YashanDB 管理员手册

v23.2.2.0

2024年5月

深圳计算科学研究院

深圳崖山科技有限公司

基本数据库管理

本章节对一些常用的基本数据库管理操作提供指导和参考，主要包括修改配置参数、数据库归档模式的调整、归档管理以及数据库实例启停等操作。

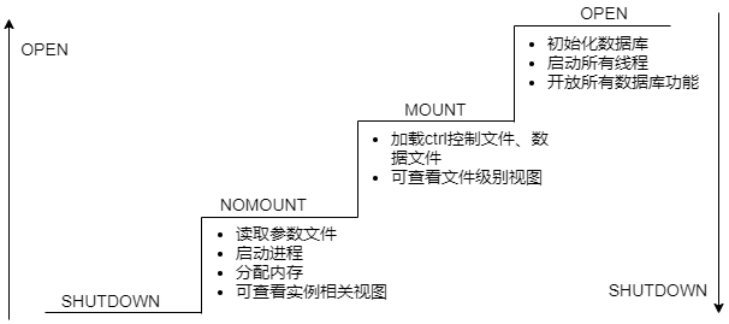
实例启停

本文将介绍单机及分布式部署的实例启停方式，分别为SQL命令方式及yasboot工具方式，如需了解共享集群的实例启停方式请查阅共享集群启停。

数据库实例阶段

数据库实例从关闭启动到正常使用，要经过NOMOUNT、MOUNT和OPEN三个阶段。

- **NOMOUNT**：启动数据库实例， 此时读取参数文件，但是不加载数据库。
- **MOUNT**：启动数据库实例，读取控制文件，加载数据库，但是数据库处于关闭状态。
- **OPEN**：启动数据库实例，加载并打开数据库。



启动数据库实例

YashanDB支持通过yasboot工具直接将数据库实例调整至NOMOUNT、MOUNT和OPEN三个阶段的任意一个，也支持通过ALTER DATABASE语句将数据库实例从NOMOUNT阶段调整到MOUNT阶段和OPEN阶段。

ALTER DATABASE语句仅适用于单机数据库对单个实例节点执行启动实例操作，即仅改变语句执行所在节点的实例。

yasboot工具支持进行如下粒度的启动实例操作：

- 单机/分布式数据库集群粒度启动节点实例
- 单机/分布式数据库节点组粒度启动节点实例
- 单机/分布式数据库启动特定节点实例

启动到NOMOUNT阶段

可通过执行如下命令将数据库启动到NOMOUNT阶段：

```
# yasboot工具：
# 方式一：先关闭然后启动数据库集群至NOMOUNT状态
$ yasboot cluster stop -c yashandb
$ yasboot cluster start -c yashandb -m nomount
# 方式二：一键重启数据库集群至NOMOUNT状态
$ yasboot cluster restart -c yashandb -m nomount

# 启动group_id为1的节点组至NOMOUNT状态
$ yasboot group start -c yashandb -g 1 -m nomount
# 启动node_id为4-1的节点至NOMOUNT状态
$ yasboot node start -c yashandb -n 4-1 -m nomount
```

操作成功后，可查询数据库状态已更新为STARTED。

```
# 请将如下password@ip:port修改成sys用户的密码及对应的ip端口号
$ yasql sys/password@ip:port

SQL> SELECT status FROM V$INSTANCE;

STATUS
```

```
---------
STARTED
```

> **Note**：
> 数据库实例启动到NOMOUNT状态后，查看V$INSTANCE视图的STATUS状态值为STARTED，而不是NOMOUNT。

启动到MOUNT阶段

支持通过yasboot工具将数据库实例从任意阶段调整至MOUNT阶段，或通过SQL命令方式将数据库实例从NOMOUNT阶段调整至MOUNT阶段。

可通过执行如下命令将数据库实例启动到MOUNT阶段：

```
# yasboot工具：
# 方式一：先关闭然后启动数据库集群至MOUNT状态
$ yasboot cluster stop -c yashandb
$ yasboot cluster start -c yashandb -m mount
# 方式二：一键重启数据库集群至MOUNT状态
$ yasboot cluster restart -c yashandb -m mount

# 启动group_id为1的节点组至MOUNT状态
$ yasboot group start -c yashandb -g 1 -m mount
# 启动node_id为4-1的节点至MOUNT状态
$ yasboot node start -c yashandb -n 4-1 -m mount

# SQL命令（此时须确保数据库实例处于NOMOUNT状态）：
$ yasql sys/password@ip:port
SQL> ALTER DATABASE MOUNT;
```

操作成功后，可查询数据库状态已更新为MOUNTED。

```
# 请将如下password@ip:port修改成sys用户的密码及对应的ip端口号
$ yasql sys/password@ip:port

SQL> SELECT status FROM V$INSTANCE;

STATUS
---------
MOUNTED
```

启动到OPEN阶段

支持通过yasboot工具将数据库实例从任意阶段调整至OPEN阶段，或通过SQL命令方式将数据库实例从NOMOUNT或MOUNT阶段调整至OPEN阶段。

可通过执行如下命令将数据库实例启动到OPEN阶段：

```
# yasboot工具：
# 方式一：先关闭然后启动数据库集群至OPEN状态
$ yasboot cluster stop -c yashandb
$ yasboot cluster start -c yashandb -m open
# 方式二：一键重启数据库集群至OPEN状态
$ yasboot cluster restart -c yashandb -m open

# 启动group_id为1的节点组至OPEN状态
$ yasboot group start -c yashandb -g 1 -m open
# 启动node_id为4-1的节点至OPEN状态
$ yasboot node start -c yashandb -n 4-1 -m open

# SQL命令（此时须确保数据库实例处于NOMOUNT或MOUNT状态）：
$ yasql sys/password@ip:port
SQL> ALTER DATABASE OPEN;
```

操作成功后，可查询数据库状态已更新为OPEN。

```
SELECT status FROM V$INSTANCE;

STATUS
---------
OPEN
```

启动实例失败场景

环境变量未设置或错误

以下为YashanDB实例成功启动的三个关键环境变量，设置错误将导致无法启动到NOMOUNT及之后阶段，请根据实际情况将环境变量调整至正确值：

- YASDB_HOME：YashanDB默认软件目录
- YASDB_DATA：YashanDB默认数据目录
- LD_LIBRARY_PATH：YashanDB依赖包目录，使用yasboot sql或yaql前必须先正确配置该环境变量

控制文件验证失败

YashanDB在实例启动时进行控制文件校验，出现如下情况时，实例将无法启动到MOUNT及之后阶段：

- **控制文件实际路径与CONTROL_FILES参数配置路径不一致**

  出现此种情况时，解决方法如下：

  - 方法一：修改控制文件路径。
  - 方法二：启动实例到NOMOUNT阶段后修改CONTROL_FILES参数，再重新启动实例。
- **控制文件中的数据库版本与实例版本不一致**

  此种情况启动实例将提示*database version x.x.x incompatible with YashanDB version x.x.x*错误，请根据实际情况选择是否进行版本升级。

- **控制文件组成员的数据库ID不一致**

  此种情况多是由于一台服务器存在多个数据库实例时，CONTROL_FILES参数配置了不同数据库的控制文件，将其修改为正确的控制文件路径即可。

数据文件丢失或损坏

- **内置表空间数据文件丢失或损坏**

  如出现内置表空间数据文件丢失或损坏的情况，实例将无法启动到MOUNT及之后阶段。

  此种情况下，通常可以通过恢复最新的可用备份集解决，若不存在备份集则只能通过找回文件、物理恢复等方式解决。

- **用户数据文件丢失或损坏**

  当用户表空间挂载的数据文件出现丢失或损坏时，通常可以通过恢复最新的可用备份集解决，若不存在备份集则可通过YashanDB提供的逃生通道进行应急处理，保证数据库能正常启动、打开和运行，具体操作见数据文件管理章节中数据文件损坏后的应急处理相关描述。

redo文件丢失或损坏

如出现redo文件丢失或损坏的情况，实例将无法启动到MOUNT及之后阶段。

此种情况下，可通过恢复最新的可用备份集解决。

相关进程未启动

使用yasboot工具调整数据库实例时须确保yasom和yasagent进程处于开启状态，否则将提示*connect： connection refused*错误。服务器重启时，yasom和yasagent进程会自动关闭，可通过注册开机自启动功能于服务器重启时自动拉起yasom和yasagent进程，具体操作见初始数据库章节描述。

数据库打开模式

YashanDB在MOUNT或NOMOUNT阶段下，准备调整至OPEN阶段时，可以通过SQL命令方式根据不同场景配置数据库的打开模式为READWRITE、RESETLOGS或者UPGRADE。

语法为：`ALTER DATABASE OPEN [READWRITE|RESETLOGS|UPGRADE]`

- READWRITE：用于正式生产环境，数据库完整事务的读写操作；数据库打开时如果不设置打开模式参数，则默认为READWRITE。
- RESETLOGS：用于数据库发生故障时，如果需要进行不完全恢复操作，请使用RESETLOGS模式打开数据库，并在打开后重新设置日志号。
- UPGRADE：用于打开数据库升级模式，该模式下不允许建立新的会话连接。

可通过数据库视图V$DATABASE查看数据库打开模式。

```
SELECT database_id,database_name,open_mode FROM V$DATABASE;

    DATABASE_ID DATABASE_NAME    OPEN_MODE
--------------- ---------------- ----------------
      569377301 yasdb            READ_WRITE
```

其中，主备模式下还需要关注主备角色的DATABASE_ROLE字段。

```
SELECT DATABASE_NAME,LOG_MODE,OPEN_MODE,PROTECTION_MODE,DATABASE_ROLE,BLOCK_SIZE,STATUS FROM  V$DATABASE;

DATABASE_NAME LOG_MODE    OPEN_MODE   PROTECTION_MODE      DATABASE_ROLE  BLOCK_SIZE STATUS
------------- ----------- ----------- -------------------- -------------- ---------- -------
yasdb         ARCHIVELOG  READ_ONLY   MAXIMUM PERFORMANCE  STANDBY              8192 NORMAL
```

> **Caution：**
> 如果执行的是完全恢复操作，此时如果使用RESETLOGS模式打开，数据库会报YAS-02184错误并关闭实例，请慎重使用该模式打开数据库。

关闭数据库实例

YashanDB支持通过执行SHUTDOWN语句或使用yasboot工具关闭数据库实例。

SHUTDOWN语句仅适用于单机/分布式数据库对单个实例节点执行关闭实例操作，即仅改变语句执行所在节点的实例。

yasboot工具支持进行如下粒度的关闭操作：

- 单机/分布式数据库集群粒度关闭节点实例
- 单机/分布式数据库节点组粒度关闭节点实例
- 单机/分布式数据库关闭特定节点实例

SQL命令方式

可在SQL命令行中指定如下三种方式关闭数据库实例：

- SHUTDOWN NORMAL：等待事务正常结束后关闭，没有时间限制，通常建议选择这种方式来关闭数据库，此方式也为YashanDB默认的关库模式。
- SHUTDOWN IMMEDIATE：立即中断当前用户的连接，同时强行终止用户的当前执行中的事务，将未完成的事务回退，并关闭数据库。
- SHUTDOWN ABORT：强制中断所有数据库操作并关闭数据库，这种关闭方式可能会丢失一部分数据，影响数据库完整性。

示例

```
SHUTDOWN NORMAL;
SHUTDOWN IMMEDIATE;
SHUTDOWN ABORT;
```

yasboot工具方式

可通过使用yasboot工具关闭数据库实例：

```
$ yasboot cluster stop -c yashandb
$ yasboot cluster stop -c yashandb -s normal
$ yasboot cluster stop -c yashandb -s immediate
$ yasboot cluster stop -c yashandb -s abort

# 关闭group_id为1的节点组
$ yasboot group stop -c yashandb -g 1
# 关闭node_id为4-1的节点
$ yasboot node stop -c yashandb -n 4-1
```

其中，`-s` 参数用于指定关库方式，可指定为NORMAL/IMMEDIATE/ABORT方式，省略则默认为IMMEDIATE方式。

> **Caution：**

> 一般当服务器宕机、断电，或者人为强制关库时才建议使用ABORT方式，否则应避免使用这种方式停止实例，防止出现数据丢失，或者数据库损坏。

归档管理

YashanDB通过开启归档模式来进行redo日志文件自动归档，用以支持生产环境中的数据热备份以及高可用主备部署场景的主备同步。当故障发生时，可以通过历史全量数据数据备份以及归档的redo日志文件重做完成数据库重建。

