星瑞格数据库管理系统 SinoDB
数据库用户手册

SINOREGAL
星瑞格软件

福建星瑞格软件有限公司
www.sinoregal.cn

■文档编号	STD-DB-11-01	■版本编号	V 16.8
■文档密级	公开	■发布日期	2023-05-16

版本记录

版本号	版本日期	状态	修改人	审核人	修改记录
V1.0	20230512	C	林雅	张道山	创建文档

状态：C-创建文档，A-增加内容，M-修改内容，D-删除内容

目 录

1. 前言

本文档描述了 SinoDB 数据库的核心概念，如实例、表空间、日志等； SinoDB 开发规范及开发指南；运维常见问题；性能调整等，为 SinoDB 数据库安装维护人员及开发人员使用提供参考。

1.1 目的

说明：本文档描述如何搭建 SinoDB，如何配置及常见问题。

1.2 范围

SinoDB 安装维护人员、使用 SinoDB 的开发人员。

2. SinoDB 概述

2.1 SinoDB 是什么

SinoDB 是星瑞格具有自主知识产权的高端国产数据库管理系统产品。产品借鉴先进技术思想与国际主流数据库产品优点，融合了安全可靠、动态扩展、分布式、弹性计算与云计算的优势。产品成熟稳定、自主可控、功能强大、具备高性能、高可用、简单易用等特性，提供了两地三中心的高可用解决方案，实现集群负载、同城灾备、异地灾备等特性。SinoDB 兼容主流软硬件平台，尤其是主流国产操作系统、芯片、中间件；支持国密算法，提供多层次安全保障机制；具备高可靠性和水平扩展能力；具有成熟的服务体系，可提供 99.99%的高水准的服务等级（SLA），能确保用户业务的连续性，为用户提供安全可靠的基础服务支撑

2.2 SinoDB 的特性

SinoDB 支持 SQL 语言标准(SQL 92,99)，它可在 Client /Server 环境下联机事务处理的需要，也支持 Internet/Web 等应用，包括数字、字符串、日期、电子文档、图片、视频等多样化数据类型。

SinoDB 使用多线程机制重新改写数据库核心，引入了虚处理器的概念，每个虚处理器就是一个 SinoDB 数据库服务器进程。

SinoDB 可瘦身为嵌入式数据库，支持时间序列，具有跨多设备的横向扩展能力提供了一个占用资源少，功能完备的企业级数据库服务器。

SinoDB 支持 JSON/BSON 数据类型，兼容 MongoDB（NoSQL 数据库）。将 RDBMS 和 NoSQL 集中在一起，通过将两种不同类型的数据和两种不同的需求集中在一起，支撑新的业务模式。

SinoDB 推出了专门针对数据仓库应用的数据仓库加速器 SWA(SinoDB Warehouse Accelerator)。它将新的数据仓库技术和传统的 SinoDB 关系数据库服务器相结合，数据经过压缩、频度分区技术全部保存在内存中，消除了磁盘 I/O，产生一种非常的性能提升。

SinoDB 提供多种高可用解决方案，例如 HDR/SDS/RSS/ER/GRID，可搭配使用，提供完善的灾备方案。

3. SinoDB 开发规范

3.1 命名规范

数据库对象名称默认采用[系统缩写_对象类型_名称]的方式命名，特殊对象的命名规则在下面的具体对象中有描述。

除命名规则外，还应遵循如下的几个规则：

1. 尽量用英文单词，这样使用者根据英文单词也可以大概知道数据库对象的用途，不用再去查找文档；

2. 由于数据库对象名不能够超过 30 个字符，所以当超过 30 个字符时，英文单词应该缩写。

（注：该规则对于目前已存在的系统或第三方的产品暂时保留原系统命名规范，但旧系统需对自己的命名规范整理成文档；对于自规范实施后的新项目一律采用该规则。）

（一）表对象命名规则

表的对象类型为 T 如：

理赔报案主表可以写成：CL_T_RegistMain。

（二）视图对象命名规则

视图对象类型为 V 如：

视图 PrpLcustView 可以写成：CL_V_CustView。

（三）序列对象命名规则

序列对象类型为 S,应遵循[系统缩写_S_表名_字段名]的规则，让使用者知道该序列对应的是某一个具体表的具体字段，如假设想在表 CL_T_RegistMain 的字段 registId 上添加序列，那这个序列应该写成如下：

CL_S_RegistMain_RegistId。

（四）触发器对象命名规则

触发器对象类型为 TRI，应遵循[系统缩写_TRI_表名]的规则。

（五）主键对象命名规则

主键对象类型为 PK，如表 CL_T_RegistMain 上的主键定义为:CL_PK_REGISTMAIN。

（六）外键对象命名规则

外键对象类型为 FK，如表 CL_T_RegistMain 的 PolicyNo 字段上定义外键，则名称为:CL_FK_REGISTMAIN_POLICYNO。

（七）唯一性索引对象命名规则

唯一性索引对象类型为 UNI，如在表 CI_T_RegistMain 表的 RegistNo 需要建立唯一索引，则名称应该为如下：

CL_UNI_REGISTMAIN_REGISTNO

（八）非唯一性索引对象命名规则

非唯一性索引对象类型为 IDX，应遵循[系统缩写_IDX_表名_字段名]的规则，如假设你需要在表 CI_T_RegistMain 的 PolicyNo 上添加一个非唯一性索引,则这个索引名应该如下：

CL_IDX_REGISTMAIN_POLICYNO

（九）临时表命名规则

表的对象类型为 TMP 如：

理赔报案保单临时主表可以写成：CL_TMP_RegistPolicy。

（十）转储表命名规则

转储表对象名称为 HIS，应遵循[原表名_HIS]的规则,如报案转储表命名为：CL_T_RegistMain_HIS。

（十一）分区表命名规则

分区表对象名称为 PT，应遵循[系统缩写_PT_分区字段_分区值]。

如：假设以表 CL_T_RegistMain 中的 inserttime 字段来按年分区，分区名如下：

CL_PT_INSERTTIME_2007

CL_PT_INSERTTIME_2008

CL_PT_INSERTTIME_2009

（十二）字段命名标准

字段名称必须用字母开头，采用有特征含义的单词或缩写，不能用双引号包含。

3.2 建库规范

（一）数据库日志模式(database logging mode)规范

请选择 unbuffer log 模式，确保宕机数据不丢失；因为 unbuffer log 模式，确保每一笔提交的交易日志以写入日志文件，因此当机日志恢复时可以确保日志重做，所有提交过的交易确保不遗漏。

（二）权限配置规范

SinoDB 数据库层级权限分为三级，分别为 CONNECT, RESOURCE 和 DBA。创建 Database 的用户为默认 DBA，且默认 grant CONNECT 给 public user。一般的使用者通常只需要 grant CONNECT 权限，开发者 grant RESOURCE 权限，数据库管理员 grant DBA 权限。

（三）对于 public user 回收 CONNECT 权限

3.3 表设计规范

(1)小表不分区，笔数小于 1000 万笔不分区

(2)大表分区，笔数大于 1000 万笔分区

- 应用场景一：以主键为分区条件

范例：如保单号码或身份证号为 PK，可采用取馀数 mod 计算做为分区条件，建表语句分区条件如下：

FRAGMENT BY EXPRESSION

MOD(id_num, 4) = 0 IN apfragdbs01,

MOD(id_num, 4) = 1 IN apfragdbs02,

MOD(id_num, 4) = 2 IN apfragdbs03,

MOD(id_num, 4) = 2 IN apfragdbs04;

- 应用场景二：以时间段为分区条件

范例：如无合适字段做为分区条件，可采用时间做为分区条件，建表语句分区条件如下：

FRAGMENT BY EXPRESSION

Month(date_column) in (1,2,3) IN apfragdbs01,

Month(date_column) in (4,5,6) IN apfragdbs02,

Month(date_column) in (7,8,9) IN apfragdbs03,

Month(date_column) in (10,11,12) IN apfragdbs04;

(3)表要具备 PK(Primary key)，如无字段可以为 PK 可用流水号做为 PK;

(4)按常用查询字段建立索引，经常做为查询字段条件，需要建索引加速查询，避免全表扫描;

(5)每个字段定义，选择正确数据类型，时间数据采用数据类型，数字采用数字型态(INT, DECIMAL)，可以避免数据类型转换;

(6)当字符串的长度大于 10 时，考虑用 varchar 替代 char，以节省存储空间;

(7)为了高度并发访问表数据，选择表的锁模式为 lock mode：row;

3.4 字段设计

（一）字符型

固定长度的字串类型采用 char，长度不固定且长度小于 256 的字串类型采用 varchar，大于 256 的变长字串类型采用 lvarchar，避免在长度不固定的情况下采用 char 类型。

（二）数字型

数字型字段采用 SMALLINT 或 INTEGER 类型。

（三）日期和时间

由数据库产生的系统时间首选数据库的日期型，如 DATE 或 DATETIME 类型。

3.5 索引规范

(1)小表索引不分区

(2)分区表索引可按表分区规则分区

(3)单表索引不超过 7 个

(4)复合字段索引，字段数不超过 5 个

(5)重复性高数据不建索引

(6)在哪些字段上建立索引

- 用于 join 条件的字段
- 用于过滤条件的字段
- 用于 ORDER BY 或 GROUP BY 的字段

3.6 SQL 设计

(1)减少返回数据量;

- 不需要的字段不返回; 明确字段，避免使用 select * ...
- 使用条件过滤不需要数据; 明确限缩数据量

范例：

```
select first 100 ...
select ..... where f1 > 100 and f1 < 200;
```

(2)尽量使用 PK 字段或有建索引字段做为查询条件;

范例：

```
select .... where pk_column = ’000010001’
```

(3)避面长事务; 大批量数据异动采分批提交事务，每 1000 笔异动提交一次，最大容许范围 10 万笔;

范例：以下为范例因开发工具不同会有所差别

```
count = 1
loop
    begin work;
    insert into tab1 values (’a’, ’b’, ’c’);
    count= count +1;
    If count >= 1000
       commit work;
    else
       count=1;
    end if
                end loop
```

(4)降低死锁发生概率; 当异动命令执行后尽快提交事务 (commit), 不要把事务拖延太长

(5)避免锁等待； 当需要对某笔数据锁定时可以执行命令限定等待时长，如 set lock mode to wait 5 限定等待 5 秒

(6)使用合理的隔离级别：数据库提供了多种不同的隔离级别（isolation level），需要根据具体的应用场景采用合理的隔离级别，以提高并发性;

隔离级别：

Dirty Read

Committed Read

Committed Read Last Committed

Cursor Stability

Repeatable Read

3.7 存储过程使用规范

(1)适用于频繁重复的操作

(2)RETURNING 字句指定返回值的个数和类型，一个存储过程可以不返回任何值，返回不要太多，尽量限制在 5 个值以内。

4. SinoDB 动态服务器基本术语

4.1 Chunk

一个 chunk 是一片连续的空间单元，数据库数据最终存放于此，因此 chunk 是数据库的物理存储实体，数据库系统所做的空间管理工作都是在 chunk 中进行的。一个典型的 chunk 可能是 UNIX 的一个原始设备(raw device)，也可能是一个 UNIX 文件。

4.2 dbspace 数据库空间

dbspace 是一个逻辑上的概念，一个 dbspace 是若干 chunk 的集合，每个 dbspace 至少有一个 chunk，最先指定给他的 chunk 称为基本 chunk。若有必要可以给一个 dbspace 指定多个 chunk。若某一个 dbspace 的空间耗尽了（即所有指定给他的 chunk 都满了），您可另外再指定 chunk 给他。数据库和表可以建立在特定的 dbspace 上，这也就决定了数据库和表最大只能增长到他所在的 dbspace 空间那么大。你不能决定你所建的库和表究竟处于该 dbspace 上的哪个 chunk 上，但你可以把库和表创建在一个指定的物理设备上（先把位于某个设备上的 chunk 指定给一个 dbspace，然后再把库和表建立在该 dbspace 上）。每个数据库系统至少有一个 dbspace（根 dbspace，root dbspace），所有控制数据库系统的系统信息都被置于其上。

4.3 page 数据页

当指定给数据库系统一个 chunk 后，该 chunk 就会被划分为一些更小的称为页（page）的单位。页是数据库系统的基本 I/O 单位，所有存储在数据库系统中的数据实际都存储在页上。当一行数据从磁盘被读入内存时，整个页上存储的数据都被读入共享内存。许多行的数据有可能被存于一个单独页中。页的大小可以指定，当指定 chunk 的大小时，必须是用能够被页大小整除的数值。一个 dbspace 的所有 chunk 必须有拥有大小相同的页。

4.4 tblspace

一个 tblspace 是只分配给一个特定表的所有页的逻辑集合（一个 tblspace 是一个数据库表的所有数据页的集合，因此一个 tblspace=一张表）。一个 tblspace 必须处在唯一的 dbspace 上。

4.5 extent

为表分配磁盘空间是以 extent 为单元进行的。一个 extent 就是在创建表时确定好的连续的磁盘空间大小（固定的几个属性所占据的空间的大小）。每个表都涉及两个 extent 大小：

extent size 为表初次分配的 extent 大小，在创建表后就分配了；next extent 依次追加到该表上的 extent 大小。

注：一个 dbspace 可看作物理 chunk 的一个逻辑组，这些 chunk 可能分布在不同的磁盘上，尽管他们都来自同一个 dbspace；一个 tblspace 可看作 extent 的一个逻辑组，来自同一个 tblspace 的 extent 可能分布在不同的 chunk 上。

4.6 物理日志

物理日志用来记录在共享内存中修改过的页的前映像，是磁盘上连续页的集合。主要用于 fast recovery。

4.7 逻辑日志

逻辑日志逻辑日志文件是磁盘上用于为服务器存储事务记录的连续页的集合。这些事务记录用于跟踪所有创建时带日志的数据库的改变。所有数据库共享同一逻辑日志文件集。每个服务器必须具有至少三个逻辑日志文件。

5. SinoDB 开发指南

5.1 SinoDB 安装与配置

5.1.1. 安装准备

本节以 centOS 6.5 上的 SinoDB 软件产品安装为例进行说明。

所有 shell 标明 sudo 的也可以直接用 root 用户操作。

5.1.1.1. 准备 SinoDB 所需的硬盘空间

数据空间的规划包括两部分工作：

1．确定需要创建几个 dbspace 以及每个 dbspace 的用途。

2．确定每个 dbspace 的初始空间大小和位置，据此创建 chunk。

这两部分工作是相互关联的。

数据空间的规划主要有以下作用：

1．提高关键数据的可靠性。root dbspace、日志文件和其他关键数据所在的 chunk 可以建在采用 RAID 技术进行容错的磁盘阵列上或进行软件镜像。

2．提高 I/O 性能。在有多个硬盘和控制器的情况下，尽量将 I/O 操作均衡地分布到各个硬盘上。在多个硬盘的性能有差异的情况下，将最频繁存取的数据放在性能最高的硬盘和分区上。

3．方便备份和恢复工作。因为备份和恢复都是以 dbspace 为最小单位的。因此将不同用途的数据放到不同的 dbspace 上可以减轻备份和恢复的工作量。

一般说来，至少要创建 4 个 dbspace，分工如下：

rootdbs：存放系统数据和物理日志。(在 oninit -i 时逻辑日志和物理日志都是放在 rootdbs 中的，一定要把逻辑日志放在 logdbs 上)

tempdbs：存放临时表。

logdbs：存放逻辑日志。

userdbs：存放用户数据。

每个 dbspace 的初始 chunk 的大小没有必要开得很大，非要把所有的硬盘空间都用完，只要够用就可以了。因为 chunk 增加容易删除难，把暂时不用的硬盘空间预留起来更加灵活。

在 OnLine 初始化时，会将该文件增大到你设定的 chunk 大小。注意要保证该文件系统中有足够的空间创建 chunk。

5.1.1.2. 创建用户

需要建立 sinodbms 用户，主目录为/home/sinodbms。
注意以下~表示 sinodbms 主目录，即/home/sinodbms。
以 root 用户执行：
创建 sinodbms 用户组和用户，并修改 sinodbms 口令

```
shell# groupadd sinodbms
shell# useradd -g sinodbms -d /home/sinodbms -s /bin/bash -m sinodbms
shell# passwd sinodbms
```

修改/home/sinodbms/.profile,增加以下几行：

```
export SINODBMSSERVER=primary
export SINODBMSDIR=/home/sinodbms
export ONCONFIG=onconfig
export SinoDBSQLHOSTS=$SINODBMSDIR/etc/sqlhosts
export PATH=$SINODBMSDIR/bin:/usr/bin:${PATH}:.
export DB_LOCALE=zh_cn.utf8
export CLIENT_LOCALE=zh_cn.utf8
export
LD_LIBRARY_PATH=${SINODBMSDIR}/lib:${SINODBMSDIR}/lib/esql:${SINODBMSDIR}/lib/cli
```

5.1.2. 安装 SinoDB 产品

5.1.2.1. 上传安装软件

将要安装的 SinoDB 产品从安装介质（磁带或光盘）上传到 sinodbms 用户的/home/sinodbms 目录下并解压。过程可能为：

```
shell# tar -xvf  /home/sinodbms/iif.16.8.FC7.linux-x86_64.tar -C /home/sinodbms
```

5.1.2.2. 执行安装

```
shell# su - root
shell# cd /home/sinodbms
shell# ./ids_install
```

5.1.3. 修改操作内核参数

因为数据库需要申请大量的共享内存和信号量，因此，一般 UNIX 系统的缺省内核参数都不能满足要求，必须进行修改。

```
shell# vi /etc/sysctl.conf
kernel.shmmax = 4398046511104
kernel.shmall = 4194304
kernel.shmmni = 4096
kernel.shm_rmid_forced = 0
kernel.msgmax = 65536
kernel.msgmni = 32768
kernel.msgmnb = 65536
kernel.sem = 250      32000      32    128

shell# vi /etc/security/limits.conf        #在文件末尾加
* soft nofile 65536
* hard nofile 65536
* soft nproc 65536
* hard nproc 65536
* soft stack 65536
* hard stack 65536
```

将该文件的内容和操作系统的当前参数进行比较，如果比当前参数大，要把当前参数修改称该文件要求的值，如果比当前参数小，一般不需要再作改动。修改成功后，reboot 系统。

不同的 UNIX 系统修改内核参数的方法差异很大。下面介绍几种常用的操作系统修改内核参数的方法。Sun Solaris 最简单，直接编辑/etc/system 文件，然后 reboot 即可；Digital UNIX 可以用 sysconfig 命令进行修改，也可以用 CDE 提供的图形界面的 Kernel Tuner 工具，因为只修改和 ipc、proc 有关的内核参数，因此不用重新连接内核，重新启动即可生效。IBM AIX 可以用集成化管理工具 smit 来修改内核参数。HP-UX 可以用维护工具 sam 来修改内核参数。SCO UNIX 可以用 idtune 命令进行修改，也可以用字符界面的集成化管理工具 sysadmsh 或 scoadmin。

5.1.4. 添加数据库文件

按空间规划执行类似以下命令添加相关数据库文件：

```
shell# su - sinodbms
shell# mkdir /home/sinodbms/dbs     #数据库数据文件所在目录
shell# touch /home/sinodbms/dbs/rootdbs
shell# touch /home/sinodbms/dbs/llogdbs
shell# touch /home/sinodbms/dbs/plogdbs
shell# touch /home/sinodbms/dbs/tempdbs1
```

```
shell# touch /home/sinodbms/dbs/datadbs1
shell# touch /home/sinodbms/dbs/idxdbs1
shell# touch /home/sinodbms/dbs/sbdbs1
shell# chmod 660 /home/sinodbms/dbs/*
shell# ls -al /home/sinodbms/dbs
```

5.1.5. 配置 sqlhost 文件

假设数据库所在机器 ip 地址为 192.168.37.46,使用的监听端口为 1526。在/etc/services 文件中为 SinoDB 增加一个端口号，只要不和已有的端口号重复即可，假设用 1526（默认端口）。

```
prilis  1526/tcp # SinoDB listen port
```

