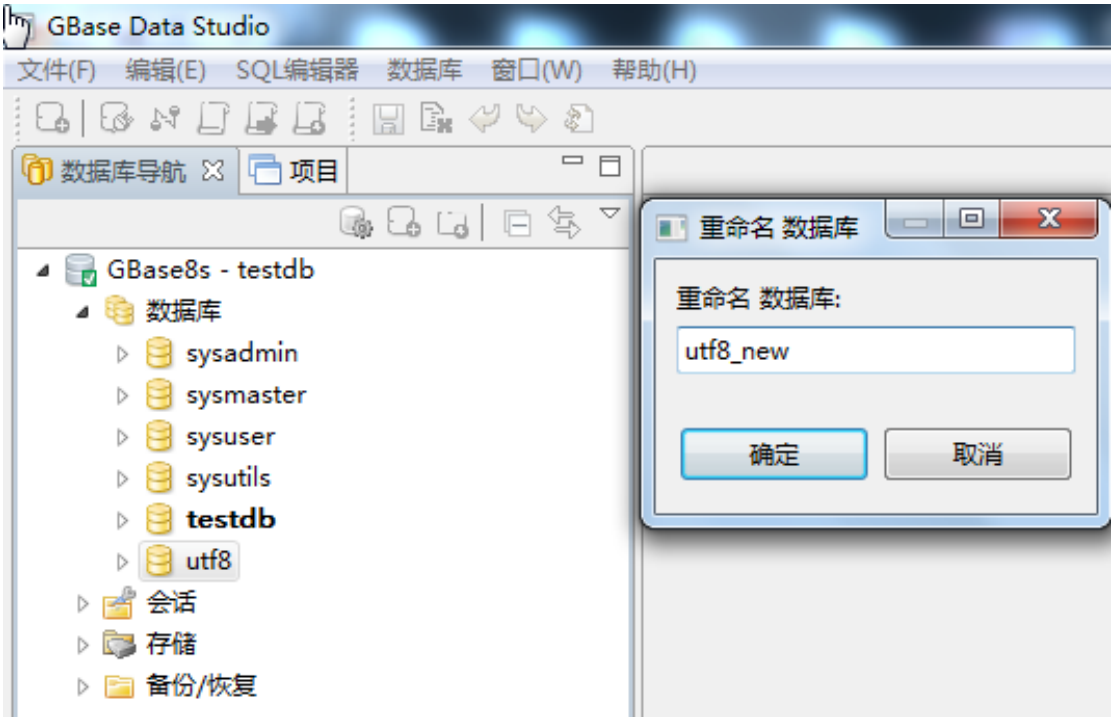
要重命名数据库，请参考以下语法：




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。

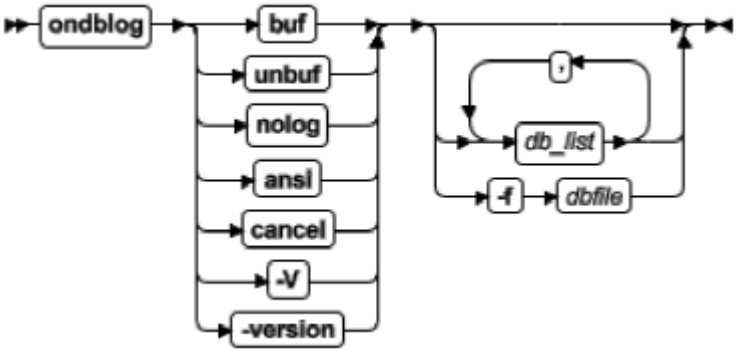



注意：无法重命名当前有连接的活动数据库。

注意：需要 DBSA 权限的用户，或者库的所有者才有权删除库。

## 3.1.6. 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
```

## 3.1.7. 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
```