查看归档配置

V$DATABASE视图中的log_mode字段表示数据库当前的归档模式配置，字段值为ARCHIVELOG表示为归档模式，NOARCHIVELOG表示非归档模式。

示例

```
SELECT database_name,log_mode,open_mode FROM V$DATABASE;

DATABASE_NAME      LOG_MODE          OPEN_MODE
------------------ ----------------- ------------
yasdb              ARCHIVELOG        READ_WRITE
```

配置归档路径

数据库默认的归档路径为$YASDB_DATA/archive，该值由ARCHIVE_LOCAL_DEST参数控制，进入到数据库可以查看并修改归档文件存放路径。

1.查看归档路径：

```
show parameter ARCHIVE_LOCAL_DEST;

NAME                    VALUE
----------------------- --------------
ARCHIVE_LOCAL_DEST      ?/archive
```

2.修改归档路径：

通过SQL语句修改归档路径，重启数据库后生效。

```
ALTER SYSTEM SET ARCHIVE_LOCAL_DEST='/home/yashan' scope=spfile;
```

3.查看配置是否生效：

```
show parameter ARCHIVE_LOCAL_DEST;

NAME                    VALUE
----------------------- --------------
ARCHIVE_LOCAL_DEST      /home/yashan
```

归档模式切换

> **Caution：**
> 归档模式切换需要关闭数据库并启动到MOUNT状态下操作，请在无业务运行时参考下列步骤完成切换。

1.关闭数据库：

```
--关闭数据库实例
SHUTDOWN IMMEDIATE;
exit;
```

2.重启数据库：

```
$ yasboot cluster restart -c yashandb -m mount
$ yasql username/password
```

3.调整到归档模式：

```
-- 从非归档模式调整到归档模式
ALTER DATABASE ARCHIVELOG;
-- 从归档模式调整到非归档模式，当数据库处于主备复制模式（单机主备部署、主备共享集群部署）时，无法从归档模式切换为非归档模式
ALTER DATABASE NOARCHIVELOG;
```

4.归档模式查看：

```
--查看当前数据库的归档模式
SELECT database_name,log_mode,open_mode FROM V$DATABASE;

DATABASE_NAME    LOG_MODE          OPEN_MODE
---------------- ----------------- -------------
yasdb            ARCHIVELOG        READ_WRITE
```

5.打开数据库：

```
--调整归档模式后打开数据库
ALTER DATABASE OPEN;
```

在线日志切换

当前正在写入的redo日志文件写满时，数据库会自动切换至下一可用redo日志文件，用户也可根据自身需求选择手动切换：

- 强制切换当前正在写入的redo日志
- 对当前正在写入的redo日志进行归档，同时切换至下一可用redo日志文件（数据库须处于归档模式）

示例

```
--强制切换
ALTER SYSTEM SWITCH LOGFILE;

--归档并进行切换（须处于归档模式否则返回错误）
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

归档日志查看

可以通过V$ARCHIVED_LOG视图查看归档日志的信息，如归档日志的序列号、归档日志名和归档时间等。

示例

```
SELECT * FROM V$ARCHIVED_LOG;
```

归档日志清理

归档空间有限的情况下，需要根据保留策略对归档日志进行清理。默认安装下，在线日志的大小为128M，则归档日志最大可为128M。

清理归档的原则：归档日志不被数据库回放需要，即小于数据库的回放点，这样的归档才可以被清理。可以从V$DATABASE视图的RCY_POINT获取数据库的当前回放点。

自动清理归档

YashanDB具有自动清理归档日志的功能，清理策略由以下三个系统配置参数决定：

- ARCH_CLEAN_IGNORE_MODE：指定清理归档文件时的忽略模式，包括如下值（默认为NONE）：
  - NONE：表示清理归档文件时不忽略备份和备库。
  - BACKUP：表示清理归档时忽略备份，此设置可能导致数据库无法恢复至任意时间点 。
  - STANDBY：表示清理归档时忽略备库，此设置可能导致备库跟不上主库，出现need repair状态，单机部署中默认忽略备库。

- BOTH：表示清理归档时忽略备份和备库，此设置可能导致如上所述的两种问题均会出现。
- 其中：
  - 忽略备份指的是无论该归档文件是否已经备份，均会被清理。
  - 忽略备库指的是无论该归档文件是否已经被所有备库获取，均会被清理。

Caution：

- YashanDB支持通过指定时间点的方式对数据库进行恢复。若在备份集后生成的归档日志和在线日志连续且完整，可通过指定时间点使数据库继续恢复至任意时间点；若备份集之后的归档日志被清理，则数据库无法继续恢复至任意时间点。
- 高可用运行过程中，备库需要通过获取主库的归档日志对主库进行同步，如清理的归档文件还未被备库所获取，会导致备库无法正常同步主库。

- ARCH_CLEAN_UPPER_THRESHOLD：指定归档清理触发的阈值，归档日志总大小超出这个值时进行归档清理，如果该值为0表示关闭自动清理。
- ARCH_CLEAN_LOWER_THRESHOLD：指定归档空间保持的最小值，触发一次自动清理归档后，保留归档日志总大小的最小值。

上述配置参数可以通过ALTER SYSTEM命令进行在线修改。ARCH_CLEAN_IGNORE_MODE参数同时也控制手动清理归档的策略。

手动清理归档

手动清理归档日志的清理条件由ARCH_CLEAN_IGNORE_MODE参数决定，具体如上所述，执行手动清理归档前先通过配置ARCH_CLEAN_IGNORE_MODE参数指定合适的忽略模式。

手动清理归档日志需要进入到数据库执行ALTER DATABASE语句，语法如下：

```
ALTER DATABASE DELETE ARCHIVELOG { ALL | UNTIL TIME date_string | UNTIL SEQUENCE asn }[FORCE]
```

- ALL：删除所有符合清理条件的归档日志。
- UNTIL TIME date_string ：删除截至date_string时间前所有符合清理条件的归档日志。
- UNTIL SEQUENCE asn： 删除序列号为ASN的归档及序列号为ASN之前的所有符合清理条件的归档。ASN可在归档文件的序列号直接查看，或者通过V$ARCHIVED_LOG视图的SEQUENCE#查看。
- FORCE：不考虑清理条件，强制清理归档。

示例（单机、共享集群部署）

```
--删除所有归档日志
ALTER DATABASE DELETE ARCHIVELOG ALL;
--删除包括2022-01-06 11:30:00之前的归档日志
ALTER DATABASE DELETE ARCHIVELOG UNTIL TIME TO_DATE('2022-01-06 11:30:00', 'yyyy-mm-dd hh24:mi:ss');
--删除序列号包括71号之前的归档日志
ALTER DATABASE DELETE ARCHIVELOG UNTIL SEQUENCE 71;
--强制归档清理
ALTER DATABASE DELETE ARCHIVELOG ALL FORCE;
```

字符集配置

YashanDB服务端和客户端支持GBK、UTF-8、GB18030、ASCII和ISO-8859-1字符集，并支持根据场景需要配置数据库字符集。

背景信息

默认安装下，YashanDB服务端、Linux客户端、JDBC客户端等均为UTF-8字符集，Windows客户端字符集为GBK。

当服务端和客户端设置的字符集不同时，数据库操作产生的最终结果可能会与预期不一致，此时在客户端输入的字符串会以服务端字符集格式进行处理。

YashanDB中国家字符集仅支持为UTF-16，建库时指定，后续无法更改。

字符集配置原则

字符集	配置原则
GBK	如数据库只需要支持中文，数据量很大，性能要求也很高，建议选择双字节定长编码的中文字符集GBK。
UTF-8	如应用程序需要处理各种各样的文字，或者需要将处理结果发布到不同语言的国家或地区，建议选择Unicode字符集，即UTF-8。此项为YashanDB推荐和默认的字符集。
ASCII	如数据库只需要支持ASCII收录的拉丁系字符，如英语和一些西欧语言，则可以选择ASCII字符集。
ISO-8859-1	此字符集为单字节编码，能表示的字符范围是0-255，仅应用于全英文场景。
GB18030	此字符集达到GB18030-2022标准的实现级别三。如果数据库有大量使用中文的场景，且对生僻字的显示、处理、输出有比较严格的要求，可以选择此字符集。

> **Note**：
> YashanDB的GB18030字符集不支持ASCII范围以外的字符进行大小写的转换。

配置方法

设置服务端字符集

服务端字符集的限制如下：

- 若数据库已创建，不允许再修改其字符集配置。
- 如需使用TAC表或LSC表，数据库服务端的字符集必须设置为UTF-8。

服务端字符集设置方法：

- 初始数据库：在安装部署过程中，可通过设置yashandb.toml配置文件中的建库参数CHARACTER_SET指定初始数据库的字符集。
- 非初始数据库：在安装完成后，如需删除初始数据库并自定义新建数据库，可以通过CREATE DATABASE语句中的CHARACTER SET字段指定新数据库的字符集。

```
CREATE DATABASE yashan CHARACTER SET utf8;
```

修改客户端字符集

1. 在YashanDB客户端文件夹中新建client文件夹，并于client文件夹中新建空文件yasc_env.ini。

```
$ mkdir client
$ cd client
$ vi yasc_env.ini
```

2. 设置环境变量。

```
$ vi ~/.bashrc
export YASDB_HOME=/home/yasdb/yashandb_client
$ source ~/.bashrc
```

> **Note**：
> YASDB_HOME路径需要指向YashanDB客户端文件夹。

3. 通过修改客户端环境变量文件yasc_env.ini设置客户端字符集。

```
$ cd $YASDB_HOME/client
$ vi yasc_env.ini
添加
CHARACTER_SET=GBK（或者ASCII，或者ISO88591）
```

参数配置

参数初始化配置

YashanDB的所有系统参数都存在一个默认值，允许在产品安装后不做任何处理也能启动实例。默认值一般基于在个人PC也能运行的最小配置给出，可能并不适用于生产环境，因此建议在安装YashanDB时也进行参数的初始化配置工作。

参数的初始化通过在产品安装过程中配置安装参数完成。

需进行初始化配置的参数范围和值应结合企业的自身业务、资源环境来确定，同时，在调优手册数据库配置调优中所列出的性能相关参数应该重点关注。

若系统中使用了过时参数，启动时会产生运行日志告警但不影响启动，可根据配置参数中具体的参数描述和告警内容，判断该参数是否因改名而过时，并且是否需要将该过时参数调整为对应的新参数。若新参数未配置，则过时参数的配置值将自动转换为新参数值。

示例

```
//当配置文件配置了过时参数BROADCAST_GTS_TIME时，run.log打印情况如下：
2024-03-15 17:10:37.606 21615 [WARN] [PARAM] parameter BROADCAST_GTS_TIME is deprecated, it will obsolete in later version
```

参数自适应配置

数据库的参数配置直接影响性能表现，对参数值的初始化和后续调整依赖于专业的DBA，为协助DBA快速决策和减少运维难度，YashanDB提供参数自适应功能，依据环境信息给出针对当前负载的调参配置推荐。

基本原理

1.可在数据库NOMOUNT、MOUNT或OPEN状态下运行，区分HEAP/TAC/LSC不同表类型。

2.获取CPU核数、内存总数、空闲内存数等环境信息。

3.结合用户指定的CPU和内存限制，计算数据库可用的资源。

4.依据数据库可用资源计算内存相关参数。

5.测试数据文件和日志文件所在磁盘性能。

6.依据磁盘性能结果计算IO相关参数。

操作步骤

1.选择无业务运行的时间，并清理其他进程，避免干扰对资源的计算和测试。

2.运行参数配置推荐程序，该程序存在可输入的参数，请查看开发手册DBMS_PARAM详细了解后再操作。

```
--使用默认参数生成推荐参数，不写入配置文件。
EXEC DBMS_PARAM.OPTIMIZE();

--使用TAC表类型，分配80%的内存，100%的CPU，生成推荐参数后，写入配置文件。
EXEC DBMS_PARAM.OPTIMIZE(True, 'TAC', 80);

--使用默认的HEAP表类型，分配100%的内存，100%的CPU，生成推荐参数后，不写入配置文件。
EXEC DBMS_PARAM.OPTIMIZE(NULL, NULL, 100, 100);

--使用LSC表类型，分配100%的内存，100%的CPU，指定datafile和redofile的路径，生成推荐参数后，不写入配置文件。
EXEC DBMS_PARAM.OPTIMIZE(NULL, 'LSC', NULL, NULL, '/home/yashan/data', '/home/yashan/redo');
```

3.查看最新的推荐参数信息。