最后，修改$SINODBMSDIR/etc/sqlhosts 文件，为每种通信方式写一行描述：

```
ipcshm      onipcshm <hostname>    ipcshm
ipcstr onipcstr          <hostname>    ipcstr
<hostname>    onsoctcp  <hostname>    prilis
```

增加一行：

```
primary          onsoctcp          192.168.37.46                prilis
```

其中每一列的含义如下：

第一列是 dbservername，是数据库 client 访问 server 时要给出的标识符。

第二列是 nettype，指出该 dbservername 采用的是哪种通信方式。

第三列是 hostname。用 hostname 命令可以查到该机的 hostname。

第四列是 servicename。对 onsoctcp 或 ontlitcp 方式的 dbservername，必须是 etc/services 文件中已经定义的一个服务名，不能直接写 TCP 端口号。在有些机器上，服务名不能超过 8 个字符。对 onipcshm 和 onipcstr 方式，建议设成和 dbservername 相同。

OnLine 的数据库用户(client)和 Server 之间，可以采用不同的通信方式：

Onipcshm

client 和 server 之间通过共享内存通信。只有当 client 和 server 位于同一台机器上时才可以采用这种方式。这种方式速度最快，而且所有的操作系统都支持。缺点是可能带来安全性和可靠性方面的问题。

Onipcstr

client 和 server 之间通过 stream pipe 通信。只有当 client 和 server 位于同一台机器上时才可以采用这种方式。这种方式不会有共享内存可能带来的安全性和可靠性方面的隐患。但是不是所有的操作系统都支持这种方式，而且在有的操作系统下，onipcstr 比 onipcshm 明显慢。

ontlitcp 或 onsoctcp

client 和 server 之间通过 TCP/IP 通信。这是当 client 和 server 不在同一台机器上时的唯一选择。OnLine 根据当前机器的操作系统支持的网络 API 是 TLI 还是 socket，决定是采用 ontlitcp 还是 onsoctcp。

每个 OnLine Server 可以同时支持多种通信方式，每种通信方式指定一个名字 dbservername。client 要用某种通信方式访问某个 server，只需要把自己的环境变量 SINODBMSSERVER 设成相应的 dbservername。文件$SINODBMSDIR/etc/sqlhosts 中描述了 dbservername 对应的通信方式的具体参数

5.1.6. 修改 onconfig 文件

首先，从配置文件模板 onconfig 中复制出初始的 onconfig 文件。

```
shell# cp $SINODBMSDIR/etc/onconfig.std $SINODBMSDIR/etc/$ONCONFIG
shell# vi   $SINODBMSDIR/etc/$ONCONFIG
```

然后进行以下修改：

```
ROOTPATH     <rootdbs 的第一个 chunk 的路径>
ROOTSIZE     <rootdbs 的第一个 chunk 的大小(以 KB 为单位)>
PHYSFILE<物理日志的大小(以 KB 为单位)>
LOGFILES     <逻辑日志文件的个数>
LOGSIZE  <逻辑日志文件的大小(以 KB 为单位)>
MSGPATH      <$SINODBMSDIR/online.log>
ALARMPROGRAM      <$SINODBMSDIR/etc/log_full.sh>
LTAPEDEV     /dev/null
LTAPESIZE    1024000
DBSERVERNAME   SinoDB
DBSERVERALIASES      <hostname>，ipcstr
MULTIPROCESSOR<单 CPU 填 0，多 CPU 填 1>
VPCLASS cpu,num=2,noage
CLEANERS 32
LOCKS    2000000
PHYSBUFF     320
LOGBUFF 320
LOGSMAX      <最大逻辑日志文件个数>
DEF_TABLE_LOCKMODE ROW    #行级锁
SHMVIRTSIZE 2048000       #内存驻留段大小
SHMADD
EXTSHMADD
```

```
    BUFFERPOOL
size=16K,buffers=100000,lrus=32,lru_min_dirty=5,lru_max_dirty=10
```

以上参数中有几个值需要作进一步说明：

1. PHYSFILE<物理日志的大小>

一般在硬盘空间足够的情况下(物理日志放在 rootdbs 中)，将物理日志的大小设成 200MB 就可以了。

2．LOGFILES<逻辑日志文件的大小>、LOGSIZE<逻辑日志文件的个数>和 LOGSMAX<最大逻辑日志文件个数>

在硬盘空间足够的情况下(逻辑日志准备放在 logdbs 中)，一般可以创建 30 个 5MB 大小的逻辑日志文件。因为 SinoDB 初始化时，只创建一个 rootdbs，因此初始化时逻辑日志文件只能放在 rootdbs 中。为了把逻辑日志文件放在我们希望的 logdbs 中，必须在初始化完成后，先创建 logdbs，然后在 logdbs 中创建原定大小和数量的逻辑日志文件，最后将 rootdbs 中的逻辑日志文件删掉。因此，在上面修改参数的时候，可以将<逻辑日志文件的个数>设成最小值 3。<逻辑日志文件的大小>设成 5MB，<最大逻辑日志文件个数>设成 100。

3．buffers<数据缓冲区的大小>

一般设成机器内存的 1/4 到 1/2。例如：对于 1GB 内存的机器，可以设为 256MB-512MB 之间的一个值。

5.1.7. 初始化数据库

在 sinodbms 用户下，执行 oninit -ivy，初始化 OnLine，创建 root dbspace。

检查初始化是否成功。在 oninit 命令结束后，等待大约 1 分钟左右，执行 onstat -，查看 OnLine 的当前状态。如果显示... OnLine ...,表示初始化成功。否则通过日志文件 $SINODBMSDIR/online.log 文件中记录的信息，分析错误原因，改正后再执行 oninit -ivy，直至成功。执行 dbaccess，如果出现终端类型不对的提示信息则需要按以下流程调整终端类型：

export TERM=vt100; 如果需要设置 TERM 或 TERMCAP，把它们加入 $SINODBMSDIR/.profile 文件中。

5.1.8. 创建其他 dbspace

在 sinodbms 用户下，用 onspaces 命令创建其他 dbspace。对于前面讨论的例子，过程如下：

```
onspaces -c -d llogdbs -p /home/sinodbms/dbs/llogdbs -o 0 -s 2048000
onspaces -c -d plogdbs -p /home/sinodbms/dbs/plogdbs -o 0 -s 204800
onspaces -c -d tempdbs1 -k 16 -t -p /home/sinodbms/dbs/tempdbs1 -o 0 -s 2048000
onspaces -c -d datadbs1 -p /home/sinodbms/dbs/datadbs1 -o 0 -s 2048000 -k 16
```

```
onspaces -c -d idxdbs1 -p /home/sinodbms/dbs/idxdbs1 -o 0 -s 2048000 -k 16
onspaces -c -S sbdbs1 -p /home/sinodbms/dbs/sbdbs1 -o 0 -s 2048000
```

使用 onstat -d 检查 dbspcace 是否创建成功，如果输错命令，可参考以下命令撤销：onspaces -d plogdbs

5.1.9. 移动逻辑日志/物理日志到对应的 dbs

在 oninit -ivy 时逻辑日志是放在 rootdbs 中的，rootdbs 主要是为了存放物理日志和系统表，而且空间相对较小，所以一定要把它移到 logdbs。定时做逻辑日志文件备份，系统只将已写满的逻辑日志文件备份到磁带上，然后清空，并释放这些日志文件。做定时备份一定要在所有逻辑日志文件被写满之前进行。所以，如果采用定时备份，要注意观察逻辑日志使用状况。另外，如果经常有长事务发生，应避免使用定时备份，采用连续备份比较安全。

使 OnLine 处于 Quiescent 模式

```
 onmode -uy
```

在 plogdbs 上添加物理日志

```
onparams -p -s 204800 -d plogdbs -y
```

在 logdbs 上创建 30 个 5MB 的逻辑日志块

```
onparams -a -d logdbs   -s   5000   -y
```

每执行一次创建一个，共执行 30 次。

进行 0 级备份，使新增的逻辑日志块可用

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

删除 rootdbs 上的 3 个逻辑日志文件

```
$nmode -l
onmode -l
onmode -l
onmode -c
onparams -d -l 1 -y
onparams -d -l 2 -y
onparams -d -l 3 -y
```

使用以下命令重启数据库：

```
onmode -ky    #关闭数据库
oninit -vy      #启动数据库
```

至此，我们成功创建了 SinoDB 数据库实例，并能正常使用了。

5.2 SinoDB 常见操作与维护命令

5.2.1. 数据库操作

创建数据库

```
--创建数据库为无日志模式数据库
CREATE DATABASE db IN db_dbs;
--创建 UNBUFFERED 日志模式数据库
CREATE DATABASE db IN db_dbs WITH LOG;
--创建 BUFFERED 日志模式数据库
CREATE DATABASE db IN db_dbs WITH BUFFERED LOG;
--创建 ANSI 日志模式数据库
CREATE DATABASE db IN db_dbs WITH LOG MODE ANSI;
```

更名数据库

```
--将数据库 stores6 重命名为 stores7
RENAME DATABASE stores6 TO stores7;
```

删除数据库

```
删除名为 db 的数据库
DROP DATABASE databasename;
```

5.2.2. 表操作

创建普通表

```
--创建名为 orders 的表
CREATE TABLE orders(
order_num    SERIAL NOT NULL,    ------字段以及字符类型
customer_num    INTEGER,
order_date    DATE)
IN dbspace1                                    ------表的存放位置
EXTENT SIZE 64                                 ------表创建时分配的 extent 大小
NEXT SIZE 32                                   ------ extent 空间使用完后下一个分配的
extent 大小
LOCK MODE ROW;                          ------锁级别
```

创建简单大对象表

```
/*创建名为 evaluation 含有简单大对象字段表,并将简单大对象字段部分存放于独立
blobspace 上*/
CREATE TABLE evaluation(
employee_num SERIAL,
```

```
    manager_num INTEGER,
    emp_eval_form TEXT IN blobspace1,   -----列字段类型与存放位置
    emp_picture BYTE IN blobspace2,     -----列字段类型与存放位置
    emp_phrase TEXT IN TABLE)           -----列字段类型与存放位置
    IN dbspace2;
```

创建智能大对象表

```
    /*创建表名为 movie 的含有智能大对象字段表，并将只能大对象字段部分存放于独立 smartblobspace 上*/
    CREATE TABLE movie(
    movie_num INTEGER,
    movie_title CHAR(50),
    video BLOB,
    audio BLOB,
    description CLOB)                   ------指定列字段类型为 CLOB
    PUT video IN (sbsp3),               ------指定字段 video 的存放位置
    audio IN (sbsp1),                   ------指定 audio 的存放位置
    description IN (sbsp5);             ------指定 description 的存放位置
```

创建临时表

```
    --创建名为 temp_order 的临时表，不记日志
    CREATE TEMP TABLE temp_order
    (order_num INTEGER)
    WITH NO LOG;                        ------指定日志模式
    --将 customer 中两个字段插入临时表 cust_temp，不记日志
    SELECT customer_num, company
    FROM customer INTO TEMP cust_temp
    WITH NO LOG;                        ------指定日志模式
```

更名表

```
    --将表名为 stock 表更名为 inventory
    RENAME TABLE stock TO inventory;
```

内存驻留

```
    --设置表常驻内存
    SET TABLE state MEMORY_RESIDENT;
    --取消表常驻内存
    SET TABLE tablename NON_RESIDENT
```

获取建表语句

```
##导出数据库 stores_demo 中 orders 表的所有表结构,sinodbms 用户登陆执行
dbschema -d stores_demo -t orders  –ss
-d          指定数据库名
-t          指定表名
-ss         同时打印出存放位置、锁模式以及 extent 信息
```

更改锁模式

```
--将 orders 表锁模式改为行锁
ALTER TABLE orders LOCK MODE (ROW);
```

更改下一个 extent 大小

```
--将 customer 表下一个 extent 设置为 20K
ALTER TABLE customer NEXT SIZE 20;
```

增加字段

```
--为 customer 表增加一个 birthday 字段，数据类型为 date
ALTER TABLE customer ADD birthday DATE;
```

删除字段

```
--删除 customer 表中 birthday 字段
ALTER TABLE customer DROP birthday;
```

更改字段数据类型

```
--更改 customer 表 birthday 字段时间类型为年至分
ALTER TABLE customer MODIFY birthday DATETIME YEAR TO MINUTE;
```

更名字段

```
--将表 invoice 表中 paid_date 字段名更改为 date_paid
RENAME COLUMN invoice.paid_date TO date_paid;
```

转化普通表为智能大对象表

```
--将 booklist 表 content 字段变更为 CLOB 类型并存放在 sbsp1 中
ALTER TABLE booklist MODIFY content CLOB, PUT content IN (sbsp1) (log);
```

删除表

```
--删除 customer 表
DROP TABLE customer;
```

5.2.3.索引操作

创建唯一索引

```
123--创建名为 ix_orders 的唯一索引并存放在 idx_dbs 中
CREATE UNIQUE INDEX ix_orders ON orders(orders_num) IN idx_dbs;
```

创建复合索引

```
--创建名为 ix_items 的复合索引
CREATE INDEX ix_items ON items(manu_code, stock_num);
```

创建簇索引

```
--创建名为 ix_manufact 的簇索引
CREATE UNIQUE CLUSTER INDEX ix_manufact ON manufact(manu_code);
```

创建可重复索引

```
--创建名为 ix_man_stk 的可重复索引
CREATE INDEX ix_man_stk ON items(manu_code desc, stock_num);
```

创建带索引页填充度的索引

```
--创建名为 state_code_idx 的可重复索引，设置填充度为 80%
CREATE INDEX state_code_idx ON state(code) FILLFACTOR 80;
```

更改、重命名、删除索引

```
--更改索引 ix_man_cd 为 CLUSTER
ALTER INDEX ix_man_cd TO CLUSTER;
--更改索引名 ix_cust 为 new_ix_cust
RENAME INDEX ix_cust TO new_ix_cust;
--删除 ix_stock 索引
DROP INDEX ix_stock;
```

禁用、启用索引

```
--禁用表上索引索引
SET INDEXES FOR tablename DISABLED;
--启用表上所有索引
SET INDEXES FOR tablename ENABLED;
```

设置、移除索引常驻内存

```
--设置索引驻留内存
SET INDEX state_ix MEMORY_RESIDENT;
--移除索引驻留内存
SET INDEX indexname NON_RESIDENT;
```

5.2.4. 分片操作

创建 ROUND ROBIN 分片表

```
--创建名为 table1 的分片表，数据轮流存放至 dbsapce1,dbspace2 中
CREATE TABLE table1(
col_1 SERIAL,
```

```
col_2 CHAR(20))
FRAGMENT BY ROUND ROBIN          -----指定存放模式为顺序存放
IN dbspace1,dbspace2                    -----指定顺序存放的位置
EXTENT SIZE 10000 NEXT SIZE 3000;
```

创建大对象 ROUND ROBIN 分片表

```
/*创建表名为 movie 表，video 数据轮流存放 sbsp3，sbsp6，sbsp7 中；audio 数据轮流存放 sbsp1，sbsp2，sbsp4 中*/
CREATE TABLE movie(
movie_num INTEGER,movie_title CHAR(50),
video BLOB,audio BLOB,description CLOB), -----指定各列名与数据类型
PUT video IN (sbsp3, sbsp6, sbsp7),          -----指定 vidoe 顺序存放位置
audio IN (sbsp1, sbsp2, sbsp4),              -----指定 audio 顺序存放位置
description IN (sbsp5);
```

创建基于表达式的分片表

```
/*创建分片表 table1 并将：
col_1<= 10000 AND col_1 >= 1 放在 dbspace1 中，
col_1 <= 20000 AND col_1 > 10000  放在 dbspace2 中；
不满足上面条件全部存放在 dbspace3 中*/
CREATE TABLE table1(
col_1 SERIAL,
col_2 CHAR(20),
...)
FRAGMENT BY EXPRESSION
col_1 <= 10000 AND col_1 >= 1 IN dbspace1,     -----指定表达式与存放位置
col_1<=20000 AND col_1>10000 IN dbspace2    -----指定表达式与存放位置
REMAINDER IN dbspace3;                          -----指定不满足条件存放位置
```

创建基于 HASH Functions 的分片表

```
CREATE TABLE table1(
customer_num SERIAL
lname CHAR(20)
...)
FRAGMENT BY EXPRESSION
MOD(customer_num, 3) = 0 IN dbspace1,          -----指定条件与存放位置
MOD(customer_num, 3) = 1 IN dbspace2,          -----指定条件与存放位置
```

```
MOD(customer_num, 3) = 2 IN dbspace3;                -----指定条件与存放位置
```

创建/修改成带 ROWID 的分片表

```
--创建带 ROWID 分片表
CREATE TABLE orders(
order_num SERIAL,
customer_num INTEGER,
part_num CHAR(20))
WITH ROWIDS                                          -----带 ROWIDS
FRAGMENT BY ROUND ROBIN IN dbs1,dbs2;
--在表中增加 ROWIS
ALTER TABLE items ADD ROWIDS;
--在表中删除 ROWIS
ALTER TABLE items DROP ROWIDS;
```

创建分片表索引

```
--通过表达式创建分片索引
CREATE INDEX idx1 ON table1(col_1)
FRAGMENT BY EXPRESSION
col_1 < 10000 IN dbspace1,
col_1 >= 10000 IN dbspace2;
```

创建不分片索引

```
CREATE INDEX idx1 ON table1(col_1)
IN dbspace1;
```

普通表与分片表转换

```
--从分片表修改为普通表
ALTER FRAGMENT ON TABLE table1 INIT IN dbspace2;

--从普通表切换为分片表
ALTER FRAGMENT ON TABLE table1 INIT                  -----init 初始化成
FRAGMENT BY ROUND ROBIN
IN dbspace1, dbspace2;

--从普通表切换为基于表达式的分片表
ALTER FRAGMENT ON TABLE table1 INIT                  -----init 初始化成
FRAGMENT BY EXPRESSION
```

```
    col_1 <= 10000 AND col_1 >= 1 IN dbspace1,
    col_1 <= 20000 AND col_1 > 10000 IN dbspace;
```

增加分片

```
    --在原条件最后添加语句
    ALTER FRAGMENT ON TABLE orders ADD                -----增加条件
    note_code > 3000 IN dbspace4;
    --在原 dbspace4 条件前添加语句
    ALTER FRAGMENT ON TABLE orders ADD                -----增加条件
    note_code <= 3000 OR note_code = 3500 IN dbspace3
    BEFORE dbspace4;                                  -----指定条件增加的位
置
    --给 Round robin 表添加 dbspace
    ALTER FRAGMENT ON TABLE customer ADD dbspace3;
```

删除分片

```
    --删除表 table1 中 dbspace1 分片
    ALTER FRAGMENT ON TABLE table1 DROP dbspace1;
    --删除分片索引 table1_idx1 中 dbspace4 分片
    ALTER FRAGMENT ON INDEX table1_idx1 DROP dbspace4;
```

更改分片规则

```
    --将原有放在 dbspace1 上的表达式更改为 col_1 > 30000
    ALTER FRAGMENT ON TABLE table1 MODIFY dbspace1 TO col_1 > 30000
    IN dbspace1;
    --将原存放在 dbspace3 上的数据库更改为存放在 dbspace5 上
    ALTER FRAGMENT ON TABLE table1 MODIFY dbspace3 TO REMAINDER
    IN dbspace5;
```

摘除/连接分片

```
    --将 table1,table2 表结合成一个 table1 的分片表
    ALTER FRAGMENT ON TABLE table1 ATTACH table1, table2;
    --将一个分片表，分裂成两个普通表
    ALTER FRAGMENT ON TABLE table1 DETACH dbspace2 table2;
```

分片跳过

```
    --数据库允许忽略不可用的分片，可以通过 DATASKIP 参数来控制
    --打开
    SET DATASKIP ON
```

```
--关闭
SET DATASKIP OFF
--SKIP 指定分片
SET DATASKIP dbspace1
```

5.2.5.统计更新操作

统计更新级别:

