# GBase 8s数据库操作指南

## 一、数据库日常管理指南

**简介：**

GBase 8s数据库日常管理包括数据库模式的切换，以及服务器的监控。监控主要分为两个方面，一方面是系统级别的监控，主要包括对系统cpu的监控，以及内存的占用情况；另一方面是数据库级别的监控，包括对磁盘空间大小的监控，对逻辑日志物理日志的监控，对数据表中数据量的监控，以及表中索引层数，顺序扫描次数等等。我们可以通过命令或者sql语句的方式，完成对数据库的监控，也可以通过图形化管理界面实现对GBase 8s数据库的监控。

**语法：**

启动数据库服务：

oninit -vy

关闭数据库服务:

onmode -ky

监控系统cpu和内存使用情况：

top

查看数据库版本和运行模式：

onstat -

监控数据库磁盘空间大小

onstat -d

监控逻辑日志，物理日志信息

onstat -l

监控GBase 8s占用系统内存大小：

onstat -g seg

监控实例整体运行情况

onstat -p

查看实例运行日志最后20行信息

onstat -m

查看数据库进程状态

onstat -g glo

查看数据库会话状态

onstat -g ses

从离线模式向静态模式转换：

oninit -s

从离线模式向管理员模式转换：

oninit -j

从管理员模式，静态模式向在线模式转换：

onmode -m

从在线模式向管理员模式转换:

onmode -j

从在线模式向静态模式转换:

onmode -s; onmode -u

## 二、SQL指南

**语法简介：**

创建数据库对象的语法都是使用 CREATE + 数据库对象类型 + 数据库对象名 的方式开 头，然后再定义其中具体的细节。这类语法属于DDL（Data Definition Language）数据库定义语言。

在GBase 8s中，有四种日志类型的数据库，分别是无日志模式，缓冲式日志模式，无缓冲式日志模式和ANSI模式。

无日志模式，数据库性能好，但是不支持逻辑恢复操作；

缓冲式日志模式，既支持数据库逻辑恢复的功能，同时又能避免频繁的I/O操作，是生产系统中常见的日志类型；

无缓冲式日志模式，会有频繁的I/O操作，优点是一旦数据库宕机，损失较小。

ANSI模式，是一种特殊的无缓冲式日志模式，要符合额外的ANSI规则。

创建库语法

create database 据库名 in dbspace名

[ with buffered log | with log | with log mode ansi ];

创建表语法

create [ temp | row ] table 表名(

列名 数据类型 列级约束,

... ... ,

[ 表级约束 ]

) [ in 表的存放路径 ];

修改表语法

alter table 表名 add 列名 数据类型 | modify 列名 数据类型 | drop column 列名;

删除表语法

drop table [ if exists ] 表名;

查询数据语法

select [ all | distinct ] 列表达式, ......

from 表名, ......

[ where 条件表达式 ]

[ group by 列名 | 选择列序号 , ...... [ having 条件表达式 ] ]

[ order by 列名 | 选择列序号 [ asc | desc ], ...... ] ;

插入数据语法

insert into 表名 [ ( 列1, 列2, 列3, ...... ) ] values ( 值1, 值2 , 值3 , ...... );

更新数据语法

update 表名 set 列1=值1, ...... [ where 条件表达式 ];

删除数据语法

delete from 表名 [ where 条件表达式 ];

**简单示例：**

创建缓冲式日志模式数据库testdb，test表，以及insertdata存储过程 ：

切换到gbasedbt用户，并生效当前实例对应的环境变量文件：

su - gbasedbt

env | grep GBASEDBTSERVER

sql编辑器dbaccess，执行SQL语句：

dbaccess - -

> create database testdb in datadbs1 with buffered log;

-- 创建库testdb

> create table test (id int , name char(20) );

-- 创建test表

> create procedure insertdata()

> define i int;

> for i in (1 to 100)

> insert into test values(i,'GBase 8s');

> end for;

> end procedure;

-- 创建insertdata()存储过程，向test表中插入100条数据

>execute procedure insertdata();