```
SELECT DBMS_PARAM.SHOW_RECOMMEND() FROM dual;
```

4.将推荐的参数配置写入yasdb.ini文件，此过程并不会生效参数。

```
EXEC DBMS_PARAM.APPLY_RECOMMEND();
```

5.重启数据库，生效参数。

修改参数配置

在系统运行过程中，可能会由于应用变化、资源扩充、性能调优等需求，手工调整某个参数的值。此时应该根据参考手册配置参数文档里对该参数的描述，评估调整参数是否对在线业务造成影响，确定执行操作的合适时机。

使用SQL命令修改参数

命令格式

参考开发手册ALTER SYSTEM和ALTER SESSION文档说明。

参数说明

alter system和alter session分别用于修改系统级别和会话级别的配置参数，而alter session修改方式默认只写到内存，对当前会话生效。

param_name和value分别为配置参数名称和值，其中值需满足系统指定的取值范围，详见参考手册配置参数描述。

scope用于设定对配置参数修改后的生效方式，默认为both。

- spfile：将参数值写入参数文件，需重启才能生效。
- memory：将参数值写入内存，立即生效，但重启后失效。
- both：将参数值同时写入内存和参数文件，立即生效，重启后也生效。

> **Note**：
> - 过时参数仍支持修改，并且修改时将在运行日志打印告警，告警内容同参数初始化配置示例内容。
> - 立即生效时，若为因改名而过时的参数且对应的新参数未配置，则过时参数修改后的值将自动转换为新参数值。
> - 重启生效的过时参数在修改后重启数据库时，运行日志也将打印告警，告警内容同上。

查看配置参数

修改后通过show命令或查询parameter相关参数视图来查看配置参数是否生效。

> **Note**：
> 过时参数仍支持查询，查询结果与正常参数区别在于查询parameter相关参数视图时过时参数的IS_DEPRECATED字段为TRUE。

```
SQL > show parameter param_name;
```

示例

```
--修改日期格式参数
--1、查看当前参数值
SQL > SHOW PARAMETER date_format;
NAME              VALUE
----------------- --------------
DATE_FORMAT       yyyy-mm-dd


--2、修改参数值，date_format为需重启生效参数
SQL > ALTER SYSTEM SET date_format='yyyy-mm-dd hh24:mi:ss' scope=spfile;

--3、重启数据库
--4、登录后查看参数值
SQL > SHOW PARAMETER date_format;
NAME            VALUE
--------------- ------------------------
DATE_FORMAT     yyyy-mm-dd hh24:mi:ss

SELECT * FROM V$PARAMETER WHERE name = 'DATE_FORMAT';
```

```
NAME                                                             VALUE
DEFAULT_VALUE                                                    IS_DEPRECATED
---------------------------------------------------------------- ----------------------------------------------------------------
---- ---------------------------------------------------------- -------------
DATE_FORMAT                                                      yyyy-mm-dd hh24:mi:ss
yyyy-mm-dd                                                       FALSE


--查看过时参数
SELECT * FROM V$PARAMETER WHERE IS_DEPRECATED = TRUE;

NAME                                                             VALUE
DEFAULT_VALUE                                                    IS_DEPRECATED
---------------------------------------------------------------- ----------------------------------------------------------------
---- ---------------------------------------------------------- -------------
DIN_RECONNECT_TIME                                               5000
5000                                                             TRUE
CGROUP_FLAG                                                      0
0                                                                TRUE
BROADCAST_GTS_TIME                                               16
5                                                                TRUE

SELECT * FROM V$PARAMETER WHERE name = 'BROADCAST_GTS_TIME';

NAME                                                             VALUE
DEFAULT_VALUE                                                    IS_DEPRECATED
---------------------------------------------------------------- ----------------------------------------------------------------
---- ---------------------------------------------------------- -------------
BROADCAST_GTS_TIME                                               16
5                                                                TRUE
```

Note：

- 无法对只读参数执行ALTER SYSTEM，只读参数只能在安装时进行配置或使用默认值。
- 通常情况下，建议一直使用SQL命令修改配置参数，修改后的参数值将持久化到yasdb.ini配置文件，不建议直接编辑该配置文件。
- 分布式部署中，上述命令将修改分布式集群中本地节点的对应配置，依据开发手册ALTER SYSTEM中type和node所描述内容可以实现多节点配置，但仍建议使用YCM图形化界面或者yasboot命令行方式来进行多节点统一的参数配置。

SSL连接配置

YashanDB启用SSL连接要求由服务器生成根证书、二级证书和DH文件，客户端获取服务器的根证书，在通讯时进行客户端到服务端的安全验证。

> **Caution**：
> - 一旦服务器开启SSL连接，所有的客户端都必须有根证书才能连接到数据库。
> - 启用SSL连接和用户密码认证无关联，用户登录数据库仍需输入密码。

服务端配置

生成证书

通过openssl工具生成证书，该工具已内置到产品安装包，无需单独安装。

以下步骤中的工具命令选项、路径、名称等无限制，用户可依据自身环境和需要替换为其他值。

1. 生成根证书（自签名，包含服务端公钥）：

   根证书用于给数据库服务器证书签名，和发送给客户端用于SSL连接验证，本手册生成证书的路径为/home/yashan/YASDB_DATA/config，以该路径为例进行介绍。

```
$ openssl req -new -x509 -days 365 -nodes -out root.crt -keyout ca.key -subj "/CN=RootCA"
```

2. 生成根证书请求文件和服务器私钥：

```
$ openssl req -new -nodes -text \
-out server.csr \
-keyout server.key \
-subj "/CN=server"
```

3. 生成二级证书并签名：

```
$ openssl x509 -req -in server.csr -text -days 5 \
-CA root.crt \
-CAkey ca.key \
-CAcreateserial \
-out server.crt
```

4. 生成DH文件：

```
$ openssl dhparam -2 -out dhparam.pem -text 2048
```

> **Note**：
> 根证书或二级证书过期后，需重新生成证书。

配置参数

1. 在数据库打开SSL连接开关，并配置证书路径。其中，路径仅可指定为绝对路径，且最长不超过254字节。

```
ALTER SYSTEM SET ssl_enable = ON SCOPE=spfile;
ALTER SYSTEM SET ssl_cert_file = /home/yashan/YASDB_DATA/config/server.crt SCOPE=spfile;
ALTER SYSTEM SET ssl_dh_param_file = /home/yashan/YASDB_DATA/config/dhparam.pem SCOPE=spfile;
ALTER SYSTEM SET ssl_key_file = /home/yashan/YASDB_DATA/config/server.key SCOPE=spfile;
```

2. 重启数据库。

> **Warn**：

如打开了SSL连接开关，但未配置证书路径，或者配置路径不正确，数据库将无法启动。

客户端配置

本手册以Linux平台为例介绍客户端配置。

1. 下载服务端根证书，放至本地路径，如/home/yasdb/cert。

2. 在YashanDB客户端文件夹中新建client文件夹，并于client文件夹中新建空文件yasc_env.ini。

```
$ mkdir client
$ cd client
$ vi yasc_env.ini
```

3. 设置环境变量。

```
$ vi ~/.bashrc
export YASDB_HOME=/home/yasdb/yashandb_client
$ source ~/.bashrc
```

Note：
YASDB_HOME路径需要指向YashanDB客户端文件夹。

4. 在{YASDB_HOME}/client/yasc_env.ini中增加如下配置：

```
ssl_root_cer = /home/yasdb/cert/root.crt
```

其中，配置中的路径可指定为绝对或相对路径，但不能为含有../的相对路径，且系统按前255字节长度进行文件读取。

操作系统身份认证配置

本文主要介绍如何配置操作系统身份认证，以便数据库管理员更便捷地登录数据库，认证方式相关的详细介绍请查阅用户及认证。

开启操作系统认证功能

该功能开关由数据库配置文件yasdb_net.ini中的ENABLE_LOCAL_OSAUTH参数的取值控制。YashanDB安装后，默认ENABLE_LOCAL_OSAUTH = on，即开启操作系统身份认证功能，且yasdb_net.ini文件不会自动生成。

1. 查询$YASDB_DATA/config路径下是否存在yasdb_net.ini文件。

```
$ echo $YASDB_DATA
/data/yashan/yasdb_data/db-1-1 # 本文以/data/yashan/yasdb_data/db-1-1为例

$ cd /data/yashan/yasdb_data/db-1-1/config
$ ll
```

   - 如不存在，则说明已开启，无需额外操作。
   - 如存在，则需执行后续操作。
2. 查看ENABLE_LOCAL_OSAUTH的值：

```
$ vi yasdb_net.ini
```

   若ENABLE_LOCAL_OSAUTH值为off，则需修改为on，保存并退出。重启数据库使配置生效。

为用户开通操作系统认证

1. 查看是否有YASDBA组，如没有则新建（需由拥有建组权限的用户操作）：

```
$ groups
$ groupadd YASDBA
```

2. 将目标操作系统账号加入YASDBA组（需由拥有建组权限的用户操作）：

```
$ usermod -a -G YASDBA yashan
$ groups yashan
```

3. 验证操作系统身份认证：

```
$ yasql / as sysdba

YashanDB SQL Enterprise Edition Release 23.2.0.2 x86_64

Connected to:
YashanDB Server Enterprise Edition Release 23.2.0.2 x86_64 - X86 64bit Linux

SQL> SELECT SYS_CONTEXT ( 'USERENV' , 'SESSION_USER' ) FROM DUAL;

SYS_CONTEXT('USERENV
----------------------------------------------------------------
SYS
```

取消用户的操作系统认证

- 方式一：将目标用户从YASDBA组移除，该用户将不再允许操作系统身份认证，立即生效。
- 方式二：删除YASDBA组，所有用户将不再允许操作系统身份认证，立即生效。

关闭操作系统认证功能

1. 查询$YASDB_DATA/config路径下是否存在yasdb_net.ini文件，如不存在则创建：

```
$ echo $YASDB_DATA
/data/yashan/yasdb_data/db-1-1 # 本文以/data/yashan/yasdb_data/db-1-1为例

$ cd /data/yashan/yasdb_data/db-1-1/config
$ vi yasdb_net.ini
```

2. 在yasdb_net.ini文件中新增或修改如下配置：

```
ENABLE_LOCAL_OSAUTH = off
```

3. 保存并退出。

4. 重启数据库使配置生效，数据库将不再执行操作系统身份认证。

数据库删除

对于一个数据库实例，通过执行删除语句DROP DATABASE可以删除该数据库实例的所有数据文件，且数据库删除后需使用CREATE DATABASE语句重新创建新的数据库。

DROP DATABASE语句不适用于分布式部署。

约束条件

- 数据库处于NOMOUNT模式，且数据库数据文件完整、能够正常启动使用。
- YashanDB采用单机部署或共享集群部署。

操作步骤

> **Warn**：
> 删除数据库将彻底删除该数据库所有相关文件，请谨慎操作。

1.重启实例至NOMOUNT模式。

```
SHUTDOWN;
exit

$ yasboot cluster start -c yashandb -m nomount
```

2.以sys用户登录系统，删除数据库。

```
$ yasql sys/sys

-- 删除除归档外的所有数据库文件
SQL> DROP DATABASE;

-- 删除包括归档在内的所有数据库文件
DROP DATABASE including ARCHIVELOG;
```

数据库托管

YashanDB支持将某个单节点数据库或数据库集群托管到指定的集群内。

注意事项

- 托管过程主要使用yasboot，请先了解该工具各命令选项含义及说明。
- 对于遵照安装部署执行安装的数据库以及已完成托管的数据库，无需执行托管操作。

操作步骤

除特殊说明，以下步骤均在yasboot工具所在的环境上操作，且命令中的路径、用户、密码等需更换为实际值。

1. 生成服务器配置文件。

   通过执行如下命令生成服务器配置文件：

   ```
   $ ./bin/yasboot package config gen --cluster yashandb -u yashan -p password --ip ip1,ip2,...,ipn --install-path
   /var/database/yashan
   ```

   其中：

   ```
   -c,--cluster    生成的集群名称，这里可以任意取一个名称
   -u,--username   服务器ssh用户名
   -p,--password   服务器ssh登录密码
   -N,--no-password   如果用户配置了免密，请使用这个参数替代-p参数
   --ip            部署的ip地址，如果涉及到多个服务器，格式为：ip1,ip2,ip3
   --port          服务器ssh连接端口
   -i, --install-path  数据库安装路径（HOME目录），这里指定的路径是新版本的home目录，必须要是空目录
   -t, --yas-type      数据库部署的架构类型：
   * SE：单机
   * DE：分布式
   --db             单机db组及节点部署的节点规模，请根据实际主备情况调整此参数
   ```

   > **Note**：
   > 该命令会生成服务器hosts.toml和数据库yashandb.toml两个配置文件，将直接替换/home/yashan/intall目录下原有的hosts.toml及yashandb.toml文件，托管只需要hosts.toml服务器配置文件，yashandb.toml可以忽略。

2. 部署主库，并将yasom和yasagent初始化。

   可通过执行如下命令进行初始化：

   ```
   $ ./bin/yasboot package install -i yashandb-22.2.0.9-linux-x86_64.tar.gz -t hosts.toml
   checking install package...
   install version: yashandb 22.2.0.9
   host0001 100% [=========================================================================]    3s
   update host to yasom...
   ```

   其中：

   ```
   -i, --install-pkg  软件包文件本地路径
   -t, --toml 服务器配置文件，即hosts.toml
   ```

3. 生成托管配置模板文件。

   可通过如下命令生成于目录中生成托管模板文件：

   ```
   $ ./bin/yasboot package config join-demo -t SE
   ```

其中：