HIGH--搜集表、索引、字段等信息而且重建所有数据库分布（比较耗时，耗资源）

MEDIUM--收集表、索引、字段等信息而且重建数据库分布，重建比例为 85%~99%。

LOW--只收集表、索引、字段等信息。

获取 SQL 语句执行计划

```
--将执行计划打印到当前文件夹下的 sqexplain.out 文件。
SET EXPLAIN ON;
Select * from systables；
OFF
```

执行统计更新

```
--为当前数据库执行统计更新，级别分为 LOW,MEDIUM,HIGH；
--建议为数据库运行 LOW 级别
UPDATE STATISTICS [LOW|MEDIUM|HIGH];

--建议为表运行 MEDIUM 级别
UPDATE STATISTICS [LOW|MEDIUM|HIGH] FOR TABLE [tabname];

--为索引首字段运行统计更新，建议为表的索引字段运行 HIGH 级别
UPDATE STATISTICS [LOW|MEDIUM|HIGH] FOR TABLE tabname (colname);
```

只统计数据分布

```
--使用 DISTRIBUTIONS ONLY 控制只重建数据分布，不统计基本信息
UPDATE STATISTICS MEDIUM FOR TABLE customer DISTRIBUTIONS ONLY;
```

删除数据分布

```
UPDATE STATISTICS LOW DROP DISTRIBUTIONS;
UPDATE STATISTICS LOW FOR TABLE customer DROP DISTRIBUTIONS;
UPDATE STATISTICS LOW FOR TABLE orders(order_num) DROP DISTRIBUTIONS;
```

5.2.6.约束操作

增加约束

```
ALTER TABLE orders
ADD CONSTRAINT PRIMARY KEY (order_num);   -----增加主键约束
ALTER TABLE items
ADD CONSTRAINT FOREIGN KEY(order_num)     -----增加外键约束
REFERENCES orders;                                -----指定外键关联表
```

约束延迟检查

```
BEGIN WORK;
SET CONSTRAINTS ALL DEFERRED;              -----设置约束延迟检查
UPDATE orders SET order_num = 1006
WHERE order_num = 1001;
UPDATE items SET order_num = 1006
WHERE order_num = 1001;
COMMIT WORK;
```

删除约束

```
--删除在 pk_orders 列上所有约束
ALTER TABLE orders DROP CONSTRAINT pk_orders;
```

5.2.7.视图操作

当视图中包括 Join 语句，运算语句时：视图将不能插入、更改、删除；当视图引用虚拟列时不能更改、删除、插入。

创建普通视图

```
CREATE VIEW ordsummary AS
SELECT order_num, customer_num, ship_date
FROM orders;


CREATE VIEW they_owe (ordno, orddate, cnum) AS
SELECT
order_num, order_date, customer_num
FROM orders
WHERE paid_date IS NULL;          -----视图语句
```

创建筛选行的视图

```
CREATE VIEW baseball AS
SELECT * FROM stock WHERE                -----视图语句
description MATCHES "*baseball*";     -----筛选条件
```

删除视图

```
DROP VIEW ordsummary;
```

创建多表联合视图

```
CREATE VIEW stock_info AS  SELECT stock.*, manu_name
FROM stock, manufact      -----指定多张表
WHERE stock.manu_code =manufact.manu_code     ----指定关联条件
```

5.2.8. 存储过程/触发器操作

创建存储过程

```
create dba Procedure sel_stu()
Returning Varchar(20);
Define tmp_name Varchar(20);
Set Debug File To "/opt/SinoDB/sel_stu.trace";
Trace On;
Let tmp_name="";
Foreach sel_stu For select name Into tmp_name
From student
Return tmp_name With Resume;
End Foreach;
Return "mamamama";
Return "babababa";
Trace off;
End Procedure
```

/*Set Debuge File To pathname 可以设置存储过程执行中的踪迹文件，利用它可调试存储过程，但已调试好的存储过程应将跟踪语句去掉，因为它会大大增加执行时的开销，Trace On 打开跟踪，Trace off 关闭跟踪。*/

创建 TRIGGER

```
CREATE TRIGGER trigger_name ...
Trigger event --什么事件将触发此 TRIGGER
INSERT ON table_name
DELETE ON table_name
UPDATE ON table_name
UPDATE OF column_name ON table_name
SELECT ON table_name
SELECT OF column_name ON table_name
```

```
REFERENCING clause--触发器将引起什么动作
CREATE TRIGGER items_upd
UPDATE OF total_price ON items
REFERENCING NEW AS post OLD AS pre
FOR EACH ROW
(UPDATE ORDERS
SET order_price = order_price +
post.total_price - pre.total_price
WHERE order_num = post.order_num);
WHEN clause--制约
CREATE TRIGGER ins_cust_calls INSERT ON cust_calls
{Flag problems for billing dept. review} REFERENCING NEW AS post
FOR EACH ROW WHEN(post.call_code = 'B')
(INSERT INTO warn_billing
VALUES(post.customer_num));
Trigger action--trigger 将做什么
BEFORE (define action here);
FOR EACH ROW (define action here);
AFTER (define action here);
```

5.2.9. 常见维护命令简介

SinoDB 提供了一整套命令行管理工具，常用的有以下这些：

命　　令	功　　能
oninit	启动 OnLine
onmode	改变模式和共享内存
onstat	通过共享内存结构监视 OnLine 的操作状态
oncheck	检查、修复、显示 OnLine 的磁盘结构
ondblog	改变 database 的 log 方式
onparams	修改逻辑和物理日志的配置参数
onspaces	修改 blobspace 和 dbspace 的配置
ontape	数据库备份和恢复工具
onarchive	比 ontape 功能更强的备份和恢复工具
dbexport	将整个 database 备份成文本文件格式
dbimport	用文本文件格式的 database 备份重建 database
dbschema	显示数据库、表的结构
dbaccess	字符窗口界面的交互式 SQL 命令执行环境

严格来说，最后四个命令不属于管理工具，但是因为在进行数据库管理时经常用到，所以也在此列出。

5.2.10. dbaccess

dbaccess 是交互式查询工具，用户可以通过它进行数据库的查询、建立、删除，数据库表的增加、删除，及表中数据的增加、删除、查询等操作。dbaccess 还可以通过 sql 语句执行数据库操作命令。

dbaccess 有两种执行方式：交互式方式和命令行方式。

在 Shell 提示符下，直接敲 dbaccess 不带任何参数，就进入交互式方式。会显示如下主菜单：

```
DBACCESS:Query-language  Connection  Database  Table  Session  Exit

Use SQL query language.

----------Press CTRL-W for Help------
```

常用的操作有：

1．选择一个数据库：

选 Database/Select 即可选择所需数据库。

2．创建一个数据库：

选 Database/Create/键入数据库名/Dbspace/选择一个数据库空间/Create-new-database

3．删除一个数据库：

选 Database/Drop/选择要删除的库(当前数据库不可删)/Yes(Y)(确认删除)

4．选择一个表，并查询字段名：

选择表所在数据库/Table/Info/选表/Column。

5．创建一个表：

选择表所在数据库/Table/Create/键入表名/Add/键入字段名/Type/Length/Index/Nulls/CTRL+C/Build-new-table

6．删除一个表：

选择表所在数据库/Table/Drop/选择要删除的表/Yes(Y)(确认删除)

7．执行 SQL 语句：

选 Query-language/New/键入 SQL 语句/CTRL+C——退出编辑状态/Run

命令行方式的 dbaccess 可以不进入窗口界面直接执行一组 SQL 语句，主要用于在 shell 程序中需要执行 SQL 命令时。其命令格式为：

```
dbaccess [<database>] [<script-file>]
```

其中：

<database>用来指定执行 SQL 命令时的 database，对于有些命令，象 create database，不需要指定这个参数。

<script-file>中存放的是要执行的 SQL 语句。如果不给出这个参数，则把标准输入作为<script-file>，这是经常用的一种情况。

例如：想删除 database stores7 中的 table stock，可以执行以下命令：

```
echo ‘drop table stock’ | dbaccess stores7
```

5.2.11. 数据库的启动和关闭

SinoDB 共有六种运行模式(Mode)：Off-Line，Quiescent，On-Line，Read-Only，Recovery 和 Shutdown。

Off-Line 模式：表示 OnLine 没有运行。

Quiescent 模式：相当于 UNIX 操作系统的单用户状态，此时不能进行数据访问，只能进行备份、增删日志文件等管理活动。

On-Line 模式：表示 OnLine 处于正常工作（在线）状态，能够向用户提供数据访问服务。

Read-Only 模式：表示当前 OnLine 处于只读状态，当使用 SinoDB 的数据复制(Data Replication)功能时，从服务器（Secondary Server）会处于这种状态。

Recovery 模式：是一种短时间的临时状态。它发生在 OnLine 从 Off-Line 向 Quiescent 模式转移的过程中，在这种模式下，主要完成数据库的快速恢复。

Shutdown 模式：是一种短时间的临时状态。它发生在 OnLine 从 On-Line 向 Quiescent 模式或从 On-Line(或 Quiescent)向 Off-Line 模式转移的过程中。

最常用的模式转换命令有两个：

从 Off-Line 模式到 On-Line 模式，即数据库的启动。

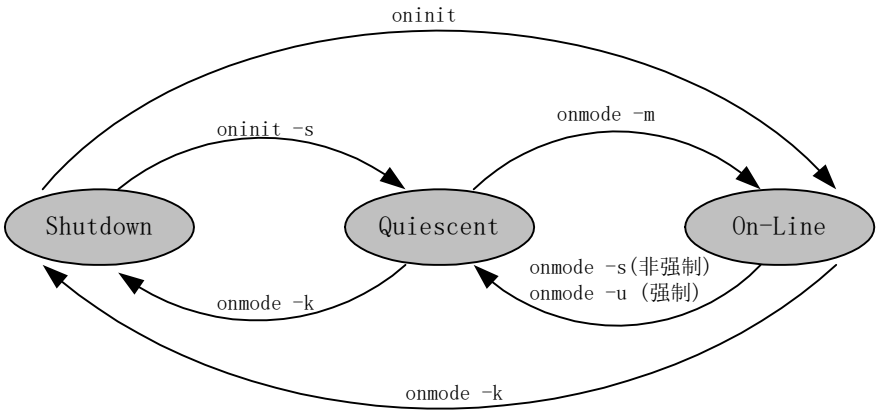
```
oninit
```

从 On-Line 模式到 Off-Line 模式，即数据库的关闭。

```
onmode -ky
```

上面的选项 y 表示当仍有用户连在 Server 上时，不再要求确认，直接断开连接。

完整的模式转换命令如下图所示：



如果需要查询当前 Server 所处的模式，可以用以下命令：

```
onstat -
```

如果当前 Server 处于 Off-Line 模式，会显示：

```
shared memory not initialized for SINODBMSSERVER 'xxx'
```

5.2.12. SinoDB 数据库运行状态查询

onstat 通过读取 OnLine 的共享内存结构，来提供关于 OnLine 的各种统计信息。这些统计信息也可以通过直接访问 sysmaster 数据库中的 SMI(System Monitoring Interface)表来得到，但是用 onstat 命令更加直观，这也是 SinoDB 的一个优点。

onstat 命令的选项非常复杂，这里只介绍最常用的。

onstat -

列出 onstat 所有选项的简要说明。

onstat -i

进入交互式状态，用命令 q 退出。

onstat -r [<秒数>] <其它选项>

每隔<秒数>重复执行<其它选项>一次，直至用 interrupt key(一般为^C)强行中断。<秒数>缺省为 5。

onstat -

显示当前 server 的版本号、所处的模式、连续运行时间和共享内存的大小。

onstat -V

显示当前 server 的版本信息和产品号。

onstat -c

显示当前 server 启动时使用的配置文件内容。因为在 server 启动后，配置文件 $ONCONFIG 可能被修改，因此可能和这里显示的内容不同。

onstat -m

显示消息日志文件 online.log 的最后 20 行。

onstat -u

显示当前用户的情况。

onstat -d

显示所有 dbspace 和 chunk 的基本情况。包括每个 dbspace 的名字、由哪些 chunk 组成、每个 chunk 的大小、可用空间、是否镜像等等。

onstat -b

显示当前 buffer 区的使用情况。在该命令输出信息的最后，会有’XXXX buffer size’的字样，这就是 OnLine 中 page 的大小（即配置文件中 BUFFERS 参数的单位）。

onstat -p

显示一些统计信息。如一共进行了多少次读写操作，cache 的命中率，消耗的 CPU 资源等。

onstat -l

查看逻辑日志和物理日志的大小，使用情况。

5.3 备份与恢复

数据库的备份包括两方面的工作：数据空间（dbspace）备份和日志文件备份。

通过定期做数据空间备份，可以在数据被破坏时恢复到最近一次备份的数据。如果想恢复该点后的数据，则需要同时做日志文件备份。也就是说，如果想在数据被破坏时能够恢复数据，数据空间备份是必须要做的。而日志文件备份在不要求恢复数据空间备份点之后的数

据时可以不做。每次执行备份命令时，备份介质（一般是磁带）都是从头开始使用的，因此介质上原有的数据就会被覆盖，这在备份的时候千万要注意。

（一）指定备份设备

在安装 SinoDB 时，缺省的备份设备指向/dev/null。在备份工作开始之前，必须把它指向真正的备份设备，一般是磁带机。

如果进行数据空间的备份，要修改 $ONCONFIG 文件的以下参数：

```
TAPEDEV <备份设备名>
TAPESIZE<备份介质的容量(KB)>
```

如果进行日志文件的备份，要修改 $ONCONFIG 文件的以下参数：

```
LTAPEDEV     <备份设备名>
LTAPESIZE    <备份介质的容量(KB)>
```

（二）数据空间的备份

数据空间的备份在 SinoDB 中称为归档（Archive），Archive 可分为三个级别：

0 级：将数据库 Server 上当前的数据全部备份。

1 级：将上次 0 级备份之后所有修改过的数据备份。

2 级：将上次 1 级备份之后所有修改过的数据备份。

当数据量不太大时（比如小于 1G），建议只做 0 级备份。这样可以简化备份工作。在发生故障时，也可以较快恢复。

0 级备份的步骤如下：

（1）以 sinodbms 用户登陆。

（2）将准备存放这次 0 级备份的磁带放入磁带机。

（3）执行 ontape -s -L 0

（4）取出磁带，标上备份日期和备份内容。

（三）逻辑日志的备份

逻辑日志备份和数据空间备份的一个重要区别是有时间性的要求。不管多长时间不进行数据空间备份，数据库 Server 仍然可以正常运行。但是逻辑日志备份就不一样了。因为硬盘上的逻辑日志文件空间是循环使用的，而且是有限的，如果很长时间不进行逻辑日志备份，就可能把逻辑日志文件空间用完。这时，Server 就会处于一种死锁状态，不能进行任何操作。因此，必须在逻辑日志文件空间用完以前及时备份。

用 onstat -l 可以看到逻辑日志文件空间的使用情况。

其中，flags 中的第一列：U 表示已经使用，F 表示没有使用。第三列中：B 表示已经备份到磁带上，-表示尚未备份。由此可知，flags 中的第一列为 U 且第三列为-的 log 文件是正在使用的，其它文件都是空闲的。

注:可以根据系统的逻辑日志计算出数据更新量。

逻辑日志文件备份的步骤如下：

（1）以 sinodbms 用户登陆。

（2）将准备存放这次逻辑日志备份的磁带放入磁带机。

（3）执行 ontape -a。

（4）取出磁带，标上备份日期和备份内容。

（四）数据库的恢复

（1）以 sinodbms 用户登陆。

（2）确认数据库未在运行。(可以用 ps -e | grep oninit)

（3）将存有备份数据的磁带放入磁带机。

（4）执行 ontape -r

注:双机恢复用 ontape -p

（5）执行 onmode -m，使数据库进入 OnLine 状态。

6. 常见问题与处理

6.1 数据库 Hang 住

可执行以下步骤：

a)采集最新状态信息

Command : onstat －a > allinfo.out

b)停止数据库

Command : onmode －ky

c)确定数据库已关闭

Command : onstat －

出现如下讯息:

shared memory not initialized for SINODBMSSERVER

d)确定系统中已无 oninit 进程

Command : ps －ef | grep oninit

e)重启数据库

Command : oninit

f)确定数据库已启动

Command : onstat －

g)收集 online.log

说明 : online.log (存放在 onconfig 配置文件中 MSGPATH 指定路径)

h)如有 assertion fail 发生收集 dump file, af.xxxxx and shmem.xxxxx

说明 : af.xxxxx and shmem.xxxxx (存放在 onconfig 配置文件中 DUMPDIR 指定路径)

6.2 查看数据库系统占用共享内存的情况

```
onstat -g seg
```

应该保证少于 3 块的共享内存区域占用。

6.3 查看数据库系统占用共享内存的情况

故障现象：数据库无法启动在 online.log 中发现如下信息

```
shmget: [EEXIST][17]: key 526a4801: shared memory already exists
mt_shm_init: can't create resident segment
```

该问题是由于操作系统中的共享内存未释放完全导致。使用 ipcs 查看 owner 为 SinoDB 的内存，然后切换到 root 用户，使用 ipcrm －m 清除掉未释放的共享内存即可。

6.4 查看各 chunk 文件的读写量和读写率

```
onstat -g iof
```

6.5 数据库目录结构检查

```
oncheck -cc <database>
```

6.6 数据库索引结构检查

```
oncheck -cI <database>:<tablename><#indexname>
oncheck -pe >a.log
```

6.7 数据库数据结构检查

```
oncheck -cD <dbname>:<tablename>
```

6.8 数据库死锁处理

故障现象：

onstat -ks|grep HDR+X　　//查询是那个表被锁

```
address  wtlist  owner  lklist  type  tblsnum  rowid  key#/bsiz
c1809510  0  d656e774  c181cb3c  HDR+X  6002e1  2c602
0
```

故障分析：

需要关注 lklist 和 type 项，从上面来看 tblsnum 为 6002e1（6292193 十六进制转换成十进制）的表被锁了。可以重查询是那个表被锁：

dbaccess ：select * from systables where partnum='6292193'（或者 select * from systables where hex(partnum)=" 6002e1"）得到

```
tabname     basetab_mvpn
owner       smpmml
partnum     6292193
tabid       12813
rowsize     464
ncols       61
nindexes    1
nrows       2984
created     12/10/2002
version     839843846
tabtype     T
```

```
locklevel    R
npused       746
fextsize     16
nextsize     16
flags        0
```

$onstat -u，将 owner(address)为 d656e774 的线程找出来

```
address    flags    sessid     user    tty     wait    tout locks nreads     nwrites
d656e774 Y--P--- 4261      smp20    -     d6ad2330 0       180     99620       16
```

$onstat -g sql d656e774 可以将这个线程执行过的 sql 语句打印出来。

故障处理：

只要用 sinodbms 用户执行 onmode -z sessid 干掉线程

```
onmode -z 4261
```

其他处理方式：

```
onstat -g ses sessid 找个进程 PID 来,然后 ps -ef|grep Pid; kill -9 pid
```

在处理这些问题时还会遇到表被锁是因为该线程还没有执行完毕，此时就不能简单的 onmode -z 杀线程

注意：分片表的 partnum 需要从 sysfragments 系统表中的 partn 中取出。

6.9 索引说明

索引应建在能有效提高 SQL 操作效率的字段上。

对经常用于连接操作的字段可创建索引。

对经常用做过滤器的字段可创建索引。

对经常出现在 ORDER BY 或 GROUP BY 字段的字段上可创建索引。

避免在高度重复的字段上建索引。

在经常变化的表上不要建过多的索引。

减少索引所占用的的空间，将索引建在长度小的字段上。

不要在同一个字段上建立升序索引和降序索引（对于 7.2 版本以上的 SinoDB 数据库而言）。

对插入操作，选择适当的 FILLFACTOR 来控制索引页的空间。

复合 index，如 abc，对 a、ab、abc 都起作用。在一个有唯一值的字段和有重复值的字段上共同建一个复合 index，有助于多重复值的字段的 insert 操作（增加唯一性）在 join 相关的多个字段上建复合 index，在 Where 条件相关的多个字段上建复合 index。