## 3.1.8. 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;
!
```

## 3.2. 用户和权限管理

### 3.2.1. 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）：

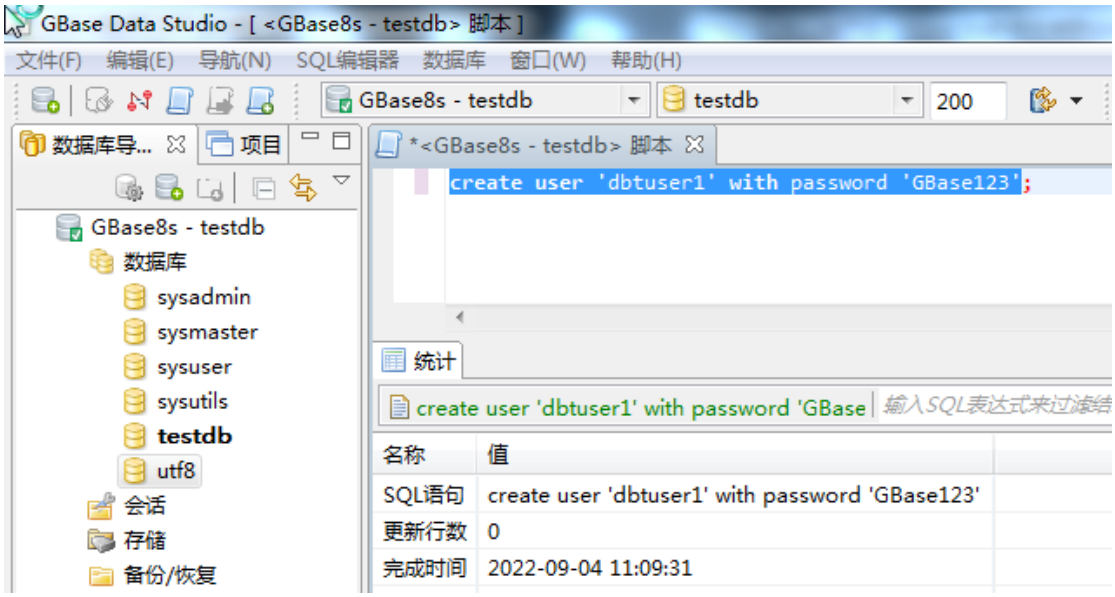
```
echo "grant connect to user1" | dbaccess testdb -
```

**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：

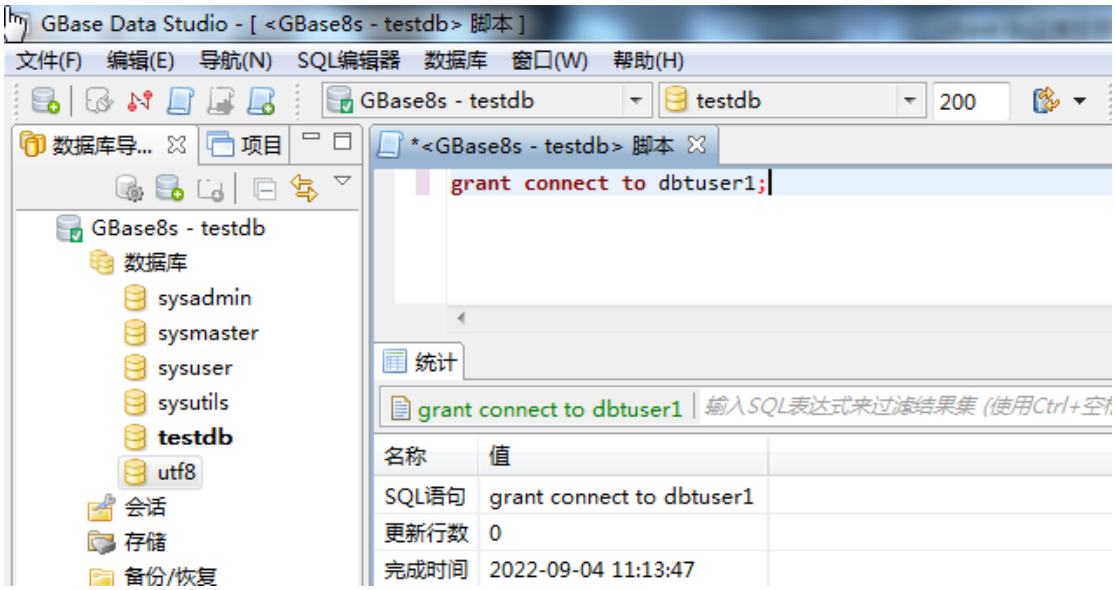
在 GBase DataStudio 中，gbasedbt 用户连接，执行 SQL 语句：