```
-t, --type 托管集群的类型，SE表示单机，DE表示分布式
```

4. 修改托管配置文件。

   如下为托管模板文件各参数描述，请根据实际情况修改模板文件中的参数值：

```
cluster = "yashandb"             --集群名称
sys_password = "yasdb_123"  --数据库sys用户的密码
yas_type = "SE"              --部署形态

[primary_config]
  manage_ip = "192.168.1.2"                                  --主节点ip
  yasdb_home = "/opt/yasom/yashandb/yashandb"                    --主节点的yasdb_home目录
  node_path = "/opt/yasom/yashandb/data/yashandb/db-1-1"         --主节点的yasdb_data目录
  node_id = "1-1:1"                                        --主节点的nodeid

[[standby_config]]                                         --备节点按照主节点的示例修改
  manage_ip = "192.168.1.2"
  yasdb_home = "/opt/yasom/yashandb/yashandb"
  node_path = "/opt/yasom/yashandb/data/yashandb/db-1-2"
  node_id = "1-2:1"

[[standby_config]]                     --备节点按照主节点的示例修改，没有就删除
  manage_ip = "192.168.1.2"
  yasdb_home = "/opt/yasom/yashandb/yashandb"
  node_path = "/opt/yasom/yashandb/data/yashandb/db-1-3"
  node_id = "1-3:1"
```

5. 执行托管命令。

   通过执行如下命令进行托管操作：

```
$ ./bin/yasboot cluster join -t SE --config join_demo.toml
```

其中：

```
-t, --type 托管集群的类型，SE表示单机，DE表示分布式
-c, --config 托管配置文件
```

命令执行结果如下所示：

```
the cluster status is as follow:
|key         |value
|------------+------
|clusterName |tp
|version     |22.2.0.9

the cluster status is as follow:
 hostid    | node_type | nodeid | pid    | instance_status | database_status | database_role | listen_address | data_path
-------------------------------------------------------------------------------------------------------------------------------------------
----------------------
 host0001 | db        | 1-1:1  | 46689 | open            | normal          | primary       | 192.168.1.2:1688   |
/home/yashan/yashandb/db-1-1
----------+-----------+--------+-------+-----------------+-----------------+---------------+----------------+--------------
----------------------
Check success
Are you sure you to add yasdb tp to yasom[yes/no]: yes
 type | uuid             | name             | hostid | index | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------
 task | e2007922a317a02c | JoinYasdbCluster | -      | tp    | SUCCESS | 0           | 100      | -
------+------------------+------------------+--------+-------+---------+-------------+----------+------
task completed, status: SUCCESS
```

6. 查看数据库状态。

   通过执行如下语句查看数据库状态：

```
$ ./bin/yasboot cluster status -c yashandb -d
 hostid   | node_type | nodeid | pid   | instance_status | database_status | database_role | listen_address | data_path
------------------------------------------------------------------------------------------------------------------------------
--------------
 host0001 | db        | 1-1:1  | 46689 | open            | normal          | primary       | 192.168.1.2:1688   |
/home/yashan/yashandb/db-1-1
----------+-----------+--------+-------+-----------------+-----------------+---------------+----------------+--------------
--------------
```

异构数据库链接配置

异构数据库链接，指从YashanDB创建到其他非YashanDB的远程链接（DBLINK）。

对于从YashanDB到Oracle数据库的链接，系统存在如下前置要求：

- YashanDB服务端已下载和安装Oracle Instant Client。

  未安装Oracle Instant Client的环境中，通过DBLINK向Oracle数据库发起远程链接，将会由于缺少必要的组件而抛出错误。

- YashanDB服务端已安装libaio库。

  YashanDB服务端缺少libaio库时，如通过DBLINK向Oracle数据库发起首次远程链接，yex_server沙箱进程可能在加载驱动时发生core dump（yasdb进程无影响）。本错误只在首次链接时影响yex_server进程，再次链接时系统可自动恢复正常。

鉴于上述要求，对于可能使用YashanDB -> Oracle数据库远程链接的数据库，管理员应按下述指导进行必要的操作。

Oracle Instant Client下载和安装

1. 以YashanDB安装用户（例如yashan）登录到数据库服务器。

2. 根据YashanDB服务器环境，从Oracle官网 下载对应的Oracle Instant Client安装包。

3. 将安装包解压到本地路径，例如 `/home/oracle-instant-client` /。

4. 设置动态库依赖路径。

```
$ vi ~/.bashrc
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/oracle-instant-client/lib
$ source ~/.bashrc
```

libaio库安装

1. 以YashanDB安装用户（例如yashan）登录到数据库服务器。

2. 以Centos为例，通过以下命令安装libaio库：

```
yum install libaio
```

文件管理

YashanDB后台包含了如下重要的物理文件：

- 参数文件：参数文件为文本文件，包含数据库实例的内存组件大小、监听端口、字符集、数据块大小与控制文件等实例配置信息。
- 密码文件：密码文件为文本文件，加密存储着SYS用户的密码，SYS用户登录数据库校验过程中使用到该文件。
- 控制文件：二进制文件，存放数据库关键信息，为数据库最核心的文件之一。
- 数据文件：二进制文件，存储数据库数据，仅可由数据库读写。
- redo日志文件：redo日志的二进制文件，用于redo日志的持久化。
- 归档日志文件：redo日志完成归档后形成的二进制文件，归档日志可用于数据恢复，也可以在主备库中的备库上应用，还原主库上的数据。

禁止手动变更数据库控制文件、数据文件、Redo文件，包括删除、移动、权限变更等，否则会导致数据库无法启动、运行异常等严重后果。

上述文件均有相应的功能与作用，缺少任何一个文件都会影响数据库的正常使用。例如缺少参数文件就无法启动数据库实例；缺少密码文件SYS用户就无法登录；缺少控制文件数据库就无法启动到mount状态。

配置参数文件与密码文件管理

YashanDB在产品安装时会创建两个重要的配置文件：参数文件和密码文件，分别控制其所在实例的数据库的系统配置参数和系统用户口令，删除或破坏这两个文件将导致所在实例无法启动或登录。

配置参数文件

配置参数文件是名为yasdb.ini的文本文件，存放在$YASDB_DATA/config路径下，产品安装时所指定的数据库初始配置参数将保存在此文件中，后续数据库运行过程中对配置参数的非memory修改也将持久化到此文件中。

> **Note**：
> 本文以单机部署为例，不同部署形态的$YASDB_DATA路径不同，具体请以实际为准。

```
$ cat yasdb.ini
_ENABLE_TAC=FALSE
_ENABLE_LSC=FALSE
_ENABLE_EPC=FALSE
LISTEN_ADDR=0.0.0.0:1688
DB_BLOCK_SIZE=8K
DATA_BUFFER_SIZE=256M
CHARACTER_SET = UTF8
CONTROL_FILES = ('/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1', '/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl2',
'/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl3')
```

> **Caution**：
> 通常情况下，不建议直接编辑yasdb.ini文件来修改配置参数，避免出现不可预计的异常。如需修改请使用SQL命令，详细操作请参考开发手册ALTER SYSTEM。

密码文件

密码文件名为yasdb.pwd，为文本文件，不可直接进行编辑，其路径受PASSWORD_FILE参数控制。

示例

```
show parameter PASSWORD_FILE;

NAME              VALUE
----------------- ----------------------
PASSWORD_FILE     ?/instance/yasdb.pwd
```

以下为yasdb.pwd内容示例：

示例

```
$ cat yasdb.pwd
M'´iSYSS:9E276A5EFA869D8DD05E6CE27CC8430194D04451D6FE9EBCAA7BE0A933800084621C2654CA20D47DFFC9
```

该文件内容为加密后信息，虽然为文本文件，但不能直接修改里面的内容，如果需要修改SYS用户的密码，可以通过以下命令完成：

示例

```
$ yaspwd file=yasdb.pwd
$ yaspwd file=yasdb.pwd input_file=yasdb_input.pwd sys=y
```

> **Note**：
> yasdb.pwd文件不存在时，才能生成新的密码文件，因此修改密码前应该先将yasdb.pwd删除或改名，详细操作请参考工具手册yaspwd。

控制文件管理

控制文件默认有3份拷贝（最多8份），具体个数和控制文件路径可以在配置参数文件中指定。

> **Note**：
> 本文以单机部署为例，不同部署形态的$YASDB_DATA路径不同，具体请以实际返回结果为准，并按需修改为实际的控制文件路径。

查看控制文件

1.SQL命令查看：

```
show parameter CONTROL_FILES;

NAME             VALUE
---------------- ------------------------------------------------------------
CONTROL_FILES    ('/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1', '/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl2',
'/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl3')
```

2.动态视图查看：（在数据库在mount或open状态执行）

```
SELECT * FROM V$CONTROLFILE;
  ID NAME                                               BLOCK_SIZE FILE_SIZE_BLKS       BYTES
---- ------------------------------------------------- ------------ -------------- -----------
   0 /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1            8192           3507    28729344
   1 /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl2            8192           3507    28729344
   2 /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl3            8192           3507    28729344
```

修改控制文件路径

修改数据库的控制文件路径至不同的路径，可以提高控制文件的安全性，示例步骤如下：

1.修改控制文件参数：

```
ALTER SYSTEM SET control_files=('/data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1','/data/yashan/yasdb_data/db-1-
1/dbfiles/ctrl2','/data/yashan/yasdb_data/db-1-1/backup/ctrl3') scope=spfile;
```

2.关闭数据库：

```
SHUTDOWN IMMEDIATE;
```

3.移动ctrl3控制文件：

```
$ mv /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl3 /data/yashan/yasdb_data/db-1-1/backup/ctrl3
```

4.文件系统层检查确认ctrl3控制文件：

```
ls -l /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1
ls -l /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl2
ls -l /data/yashan/yasdb_data/db-1-1/backup/ctrl3
```

5.启动数据库：

```
$ yasboot cluster start -c yashandb
```

如果数据库启动成功并没有报错，则修改控制文件多路径成功。

6.数据库启动后，yasql登录到数据库检查确认ctrl3控制文件（数据库在mount或open状态执行）：

```
SELECT * FROM V$CONTROLFILE;
  ID NAME                                                BLOCK_SIZE FILE_SIZE_BLKS       BYTES
---- -------------------------------------------------- ---------- -------------- -----------
   0 /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl1             8192           3507    28729344
   1 /data/yashan/yasdb_data/db-1-1/dbfiles/ctrl2             8192           3507    28729344
   2 /data/yashan/yasdb_data/db-1-1/backup/ctrl3              8192           3507    28729344
```

Warn：

若要移动控制文件至其他路径，必须先手动修改控制文件路径，在移动对应控制文件至目标位置。 不可在生产环境正常执行状态下单独移动控制文件至其他位置。

数据文件管理

数据文件的存放位置

默认情况下，数据文件存放在`$YASDB_DATA/dbfiles`目录下（SCOL数据默认存放在与data同级的local_fs目录下）。

共享集群部署模式下的数据文件存放在`+DG0/dbfiles`目录下，访问方式请参考YFS文件管理。