聚类（clster）index 对相对稳定的表较为有用，能加快查询。聚类和生聚类都需花费大量磁盘空间和时间数据录入时就是有序的，则无需聚类。

对大批量 update 操作，如 load，首先 drop index，再 update,然后再建 index，能提高性能。

6.10 给 dbspace 增加新的 chunk

1. 应该定期用 onstat -d 命令检查每个 dbspace 的剩余空间。当发现某个 dbspace 空间快要用完时，就要及时给它增加新的 chunk。

2. 创建新的 chunk 。

如果采用普通文件作为 chunk(cooked file 方式)，创建一个空文件。然后修改文件的属主和权限，命令如下：

```
# cat /dev/null > <chunk_pathname>
# chown SinoDB <chunk_pathname>
# chgrp SinoDB <chunk_pathname>
# chmod 660 <chunk_pathname>
```

如果采用块设备作为 chunk(raw disk 方式)，最好不直接引用 chunk 的设备名，而是创建一个符号链。

```
/* 首先创建所需大小的 chunk */
# chown SinoDB <chunk_pathname>
# chgrp SinoDB <chunk_pathname>
# chmod 660 <chunk pathname>
# su - sinodbms
$ cd chunklink    /* 如果该目录不存在则先创建它 */
$ ln -s <chunk_pathname> <linkname>
```

3. 将新 chunk 添加到指定 dbspace 中，命令语法为：

```
onspaces -a <dbspace> -p <pathname> -o 0 -s <size>
```

其中：

<dbspace> 为要增加 chunk 的 dbspace 名。

<pathname>为新增 chunk 的文件名或设备(符号链)名。

<size>是新增 chunk 的大小，以 KB 为单位。

4. 再执行 onstat -d 命令，检查 chunk 是否成功加入。

6.11 数据移动

1. 获取 table 的结构

```
dbschema [ -t <tablename> ] -d <dbname>
```

以上命令执行后，会将创建指定 table（包括索引）所需的 SQL 命令显示出来。如果没有指定 tablename，则是指该 database 中的所有 table。

2. 将一个 table 中的数据以文本格式输出到一个文件中。

```
unload to <filename> select * from <tablename>
```

这是一条 SQL 命令，必须在 inaccess 中执行。

3. 将 unload 得到的文本文件格式的数据插入到指定的 table 中。

```
load from <filename> insert into <tablename>
```

这是一条 SQL 命令，必须在 inaccess 中执行。

4．将整个 database 备份成文本格式

```
dbexport <database>
```

这条命令执行后，会在当前目录下创建一个<database>.exp 目录，其中包含了重建该 database 的所有数据。但是其中没有记录 database 所在的 dbspace 和 log 方式。

5．用<database>.exp 重建整个 database

```
dbimport <database> [-d <dbspace>] [-l [buffered]]
```

这条命令应该在执行 dbexport 命令的同一目录下执行。

-d 选项用来指定 database 建在哪个 dbspace 下，缺省值为 root dbspace。

-l 选项用来指定 database 的 log 方式，-l 为 Unbuffered Log，-l buffered 为 Buffered Log，缺省为 No Log。

6.12 finderr 命令

当数据库出错时，一般带有错误号，利用 finderr 命令可以查出数据库系统出错原因及处理建议提示。

6.13 使用 UPDATE STATISTICS 命令优化数据库

每天或每周有大量删除操作后，在业务已全部作完时运行该命令。

6.14 当出现系统性能问题时要从哪些方面考虑？

系统运行效率是多个方面决定的，当出现效率低时，可以从以下几个方面考虑：提高硬件配置，调整数据库配置参数，优化应用程序。运行 UPDATE STATISTICS 优化数据库设计，如 INDEX，FRAGMENTATION 等。

这是由于在该表上，设置了记录级只读锁，如果其他用户正在操作该记录，则其他用户不能操作。

```
1)用 sinodbms 用户登陆
2)dbaccess   Query Lanuage   Database   New
set lock mode to wait 10;
delete from table_name(具体删除语句)
```

6.15 长事务问题处理

系统出现长事务是由于可用的逻辑日志不能满足一个事务的需要，请从以下几个方面考虑：

1）是否及时备份逻辑日志

2）如及时备份了逻辑日志，则需要增加逻辑日志个数，以满足事务的需要。

请从下面两个方面解决所发现的错误：

1）检查是否有足够的锁资源：

用 SinoDB 登陆，运行命令 onstat -p，将出现信息：

```
SinoDB Dynamic Server Version 7.31.UC5      -- On-Line -- Up 1 days 01:50:07 --
28672 Kbytes
Profile
```

```
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
235      250      7175     96.72    187      258      1808     89.66
isamtot  open     start    read     write    rewrite  delete   commit   rollbk
6176     892      1166     1856     443      171      18       45       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        6.04     39.98    4        614
bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
11       0        4035     0        0        0        5        75
ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
4        0        1        5          0
```

如果 lokwait/lockreqs 大于 1%，则需要增加 LOCKS 数.

2）检查磁盘是否还有可用空间：

用命令 onstat -d 检查 dbspace 的空间是否已满？

用操作系统命令 df -v 命令检查文件系统是否已满。

6.16 如何得到错误号?如何查询错误号?

1）参考如下方法获得错误号：SQL 错误号存放在结构 sqlca.sqlcode 中 ISAM 错误号存放在结构 sqlca.sqlerrd[l]中在 ESQL/C 中得到错误号采用如下方法：

prinrf(“SQL 错误号为：%d\nISAM 错误号为：%d\n”,sqlca.sqlcode,sqlca.sqlerrd[1]);

2)使用 finderr error-number 得到错误号参考；

SinoDB 常见错误号有哪些，碰到这些错误号该如何处理？

—201 语法错

SQL 命令中出现不正确的语法时，系统提示该错误号。请检查是否有拼写方面的错误。

—202 语句中有非法字符

该字符无法被正确地解释为 SQL 语句中的一部分，如果出现在执行程序中则有可能是不可打印字符，若如此则请删除该不可打印字符重新执行程序，看是否还有该错误。也可考虑改变目前的字符集，如 export LANG=en_US.8859-1 解决此问题。

—239 无法插入新记录

该表某列上建有唯一性索引（UNIQUE INDEX）因此，插入新记录时，如果存在重复键值，则系统提示该错误号。

—243 请使用 oncheck 检查数据库索引是否出现错误，数据库系统是否有不一致的现象：

用 sinodbms 用户登陆，执行命令

$ oncheck -cl cleardb

—271 表中无法插入新记录

该错误可能有多种产生的可能性，如表被锁住或 dbspace 满，文件系统满，请检查相应的 ISAM 错误号。

—272 无 SELECT 权限

建表者没有向你的帐户或 PUBLIC 授权，作 SELECT 前请让建表者或 DBA 用户向你授权。

—273　　无 UPDATE 权限同上

—274　　无 DELETE 权限

—275　　无 INSERT 权限

—329　　数据库不存在或无系统权限

你所准备访问的数据库对服务器不可见，请检查是否有拼写错误，或忘记写全服务器名。

—349　　数据库还没有选择。

当前命令不能执行，因为没有当前数据库，要么数据库还没建立要么数据库被 CLOSE DATABASE 关闭了。

—359　　不能 DROP 当前数据库

当前被打开的数据库不能被 DROP，请先 CLOSE　DATABASE 再 DROP DATABASE。

—369　　非法的序列号

安装产品时，出现误操作，请检查是否设置了正确的环境变量如 SINODBMSDIR，咨询相关安装人员并检查安装情况。

—378　　记录目前被其他用户锁住

当前命令无法访问所需要的记录。因为被锁住，在程序中可以通过设置 SET LOCK MODE TO WAIT 防止部分这种错误的产生；

—387　　无连接权限

命令中所需访问的数据库无法访问，因为还没有被授予 CONNECT 权限，请与数据库管理员联系，并让他向你授予 CONNECT 权限。

—388　　无资源权限

CREATE 命令无法执行，因为你的账户还没有被授予 RESOURCE 权限，在建永久表和索引时必需具备 RESOURCE 权限，请与数据库管理员联系，并请其向你授予 RESORCE 权限。

—389　　无 DBA 权限

—391　　某无法插入 NULL 值列

命令准备向已定义成非空的字段插入空值。

—425　　数据被其他用户使用

请求的数据库被其他用户以排他方式打开，等一会儿再执行该命令或等数据库空闲时再试。

—457　　应用程序正在工作的进程或线索被意外中止，可能 DBA 关掉了系统，请查看操作系统信息。

—458　　长事物中断

—459　　SinoDB-Online 被关闭

应用程序正在使用的进程被操作员关闭，当前事务会在数据库服务器再次启动时自动回滚。

—535　　已经在事务中

BEGIN WORK 冗余，一个事务已经在。

—668　　系统调用出错，导致该错误的可能如下：

* 通过系统调用的系统程序不存在
* 系统程序不在可见的目录下
* 系统程序运行时出错,如对某些目录的写权限等。

—753　　访问失败，单用户限制被超出。

授权增强配置成单用户形式，多用户不能在同一时间使用该数据库，如果从其他计算机访问，则该产品仅限非网络环境。

—931　　在/etc/services 文件中无法定位端口号。

/etc/services 文件中无此端口号，请检查$SINODBMSDIR/etc/sqlhosts 文件中的端口号与/etc/services 文件中的端口号是否一致。

—932　　错误的网络连接

请检查网络配置文件的正确性。

—952　　远程主机的口令错

其他计算机上的数据库不接受你输入的口令，检查是否输入了正确的口令。

6.17　观察共享内存的 BUFFER 参数

运行命令 onstat　-p，将出现以下信息：

```
Sinoregal SinoDB Dynamic Server Version 16.8.FC7 -- On-Line -- Up 34 days 20:07:18
-- 194056 Kbytes
Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
87       105      523      94.29    7        7        2        0.00
isamtot open     start    read     write    rewrite  delete   commit   rollbk
251      47       46       81       0        0        0        0        0
gp_read  gp_write gp_rewrt gp_del   gp_alloc gp_free  gp_curs
4        0        0        0        0        0        2
ovlock   ovuserthread ovbuff   usercpu  syscpu   numckpts flushes
0        0               0      3.83     0.22     1        2
bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
6        0        177      0        0        0        0        5
ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
2        0        0        2          0
```

结果分析：持续观察 bufreads %cached 和 bufwrits %cached 值，如前者小于 95%，后者小于 87%，则需要增加 BUFFERS 值。

6.18　批处理时系统 Check-Point 时间很长，怎么办？

用 sinodbms 用户登陆改配置文件$SINODBMSDIR/etc/$ONCONFIG，调整参数 LRU_MAX_DIRTY 和 LRU_MIN_DIRTY，将参数 LRU_MAX_DIRTY 60 调整为 30，参数 LRU_MIN_DIRTY 50 调整为 20。

另 buffers 参数对做 checkpoint 的时间影响也很大。

6.19　观察共享内存的使用情况

用 sinodbms 用户登陆，执行命令：$ onstat　-g　seg 将出现信息

```
Sinoregal SinoDB Dynamic Server Version 16.8.FC7 -- On-Line -- Up 34 days 20:07:18
-- 194056 Kbytes
Segment Summary:
id          key          addr     size       ovhd     class blkused  blkfree
101         1381386241 10000000  20971520   1172     R       2412       148
102         1381386242 11400000  8388608    720      V       425        599
103         1381386243 11c00000  4194304    656      M       42         470
Total:      -            -       33554432   -        -       2879       1217
```

其中：class 表示共享内存的内容

R—驻留部分 V—虚拟部分部分 M—信息部分 size—表示各部分的大小 blkused—表示已经用的块数量 Blkfree—空闲块的数量

结果分析：持续观察共享内存的情况，如果 V 部分 blkfree 值很小，则需要增加 SHMVIRTSIZE 值。

6.20 检查 DBSPACE 的使用情况

查看数据库系统的 Dbspaces 和 Chunks 的分布和使用情况。

以 sinodbms 用户登陆，执行以下命令：$ onstat -d

```
Sinoregal SinoDB Dynamic Server Version 16.8.FC7 -- On-Line -- Up 34 days 20:07:18
-- 194056 Kbytes
Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
804178a2 1        1        1        1        N        SinoDB
rootdbs
804178a4 2        1        2        2        N        SinoDB
cleardbs
804178b1 3        1        3        1        N        SinoDB
logdbs
804178a9 4        1        4        1        N        SinoDB   tmpdbs
active, 8 total
Chunks
address  chk/dbs offset   size     free     bpages   flags    pathname
8041730a 1       1        0        5000     2261              PO-
/home/sinodbms/rootdbs
8041730c 2       2        0        5000     4947              PO-
/home/sinodbms/cleardbs
```

```
    804173a4 3    2          0          500       497                    PO-
/home/sinodbms/cleardbs1
    8041730c 4    3          0          5000      4947                   PO-
/home/sinodbms/logdbs
    804173a4 5    4          0          500       497                    PO-
/home/sinodbms/tmpdbs
    active, 8 total
```

观察结果：

chk/dbs 前一数字 chk 表示 chunk 编号，后一数字 dbs 表示 dbspace 编号，如 dbspace 编号相同则表示其相应前面编号 chunk 属同一 dbspace

size 该 chunk 的大小，单位为 page

free 该 chunk 的空闲空间

flags 表示目前 chunk 状态

注意事项：

A：如某一dbspace 的所有 chunk 空闲很小，则需要增加 chunk

B：flags 正常标志为 PO-，否则为异常.出现异常情况时，flags 标志的第二位为：

D：表示该 chunk down 了

I：表示数据不一致（Inconsistent）

P：表示 primary O：表示 Online

6.21 监测数据库日志文件

数据库日志文件 Message Log，即 online.log 文件包含了一些系统运行状态信息，其中一些是正常信息，另外一此是异常信息。常见的正常信息包括：

A．状态的改变

```
    09:29:07    DR: DRAUTO is 0 (Off)
    09:29:08    SinoDB-Onlline Initialized -Shared Memory Initialized.
```

B．快速恢复信息

```
    14:42:46    Physical Recovery Started.
    14:42:46    Physical Recovery Complete: 0 Pages Restored.
    14:42:46    Logical Recovery Started.
    14:42:50    Logical Recovery Complete.
    0 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks
```

C．检查点的记录和间隔时间

```
    14:47:05    Checkpoint Completed:    duration was 3 seconds.
```

D．配置参数的改变信息

```
    18:42:54    Onconfig parameter SHMVORTSIZE modified from 200000 to    8000.
```

E．动态分配内存信息

```
    18:42:54 Dynamically allocated new shared memory segment (size 8388608)
```

6.22 如何找对应应用占用内存较多的进程和长时间执行的 sql 语句

```
$ onstat -g ses

Sinoregal SinoDB Dynamic Server Version 16.8.FC7 -- On-Line -- Up 34 days 20:07:18
-- 194056 Kbytessession                                          #RSAM    total
used
id          user        tty       pid     hostname threads  memory     memory
54          SinoDB -          0       -          0          8192       4848
41          smctest   3           1102    sun        1          57344      33688
40          smctest   3           1104    sun        1          49152      40728
39          smctest   3           1103    sun        1          172032     91200
3           SinoDB -          0       -          0          24576      16704
2           SinoDB -          0       -          0          8192       4848

$ onstat -g ses 39

SinoDB Dynamic Server Version 7.30.UC7      -- On-Line -- Up 00:41:35 -- 842160
Kbytes

session                                              #RSAM    total      used
id          user        tty       pid     hostname threads  memory     memory
39          smctest   3           1103    sun        1          172032     107992

tid         name        rstcb       flags       curstk     status
58          sqlexec   3cedd2e8 Y--P---   1400        3cedd2e8 cond wait(netnorm)

Memory pools       count 1
name              class addr        totalsize freesize #allocfrag #freefrag
39                V       3d196018 172032      64040     473          29

name              free        used             name              free        used
overhead          0           120              scb               0           96
```

```
opentable       0        9800        filetable       0        536
ru              0        224         log             0        2152
temprec         0        1608        keys            0        864
ralloc          0        65536       gentcb          0        11856
ostcb           0        2008        sqscb           0        8008
rdahead         0        160         hashfiletab     0        280
osenv           0        1744        sqtcb           0        2248
fragman         0        752

Sess    SQL             Current          Iso Lock        SQL    ISAM F.E.
Id      Stmt type       Database         Lvl Mode        ERR    ERR    Vers
39      -               smc              CR  Wait 20     0      0      7.24

Last parsed SQL statement :
UPDATE deliveried_msgs SET status = 'ENROUTE' , last_state_time =
        '0307031337' WHERE ( message_id = '00001683' )
```

6.23 检查大表和对大表的维护

先做 update statistics；然后，执行语句：

```
select tabname, nrows, rowsize, fextsize, nextsize
from systables order by nrows desc
```

找到较大的表的表名，并检查其 fextsize, nextsize 的值如果记录数很多（百万级以上），但是 fextsize 为 16，则表示此表需要重建；对这样的表先进行数据备份和表结构备份；修改导出的表结构备份脚本，设置 fextsize 和 nextsize，并对表进行分片；SQL 语法如下：

```
create table xxx(a int, … …) extent size A next size B
init fragment by round robin in
pdbsx1, pdbsx2, pdbsx3, pdbsx4, pdbsx5, pdbsx6, pdbsx7, pdbsx8;
```

其中 A 为上面语句中查出的 nrows*rowsize，单位为 K，B 的大小通常定义为 A 的一半，x 为 1 或 2（分别表示第一组分片表空间和第二组分片表空间）；按照新的脚本重建表并导入数据。

6.24 STMT_CACHE_DEBUG 报-19817 错误

数据库报以下错误：

```
[-19817]: Invalid syntax for STMT_CACHE_DEBUG environment variable.
```

故障处理：配置 STMT_CACHE_DEBUG 参数

```
export STMT_CACHE_DEBUG=1:/tmp/file.out
```

6.25 双机由于网络中断导致都变为主机状态如何处理？

确定数据为最新的状态正常的主机(prim)，down 掉不正常的一台主机。(如果这时直接恢复很有可能不成功)。切断主机的主从关系。Onmode -d standard 如果不成功，需要用 onmode -ky 关闭数据库。从新启动后，用 oninit -D 启动正常后，做 0 级备份，重新建立主从关系，恢复主从机

6.26 双主机故障的解决建议

网络情况不好的 DRINTERVAL=30，异步更新。如果出现以下提示，建议如果不能正常恢复主从机，先把主从关系断开，做备份，建立主从关系，恢复从机。

```
DR: Server state incompatible
```

7. 性能调整

7.1 影响 CPU 的性能

7.1.1. 影响 CPU 使用率的配置参数和环境变量

Onconfig 配置文件中的下列参数对 CPU 的利用率有明显的影响：

NUMCPUVPS

SINGLE_CPU_VP

MULTIPROCESSOR

AFF_NPROCS

AFF_SPROC

NUMAIOVPS

OPTCOMPAND

NETTYPE

1．NUMCPUVPS、MULTIPROCESSOR 和 SINGL_CPU_VP

NUMCPUVPS 参数规定了数据库启动时 CPU VP 的数量。分配的 CPU VP 的个数不要超过可以为它们服务的 CPU 的个数。对于单处理器的计算机系统，SinoDB 建议使用一个 CPU VP。对于有 4 个以上 CPU、主要用做数据库服务器的多处理器系统，SinoDB 建议设置 NUMCPUVPS 的值等于处理器总数减一。对于双处理器系统，运行两个 CPU VP 可能会改善性能。这需要监控操作系统的 CPU 使用情况。可以使用操作系统命令 sar 或 vmstat。

如果运行多个 CPU VP，应将 MULTIPROCESSOR 设置为 1，当设置 MULTIPROCESSOR 为 1 时，Online 以对应于多处理器的方式执行锁定。否则，设置该参数为 0。

注意：如果设置 SINGLE_CPU_VP 参数为 Y，则 NUMCPUVPS 参数的值也必须是 1，如果后者大于 1，Online 就不能初始化并显示下面的错误信息：