```
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。

## 3.2.2. 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。

## 3.2.3. 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<br>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：<br>• 使用 connect 语句或另外的语句连接至数据库<br>• 执行 SELECT、INSERT、UPDATE 和 DELETE 语句，如果用户具有必要的表级特权<br>• 创建视图，如果用户对底层表有 selelct 特权<br>• 创建同义词<br>• 创建临时表以及创建临时表的索引<br>• 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）<br>• 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：<br>• 创建新表<br>• 创建新索引<br>• 创建新 UDR<br>• 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：<br>• 将任何数据库特权（包括 DBA 特权）授予其他用户<br>• 将任何表级特权授予其他用户或角色<br>• 将角色授予用户或其他角色<br>• 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。<br>• 当注册 UDR 时，限制对 DBA 的 Execute 特权<br>• 执行 SET SESSION AUTHORIZATION 语句<br>• 创建任何数据库对象<br>• 创建表、视图和索引，指定其他用户作为这些对象的所有者<br>• 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<br>• 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;
```

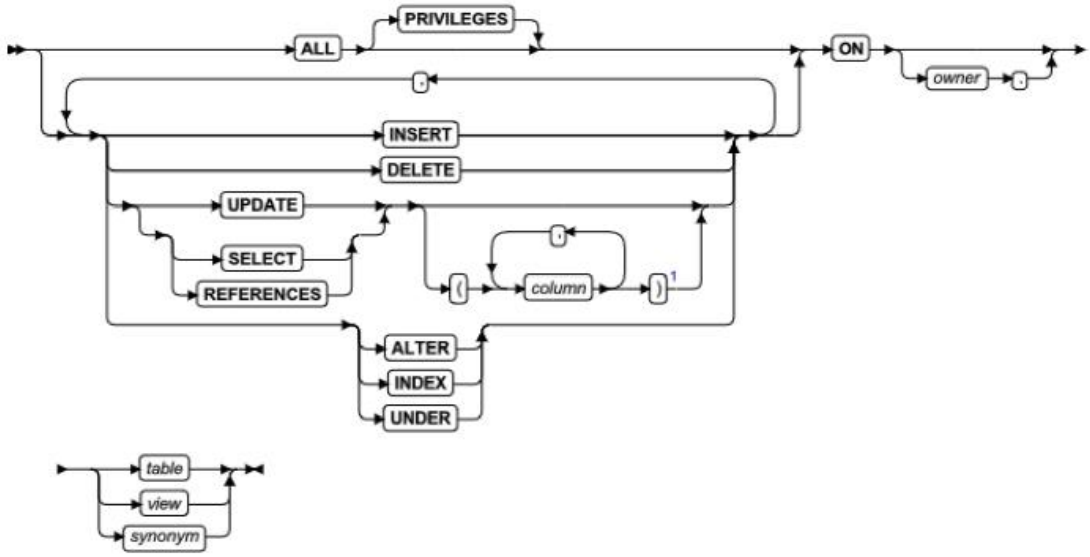
不能将数据库级特权授予角色。

**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;
```

# 3.3. 空间管理

## 3.3.1. 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。

## 3.3.2. 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。

### 3.3.3. 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。

## 3.4. 数据迁移

### 3.4.1. 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。

注意：如果使用了分片表，且创建绑定空间的函数，dbimport 前需要修改 *dbname*.exp/*dbname*.sql 文件内容，将 create function 内容移动到 create table 前。

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
```

## 3.4.2. 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|
......
```

## 3.4.3. 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
```

### 3.4.4. 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
```

### 3.4.5. 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。

注意：onunload 和 onload 不支持大对象和智能大对象。

# 4. 其他

## 4.1. 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
```

## 4.2. 售后电话

```
400-013-9696
```

## 4.3. 技术支持社区

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