```
$ cd $YASDB_DATA/dbfiles
$ ls -lrt
total 1122904
...
-rw-r----- 1 yashan yashan 134217728 Oct 19 11:50 redo1
-rw-r----- 1 yashan yashan 134217728 Oct 19 14:15 redo2
-rw-r----- 1 yashan yashan 134217728 Oct 19 11:50 redo3
-rw-r----- 1 yashan yashan 134217728 Oct 19 11:50 redo4
-rw-r----- 1 yashan yashan  67108864 Oct 19 11:50 temp
-rw-r----- 1 yashan yashan  67108864 Oct 19 11:50 swap
-rw-r----- 1 yashan yashan  67108864 Oct 19 11:50 users
-rw-r----- 1 yashan yashan  67108864 Oct 19 14:00 sysaux
-rw-r----- 1 yashan yashan  67108864 Oct 19 14:15 system
-rw-r----- 1 yashan yashan 134217728 Oct 19 14:15 undo
-rw-r----- 1 yashan yashan  67108864 Oct 19 14:15 dwf
-rw-r----- 1 yashan yashan  25370624 Oct 19 14:15 ctrl1
-rw-r----- 1 yashan yashan  25370624 Oct 19 14:15 ctrl2
-rw-r----- 1 yashan yashan  25370624 Oct 19 14:15 ctrl3
...
```

各类文件在不同部署形态中略有差异：

- redo：redo日志文件
- temp、swap、users、sysaux、system、undo、users_aim-0（分布式部署独有）：表空间数据文件；TSS_*CHUNK*_FILE_0（分布式部署独有）：表空间集下包含了chunk的表空间的数据文件
- dwf、double_write（分布式部署）：双写文件
- ctrl：控制文件

查看数据文件

除了直接在操作系统进入到数据文件存放的目录查看外，还可以通过数据库的DBA_DATA_FILES和V$DATAFILE视图查看，如下：

示例（单机部署）

```
SELECT file_name FROM DBA_DATA_FILES;

FILE_NAME
----------------------------------------------------------------
/data/yashan/yasdb_data/db-1-1/dbfiles/system
/data/yashan/yasdb_data/db-1-1/dbfiles/sysaux
/data/yashan/yasdb_data/db-1-1/dbfiles/undo
/data/yashan/yasdb_data/db-1-1/dbfiles/temp
/data/yashan/yasdb_data/db-1-1/dbfiles/swap
/data/yashan/yasdb_data/db-1-1/dbfiles/users

SELECT name FROM V$DATAFILE;

NAME
----------------------------------------------------------------
/data/yashan/yasdb_data/db-1-1/dbfiles/system
/data/yashan/yasdb_data/db-1-1/dbfiles/sysaux
/data/yashan/yasdb_data/db-1-1/dbfiles/undo
/data/yashan/yasdb_data/db-1-1/dbfiles/temp
/data/yashan/yasdb_data/db-1-1/dbfiles/swap
/data/yashan/yasdb_data/db-1-1/dbfiles/users
```

示例（分布式部署）

```
SELECT file_name FROM DBA_DATA_FILES;

FILE_NAME
----------------------------------------------------------------
/data/yashan/yasdb_data/cn-2-1/dbfiles/system
/data/yashan/yasdb_data/cn-2-1/dbfiles/sysaux
/data/yashan/yasdb_data/cn-2-1/dbfiles/temp
/data/yashan/yasdb_data/cn-2-1/dbfiles/swap
/data/yashan/yasdb_data/cn-2-1/dbfiles/users
/data/yashan/yasdb_data/cn-2-1/dbfiles/undo
/data/yashan/yasdb_data/cn-2-1/dbfiles/users_aim-0

SELECT name FROM V$DATAFILE;

NAME
----------------------------------------------------------------
/data/yashan/yasdb_data/cn-2-1/dbfiles/system
/data/yashan/yasdb_data/cn-2-1/dbfiles/sysaux
/data/yashan/yasdb_data/cn-2-1/dbfiles/temp
/data/yashan/yasdb_data/cn-2-1/dbfiles/swap
/data/yashan/yasdb_data/cn-2-1/dbfiles/users
/data/yashan/yasdb_data/cn-2-1/dbfiles/undo
/data/yashan/yasdb_data/cn-2-1/dbfiles/users_aim-0
```

示例（共享集群部署）

```
SELECT file_name FROM DBA_DATA_FILES;

FILE_NAME
----------------------------------------------------------------
+DG0/dbfiles/system
+DG0/dbfiles/sysaux
+DG0/dbfiles/temp
+DG0/dbfiles/swap
+DG0/dbfiles/users
+DG0/dbfiles/undo1
+DG0/dbfiles/undo2
+DG0/dbfiles/undo3

SELECT name FROM V$DATAFILE;

NAME
----------------------------------------------------------------
+DG0/dbfiles/system
+DG0/dbfiles/sysaux
+DG0/dbfiles/temp
+DG0/dbfiles/swap
+DG0/dbfiles/users
+DG0/dbfiles/undo1
+DG0/dbfiles/undo2
+DG0/dbfiles/undo3
```

通过DBA_DATA_FILES和V$DATAFILE视图还可以查看数据文件的其他属性，例如文件号、文件大小、文件对应的表空间和文件状态等信息。

数据文件的维护

添加数据文件

一个表空间可以有多个数据文件组成，当表空间的空间使用完或者数据文件达到最大值时，可以通过给表空间扩容的方式添加数据文件。

新建表空间方式添加数据文件：

示例（单机、分布式部署）

```
CREATE TABLESPACE ts_yashan DATAFILE '/data/dbfiles/data01' SIZE 100M AUTOEXTEND OFF;
```

示例（共享集群部署）

```
CREATE TABLESPACE ts_yashan DATAFILE '+DG0/ts_yashan' SIZE 100M AUTOEXTEND OFF;
```

表空间扩容方式添加数据文件：

示例

```
ALTER TABLESPACE yashan ADD DATAFILE;
```

离线数据文件

某些场景下，例如数据文件损坏、不再使用或者需进行安全隔离，可能需要对指定的数据文件做离线（offline）处理（内置表空间的数据文件无法执行此操作）。

在文件正常使用的情况下，文件offline后其数据仍将被备份，不能对文件进行更名或外部删除等非法操作，否则会导致备份失败。

在文件损坏等异常情况，只能在数据库为MOUNT状态时offline，其他情况建议在OPEN状态下进行offline。

在主备环境中，建议在主库上执行offline，备库会自动同步offline操作。若在备库上执行offline会导致备库出现need repair异常。

> **Warn**:
> 生产系统上请谨慎操作，如果数据文件的offline时间过长，且在此期间生成的归档文件丢失，会导致该文件无法online，后续无法正常使用。

单机部署和共享集群部署中可直接使用ALTER DATABASE语句，分布式部署中推荐使用yasboot运维工具。

示例（单机、共享集群部署）

```
ALTER DATABASE DATAFILE 'datafile' OFFLINE;

-- 数据库未开启归档时，必须指定DROP
ALTER DATABASE DATAFILE 'datafile' OFFLINE DROP;
```

数据文件损坏后的应急处理

数据文件损坏，通常可以通过恢复最新的可用备份集解决，若不存在备份集，则需参考本操作进行应急处理。

常见场景

数据库中表空间下的数据文件发生丢失、损坏等情况导致数据库无法正常启动，且无可用备份集。

> **Note**：
> 本文所述常见场景、解决措施及操作步骤不涵盖内置表空间的数据文件，当内置表空间的数据文件发生丢失或损坏且无可用备份集时，应使用物理恢复等其他手段。

解决措施

启用逃生通道，在数据库处于mount模式下将丢失或损坏的数据文件offline，然后打开数据库。

操作步骤

在主备环境中，需在主库上执行相关操作，通过主备同步使备库上相应的数据文件也被offline。若直接在备库上执行相关操作，会导致其状态被置为need repair，无法正常使用。

1. 通过数据库启动时的报错信息或日志，定位出异常的数据文件：

```
$ open file /home/yashan/YASDB_DATA/dbfiles/SALES0 failed, errno 2, error message "No such file or directory"
Failed to start instance
```

或者在发现数据库为abnormal状态时，查询V$DIAG_INCIDENT排查是否因某个文件损坏导致abnormal。

2. 启动数据库到mount状态：

```
$ yasboot cluster restart -c yashandb -m mount
```

3. 将异常的数据文件offline：

当数据库未开启归档时，必须指定DROP选项，否则无须指定。

```
ALTER DATABASE DATAFILE 'SALES0' OFFLINE DROP;
```

4. 此时可以正常打开数据库，数据文件对应表空间被offline：

```
ALTER DATABASE OPEN;

SELECT tablespace_name, status FROM DBA_TABLESPACES WHERE tablespace_name='SALES';
TABLESPACE_NAME                                                  STATUS
---------------------------------------------------------------- ---------
SALES                                                            OFFLINE
```

5. 对于异常数据文件的读取提示错误：

```
SELECT * FROM area;
YAS-02247 file 6 can not be read at this time
```

redo日志文件管理

YashanDB的redo日志文件用于记录数据库产生的物理日志，可被用于数据库宕机重演和主备复制。

redo日志文件有如下4种状态：

- NEW：表示新创建且未使用的redo日志文件。
- CURRENT：表示当前正在写入的redo日志文件，当current redo日志文件写满，或者手动执行切换操作后，会将下一个可用的redo日志文件状态设置为CURRENT，原current redo日志文件状态变为ACTIVE。
- ACTIVE：该状态下的redo日志文件包含的redo对应的页面存在未写入磁盘的情况，或者该redo日志文件没有被归档，不可复用。ACTIVE状态的redo日志文件，在执行checkpoint并等待归档完成（如果已开启归档模式）后会变成INACTIVE状态。
- INACTIVE：该状态下的redo日志文件包含的redo对应的页面都已经写入磁盘，可复用。

> **Note**：
> - redo日志文件切换时，会将处于INACTIVE或NEW状态的redo日志文件作为下一个文件，如不存在INACTIVE或NEW状态的redo日志文件则会触发checkpoint。
> - 分布式部署中无法添加和删除redo文件。

查看redo日志文件

> **Note**：
> 本文以单机部署为例，不同部署形态的查看方法相同但$YASDB_DATA路径不同，具体请以实际为准。

方式一：

通过V$LOGFILE视图查看redo日志的信息，包括它的组号、路径、成员、块大小、已用大小与状态等信息。

示例

```
SELECT * FROM V$LOGFILE;

ID NAME                                            BLOCK_SIZE  BLOCK_COUNT  USED_BLOCKS SEQUENCE# STATUS
-- ----------------------------------------------  ----------- ------------ ------------ --------- --------
 0 /data/yashan/yasdb_data/db-1-1/dbfiles/redo0         4096        25600         2961         1 INACTIVE
 1 /data/yashan/yasdb_data/db-1-1/dbfiles/redo1         4096        25600          840         2 CURRENT
 2 /data/yashan/yasdb_data/db-1-1/dbfiles/redo2         4096        25600            0         0 INACTIVE
 3 /data/yashan/yasdb_data/db-1-1/dbfiles/redo3         4096        25600            0         0 INACTIVE
```

方式二：

直接进入到$YASDB_DATA/dbfiles目录查看。

```
$ cd /data/yashan/yasdb_data/db-1-1/dbfiles
$ ls -lrt redo*
-rw-r----- 1 yashan yashan 104857600 Jun 16 09:49 redo2
-rw-r----- 1 yashan yashan 104857600 Jun 16 09:49 redo3
-rw-r----- 1 yashan yashan 104857600 Jun 17 11:45 redo0
-rw-r----- 1 yashan yashan 104857600 Jun 18 15:15 redo1
```

维护redo日志文件

添加redo日志

YashanDB中添加redo日志主要是用于提高数据库性能，具体见数据库配置调优章节描述。

可通过ALTER DATABASE语句添加redo日志。

示例（单机、共享集群部署）

```
ALTER DATABASE ADD LOGFILE ('redo6a' size 200M,'redo6b' size 200M);
-- 或者
ALTER DATABASE ADD LOGFILE ('redo7a' size 200M  BLOCKSIZE 512,'redo7b' size 200M BLOCKSIZE 512);
```

Note：

- 添加redo日志时，属性size需放在BLOCKSIZE前才能正常添加，默认块大小为4096Byte，这与磁盘类型有关系，有些磁盘并不支持512Byte的块大小，只能设为4096Byte。
- YashanDB无法直接扩展redo日志的大小，需通过添加新的大redo日志并将原有的小redo日志删除的方式实现。

切换redo日志

可通过ALTER SYSTEM语句手动切换redo日志。

示例（单机、共享集群部署）

```
-- 只切换redo
ALTER SYSTEM SWITCH LOGFILE;

-- 切换redo，并且等待该redo归档完成（归档模式下）
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