Cannot have 'SINGLE_CPU_VP' now-zero and 'NUMCPUVPS' greater than 1

2．AFF_NPROCS 和 AFF_SPROC

在支持数据库和客户应用的系统上，可以通过操作系统把应用连接到某些特定的 CPU。这样做可以有效地保留剩余的 CPU 给数据库 U VP 使用，它们是用 AFF_NPROCES 和 AFF_SPROC 配置参数连接到剩余 CPU 的。

AFF_NPROCS 指定了连接到数据库 U VP 上的 CPU 的个数。连接一个 CPU VP 到一个 CPU 会引起该 CPU VP 在这个 CPU 上的排它性运行。

AFF_SPROC 指定了数据库把 CPU VP 连接到 CPU 上时所启动的 CPU。

AFF_NPROCS 规定了计算机上可以绑定 CPU VP 的 CPU 的数目。NUMCPUVPS 参数指定了 Online 将启动的 CPU VP 的数目，AFF_SPROC 参数指定了 Online 连接第一个 CPU 序号。例如，某个数据库系统所在的硬件平台有 8 个 CPU，AFF_NPROCS 设置为 8（即可用于绑定 CPU VP 的 CPU 有 8 个），NUMCPUVPS 设置为 3，AFF_SPROC 设置为 5，则 3 个 CPU VP 需要绑定到 CPU 上，是从第五个 CPU 开始，绑定到第五、六、七个 CPU 上。需要注意的是，AFF_SPROC 的取值是在 0 和（AFF_NPROCS - NUMCPUVPS + 1)这两个值之间的，不能大于后者。

3．NUMAIOVPS

参数 NUMAIOVPS 指定最初产生的 AIO VP 的数目。如果所在的操作系统不支持核心异步 I/O（KAIO），Online 使用 AIP VP 来处理所有数据库 I/O 请求。推荐的 AIP VP 数目取决于 Online 使用的硬盘个数。如果所在操作系统不支持或没有使用 KAIO，则 SinoDB 建议对包含数据库表的每一个磁盘分配一个 AIO VP。可以对 Online 频繁访问的每六块增加额外的 AIO VP。如果所在的操作系统使用 KAIO VP，CPU VP 将直接向操作系统发出原始的 I/O 请求。在这种情况下，可以只配置一个 AIO VP，此时 AIO VP 只处理文件系统方式的 chunk。如果文件系统方式的 chunk 有增加时，可以增大 AIO VP 的数目。

分配 AIO VP 的目的是要分配足够的 AIO VP 以便 I/O 请求队列的长度保持很短，即队列中保持尽可能少的 I/O 请求。

4．OPTCOMPIND

OPTCOMPIND 参数帮组优化程序为应用选择合适的访问方法。

如果 OPTCOMPIND 等于 0，优化程序给予现存索引优先权，即使在表扫描比较快时。

如果 OPTCOMPIND 设置为 1，给定查询的隔离级设置为 Repeatable Read 时，优化程序才使用索引。

如果 OPTCOMPIND 等于 2，优化程序选择基于开销选择查询方式，即使表扫描可以临时锁定整个表。

5. NETTYPE

NETTYPE 参数为数据库实例支持的每个连接类型配置轮询线索。如果 sqlhosts 文件中支持一个以上的接口或协议的连接，就必须对每个连接类型规定独立的 NETTYPE 参数。也即，每中与数据库服务器名字有关的连接类型都需要单独指定一个 NETTYPE 参数。

每个用 NETTYPE 表项配置或动态加入的轮询线索在不同的 VP 上运行，轮询线索可以在两类 VP 上运行：NET VP 和 CPU VP。为得到最佳性能，SinoDB 建议使用 NETTYPE 表项

为 CPU VP 类只分配一个轮询线索，将其余轮询线索轮询线索分配给 NET VP。分配给任何一种连接类型的轮询线索不得超过 NUMCPUVPS 的取值。

单 CPU 计算机上每个轮询线索的最佳连接个数不超过 300，多 CPU 机上可多达 350 个。但一个轮询线索最多支持 1,024 甚至更多的连接。

NETTYPE 的配置格式如下：

```
NETTYPE        connection_type，poll_threads，c_per_t，vp_class
```

connection_type　　标识轮询线索分配的连接协议。

poll_threads　　是分配给该连接类型的轮询线索数目。对任何连接类型，这个值不能超过 NUMCPUVPS 值。

c_per_t　　是每个轮询线索的连接数目。可以用如下公式计算这个值：

c_per_t = connections / poll_threads

connections　　是所希望指定的连接类型支持的最大连接数。对于共享内存连接（ipcshm），该值应该加倍以获得最好的性能。

vp_class　　是可运行轮询线索的 VP 类。如果 CPU VP 上只运行一个轮询线索，那么指定为 CPU VP。为了达到最好性能，当要求多个轮询线索时应该指定为 NET VP。

如果 c_per_t 的值超过了 350，而当前连接的轮询线索数小于 NUMCPUVPS，可以增加轮询线索数目，但不能超过 NUMCPUVPS，然后重新计算 c_per_t 的取值。

注意：每个 ipcshm 连接需要一个信号量。当 c_per_t 的值很大时，对于某些操作系统要相应增加信号量。

7.1.2. 监控系统 CPU 的使用状况

1. 使用 UNIX 的监控工具 sar 或 vmstat 来监控 CPU 的使用情况。

例：sar -u 5　3

```
                %usr      %sys      %wio      %idle
10:06:22  34      1         0         65
10:06:27        34        2         0         64
10:06:32        34        1         0         65
```

连续监控%idle 来确认 CPU 没有超载。如果%sys 的值很大则可能应用有问题。

2. 监控 CPU vp 的方法

```
onstat   -g   glo
```

将出现信息：

```
Virtual processor summary:
class          vps          usercpu     syscpu      total
```

```
cpu              1            5.90         2.01         7.91
aio              6            0.07         15.69        15.76
lio              1            0.01         6.08         6.09
pio              1            0.01         0.43         0.44
adm              1            0.06         15.91        15.97
msc              1            0.00         0.01         0.01
total            11           6.05         40.13        46.18

Individual virtual processors:
vp      pid          class          usercpu      syscpu       total
1       6534         cpu            5.90         2.01         7.91
2       6535         adm            0.06         15.91        15.97
3       6536         lio            0.01         6.08         6.09
```

可以通过该监控看出 CPU 忙占用的时间(隔 60 秒分别监控结果）。如果非常忙，则需要增加 CPU VP。

```
onstat -g rea
```

将出现信息：

```
Sinoregal SinoDB Dynamic Server Version 16.8.FC7 -- On-Line -- Up 34 days 20:07:18 -- 194056 Kbytes
Ready threads:
 tid        tcb          rstcb        prty status                    vp-class        name
```

如果有大量的线索在等待队列中，则说明需要增加 CPU VP。

7.2 影响内存的性能

7.3.1. 影响内存使用效率的 Online 配置参数

（一）SHMVIRSIZE

SHMVIRTSIZE 参数规定了初始分配给数据库的共享内存的虚拟区的大小。共享存储器的虚拟区存储与会话、请求有关的数据及其它信息。虽然数据库按处理大型查询或高峰负荷的需要增加共享内存给虚拟区，但共享内存的分配增加事务处理的时间，SinoDB 建议设置 SHMVIRTSIZE 以提供一个满足一般日常操作需要的虚拟接口。

（二）SHMADD

SHMADD 参数规定数据库自动加到虚拟区的共享内存增量的大小。在决定该值的大小时有些折中因素。增加共享内存要占用 CPU 周期：每次的增加量越大，增加次数就越少，留给

其它的进程的内存也越少。通常采用大增加量，但当内存负荷很重时，少量增加使其他程序更好的共享内存资源。SinoDB 有如下建议：

内存大小	SHMADD
<=256MB	8192KB(default)
256--512MB	16,384KB
>=512	32,768KB

（三）BUFFERS

BUFFERS 是可以用于 Online 的数据缓冲区数。这些缓冲区驻留在驻留区，用来缓存主存中的数据库的数据页。可用的缓冲区越多，所需的数据页就越可能用于前一次请求而已经在内存里。这个参数对数据库 I/O 和事务处理吞吐量有明显的影响。但是，分配过多的缓冲区会影响内存系统并导致过多的页面活动。SinoDB 建议设置 BUFFERS 为物理内存(以 MB 为单位)的 20%到 25%。实际 BUFFERS 的单位为页，不同操作系统的页大小是不同的，因此需要计算。使用 onstat -p 监控读缓存的频率。这个频率代表一个查询请求的数据库页已经在共享内存里的百分比。（还没有存在的页必须从磁盘拷贝到内存中）。如果此值很低，可增加 BUFFERS 并重新启动 Online。在增加 BUFFERS 值时，到达某一点后，增加 BUFFERS 也不再明显改善读缓存的频率，或者达到操作系统共享内存分配的上限。如果读高速缓存的比率很高，则应下调 BUFFERS 并重启动 Online。

（四）RESIDENT

RESIDENT 参数规定是否强制共享内存驻留作为 Online 共享内存驻留区。这个参数只对支持强制驻留的机器有效。Online 中的驻留区，包含用于数据库读写作业的 LRU 队列。

（五）LOCKS

参数 LOCKS 设置任意时刻可用的锁的最大数量。Online 中每个锁需要占用驻留区段的 44 个字节，分配共享内存时要考虑锁所用的资源。一般锁可以分配的大些，对应用比较忙的系统可以到 800 万以上。

（六）LOGBUFF

参数 LOGBUFF 指定为三个用来保存逻辑日志记录的缓冲区分别保留的共享内存的数量。这些缓冲区保存着逻辑日志记录，直到它们被刷新到硬盘上的逻辑日志文件。缓冲区的大小决定了它被添满的频率，从而决定了它必须被刷新到硬盘上的逻辑文件中的频率。

（七）PHYSBUFF

参数 PHYSBUFF 指定为两个用来暂时保存将被修改的数据页的缓冲区分别保留的共享内存的数量。缓冲区的大小决定了它被添满的频率，从而也决定了它被写到硬盘上的物理日志的频率。

7.3.2. 如何监控内存使用情况

1. 使用 onstat -g seg 命令监控共享内存的 segments。

```
$onstat -g seg    （结果参阅前面介绍）
```

这里三行分别代表了驻留内存段（class 为 R)、虚拟内存段(class 为 V)、消息内存段(class 为 M)。blkused 和 blkfree 分别代表使用空间和空闲空间。如果虚拟内存段的 blkused 频繁增加，则需要将 SHMVIRTSIZE 和 SHMADD 相应调大，调整后重新启动 Online。

2. 使用 onstat -p　（结果参阅前面介绍）

1）ovlock 指出分配的 locks 的不足量，如果该值持续增长，则需要增大参数 LOCKS 的值。

2）ovbuf 指出分配的 buffers 的不足量，如果该值持续增长，则需要增大参数 BUFFERS 的值。

3）lockwaits/lockreqs * 100 应该小于 1%，如果这个计算值比较高，则应有如下考虑：

a. 是否用了太多的 page level locks。如果是，可以考虑用 row level locks。

b. 考虑用了 table level lock 的应用是否可以用其它类型的 lock。

c. 是否有太多的 isolation 设置为 Repeatable Read 和 Cursor Stability。确定是否可以使用更多的 Dirty Read 来替代。

4）bufreads %cached 的值指出 buffer 读的百分比，该值建议大于 95%，否则增大 BUFFERS，bufwrits %cached 的值指出 buffer 写的百分比，该值建议大于 85%，但太大如大于 97%则可以将 BUFFERS 相应减少些。

7.3 影响 I/O 的性能

7.3.1. 影响 I/O 配置参数

（一）CKPINTVL，PHYSFILE

CKPINTVL 参数指定检查点之间的时间间隔。当检查点间隔到了，则系统执行检查点操作。但如果这期间的所有数据物理上是一致的，Online 可以跳过检查点操作。另外，一旦物理日志（PHYSFILE）的 75%已满，检查点也会发生。通过设置 CKPTINTVL 为长时间间隔，可以利用物理日志容量来触发基于实际数据库活动而不是任意时间单位的检查点操作。但是，使用长检查点间隔回增加失败事件之后的恢复时间。

（二）LRUS、LRU_MAX_DIRTY 和 LRU_MIN_DIRTY

LRUS 参数指示共享内存缓冲池中设置的最近最少使用（LRU）队列数目。配置较多的 LRU 队列将允许有更多的页清除器操作，并减少每个 LRU 队列的大小。对于单 CPU 系统，

SinoDB 建议设置 LRUS 参数为最小值 4。对于多 CPU 系统，SinoDB 建议设置 LRUS 为最小值 4 和 NUMCPUVPS 的取值之中较大的一个。

可以用 LRUS 和 LRU_MAX_DIRTY 及 LRU_MIN_DIRTY 来控制在满的检查点之间页被刷新到磁盘的频度。在某些情况下，通过设置这些参数，使得在检查点发生时需要刷新的修改的页数量很少，可以达到高的吞吐量；这样，检查点的主要功能是更新物理日志和逻辑日志文件。

（三）CLEANERS

CLEANERS 参数指定执行的页清除线索的数目。对于少于 20 磁盘的系统，SinoDB 推荐 CLEANERS 的取值为磁盘的个数。对于 20 至 100 的磁盘的系统，SinoDB 推荐每两个磁盘分配一个 CLEANERS。对于更多的磁盘系统，SinoDB 推荐每四个磁盘分配一个 CLEANERS。

7.3.2. 监控系统的 I/O 情况

使用 onstat -g ioq，onstat -g iof, onstat -d 监控磁盘的负载情况：

1.运行命令$onstat -g ioq

```
AIO I/O queues:
q name/id    len  maxlen  totalops  dskread  dskwrite  dskcopy
  adt    0     0       0         0        0         0        0
  msc    0     0       1         8        0         0        0
  aio    0     0       4         75       42        2        0
  pio    0     0       1         3        0         3        0
  lio    0     0       1         9        0         9        0
gfd    3     0       16        410      235       175      0
```

如果 aio 队列很大，则可增加一个 AIO VP。

如果某些 class 为 gfd 所对应的 len 和 maxlen 非常大，则需要考虑你的数据分布是否合理，记住这些 gfd 所对应的 hvp-id 的值，再通过 onstat -g iof 查出是那几个设备，

运行命令 onstat -g iof

将出现信息：

```
AIO global files:
gfd    pathname        totalops    dskread    dskwrite    io/s
  3    rootdbs           303         235         68         0.0
```

这里 gfd 的值等于 onstat -g ioq 中那几个 name/id 的值所对应的 pathname，就是 I/O 负载较大的设备。用 onstat -d 可确定是哪个 dbspace。则可以考虑重新分配磁盘或给表分片。

8. SQL 监测与优化建议

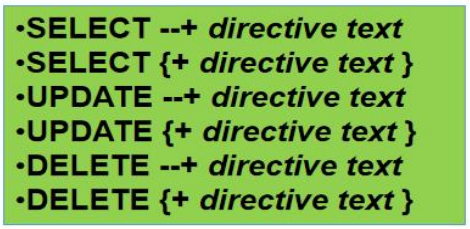
8.1 优化单条查询语句性能

8.1.1. PDQ & Fragment

1、启用 PDQPRIORITY :
查询语句应用分片表
查询语句应用更新统计排序，索引创建；
2、表分片
应用分片于高活跃性表
监控高活跃性表
onstat －D、onstat －g iof、onstat －g iov、onstat －g ppf

8.1.2. 指示符(directives)

使用 SQL 语句指示符(directives)，以达到更好的性能，强制优化器使用特定的索引或连接路径，使用以下指令将得到更好的性能；



解释模式指令：

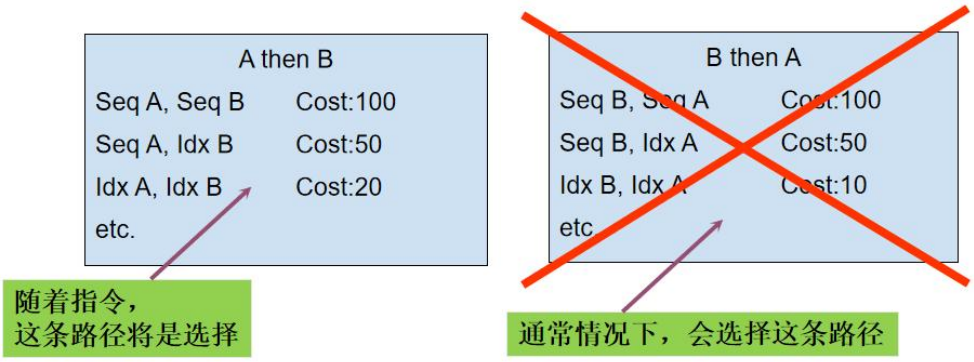
- EXPLAIN：打开 SET EXPLAIN ON 为指定的查询
- EXPLAIN AVOID_EXECUTE：查询计划只是印出并不执行查询，也可以不执行指令
- SET EXPLAIN ON AVOID_EXECUTE;

```
With AVOID_EXECUTE:
SET EXPLAIN ON;
DELETE /*+ EXPLAIN AVOID_EXECUTE */
FROM x WHERE y=10;
```

Delete 将不会执行，但执行计划将被写入。
优化指示符-示例

```
select --+ORDERED * from A, B where A.join_col = B.join_col
```

随着指令(directives)，ORDERED，优化器 从 A 先读，再则 B。若无指令(directives)，最低的成本路径，优化器 从 B 先读，再则 A。



8.1.3. 连接(Join)和排序

1、避免或减少重复的大表的排序：如果顺序的字段在索引之中，无需另外排序。

2、通过设置 PSORT_NPROCS 的值 > 1 或 0 并行排序：并发排序并非仅限于 PDQ 查询。

3、使用临时表，以减少排序的数据集：为 hash joins and aggregates 查询分配更多的内存：

- DS_NON_PDQ_QUERY_MEM
- 默认值是 128KB
- 最大允许 25%的 DS_TOTAL_MEMORY

8.1.4. 语句缓存

如果多个用户执行相同的 SQL 语句，可以考虑使用 SQL 语句缓存。

例如：如果 100 人同一天执行同一个应用程序，他们都执行了这条查询语句：

```
SELECT * FROM ORDERS WHERE order_num = :hostvar
```

语句应该在准备（Prepare）时完全符合，不完全符合如下：

```
SELECT * FROM customer, orders
WHERE customer.customer_num = orders.customer_num
AND order_date > "01/01/97"
```

```
SELECT * FROM customer, orders
WHERE customer.customer_num = orders.customer_num
AND order_date > "01/01/1997“
```

8.2 SQL 优化

8.2.1. 字符串的数据类型

1、当字符串的长度小于或等于 10 时，考虑用 char 替代 varchar 或 lvarchar。

```
CREATE TABLE new_table
(
        COL1    INTEGER NOT NULL,
        COL2    VARCHAR(4)                      (实际是 4 bytes + 1 bytes ending)
)
```

```
CREATE TABLE new_table
(
        COL1    INTEGER NOT NULL,
        COL2    CHAR(4)
)
```

2、当字符串的长度大于 10 时，考虑用 varchar 或 lvarchar 替代 char

```
CREATE TABLE new_table
(
        COL1    INTEGER NOT NULL,
        COL2    CHAR(200)
)
CREATE TABLE new_table
(
        COL1    INTEGER NOT NULL,
        COL2    VARCHAR(200)
)
```

8.2.2. 改写过滤条件

示例 1 如下：

```
select compress_date, s_hour, gz_nw (ne_id, ne_type)
from tpa_sum
where s_hour in (10, 20)
and compress_date >= "2003-01-01"
and compress_date <= "2008-12-12"
and compress_date >= "2006-01-01"
and compress_date <= "2009-01-08"
```

```
order by compress_date and s_hour
```

compress_date 相关的过滤条件可以精简成：

```
compress_date <= "2008-12-12"
and compress_date >= "2006-01-01"
```

示例 2 如下：

```
select distinct line_number
from line_log
where (send_time > "20070516000000")
and (customer_type <> "u")
and (customer_type <> "v")
and (customer_type <> "x")
and (customer_type <> "y")
and (customer_type <> "z");
```

与 customer_type 相关的过滤条件可以改写成：

```
Customer_type not in ("u", "v", "x", "y", "z")
```

8.2.3. 子字符串

尽量避免不包含首位的子字符串，基于某一列上不包含首位的子字符串的过滤器需要数据库服务器测试该列上的每一个值。非首位的子字符串无法使用索引。示例如下：

```
SELECT * FROM customer WHERE zipcode[4,5] > '50'
```

上面这条 SQL 语句的运行时间要大大长于下面这条 SQL 语句

```
SELECT * FROM customer WHERE zipcode[1,2] > '50'
```

8.2.4. Union

尽在可以确定结果集不会有重复的情况下，使用 union all 而不是 union 。如下：

删除重复必须排序：

```
select employee_num from red_region
      union
select employee_num from green_region
```

不删除重复不须排序：

```
select employee_num from red_region
      union all
select employee_num from green_region
```

合并 union all 的各部分

```
select * from u_product
where business_id = 302
union all select * from u_product
where business_id = 303
union all select * from u_product
where business_id = 304
```

```
union all select * from u_product
where business_id = 305
```

合并 union all 的各部分，得到如下的 SQL 语句：

```
select * from u_product
where business_id in (302, 303, 304, 305)
```

8.3 SQLTRACE

8.3.1. SQL 跟踪的开启与执行

SQL 跟踪数据存放于数据库共享内存中的 SQLTRACE 缓存中，SQL 跟踪功能可通过设置 SQLTRACE 配置参数开启；

```
SQLTRACE level=(low|med|high),ntraces=<#>,
        size=<# (# is in kbytes)>,mode=(global|user)
```

使用 SQL Admin API 开启 SQL 跟踪功能

```
execute function sysadmin:task("set sql tracing on",      4000, "2k", "high", "global");
```

注释：每个跟踪缓存放的大小设置为:~ 2 kbytes (~ 2048 bytes).