-- 执行insertdata()存储过程

>select a.cf\_original, DBINFO('dbname') dbname,user from

>sysmaster:sysconfig a where cf\_name = 'testdb'; -- 查看实例名,数据库名,用户名

> select name from sysmaster:sysdatabases; -- 查看实例中所有数据库名

> select dbsname,tabname from sysmaster:systabnames where

> dbsname='testdb'; -- 查看数据库中所有表名

## 三、数据装载与卸载指南

**语法简介：**

GBase 8s中根据装载卸载粒度的不同，有不同的装载卸载工具。

load/unload：装卸载粒度表级，输出结果为明文.

unload to 文件名 [DELIMITER 'delimiter'] select 语句;

load from 文件名 [DELIMITER 'delimiter'] insert into 表名 | 同义词名 | 视图 [(字段...)];

onunload/onload：装卸载粒度为数据库级或表级，输出结果为二进制文件。

onunload [-l] [-t <tape device>] <database>[:[<owner>.]<table>]

onload [-l] [-t <tape device>] [-d <DBspace>] <database>[:[<owner>.]<table>]

dbload：装载粒度为表级，功能比load方式强大，使用起来也相对复杂。

dbload [-d dbname] [-c cfilname] [-l logfile] [-e errnum] [-n nnum]

dbexport/dbimport：装卸载粒度为数据库级，输出结果为明文。

dbexport <database> [-ss] [{ -o <directory> | -t <tapedevice> }]

dbimport <database> [-d <dbspace>] [-l [{ buffered }] [-ansi]] [-ci] [-nv] [-D]

[{ -i <dir> | -t <tapedev> }]

**简单示例：**

（1）通过unload将test表数据卸载到test.unl文件，用竖线做列的分隔符

unload to /tmp/test.unl select \* from test;

通过load将test.unl中的数据装载到test表中

load from /tmp/test.unl insert into test;

（2）通过dbload工具将test.unl中的数据装载到test表中

先建立命令文件：例如/tmp/test.ctl 在文件中 写入以下命令

vi /tmp/test.ctl (添加如下两行内容)

FILE /tmp/test.unl delimiter "|" 2;

insert into test;

(/tmp/test.ctl 是数据文件，2代表两个字段)

然后用dbload命令导入数据：

dbload -d testdb -c /tmp/test.ctl -l error.log

（3）使用dbexport/dbimport迁移数据库

使用dbexport将testdb数据库导出 -ss生成服务器特定的语法

dbexport testdb -ss

删除testdb数据库

dbaccess - -

>drop database testdb;

使用dbimport重新导入testdb数据库,数据库创建在datadbs1数据库空间中。导入过 程使用缓冲日志模式

dbimport testdb -d datadbs1 -l buffered

## 四、备份与恢复操作指南

**语法简介：**

GBase 8s数据库中，备份与恢复分为三个级别，分别是0级备份，1级备份，以及2级备份。

0级备份指的是备份数据库中所有的数据。

1级备份是在0级备份基础上，备份被修改过的数据页。

2级备份是在1级备份基础上，备份被修改过的数据页。

主要的备份命令有两个，分别是ontape和onbar 。

其中，onbar命令功能强大，可以恢复到具体时间点，可以选择具体存储空间，可以并行执行备份操作，但是相对配置比较复杂。

ontape命令简单易用，虽然没有onbar功能强大，但是使用ontape命令在备份的同时，可以更改数据库日志模式，这也是比较实用的功能。

**ontape语法：**

ontape -a [-d] //备份逻辑日志

ontape -s [[-L archive\_level]] [-A database\_list] [-B database\_list]

[-N database\_list] [-U database\_list] [-t tape\_device\_path ]

//备份数据库

ontape -p //恢复数据库

**onbar 语法：**

onbar -b -l [-c | -C | -s] [-O] //备份逻辑日志

onbar -b [-L <level>] [-w | -f <filename> | <spaces>] [-O]

//备份数据库

onbar -r -l [ -t "<time>" | -n <log>] //恢复逻辑日志

onbar -r [-t "<time>" | -n <log>] [-f <filename> | <spaces>]

//恢复数据库

**简单示例：**

（1）使用ontape命令执行零级数据库备份

ontape -s -L 0

通过onstat -m命令查看日志中备份信息。

通过onstat -g arc命令查看数据库近期备份情况。

（2）使用ontape命令恢复数据库

关闭实例：

onmode -ky

执行ontape命令

ontape -r //从离线模式到静态模式

onmode -m //转换到在线模式