删除redo日志

如发生下述情况，可通过ALTER DATABASE语句删除redo日志：

- 磁盘发生故障时，可通过删除故障磁盘上redo日志文件避免写入不可访问的文件。
- redo日志文件存储在不恰当的位置。
- 通过增加更大的redo文件，删除原有的redo，从而实现redo文件扩展。

Caution：

为了保证数据库正常运行以及数据安全，仅允许直接删除INACTIVE或NEW状态的redo日志。

如需删除CURRENT状态的redo日志，操作如下：

1. 切换redo日志，使其变为ACTIVE状态。
2. 执行checkpoint并等待归档完成（如果已开启归档模式），使其变为INACTIVE状态。
3. 执行删除操作。

```
-- 建议先执行checkpoint，以便redo文件变成INACTIVE状态
ALTER SYSTEM CHECKPOINT;

-- 本文以单机部署为例，具体请修改为实际的redo日志文件路径
ALTER DATABASE DROP LOGFILE '/data/yashan/yasdb_data/db-1-1/dbfiles/redo5a';
```

归档日志文件管理

归档日志文件默认存放在$YASDB_DATA/archive目录下。

> **Note**：
> 本文以查询单机部署中的归档日志文档为例，不同部署形态查询方法相同但$YASDB_DATA路径不同，具体请以实际为准。

```
$ cd /data/yashan/yasdb_data/db-1-1/archive
$ ls -lrt
total 15456
-rw-r----- 1 yashan yashan 12128256 Jun 17 11:45 arch_0_1.ARC
-rw-r----- 1 yashan yashan  3698688 Jun 18 17:14 arch_0_2.ARC
```

单机和分布式部署形态下的归档日志默认的文件格式为arch_{resetlogs_id}_{sequence}.ARC。其中resetlogs_id对于没有做过不完全恢复的数据库该值一般为0，sequence为归档日志的序列号。

共享集群部署形态下的归档日志默认的文件格式为arch{instanceId}*{resetlogs_id}*{sequence}.ARC。相比较单机，归档日志文件格式中新增instanceId字段，标记该归档所属的节点ID。其中1号集群节点的归档格式与单机一致，省略了instanceId字段。

上述信息也可以通过V$ARCHIVED_LOG视图查看：

```
SELECT NAME,SEQUENCE#,COMPLETION_TIME FROM V$ARCHIVED_LOG;

NAME                                                    SEQUENCE# COMPLETION_TIME
------------------------------------------------------- ---------- ------------------
/data/yashan/yasdb_data/db-1-1/archive/arch_0_1.ARC              1 2022-06-17
/data/yashan/yasdb_data/db-1-1/archive/arch_0_2.ARC              2 2022-06-18
```

归档日志文件的增加和删除由归档管理操作控制，具体请查看归档管理文档描述。

表空间管理

用户表空间管理准则

建议为业务用户创建特定的用户表空间，而不是使用默认的USERS表空间。

对于用户表空间，建议按以下准则进行创建和维护：

- 将用户数据与数据字典数据分开，以减少I/O争用。
- 将一个应用程序的数据与另一个应用程序的数据分开，以防止在表空间必须脱机时多个应用程序受到影响。
- 将不同表空间的数据文件存储在不同的磁盘驱动器上，以减少I/O争用。
- 单个表空间脱机时，其他表空间可以继续保持联机，无需全部表空间脱机或关闭数据库，从而提供更好的可用性。
- 通过为特定类型的数据库使用保留表空间来优化表空间的使用，例如高更新活动、只读活动或临时段存储。
- 按单个表空间执行逻辑备份。

YashanDB默认表空间

YashanDB内置了如下表空间，其属性可以默认或者在建库时指定：

- 永久表空间，包括SYSTEM表空间、SYSAUX表空间和USERS表空间
- 临时表空间
- UNDO表空间
- SWAP表空间
- USERS_AIM表空间（仅存在于分布式部署）

SYSTEM表空间管理

SYSTEM表空间是创建数据库时包含在数据库中的一个必需的管理表空间，也是建库时创建的第一个表空间，YashanDB使用SYSTEM表空间来管理数据库。

SYSTEM表空间包含以下信息：

- 数据字典
- 包含数据库管理信息的表和视图
- 已编译的存储对象，如触发器、过程和包

SYSTEM表空间像任何其他表空间一样被管理，但是需要更高级别的特权，并且在某些方面受到限制。例如，不能重命名或删除SYSTEM表空间，也不能将SYSTEM表空间脱机。

为SYSTEM表空间添加数据文件

示例（单机、分布式部署）

```
ALTER TABLESPACE SYSTEM ADD DATAFILE '?/dbfiles/system01' SIZE 10G AUTOEXTEND OFF;
```

示例（共享集群部署）

```
ALTER TABLESPACE SYSTEM ADD DATAFILE '+DG0/system01' SIZE 10G AUTOEXTEND OFF;
```

为SYSTEM表空间删除数据文件

示例（单机部署）

```
ALTER TABLESPACE SYSTEM DROP DATAFILE '?/dbfiles/system01';
```

示例（共享集群部署）

```
ALTER TABLESPACE SYSTEM DROP DATAFILE '+DG0/system01';
```

为SYSTEM表空间调整数据文件大小

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE DATAFILE '?/dbfiles/system01' RESIZE 20G;
```

> **Note**：
> 在上述语句中指定数据文件路径时，单机部署中可按绝对或相对路径指定，分布式部署中只允许按相对路径指定，共享集群部署中按绝对路径且只能是YFS路径指定。

查看SYSTEM表空间中空间占用情况

示例

```
SELECT * FROM
(SELECT TABLESPACE_NAME,SEGMENT_NAME,SEGMENT_TYPE,SUM(BYTES)/1024/1024 SIZE
FROM DBA_SEGMENTS
WHERE TABLESPACE_NAME='SYSTEM'
GROUP BY TABLESPACE_NAME,SEGMENT_NAME,SEGMENT_TYPE)
ORDER BY SIZE DESC;

TABLESPACE_NAME      SEGMENT_NAME              SEGMENT_TYPE            SIZE
```

```
------------------- ------------------------- ---------------- -----------
SYSTEM              WRH$_SQLSTAT_PK           INDEX PARTITION            1
SYSTEM              WRH$_SQLTEXT_PK           INDEX PARTITION        .5625
SYSTEM              WRH$_SQLTEXT_INDEX        INDEX PARTITION        .3125
SYSTEM              WRH$_SQLSTAT_INDEX        INDEX PARTITION        .3125
SYSTEM              COL$                      TABLE                   .25
SYSTEM              I_COL2                    INDEX                  .1875
SYSTEM              VIEW$                     TABLE                  .1875
SYSTEM              I_OBJ3                    INDEX                   .125
SYSTEM              WRH$_SYSTEM_EVENT_PK      INDEX                   .125
SYSTEM              I_COL1                    INDEX                   .125
SYSTEM              I_DEPENDENCY2             INDEX                   .125
SYSTEM              I_DEPENDENCY1             INDEX                   .125
SYSTEM              DEPENDENCY$               TABLE                   .125
SYSTEM              OBJ$                      TABLE                   .125
SYSTEM              CDEF$                     TABLE                  .0625
SYSTEM              SEQ$                      TABLE                  .0625
SYSTEM              PARTCOL$                  TABLE                  .0625
SYSTEM              SYN$                      TABLE                  .0625
SYSTEM              ARGUMENT$                 TABLE                  .0625
SYSTEM              USERAUTH$                 TABLE                  .0625
SYSTEM              PACKAGE_ITEMS$            TABLE                  .0625
SYSTEM              I_IND1                    INDEX                  .0625
SYSTEM              I_ICOL1                   INDEX                  .0625
SYSTEM              I_USER2                   INDEX                  .0625
SYSTEM              I_CDEF1                   INDEX                  .0625
SYSTEM              I_CDEF3                   INDEX                  .0625
SYSTEM              I_CDEF5                   INDEX                  .0625
SYSTEM              I_SEQ1                    INDEX                  .0625
SYSTEM              I_PARTCOL                 INDEX                  .0625
SYSTEM              I_TABPART_OBJ             INDEX                  .0625
SYSTEM              I_INDPART_OBJ             INDEX                  .0625
SYSTEM              I_SYN1                    INDEX                  .0625
SYSTEM              I_PROC1                   INDEX                  .0625
SYSTEM              I_ARG2                    INDEX                  .0625
SYSTEM              I_OBJ_ROLE_PRIVS          INDEX                  .0625
```

SYSAUX表空间管理

SYSAUX表空间为SYSTEM表空间的辅助表空间，是YashanDB许多特性（如快照信息）的默认表空间。其总大小由这些特性组件占用的空间决定，组件占用的空间取决于特性正在使用的功能，以及数据库工作负载的状况。

SYSAUX表空间容纳多个组件所需空间，可以减少数据库所需的表空间数量，以及减少SYSTEM表空间上的负载。

YashanDB不允许删除或重命名SYSAUX表空间，当SYSAUX表空间由于异常变得不可用时，核心数据库功能仍保持运行，但使用SYSAUX表空间的相关功能可能会失败或者受限。

如在默认安装时未对SYSAUX表空间做任何配置，随着时间的推移，该表空间会越来越大，建议数据库管理员进行日常监控和定期清理。

为SYSAUX表空间添加数据文件

示例（单机、分布式部署）

```
ALTER TABLESPACE SYSAUX ADD DATAFILE '?/dbfiles/sysaux01' SIZE 10G AUTOEXTEND OFF;
```

示例（共享集群部署）

```
ALTER TABLESPACE SYSAUX ADD DATAFILE '+DG0/sysaux01' SIZE 10G AUTOEXTEND OFF;
```

为SYSAUX表空间删除数据文件

示例（单机部署）

```
ALTER TABLESPACE SYSAUX DROP DATAFILE '?/dbfiles/sysaux01';
```

示例（共享集群部署）

```
ALTER TABLESPACE SYSAUX DROP DATAFILE '+DG0/sysaux01';
```

为SYSAUX表空间调整数据文件大小

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE DATAFILE '?/dbfiles/sysaux01' RESIZE 20G;
```

> **Note**：
> 在上述语句中指定数据文件路径时，单机部署中可按绝对或相对路径指定，分布式部署中只允许按相对路径指定，共享集群部署中按绝对路径且只能是YFS路径指定。

查询SYSAUX表空间中表的占用情况

示例

```
SELECT D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE,SUM(BYTES)/1024/1024 SIZE_M FROM DBA_SEGMENTS D
WHERE D.TABLESPACE_NAME = 'SYSAUX' GROUP BY D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE ORDER BY SIZE_M DESC;

TABLESPACE_NAME    SEGMENT_NAME                   SEGMENT_TYPE           SIZE_M
------------------ ------------------------------ ---------------- -----------
SYSAUX             WRH$_SQLTEXT                   TABLE PARTITION            4
SYSAUX             WRH$_SQLSTAT                   TABLE PARTITION            4
SYSAUX             WRH$_SYSSTAT_PK                INDEX PARTITION         .625
SYSAUX             WRH$_SYSSTAT                   TABLE PARTITION         .625
SYSAUX             WRH$_SYSTEM_EVENT              TABLE PARTITION         .375
SYSAUX             WRH$_MEM_USED_COMP_PK          INDEX PARTITION         .125
```

```
SYSAUX          WRH$_OSSTAT_PK                  INDEX PARTITION         .125
SYSAUX          WRH$_SERVICE_WAIT_CLASS_PK      INDEX PARTITION         .125
SYSAUX          WRM$_SNAPSHOT                   TABLE                   .125
SYSAUX          WRH$_OSSTAT                     TABLE PARTITION         .125
SYSAUX          WRH$_SERVICE_WAIT_CLASS         TABLE PARTITION         .125
SYSAUX          WRH$_MEM_USED_COMP              TABLE PARTITION         .125
SYSAUX          WRM$_DATABASE_INSTANCE          TABLE                   .125
SYSAUX          WRM$_WR_CONTROL                 TABLE                   .125
```

SWAP表空间管理

数据库操作（例如order by，hash join，统计信息收集等）首先会通过数据库虚拟内存（通过VM_BUFFER_SIZE参数控制）缓存计算的中间结果，但如果虚拟内存不足时，需要通过将虚拟内存交换到SWAP表空间来释放内存，必要时再将内存从SWAP表空间换入。