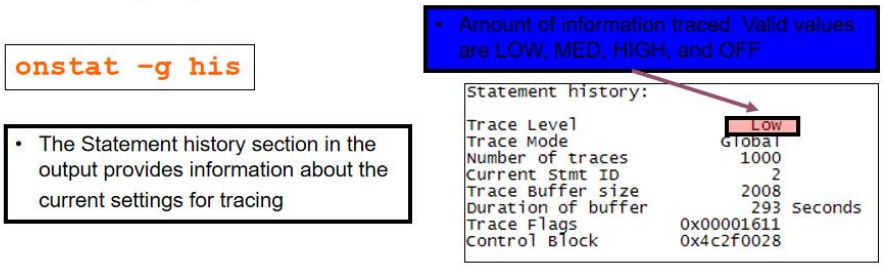
SQLTRACE 分配的内存大小为:2048 * 4000 = 7.8 Mb

8.3.2. SQL 跟踪取得历史数据 - onstat -g his

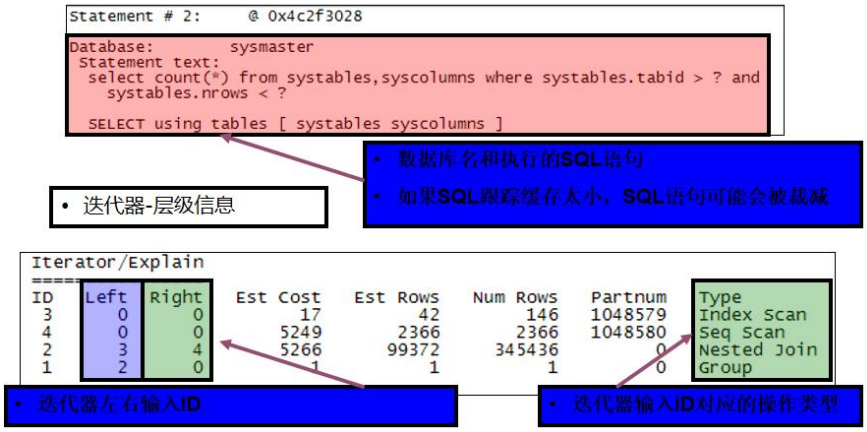
onstat－g his 命令显示 SQL 跟踪信息，这些信息保存在 SYSMASTER 数据库中的一组系统表中 (syssqltrace, syssqltrace_info, syssqltrace_hvar and syssqltrace_itr)

保存在系统表中的 SQL 历史跟踪数据是由所设置的 SQL 跟踪级别决定的。设置不同的 SQL 跟踪级别将影响 onstat －g his 命令所显示的历史跟踪信息。

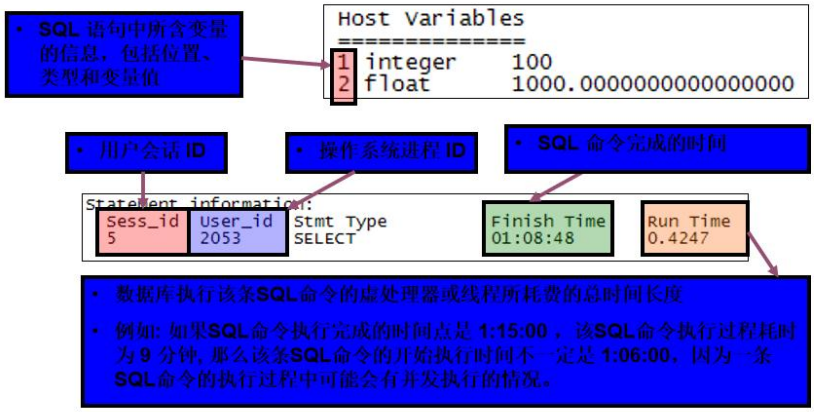
SQL 跟踪系统表 syssqltrace 中的每一行数据描述的是已经完成的 SQL 执行信息。



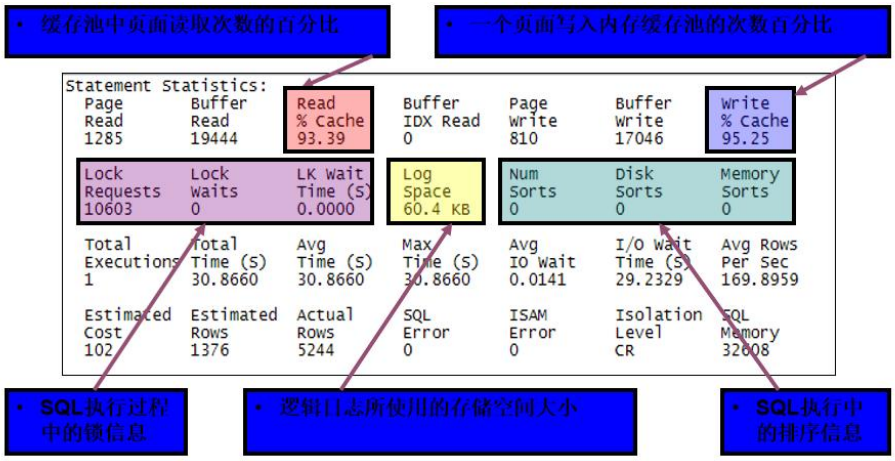
在每次执行完一条 SQL 语句后，下面的 SQL 跟踪信息段都会重复记录，在这个例子中 SQL 命令包含了两个 host variables：

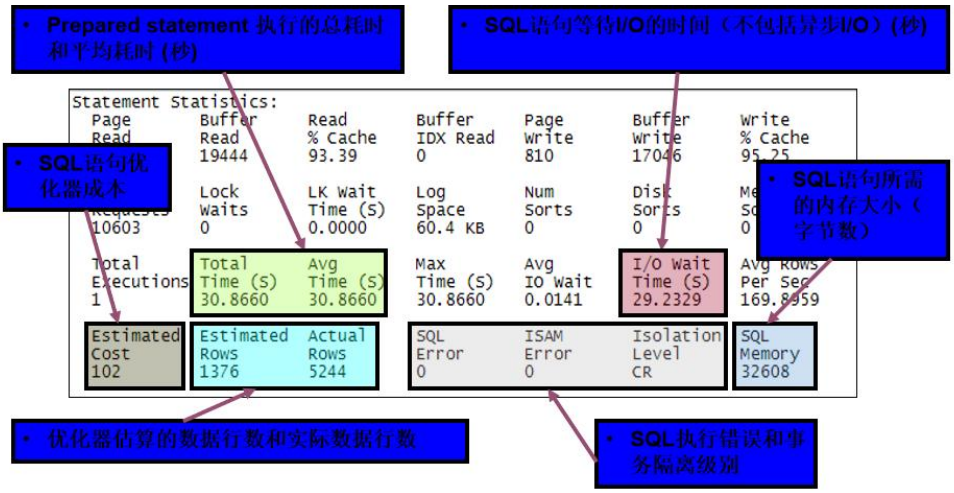


如果 SQL 语句含有一个或多个变量，且 SQL 跟踪级别设置为 High，则 SQL 跟踪的输出信息中将包括变量信息段：



SQL 跟踪信息输出中的 Statement Statistics 部分是关于 SQL 语句执行过程中的特定统计信息：





8.3.3. SQLTRACE 举例 - 开启和关闭跟踪

onstat－g his 命令显示 SQL 跟踪信息，这些信息保存在 SYSMASTER 数据库中的一组系统表中 (syssqltrace, syssqltrace_info, syssqltrace_hvar and syssqltrace_itr)

开启 SQL 跟踪（SQLTRACE）

从当前跟踪的数据库名单中清除所有的数据库

```
execute function sysadmin:task(“set sql tracing database CLEAR”);
```

指定跟踪一个或者多个数据库，而不是跟踪所有的数据库

```
execute function sysadmin:task(“set sql tracing database ADD”, “stores7”);
```

指定跟踪一个特定用户

```
execute function sysadmin:task(“set sql tracing user ADD”, sinodbms”);
```

列出当前被跟踪的数据库和用户名

```
execute function sysadmin:task(“set sql tracing database LIST”);
execute function sysadmin:task(“set sql tracing user LIST”);
```

开启已指定的跟踪并配置跟踪级别和跟踪使用的内存量

```
execute function sysadmin:task(“set sql tracing on”, 4000, “2k”,“high”, “user”);
```

关闭 SQL 跟踪（SQLTRACE）

```
execute function sysadmin:task(“set sql tracing off”);
```

8.3.4. SQLTRACE 监控—查询语句向下展开

```
select * from syssqltrace where sql_id = 5678;
```

```
sql_id              5678
sql_address         4489052648
sql_sid             55
sql_uid             2053
sql_stmttype        6
sql_stmtname        INSERT
sql_finishtime      1140477805
sql_begintxtime     1140477774
sql_runtime         30.86596333400
sql_pgreads         1285
sql_bfreads         19444
sql_rdcache         93.39127751491
sql_bfidxreads      5359
sql_pgwrites        810
sql_bfwrites        17046
sql_wrcache         95.24815205913
sql_lockreq         10603
sql_lockwaits       0
sql_lockwttime      0.00
sql_logspace        60400
sql_sorttotal       0
sql_sortdisk        0
sql_sortmem         0
sql_executions      1
sql_totaltime       30.86596333400
sql_avgtime         30.86596333400
sql_maxtime         30.86596333400
sql_numiowaits      2080
sql_avgiowaits      0.014054286131
sql_totaliowaits    29.23291515300
sql_rowspersec      169.8958799132
sql_estcost         102
sql_estrows         1376
sql_actualrows      5244
sql_sqlerror        0
sql_isamerror       0
sql_isollevel       2
sql_sqlmemory       32608
sql_numiterators    4
sql_database        db3
sql_numtables       3
sql_tablelist       t1
sql_statement       insert into t1 select {+ AVOID_FULL(sysindices) } 0, tabname
```

8.3.5. SQLTRACE 监控 —跟踪数据存储

1、跟踪数据是从共享内存中的 SQLTACE 缓存中收集的，SQL 跟踪缓存的大小可配置；

2、SQLTRACE 的信息可在 SYSMASTER 里的伪表中查看；

sysssqltrace – SQL 语句的主要信息和统计信息

syssqltrace_iter – 优化器统计信息 (OAT 中的 Query Tree 标签页)

sysqltrace_hvar – host 变量 (OAT 中的 Host Variables 标签页)

syssqltrace_info – 当前 SQLTRACE 设置 (OAT 中的 Tracing Admin 标签页)

3、OAT 中的 “SQL Explorer”

OAT 使用 SQL 命令来设置 SQL 跟踪的选项

OAT 通过查询 sysmaster:syssqltrace* 系统表获取实时跟踪数据

OAT 通过查询 sysadmin:mon_syssqltrace* 系统表获取保留的跟踪数据

8.4 优化查询 - SQL 最佳实践

1、识别使用下列功能的字段数据类型和其上的索引，对优化 SQL 语句是很重要的：

正用于查询的字段

用于过滤器的字段

用于连接的字段

用于排序的字段

2、了解字段上的限制，如主键，检查等

某些限制是索引强制的
主键和唯一索引限制可以帮助识别数据库何时会返回唯一的数据行
检查限制（Check constraints） 可以提示数据的分布情况
3、考虑将检查过的数据行与实际返回的数据行进行对比，以决定过滤字段的分布情况

```
dbschema -hd <tablename> -d <database>
```

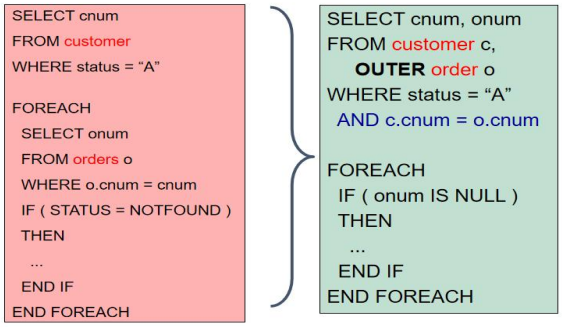
被描述的数据行
唯一值的行数
最大值
4、关注连接表之间的关系
一对一
一对多
多对多

8.4.1. 过滤器先行

1、如非获取全表数据，应避免查询表中的所有字段 (select *)，对于远程用户，可以减少网络传输流量。可用减少数据库计算和存储资源的使用
2、在表字段进行连接之前可以指定字段过滤器
3、最起码应该尝试在 SQL 语句中至少包含一个 INNER JOIN 子句
在一个 INNER JOIN 中，结果中只会包含表中满足连接条件的复合数据行，INNER JOIN 子句应置于被连接表之前，这将有助于优化器决定连接顺序，能够通过减少连接数据行来帮助提高可查询性。

8.4.2. 少用 OUTER

减少 OUTER 连接表的数量：
1、一个 OUTER JOIN 的结果将保留主导表中的数据行，否则会因为从属表中没有匹配的数据行而将这些主导表中的数据行抛弃，主导表中那些没有在从属表中得到数据匹配的数据行，会将从属表中的字段赋予 NULLS 值。
2、如果从 OUTER 表中进行连接的数据行不多，连接操作会造成没有必要的高成本！只有在连接条件应用之前，连接过滤器被允许用于 INNER 表的情况下，才可使用 OUTER JOINS。
3、当两个以上的表参与 OUTER JOIN，将会有多种方式来处理这些 OUTER JOINS，这导致 SQL 优化器无所适从。例如，在三个表之间有五个逻辑上清晰的连接，而其中四个连接涉及到 OUTER JOINS!
4、在频繁使用 OUTER JOINS 之前，一定要明确是一个或多个简单的 INNER 连接能够替换那些 OUTER JOINS。
只有在需要的时候才用 OUTER Joins。如下图



8.4.3. 复合式索引的首部

复合索引的首部是非常重要的，如果复合索引的首部不能使用，而索引中的其它字段可以，这个复合索引就不会被选择。在每一个 ORDER BY 子句中选择索引字段，可以节约可观的结果集排序时间。可以尝试用 UPPER 和 LOWER 索引过滤符来限制查看的数据量。检查查询语句：1 打开 SET EXPLAIN；2 尝试使用优化器指示符（Optimizer Directives）。

8.4.4. 索引读与排序

执行有索引的读取进行排序，有索引的数据行读取是以索引字段的顺序进行的。用于过滤符的索引字段有更高的优先级。

索引不会用于排序的原因：1、在排序标准中的字段不包括在索引中；2、在排序标准中的字段顺序与索引不同；3、在排序标准中的字段来自于不同的数据表。

8.4.5. 避免顺序扫描大表

在大数据表上应避免顺序扫描，在大表上执行顺序扫描会加剧计算资源的消耗，如果可能，可以使用轻扫描（light scans），小表的顺序扫描没有害处，尽量使用永久索引一避免顺序扫描，为批量查询创建临时索引。可以用实际索引替换自动索引，如果查询计划包含大表的自动索引路径，通常可在相同的字段上创建一个实际的索引以提高查询性能。

8.4.6. 哈希连接

以下情况，尽量使用哈希连接：

当需要连接来自多个表中的数据行时，使用哈希连接（Hash Joins）

当连接来自多个表中的大量数据行时最好使用哈希连接；

当大表连接中没有索引时自动使用哈希连接；

在内存中为一个表建立哈希表，扫描第二张表并在内存中进行哈希连接；

典型的连接是 NESTED LOOP，不断地反复执行索引扫描会有较高的代价；

必须打开 PDQ 并发执行功能；
DS_TOTAL_MEMORY 应该设置一个较高的值；
可以有效地结合优化指示符（optimizer directives）来使用。

8.4.7. 避免循环子句

相互连接的子查询通常对性能会产生不利影响，相互连接的子查询是指一个表中的字段不在 FROM 子句中列出，但出现在 SELECT 或 WHERE 子句中，在内查询中引用的外查询，对应外查询中的每一行数据，内查询都必须重复执行一次。如下

```
select c.* from customers  c
Where  exists  （select “X”
from orders o
where o.cstid = c.custid
and o.stat =“OPEN”     )
```

从 customers 表中读取的每一行数据，orders 表上的子查询都重复执行了一次！

SinoDB 在执行相互连接查询时不会使用 PDQ 并发功能，另外建议在子查询中用表名或表的别名来限制字段名，这样可以明确定义这些字段是属于哪一张表的，避免不必要的疑惑。

8.4.8. 复合式索引的首部

相尽量避免使用通配符：避免难度大的正则表达式，MATCHES 和 LIKE 关键字支持通配符匹配(正则表达式)。在初始位置上的通配符（‘%y’）强制 SinoDB 评估字段中的每一个值。这种过滤器不能使用索引，所以必须顺序访问该数据表。

如果必须使用正则表达式来判断，应避免和连接一起使用，处理最初不含正则表达式的查询语句，然后在之前查询语句返回的数据行上使用正则表达式判断，将结果保存在一个临时表中，然后再用临时表连接其他数据表。

在索引存在的情况下，运算元中间或末端包含通配符的正则表达式判断不会阻止使用索引。

避免非初始子字符串的搜索：任何一个基于字段非初始子字符串的过滤器也会导致 SinoDB 对字段的每一个值进行判断，如下例所示:

```
SELECT * FROM customer
WHERE zipcode[4,5] > '50‘
```

SinoDB 不能使用索引来评估这样的一个过滤器。

对于一个有索引的字段，优化器使用索引来处理一个判断该字段初始子字符串的过滤器。但是，子字符串判断的存在可能会影响使用复合索引来判断该子字符串字段和另外一个字段。

9. 数据库安全

9.1 用户身份标识与鉴别

9.1.1. 多重鉴别-第三方鉴别

9.1.1.1. pam 文件配置

确认$SINODBMSDIR/lib 下是否存在 pam_auth.so

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ ls  -l $SINODBMSDIR/lib/pam_auth.so
-rwxr-xr-x 1 sinodbms sinodbms 26336 4月  23 17:56 /data/sfc8_0424bak//lib/pam_auth.so
sinodbms@server160:/data/sfc8_0424bak$
```

创建/etc/pam.d/pam_auth，修改路径到对应的$SINODBMSDIR/lib
auth required $SINODBMSDIR/lib/pam_auth.so
account required $SINODBMSDIR/lib/pam_auth.so

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ sudo  more  /etc/pam.d/pam_auth
auth required /data/sfc8_0424bak/lib/pam_auth.so
account required /data/sfc8_0424bak/lib/pam_auth.so
sinodbms@server160:/data/sfc8_0424bak$
```

修改 sqlhosts 文件
ol_sinodb1210_20230425101502 onsoctcp 192.168.36.160 port s=4,pam_serv=pam_auth,pamauth=password

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ vi  etc/sqlhosts.ol_sinodb1210_20230425101502

ol_sinodb1210_20230425101502 onsocssl server160 ol_sinodb1210_20230425101502 s=4,pam_serv=pam_auth,pamauth=password
```

该配置文件修改说明如下:
增加“s=4,pam_serv=pam_auth,pamauth=password”
pam_serv 配置项的 pam 服务名“pam_auth”要与/etc/pam.d/pam_auth 一致

增加$SINODBMSDIR/etc/pam_auth.cfg 配置文件,该文件内容如下所示。

```
sinodbms@server160:/data/sfc8_0424bak$ more  etc/pam_auth.cfg
LOG_FLAG        1
IP      192.168.84.190
PORT    389
URL     ldaps://vm190
DN      uid=?,ou=People,dc=my-domain,dc=com
#NETWORK_TIMEOUT        10
#SSL    ON
#SSL_CACERTFILE         /etc/openldap/cacerts/ca.cert.pem
#SSL_REQCERT            allow

sinodbms@server160:/data/sfc8_0424bak$
```

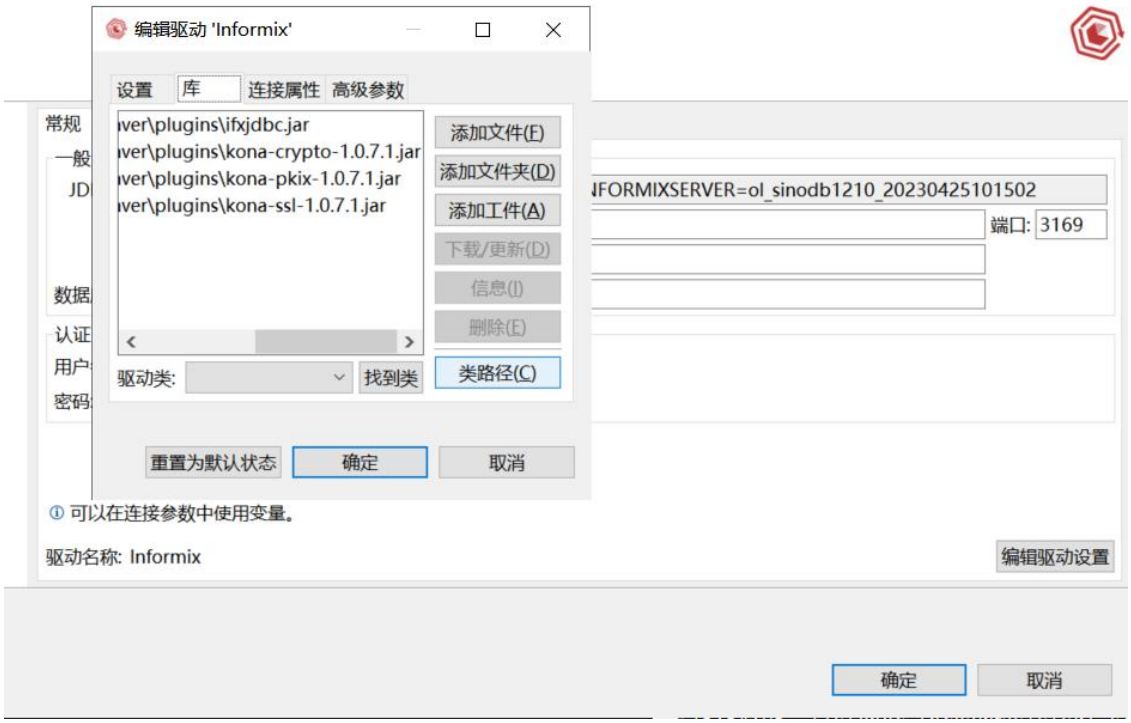
IP 192.168.84.190
PORT 389
URL ldaps://vm190
DN uid=?,ou=People,dc=my-domain,dc=com
NETWORK_TIMEOUT 10
SSL ON
SSL_CACERTFILE /etc/openldap/cacerts/ca.cert.pem
SSL_REQCERT allow

9.1.1.2. 第三方用户登录

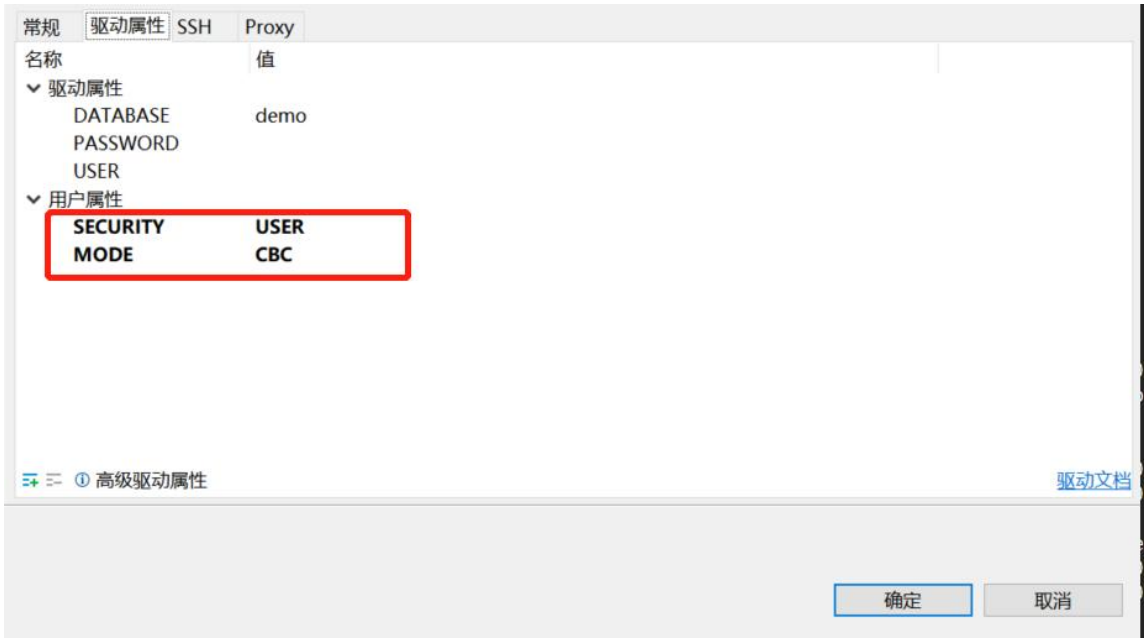
9.1.2. 鉴别信息保护

9.1.2.1. Dbeaver 驱动设置

增加 4 个文件



Dbeaver 作为客户端，进行用户鉴别时，需要增加用户参数



9.1.2.2. 设置环境变量

(1) export SECURITY=USER
(2) export SECURITY=NO
(3) export SECURITY=

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ more  ol_sinodb1210_20230425101502.ksh
 SINODBMSDIR=/data/sfc8_0424bak
 SINODBMSSERVER=ol_sinodb1210_20230425101502
 ONCONFIG=onconfig.ol_sinodb1210_20230425101502
 SINODBMSSQLHOSTS=/data/sfc8_0424bak/etc/sqlhosts.ol_sinodb1210_20230425101502
 GL_USEGLU=1
 PATH=${SINODBMSDIR}/bin:${SINODBMSDIR}/bin/onutilities:${SINODBMSDIR}/extend/krakatoa/jre/bin:${PATH}
export SINODBMSDIR SINODBMSSERVER ONCONFIG SINODBMSSQLHOSTS GL_USEGLU PATH
SECURITY=NO
sinodbms@server160:/data/sfc8_0424bak$
```

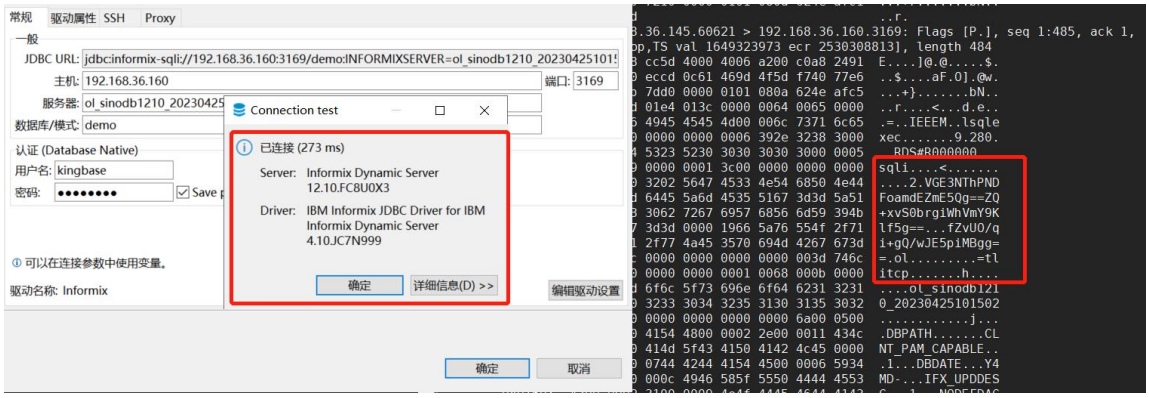
9.1.2.3. 抓包命令输入

Root 环境输入

tcpdump -i enp8s0f3 port 3169 -nn -X -s 10240 客户端抓包命令

tcpdump -i lo port 3169 -nn -X -s 10240 本地服务端抓包命令

9.1.2.4. 信息保护输出



9.1.3. 密码安全策略

9.1.3.1. 密码强度

onmode -wf PASSWORD_COMPLEXITY=1 密码强度 1

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf   PASSWORD_COMPLEXITY=1
Value of PASSWORD_COMPLEXITY has been changed to 1.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.1.3.2. 密码长度

onmode -wf MIN_PASSWORD_LEN=8 密码长度 8

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf  MIN_PASSWORD_LEN=8
Value of PASSWORD_COMPLEXITY has been changed to 1.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.1.3.3. 密码修改周期

onmode -wf ENFORCE_PASSWORD_HISTORY=2 密码修改周期

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$onmode  -wf  ENFORCE_PASSWORD_HISTORY=2
Value of ENFORCE_PASSWORD_HISTORY has been changed to 2.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.1.3.4. 密码最小使用周期

onmode -wf MIN_PASSWORD_AGE=1 最小周期（过一天才能修改）

9.1.3.5. 密码失效周期

onmode -wf MAX_PASSWORD_AGE=10 最大周期（过 10 就失效）

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf  MAX_PASSWORD_AGE=10
Value of MAX PASSWORD AGE has been changed to 10.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.1.3.6. 用户验证

onmode -wf CHECKSUM_FLAG=1 （用户信息校验）

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf  CHECKSUM_FLAG=1
Value of CHECKSUM_FLAG has been changed to 1.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.1.3.7. 密码试错锁定

onmode -wf RETRY_LIMIT=2 密码试错 2 次锁定

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf  RETRY_LIMIT=2
Value of RETRY_LIMIT has been changed to 2.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

9.2 访问控制

9.2.1. 黑白名单控制

9.2.2.1. 用户黑名单配置

onmode -wf USER_BLACKLIST=user_1,sinodbms

```
sinodbms@server160:/data/sfc8 0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf   USER_BLACKLIST=user_1,sinodbms
Value of USER_BLACKLIST has been changed to user_1,sinodbms.
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8 0424bak$
```

```
sinodbms@server160:/data/sfc8_0424bak$ dbaccess  demo  -

Database selected.

> connect  to  'demo'  user  'sinodbms';
   ENTER PASSWORD:

Disconnected.


 8610: User is in blacklist.


Invalid argument
Error in line 1
Near character position 1
>
```

9.2.2.2. 用户白名单配置

onmode -wf USER_WHITELIST=user_1,snodbms

```
sinodbms@server160:/data/sfc8_0424bak$ onmode  -wf   USER_WHITELIST=user_1,sinodbms
Value of USER_WHITELIST has been changed to user_1,sinodbms.
sinodbms@server160:/data/sfc8_0424bak$
```

```
> ^C sinodbms@server160:/data/sfc8_0424bak$ dbaccess  demo  -

Database selected.

>  connect  to 'demo'  user 'testuser';
   ENTER PASSWORD:

Disconnected.


 8608: User isn't in whitelist.
Error in line 1
Near character position 2
>
```

9.2.2.3. IP 黑名单配置

onmode -wf IP_WHITELIST=192.168.36.160

```
testuser@server160:~$ dbaccess  demo  -

Database selected.

> connect to  '@ol_sinodb1210_20230425101502'  user  'testuser';
   ENTER PASSWORD:

Disconnected.


 8609: Ip is in blacklist.

Invalid argument
Error in line 1
Near character position 1
>
```

9.2.2.4. IP 白名单配置

onmode -wf IP_WHITELIST=192.168.36.160

```
> ^Csinodbms@server162:/data2/sfc8_aqkk$ dbaccess  demo@ol_sinodb1210_20230425101502
-

Database selected.

> connect to  '@ol_sinodb1210_20230425101502'  user  'testuser';
   ENTER PASSWORD:

Disconnected.


 8606: Ip isn't in list.

Invalid argument
Error in line 1
Near character position 1
>
```

9.2.2. 三权分立

9.2.2.1. 设置角色分离

安装时
选择 1- Enable role separation

Group for security-related tasks: (Default: sinodbms): dbsso
Group for audit-administration tasks: (Default: sinodbms): dbaao
Group for database users (leave blank to allow all users): (Default:):

```
================================================================================
Get Role Separation choice
--------------------------

Enable role separation for auditing procedures.

If you enable role separation, you can assign existing groups of users to
specific roles.

If you do not enable role separation, the database server administrator
performs all administration tasks.

    1- Enable role separation
  ->2- Do not enable role separation

ENTER THE NUMBER FOR YOUR CHOICE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT:: 1




================================================================================
Role Separation groups selection
--------------------------------

Assign a group of users to each of the following roles by specifying group
identifiers (group IDs). The group IDs specified must already exist on your
system.

Group for security-related tasks: (Default: sinodbms): dbsso

Group for audit-administration tasks: (Default: sinodbms): dbaao

Group for database users (leave blank to allow all users): (Default: ):
```

9.2.2.2. 查看权限

dbaao 用户启停审计 onaudit -l

```
[dbaao@dcmaster ~]$ onaudit  -l  1
Onaudit -- Audit Subsystem Configuration Utility

[dbaao@dcmaster ~]$ onaudit  -c
Onaudit -- Audit Subsystem Configuration Utility

Current audit system configuration:
    ADTMODE    = 1
    ADTERR     = 0
    ADTPATH    = /data/sinodb_0424/audit_log
    ADTSIZE    = 256000000
    Audit file = 0
    ADTROWS    = 0
[dbaao@dcmaster ~]$
```

dbaao 用户查看修改审计配置 onaudit -c/onaudit -p

```
[dbaao@dcmaster ~]$
[dbaao@dcmaster ~]$ onaudit  -c
Onaudit -- Audit Subsystem Configuration Utility

Current audit system configuration:
    ADTMODE    = 0
    ADTERR     = 0
    ADTPATH    = /usr/sinodbms/aaodir
    ADTSIZE    = 50000
    Audit file = 0
    ADTROWS    = 0
[dbaao@dcmaster ~]$
```

dbsso 用户查看修改审计掩码 onaudit -a,-d,-m,-o

```
[dbsso@dcmaster ~]$
[dbsso@dcmaster ~]$ onaudit  -o
Onaudit -- Audit Subsystem Configuration Utility

    This will list every row in the audit mask table


Do you wish to continue? [y/N]: y


[dbsso@dcmaster ~]$
```

9.3 数据存储安全

9.3.1. 存储加密算法管理

9.3.1.1. 国密存储配置

1. $SINODBMSDIR/lib 目录下存在 libsino8crypt_64.so
2. 修改数据库 onconfig 配置文件，配置成国密的 sms4 算法
3. 添加： DISK_ENCRYPTION keystore=yangyp,cipher=sms4
4. keystore=yangyp，指明证书的文件名，oninit 第一次启动时会在$SINODBMSDIR/etc 目录生成的对应的 sec 和 ssc 后缀的证书
5. 需要重新初始化数据库 改了存储方式
   onconfig 配置文件 FULL_DISK_INIT 改成 1
   onmode -ky
   onclean -ky
   oninit -ivy

9.3.1.2. 国际存储配置

1. $SINODBMSDIR/lib 目录下存在 libsino8crypt_64.so
2. 修改数据库 onconfig 配置文件，配置成国际的 aes128 算法
3. 添加： DISK_ENCRYPTION keystore=yangyp,cipher=aes128

4. keystore=yangyp，指明证书的文件名，oninit 第一次启动时会在$SINODBMSDIR/etc 目录生成的对应的 sec 和 ssc 后缀的证书

5. 需要重新初始化数据库　改了存储方式

onconfig 配置文件 FULL_DISK_INIT 改成 1

onmode -ky

onclean -ky

oninit -ivy

9.3.2. 存储加密（国密支持）

9.3.2.1. 国密存储

oncheck -pr | grep 'at-rest'

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onstat  -d

Sinoregal SinoDB Dynamic Server Version 16.8.FC8U0X3 -- On-Line -- Up 21 days 03:59:45 -- 664220 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
44d19028         1        0x10000001 1        2        4096     N  BAE   sinodbms rootdbs
44e757e0         2        0x10000001 2        1        4096     N  BAE   sinodbms datadbs1
 2 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
44d19268         1      1      0          41984      13973                 PO-B-D /data/sfc8_0424/storage/rootdbs
44e59050         2      2      0          512000     510761                PO-B-D /data/sfc8_0424/storage/datadbs1
46fd8028         3      1      0          51200      48969                 PO-B-D /data/sfc8_0424/storage/datadbs2
 3 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

sinodbms@server160:/data/sfc8_0424bak$ oncheck  -pr |grep 'at-rest'
        Encryption-at-rest is enabled using cipher 'sms4'
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

dd if=/data/sfc8_0424/storage/rootdbs bs=4096 skip=1762 count=2|od -x　查看加密情况

```
sinodbms@server160:/data/sfc8_0424bak$ dd if=/data/sfc8_0424/storage/rootdbs bs=4096 skip=1762 count=2|od  -x
0000000 62cf 11ba c113 4861 fceb 4d97 c689 3dcc
0000020 8c87 5fe2 e980 cf1b 1d3c 3760 6f25 c8dd
0000040 ac05 9725 8c11 a1b4 4157 333e 06b6 f460
0000060 c0a0 97ec cfa2 d6f3 1816 c164 d543 4961
0000100 735e 244a 1446 f11d 5445 62be 3c25 6e60
0000120 08c4 2e7f 9b0c 1813 0708 23b7 d1d2 e725
0000140 a1f8 3281 443f c59a 03dd 8b39 0ee9 18ae
0000160 9845 2ea6 6dd0 2b9d 6bb3 888e ebed 2261
0000200 90a4 fe88 fb82 1eb5 03e8 9791 1687 4884
0000220 a619 b483 e00b d681 a9ae de92 af7a 8ab8
0000240 3a9c d38a d5ec 35ad c8e5 2742 c708 def7
0000260 c099 3728 1a88 26bd 130d f47a d59e 60f4
0000300 e2a8 c263 047e 4bf4 eea7 f1c2 7879 04fb
0000320 2329 7aea 3296 0dc2 72db cb37 e9f5 71f8
0000340 14c6 3757 cf8b 71a8 3cf8 1587 abd3 1c72
0000360 849d 4280 93f4 3e54 cb30 145b 209a 0869
0000400 235f 0095 e767 2d58 a605 9d26 f637 cb85
0000420 d06d 1621 2aee c42c a4fa 8a3a 1fde 08a2
0000440 937e b6be 1655 f5f8 8a42 c3ed 4141 da65
0000460 c666 eb15 c281 26d0 9535 9634 3049 f9eb
0000500 2a6f 72df 70c0 ceb3 04fc 228d abbb 0423
0000520 ac18 33ef 3767 bcab 2e8e daf6 4cea 551c
0000540 52c9 e467 08f2 7a49 8b05 3966 4bfb a7cf
0000560 13d5 f1bf d29a e5a6 3130 ed02 4458 bcee
记录了2+0 的读入
记录了2+0 的写出
0000600 4c4a 84a6 cef2 1537 48f3 0001 1c17 b454
0000620 5d59 0c06 277c 71d7 5452 dc6e d721 fde9
0000640 2f22 6a10 9504 0448 f1ee fe89 a0e6 2de4
8192 bytes (8.2 kB, 8.0 KiB) copied, 0.00719795 s, 1.1 MB/s
0000660 a6a8 9e60 11c9 8a2d 7ed9 fa57 ae32 6bae
0000700 83c0 6773 a95e 06ab a4b4 8b08 1d55 3129
0000720 2db7 cba9 3ee5 4244 0ff0 0da6 5fb8 6f68
0000740 2f01 7773 1387 b7fb 317e 48c1 2be8 fb6a
```