SWAP表空间是非持久化表空间，只用于数据库虚拟内存的换入换出，因此持久化对象（例如表、索引等）不能创建在SWAP表空间。

SWAP表空间空间不足时，可能会导致产生大量中间结果的操作失败，因此需要根据实际业务合理规划SWAP表空间大小或打开自动扩展。

分布式部署中，CREATE DATABASE时内置了SWAP表空间，不允许手动创建新的SWAP表空间。

创建SWAP表空间

示例（单机部署）

```
CREATE SWAP TABLESPACE swap_shared TEMPFILE '?/dbfiles/swap_shared' SIZE 4M;
```

示例（共享集群部署）

```
CREATE SWAP TABLESPACE swap_shared TEMPFILE '+DG0/dbfiles/swap_shared' SIZE 4M;
```

创建本地SWAP表空间

示例（共享集群部署）

```
CREATE LOCAL SWAP TABLESPACE swap_local TEMPFILE '?/dbfiles/swap_local' SIZE 4M;
```

为SWAP表空间添加数据文件

示例

```
ALTER TABLESPACE swap_shared ADD TEMPFILE 'swap01' SIZE 10M AUTOEXTEND ON;
```

> **Note**：
> 共享集群部署下，只有本地SWAP表空间可以添加本地磁盘路径的文件。

示例（共享集群部署）

```
ALTER TABLESPACE SWAP ADD TEMPFILE '+DG0/dbfiles/swap01' SIZE 10M AUTOEXTEND ON;
```

> **Note**：
> 共享集群部署下，SWAP表空间与本地SWAP表空间均支持添加YFS路径的文件。

为SWAP表空间调整数据文件大小

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE TEMPFILE 'swap01' RESIZE 20M;
```

为SWAP表空间删除数据文件

示例

```
ALTER TABLESPACE swap_shared DROP TEMPFILE 'swap01';
```

示例（共享集群部署）

```
ALTER TABLESPACE SWAP DROP TEMPFILE '+DG0/dbfiles/swap01';
```

> **Note**：
> 共享集群部署下，必须全部实例在线才能删除文件。

删除（本地）SWAP表空间

示例（单机、共享集群部署）

```
DROP TABLESPACE swap_shared INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS;
```

> **Note**：
> 共享集群部署下，必须全部实例在线才能删除SWAP表空间。

查询当前正在使用的SWAP表空间名称

示例

```
show parameter DEFAULT_SWAP_TABLESPACE;
```

切换SWAP表空间

示例

```
ALTER SYSTEM SET DEFAULT_SWAP_TABLESPACE = 'SWAP';
```

查询SWAP表空间信息

示例

```
SELECT ID,TABLESPACE_NAME,BLOCK_SIZE,MAX_SIZE/1024/1024 MAX_SIZE,TOTAL_BYTES/1024/1024
TOTAL_SIZE,STATUS,CONTENTS,LOGGING,ALLOCATION_TYPE,
SEGMENT_SPACE_MANAGEMENT,ENCRYPTED FROM DBA_TABLESPACES;

 ID TABLESPACE_NAME    BLOCK_SIZE    MAX_SIZE  TOTAL_SIZE STATUS    CONTENTS    LOGGING       ALLOCATION_TYPE
SEGMENT_SPACE_MANAGEMENT ENCRYPTED
 -- --------------- ------------ ----------- ----------- --------- ----------- ------------- --------------- -----------------
-------- ---------
  0 SYSTEM                 8192      524288          64 ONLINE    PERMANENT   LOGGING       AUTO            BITMAP
N
  1 SYSAUX                 8192      524288          64 ONLINE    PERMANENT   LOGGING       AUTO            BITMAP
N
  2 UNDO                   8192       65536          64 ONLINE    UNDO        LOGGING       UNIFORM         BITMAP
N
  3 TEMP                   8192      524288          94 ONLINE    TEMPORARY   NOLOGGING     AUTO            BITMAP
N
  4 SWAP                   8192      524288          64 ONLINE    SWAP        NOLOGGING     UNIFORM         BITMAP
N
  5 USERS                  8192      524288         124 ONLINE    PERMANENT   LOGGING       AUTO            BITMAP
N
```

查询SWAP表空间的数据文件信息

可以查询`DBA_DATA_FILES`和`DBA_TEMP_FILES`视图获取SWAP表空间数据文件的基本信息。

示例

```
SELECT FILE_NAME,FILE_ID,TABLESPACE_NAME,BYTES,BLOCKS,STATUS,MAXBYTES,MAXBLOCKS,AUTO_EXTEND FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME='SWAP';
```

```
FILE_NAME                                        FILE_ID TABLESPACE_NAME     BYTES       BLOCKS STATUS        MAXBYTES
MAXBLOCKS AUTO_EXTEND
------------------------------------------    -------- --------------- --------- ------------ --------- ------------ -------
---- -----------
/usr/local/yashandb/yasdb_data/dbfiles/swap           4 SWAP             67108864         8192 ONLINE    549755813888
67108864 ON
/usr/local/yashandb/yasdb_data/dbfiles/swap01         8 SWAP             20971520         2560 ONLINE    549755813888
67108864 ON

SELECT FILE_ID,FILE_NAME,STATUS,BYTES,BLOCKS,RELATIVE_FNO,AUTOEXTENSIBLE,TABLESPACE_NAME,MAXBYTES,MAXBLOCKS
FROM DBA_TEMP_FILES WHERE TABLESPACE_NAME='SWAP';

FILE_ID FILE_NAME                                      STATUS     BYTES  BLOCKS RELATIVE_FNO AUTOEXTENSIBLE TABLESPACE_NAME
MAXBYTES  MAXBLOCKS
------- ---------------------------------------------- ------- -------- ------- ------------ -------------- --------------- --
---------- ----------
      4 /usr/local/yashandb/yasdb_data/dbfiles/swap   ONLINE  67108864    8192            0 ON             SWAP
549755813888   67108864
      8 /usr/local/yashandb/yasdb_data/dbfiles/swap01 ONLINE  20971520    2560            1 ON             SWAP
549755813888   67108864
```

此外，还可查询 `V$TEMP_EXTENT_POOL` 动态视图获取当前SWAP表空间的EXTENT分配情况。

示例

```
SELECT TABLESPACE_NAME,FILE_ID,EXTENTS_CACHED,EXTENTS_USED,BLOCKS_CACHED,BLOCKS_USED,BYTES_CACHED,BYTES_USED,INTER_FNO
FROM V$TEMP_EXTENT_POOL
WHERE TABLESPACE_NAME='SWAP';

TABLESPACE_NAME                                                  FILE_ID EXTENTS_CACHED EXTENTS_USED BLOCKS_CACHED
BLOCKS_USED BYTES_CACHED  BYTES_USED    INTER_FNO
---------------------------------------------------------------- ------- ------------ ------------ ------------- -----
------ ------------ ----------- ------------
SWAP                                                                   5         4747         4747         37976
37976    311099392   311099392            0
SWAP                                                                   6         4592         4592         36736
36736    300941312   300941312            1
```

UNDO表空间管理

UNDO表空间用于YashanDB创建和管理回滚（撤销数据库更改）信息，这种信息包括交易行为的记录，且主要是在交易提交之前，统称为undo。

undo记录用于：

- 执行ROLLBACK语句回滚事务
- 恢复数据库
- 提供读取一致性
- 使用闪回查询分析较早时间点的数据
- 使用闪回功能从逻辑损坏中恢复

设置最短撤销保留期

YashanDB提供一种自动化的机制，称为自动撤销管理，用于管理undo信息和空间。

当启用自动撤销管理时，始终存在一个当前撤销保持期，这是YashanDB在覆盖旧的撤销信息之前尝试保留该信息的最短时间。

通过UNDO_RETENTION参数指定最小撤销保持期（以秒为单位）：

```
ALTER SYSTEM SET UNDO_RETENTION = 2400;
```

为UNDO表空间添加数据文件

示例（单机、分布式部署）

```
ALTER TABLESPACE UNDO ADD DATAFILE '?/dbfiles/undo02' SIZE 10M AUTOEXTEND ON;
```

示例（共享集群部署）

```
ALTER TABLESPACE undo0 ADD DATAFILE '+DG0/undo02' SIZE 10M AUTOEXTEND ON;
```

为UNDO表空间调整数据文件大小

UNDO表空间的数据文件只能扩大，不能缩小。

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE DATAFILE '?/dbfiles/undo02' RESIZE 20M;
```

删除UNDO表空间

分布式部署中无法使用此功能。

示例（单机、共享集群部署）

```
DROP TABLESPACE undo0 INCLUDING CONTENTS AND DATAFILES;
```

> **Note**：
> - 只有当前未被任何实例使用时，才能删除UNDO表空间。
> - 在上述语句中指定数据文件路径时，单机部署中可按绝对或相对路径指定，分布式部署中只允许按相对路径指定，共享集群部署中按绝对路径且只能是YFS路径指定。

查看UNDO相关统计信息

示例

```
SELECT ID,BLK_REUSE,BALANCE_TIME,BALANCE,BALANCE_BLK,RECYCLE_TIME,RECYCLE_LIST,RECYCLE_LIST_BLK FROM V$UNDOSTAT;

  ID    BLK_REUSE BALANCE_TIME          BALANCE  BALANCE_BLK RECYCLE_TIME      RECYCLE_LIST RECYCLE_LIST_BLK
---- ------------ ---------------- ------------ ------------ ---------------- ------------ ----------------
   0            9 2022-07-04                425          112 1970-01-01                 0                0
   1            6 2022-07-04                425          216 1970-01-01                 0                0
   2            8 2022-07-04                425          326 1970-01-01                 0                0
   3            9 2022-07-04                425          437 1970-01-01                 0                0
   4            9 2022-07-04                425          564 1970-01-01                 0                0
```

查看UNDO SEGMENT信息

示例

```
SELECT ID,USED_TIME,FIRST_UBAFIL,FIRST_UBABLK,LAST_UBAFIL,LAST_UBABLK,UFB_COUNT,FIRST_UFBFIL,FIRST_UFBBLK FROM
V$UNDO_SEGMENTS;

 ID USED_TIME      FIRST_UBAFIL FIRST_UBABLK  LAST_UBAFIL  LAST_UBABLK    UFB_COUNT FIRST_UFBFIL FIRST_UFBBLK
--- -------------- ------------ ------------ ------------ ------------ ------------ ------------ ------------
  0 2022-07-04                0         7012            0         7012           15            0         7013
  1 2022-07-04                0         6472            0         6472            7            0         6473
  2 2022-07-04                0         6510            0         6510            1            0         6511
  3 2022-07-04                0         6883            0         6883            0            0         6885
  4 2022-07-04                0         6527            0         6527            0            0         6528
```

TEMP表空间管理

TEMP临时表空间主要用于临时表的段分配以及与临时表相关undo空间分配，并存储临时表数据和临时表相关的undo。

临时表的及其相关的undo记录被存储在数据库的临时表空间中，并且不生成redo信息，从而减少了UNDO表空间中存储的undo数量，也减少了数据库redo日志的大小；提升数据库性能。

分布式部署中，CREATE DATABASE时内置了TEMP临时表空间，不允许手动创建新的TEMP临时表空间。

创建TEMP表空间

示例（单机、共享集群部署）

```
CREATE TEMPORARY TABLESPACE temp_shared TEMPFILE '?/dbfiles/temp_share.dbf' SIZE 4M;
```

创建本地TEMP表空间

示例（共享集群部署）

```
CREATE LOCAL TEMPORARY TABLESPACE FOR ALL temp_local TEMPFILE '?/dbfiles/temp_local.dbf' SIZE 4M;
```

为TEMP表空间添加数据文件

示例

```
ALTER TABLESPACE temp ADD TEMPFILE '?/dbfiles/temp01.dbf' SIZE 10M AUTOEXTEND ON;
```

注意：共享集群部署下，只有本地TEMP表空间可以添加本地磁盘路径的文件。

示例（共享集群部署）

```
ALTER TABLESPACE temp ADD TEMPFILE '+DG0/dbfiles/temp01.dbf' SIZE 10M AUTOEXTEND ON;
```

注意：共享集群部署下，TEMP表空间与本地TEMP表空间均支持添加YFS路径的文件。

为TEMP表空间删除数据文件

示例（单机部署）

```
ALTER TABLESPACE temp DROP TEMPFILE '?/dbfiles/temp01.dbf';
```

示例（共享集群部署）

```
ALTER TABLESPACE temp DROP TEMPFILE '+DG0/temp01.dbf';
```

删除TEMP表空间

示例（单机、共享集群部署）

```
DROP TABLESPACE temp_local INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS;
```