9.3.2.2. 国际存储

oncheck -pr | grep 'at-rest'

```
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$ onstat  -d

Sinoregal SinoDB Dynamic Server Version 16.8.FC8U0X3 -- On-Line -- Up 21 days 04:02:01 -- 664220 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
44d19028         1        0x10000001 1        2        4096     N  BAE   sinodbms rootdbs
44e757e0         2        0x10000001 2        1        4096     N  BAE   sinodbms datadbs1
 2 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
44d19268         1      1      0          41984      13973                 PO-B-D /data/sfc8_0424bak/storage/rootdbs
44e59050         2      2      0          512000     510761                PO-B-D /data/sfc8_0424bak/storage/datadbs1
46fd8028         3      1      0          51200      48969                 PO-B-D /data/sfc8_0424bak/storage/datadbs2
 3 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

sinodbms@server160:/data/sfc8_0424bak$ oncheck  -pr |grep 'at-rest'
        Encryption-at-rest is enabled using cipher 'aes128'
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
sinodbms@server160:/data/sfc8_0424bak$
```

dd if=/data/sfc8_0424/storage/rootdbs bs=4096 skip=1762 count=2|od -x 查看加密情况

9.4 通信安全

9.4.1. 传输加密算法管理

9.4.1.1. 国密传输配置

1. lib 目录 libsino8ssl_64.so
2. ssl 目录 4 个文件 $SINODBMSSERVER_enc.crt $SINODBMSSERVER_enc.key $SINODBMSSERVER_sign.crt $SINODBMSSERVER_sign.key
3. etc 下 4 个文件 client_enc.crt client_enc.key client_sign.crt client_sign.key
4. etc 下 conssl.cfg 内容

   SSL_CIPHER_SUITE ECC_SM4_CBC_SM3
5. etc 下 sqlhost 内容

   ol_sinodb1210_20230424100159 onsocssl server160 ol_sinodb1210_20230424100159
6. etc 下 onconfig 内容

   onstat -c |grep ^NETTYPE

   NETTYPE ipcshm,1,50,CPU

   NETTYPE onsocssl,1,150,NET

   NETTYPE drsoctcp,1,150,NET

   onstat -c |grep ^SSL

   SSL_CIPHER_SUITE ECC_SM4_CBC_SM3

DBEAVER 配置

SSLCONNECTION 是否采用安全通信 false 不采用 true 采用

SSLGM true 国密 false 国际

javax.net.ssl.encCert 客户端存放路径

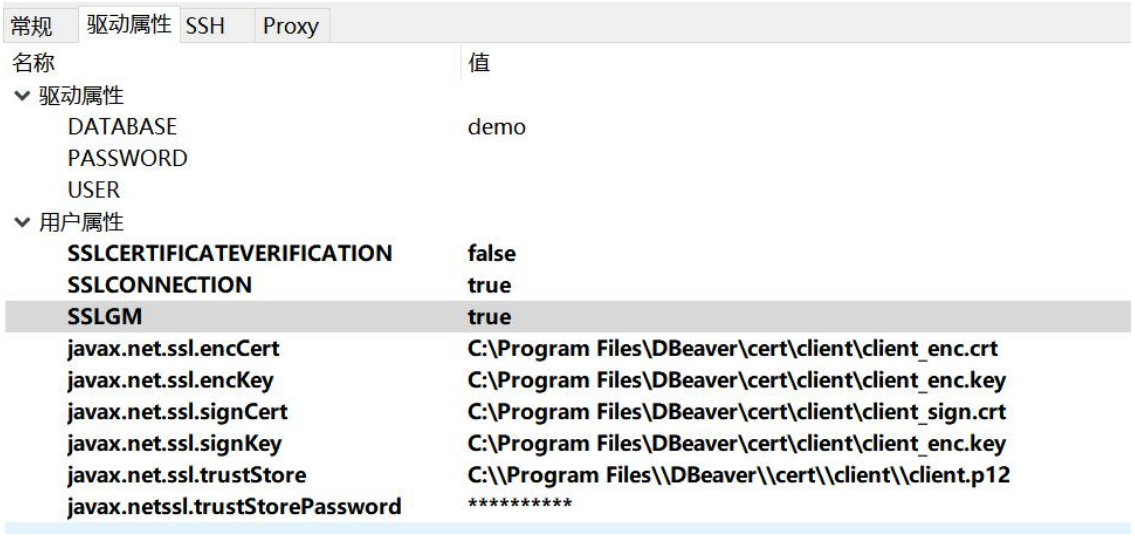
javax.net.ssl.encKey 客户端存放路径

javax.net.ssl.signCert 客户端存放路径

iavax.net.ssl.signKey 客户端存放路径

avax.netssltrustStore 客户端存放路径

javax.netssl.trustStorePassword 保存密码

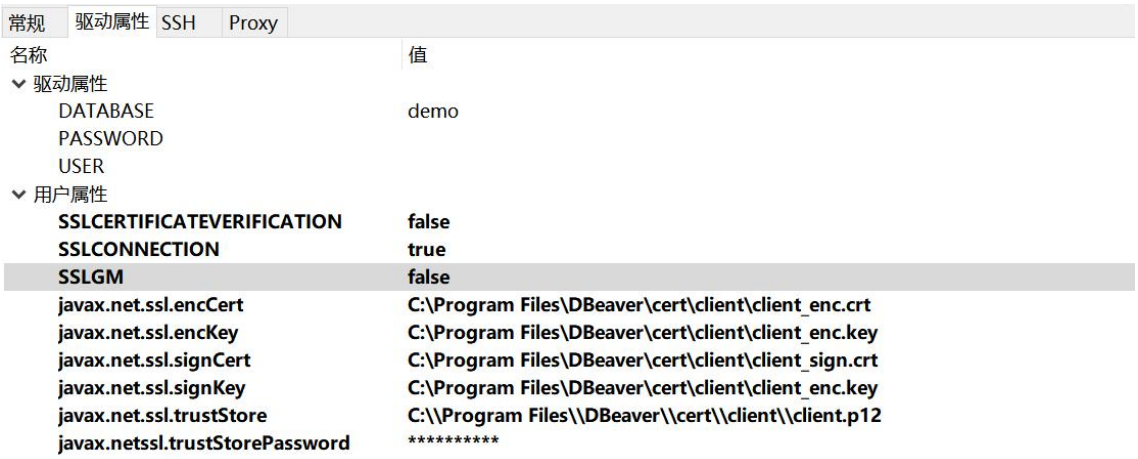


9.4.1.2. 国际传输配置

1. lib 目录 libsino8ssl_64.so 不能有，存在这个文件，会检测国密
2. ssl 目录 2 个文件
   $SINODBMSSERVER.p12 $SINODBMSSERVER.sec
3. etc 下 2 个文件 client.p12 client.sec
4. etc 下 conssl.cfg 内容
   SSL KEYSTORE FILE /data/sfc8_0424bak/etc/client.p12
   SSL KEYSTORE STH /data/sfc8_0424bak/etc/client.sec
5. etc 下 sqlhost 内容
   ol_sinodb1210_20230424100159 onsocssl server160
   ol_sinodb1210_20230424100159
6. etc 下 onconfig 内容
   onstat -c |grep ^NETTYPE
   NETTYPE ipcshm,1,50,CPU
   NETTYPE onsocssl,1,150,NET
   NETTYPE drsoctcp,1,150,NET
   onstat -c |grep ^SSL
   SSL_KEYSTORE_LABEL ol_sino1210_ssl

DBEAVER 配置
SSLCONNECTION 是否采用安全通信 false 不采用 true 采用
SSLGM true 国密 false 国际
javax.net.ssl.encCert 客户端存放路径
javax.net.ssl.encKey 客户端存放路径
javax.net.ssl.signCert 客户端存放路径
iavax.net.ssl.signKey 客户端存放路径
avax.netssltrustStore 客户端存放路径

javax.netssl.trustStorePassword 保存密码

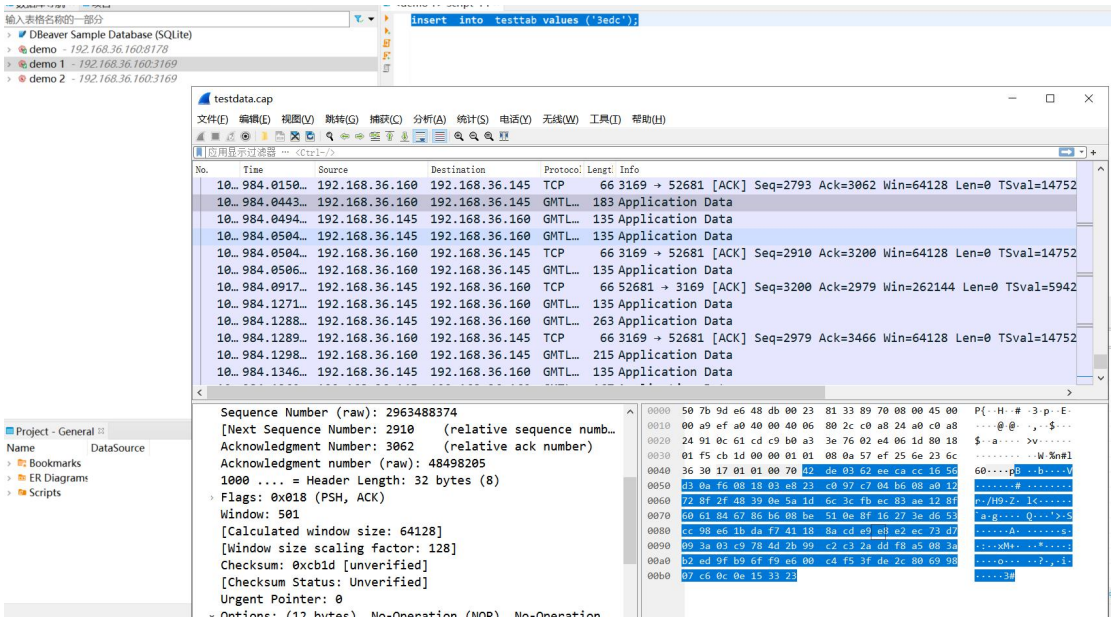


9.4.2. 传输加密（国密支持）

tcpdump -i enp8s0f3 port 3169 -nn -X -s 10240 -w testdata.cap 客户端抓包命令
tcpdump -i lo port 3169 -nn -X -s 10240 -w testdata.cap 本地服务端抓包命令

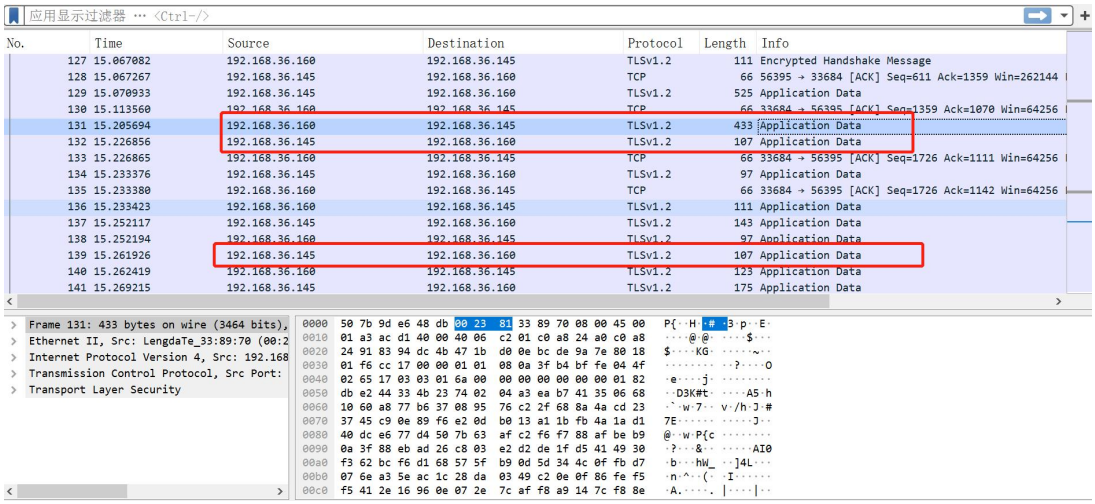
9.4.2.1. 国密传输

dbeaver 执行 sql



9.4.2.2. 国际传输

dbeaver 执行 sql



9.5 安全审计

9.5.1. 审计记录存储配置

9.5.2.1. 审计开始配置

1. 设置 onconfig 配置文件

在 onconfig 配置文件中增加配置项“AUDITON 2”，将操作写到审计文件以及审计表 sysauditlog 中。

2. 创建审计文件所在目录

通过 sinodbms 用户登录操作系统，执行以下命令：

mkdir $SINODBMSDIR/audit_log

3. 配置审计文件路径和大小

指定审计文件路径为$SINODBMSDIR/audit_log，大小 256MB，执行以下命令：

onaudit -p $SINODBMSDIR/audit_log -s 256000000

4. 指定用户缺省下需要审计的事件

onaudit -a -u tester -e +UPRW

Onaudit -m -u tester -e +INRW

9.5.2.2. 审计记录配置

onmode -wf AUDITON=0 只写到审计 log 文件

onmode -wf AUDITON=1 只写到 sysauditlog 表

onmode -wf AUDITON=2 审计 log 文件和 sysauditlog 表都写

9.5.2. 审计记录保存到数据库置

9.5.2.1. 审计 log 文件查看

```
ONLN|2023-06-01 09:42:05.000|server160|37590|ol_sinodb1210_20230425101502|tester|-580:RVTB:testdb:102:1:tester:user32:0
ONLN|2023-06-01 09:42:05.000|server160|37590|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:07.000|server160|37718|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:07.000|server160|37718|ol_sinodb1210_20230425101502|tester|-389:RVDB:testdb:256:user12
ONLN|2023-06-01 09:42:07.000|server160|37718|ol_sinodb1210_20230425101502|tester|-389:RVDB:testdb:1024:user3
ONLN|2023-06-01 09:42:07.000|server160|37718|ol_sinodb1210_20230425101502|tester|-389:RVDB:testdb:512:user4
ONLN|2023-06-01 09:42:07.000|server160|37718|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|-9662:DRUR:public
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|-9662:DRUR:user4
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|-9662:DRUR:user12
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|-9662:DRUR:user3
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|-9662:DRUR:user32
ONLN|2023-06-01 09:42:10.000|server160|37819|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:15.000|server160|38071|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:15.000|server160|38071|ol_sinodb1210_20230425101502|tester|-8210:CRLB:testdb:megacerp.techniciar
ONLN|2023-06-01 09:42:15.000|server160|38071|ol_sinodb1210_20230425101502|tester|0:CRLB:testdb:megacorp.technician
ONLN|2023-06-01 09:42:15.000|server160|38071|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:17.000|server160|38184|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:17.000|server160|38184|ol_sinodb1210_20230425101502|tester|0:GRLB:testdb:megacorp.technician:user3:A
ONLN|2023-06-01 09:42:17.000|server160|38184|ol_sinodb1210_20230425101502|tester|0:GRLB:testdb:megacorp.technician:user13:A
ONLN|2023-06-01 09:42:17.000|server160|38184|ol_sinodb1210_20230425101502|tester|-8210:GRLB:testdb:megacerp.technician:user3:A
ONLN|2023-06-01 09:42:17.000|server160|38184|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:20.000|server160|38234|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:20.000|server160|38234|ol_sinodb1210_20230425101502|tester|0:RVLB:testdb:megacorp.technician:tester3:A
ONLN|2023-06-01 09:42:20.000|server160|38234|ol_sinodb1210_20230425101502|tester|-8210:RVLB:testdb:megacerp.technician:tester3:A
ONLN|2023-06-01 09:42:20.000|server160|38234|ol_sinodb1210_20230425101502|tester|0:DISC
ONLN|2023-06-01 09:42:25.000|server160|38441|ol_sinodb1210_20230425101502|tester|0:CONN
ONLN|2023-06-01 09:42:25.000|server160|38441|ol_sinodb1210_20230425101502|tester|-8210:DRLB:testdb:megacerp.technician
ONLN|2023-06-01 09:42:25.000|server160|38441|ol_sinodb1210_20230425101502|tester|0:DRLB:testdb:megacorp.technician
ONLN|2023-06-01 09:42:25.000|server160|38441|ol_sinodb1210_20230425101502|tester|0:DISC
```

9.5.2.2. 审计 sysauditlog 表查看

```
accesstime  2023-04-18 15:48:49
clientname  server160
username    tester
event       DRUR
result      -26700
detail      -26700:DRUR:user32

accesstime  2023-04-18 15:48:50
clientname  server160
username    tester
event       CRLB
result      -8210
detail      -8210:CRLB:testdb:megacerp.techniciar

accesstime  2023-04-18 15:48:50
clientname  server160
username    tester
event       CRLB
result      0
detail      0:CRLB:testdb:megacorp.technician

accesstime  2023-04-18 15:48:51
clientname  server160
username    tester
event       GRLB
result      0
detail      0:GRLB:testdb:megacorp.technician:user3:A

accesstime  2023-04-18 15:48:51
clientname  server160
username    tester
event       GRLB
result      0
detail      0:GRLB:testdb:megacorp.technician:user13:A

accesstime  2023-04-18 15:48:51
clientname  server160
username    tester
event       GRLB
result      -8210
detail      -8210:GRLB:testdb:megacorp.technician:user3:A
```

9.6 语法增强

9.6.1. Create Or Replace View

1. create or replace view

create table tab40 (id int ,col char(10));
CREATE or replace VIEW v1 AS SELECT * FROM tab40;
CREATE or replace VIEW v1 AS SELECT * FROM tab40;
create table tab41 (id int ,col char(10));
CREATE or replace VIEW palo_alto AS SELECT * FROM tab41 WHERE col = 'Palo Alto' WITH CHECK OPTION;

```
>  create  table  tab40 (id  int ,col char(10));
CREATE or replace VIEW v1 AS SELECT * FROM tab40;
CREATE or replace VIEW v1 AS SELECT * FROM tab40;
create  table  tab41 (id  int ,col char(10));
 CREATE or replace VIEW palo_alto AS  SELECT * FROM tab41 WHERE col = 'Palo Alto' WITH CHECK OPTION;
Table created.

>
View created.

>
View created.

>
Table created.

>

View created.
```

9.6.2. Create Or Replace Trigger

1. Create or replace trigger
   create table table_trigger1 (
   id1 char(10),
   name1 char(20),
   primary key(id1)
   );

```
create or replace trigger trigger_select
select on table_trigger1
referencing Old as old
for each row
(
insert into table_trigger2 (id2,kind,time2)
values (old.id1,'S',current)
);


create or replace trigger trig
Trigger created.

> > > > > > > > > >
Trigger created.
```