为TEMP表空间调整数据文件大小

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE TEMPFILE '?/dbfiles/temp02.dbf' RESIZE 20M;
```

> **Note**：
> 在上述语句中指定数据文件路径时，单机部署中可按绝对或相对路径指定，分布式部署中只允许按相对路径指定，共享集群部署中按绝对路径且只能是YFS路径指定。

查看tempfile信息

可以查询 `DBA_TEMP_FILES` 视图获取临时表空间数据文件基本信息。

> **Note**：
> 本文以单机部署为例，不同部署形态的查看方法相同但$YASDB_DATA/dbfiles/temp路径不同，具体请以实际返回结果为准。

示例

```
SELECT FILE_ID,FILE_NAME,STATUS,BYTES/1024/1024,AUTOEXTENSIBLE,TABLESPACE_NAME,MAXBYTES/1024/1024,INCREMENT_BY FROM
DBA_TEMP_FILES;

     FILE_ID FILE_NAME                                          STATUS    BYTES/1024/1024 AUTOEXTENSIBLE TABLESPACE_NAME
MAXBYTES/1024/1024 INCREMENT_BY
------------ ------------------------------------------------- --------- --------------- -------------- ------------------ --
--------------- ------------
           3 /data/yashan/yasdb_data/db-1-1/dbfiles/temp     ONLINE                64 ON             TEMP
524288          8192
           4 /data/yashan/yasdb_data/db-1-1/dbfiles/swap     ONLINE                64 ON             SWAP
524288          8192
```

此外，还可查询 `V$TEMP_EXTENT_POOL` 动态视图获取当前TEMP表空间的EXTENT分配情况。

示例

```
SELECT TABLESPACE_NAME,FILE_ID,EXTENTS_CACHED,EXTENTS_USED,BLOCKS_CACHED,BLOCKS_USED,BYTES_CACHED,BYTES_USED,INTER_FNO
FROM V$TEMP_EXTENT_POOL
WHERE TABLESPACE_NAME='TEMP';

TABLESPACE_NAME                                                  FILE_ID EXTENTS_CACHED EXTENTS_USED BLOCKS_CACHED
BLOCKS_USED BYTES_CACHED  BYTES_USED    INTER_FNO
---------------------------------------------------------------- ------------ -------------- ------------ -------------- -----
------ ------------ ----------- ------------
TEMP                                                                        2             64           64            512
512      4194304     4194304            0
TEMP                                                                        3              0            0              0
0            0           0            1
TEMP                                                                        4              0            0              0
0            0           0            2
```

USERS表空间管理

USERS表空间是默认的用户表空间，用于存储永久用户对象和私有信息。

在创建一个用户而没有指定其使用的表空间时，该用户下除分布表外的所有信息都会放入到USERS表空间中，分布表相关信息存储在表空间集中。

为USERS表空间增加数据文件

示例（单机、分布式部署）

```
ALTER TABLESPACE users ADD DATAFILE '?/dbfiles/users02' SIZE 100M;
```

示例（共享集群部署）

```
ALTER TABLESPACE users ADD DATAFILE '+DG0/users02' SIZE 100M;
```

为USERS表空间调整数据文件大小

分布式部署/共享集群部署中无法使用此功能。

示例（单机部署）

```
ALTER DATABASE DATAFILE '?/dbfiles/users02' RESIZE 50M;
```

为USERS表空间删除数据文件

示例（单机部署）

```
ALTER TABLESPACE users DROP DATAFILE '?/dbfiles/users02';
```

示例（共享集群部署）

```
ALTER TABLESPACE users DROP DATAFILE '+DG0/users02';
```

> **Note**：
> 在上述语句中指定数据文件路径时，单机部署中可按绝对或相对路径指定，分布式部署中只允许按相对路径指定，共享集群部署中按绝对路径且只能是YFS路径指定。

查看使用USERS表空间的对象信息

示例

```
SELECT D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE, SUM(BYTES)/1024/1024 SIZE_M FROM DBA_SEGMENTS D
WHERE D.TABLESPACE_NAME = 'USERS'
GROUP BY D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE ORDER BY SIZE_M DESC;

TABLESPACE_NAME            SEGMENT_NAME         SEGMENT_TYPE           SIZE_M
-------------------------- -------------------- ----------------- -----------
USERS                      SYS_C_21             INDEX                     .125
USERS                      AREA                 TABLE                     .125
```

USERS_AIM表空间管理

USERS_AIM表空间是分布式内置的memory mapped表空间，memory mapped类表空间包含文件的所有页面在系统运行时都映射在内存中。

USERS_AIM表空间只用于存储复制表，分布表则存储在表空间集中。

初始的USERS_AIM表空间大小由`文件数量*文件大小`计算得到，其中文件数量和文件大小分别来自于建库参数中的MMS_DATA_FILE_NUM和MMS_DATA_FILE_SIZE参数，用户可以在建库时根据自身资源需求自行配置这些参数的值。

为USERS_AIM表空间增加数据文件

示例（分布式部署）

```
ALTER TABLESPACE users_aim ADD DATAFILE '?/dbfiles/users01' SIZE 16M;
```

为USERS_AIM表空间删除数据文件

示例（分布式部署）

```
ALTER TABLESPACE users_aim DROP DATAFILE '?/dbfiles/users01';
```

> **Note**：
> 在上述语句中指定数据文件路径时，只允许按相对路径指定。

查看使用USERS_AIM表空间的对象信息

示例（分布式部署）

```
SELECT D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE,SUM(BYTES)/1024/1024  SIZE_M FROM DBA_SEGMENTS D
WHERE D.TABLESPACE_NAME = 'USERS_AIM'
GROUP BY D.TABLESPACE_NAME, D.SEGMENT_NAME, D.SEGMENT_TYPE ORDER BY SIZE_M DESC;

TABLESPACE_NAME          SEGMENT_NAME          SEGMENT_TYPE            SIZE_M
------------------------ --------------------- ----------------- ------------
USERS_AIM                AREA0                 TABLE                   2.4375
```

用户表空间管理

关于表空间管理的详细语法描述请参考开发手册CREATE TABLESPACE、ALTER TABLESPACE、DROP TABLESPACE。

创建表空间

数据库管理员在接收到一个创建表空间的申请时，需要从以下方面进行考量：

- 表空间所服务的业务属性：
  - HEAP表和TAC表采取段页式结构，挂载普通数据文件

```
-- 不指定DATAFILE将默认创建一个数据文件
CREATE TABLESPACE tablespace_name;
```

  - LSC表的稳态数据采取对象结构，需挂载数据桶

```
-- LSC的表空间必须指定DATABUCKET
CREATE TABLESPACE tablespace_name DATABUCKET 'lscfile1','lscfile2';
```

  - 对敏感数据或核心数据可定义为加密表空间

```
-- 加密属性在创建表空间时确定，后续不可更改
CREATE TABLESPACE tablespace_name DATABUCKET 'lscfile1','lscfile2' ENCRYPTION ENCRYPT;
```

  - 对极高性能要求的表空间可定义为MMS表空间，同时配置MMS_DATA_LOADERS参数确定预加载策略

```
CREATE TABLESPACE tablespace_name MEMORY MAPPED;
```

- 表空间的存储属性：
  - 确定数据文件初始值、扩展值和最大值，默认由系统自动决定

```
CREATE TABLESPACE yashan1 DATAFILE 'yashan1' SIZE 4M AUTOEXTEND ON NEXT 4M MAXSIZE 1G;
```

  - 确定extent大小，默认由系统自动决定

```
CREATE TABLESPACE yashan1 DATAFILE 'yashan1' SIZE 4M AUTOEXTEND ON NEXT 4M MAXSIZE 1G EXTENT UNIFORM SIZE 64K;
```

  - 对于MMS表空间，确定是否启用大页内存，启用大页内存的前提是操作系统配置大页内存

```
-- 重启生效
ALTER SYSTEM SET MMS_USE_LARGE_PAGES=true SCOPE=BOTH;
```

表空间运维

offline与online

当某个表空间不再使用，或者不想被访问到时，可将其进行脱机：

```
-- 默认要求脱机的表空间中的所有数据文件均为脱机
ALTER TABLESPACE yashan OFFLINE;

-- TEMPORARY选项允许脱机的表空间中有数据文件已脱机
ALTER TABLESPACE yashan OFFLINE TEMPORARY;
```

```
-- IMMEDIATE立即脱机，存在数据风险，后续不能被online
ALTER TABLESPACE yashan OFFLINE IMMEDIATE;
```

通过online操作可以恢复被脱机的表空间：

```
ALTER TABLESPACE yashan ONLINE;
```

增删数据文件

表空间可挂载的数据文件数量和数据桶数量存在规格约束（YashanDB为64），管理员需关注该限制，避免达到上限（达到上限后将无法增加数据文件，且无法通过删除数据文件的方式释放数量）。当已达到上限仍需增加数据文件或数据桶时，通过创建新的表空间来解决。

增加数据桶时还需考虑BUCKET_RESERVED_SPACE参数，该参数用于控制数据桶占用所在磁盘的空间上限。

```
-- 默认由系统决定新增的数据文件的属性，临时文件需使用TEMPFILE关键字
ALTER TABLESPACE yashan ADD DATAFILE;

ALTER TABLESPACE yashan DROP DATAFILE 'yashan1';

ALTER TABLESPACE lsc_tb ADD DATABUCKET 'lscfile3','lscfile4';

ALTER TABLESPACE lsc_tb DROP DATABUCKET 'lscfile3';
```

空闲空间回收

对单机部署和分布式部署中的表空间进行收缩，可以回收其空闲空间，供其他表空间使用。

示例（单机、分布式部署）

```
-- 不指定KEEP时，系统将最大化收缩表空间
ALTER TABLESPACE SYSTEM SHRINK SPACE KEEP 100M;
ALTER TABLESPACE SYSTEM SHRINK SPACE;
```

分布式表空间故障恢复

分布式部署中执行表空间DDL时出现节点故障，YashanDB提供自动恢复功能。

```
-- 为确认系统是否正常，可通过在CN执行命令查询DV$TABLESPACE，DV$DATAFILE，DV$DATABUCKET视图获取节点tablespace和datafile，bucketfile文件状态
SELECT * FROM DV$TABLESPACE WHERE name = UPPER('tablespace名称') ORDER BY group_id;
SELECT * FROM DV$DATAFILE WHERE name LIKE '%datafile名称' ORDER BY group_id;
SELECT * FROM DV$DATABUCKET WHERE url LIKE '%bucketfile名称' ORDER BY group_id;
```

Datafile，Bucketfile文件目录：

1. Datafile 文件默认目录：`${YASDB_DATA}/节点名/dbfiles/`
2. Bucketfile 文件默认目录：`${YASDB_DATA}/节点名/local_fs/`
3. 如果指定了目录路径，文件目录：`${YASDB_DATA}/节点名/指定目录/`

CREATE TABLESPACE

创建表空间时出现节点故障，具体情况和解决方案如下所示：

- 仅在MN节点（group_id为1）有Tablespace和Datafile或Databucket记录，DV$PUB_STAT视图没有异常Tablespace相关DDL记录，发现仅有MN节点出现故障：

  1. 需重启MN节点并手动在CN节点上执行如下语句：

     ```
     DROP TABLESPACE IF EXISTS tablespace_name including contents AND datafiles;
     ```

  2. 查询`DV$TABLESPACE`，`DV$DATAFILE`和`DV$DATABUCKET`视图，确认没有对应的Tablespace。

3. 查询MN节点下的Datafile和Databucket文件夹，确认没有对应的Tablespace文件残留，若有则需手动清除该文件夹。
4. 重启后，再次执行 `CREATE TABLESPACE` 命令即可。

- MN节点，部分CN/DN节点有Tablespace和Datafile或Databucket记录，MN节点中DV$PUB_STAT视图有异常Tablespace相关DDL推送记录，发现有部分CN/DN节点出现故障：
  1. 重启故障节点，等待片刻后于MN节点中查询DV$PUB_STAT视图，如没有发现异常Tablespace相关的DDL记录，则表示恢复成功；如仍查询到异常DDL记录，可通过ERR_MSG字段查看失败原因：若显示DN资源不足，则需要重新合理分配资源。
- 所有节点都没有Tablespace和Datafile或Databucket记录，MN节点中DV$PUB_STAT视图没有异常Tablespace相关DDL记录，发现仅有MN节点出现故障：
  1. 重启MN节点，再次执行 `CREATE TABLESPACE` 命令即可。

DROP TABLESPACE

删除表空间时出现节点故障，具体情况和解决方案如下所示：

- MN节点有Tablespace和Datafile或Databucket记录，仅有MN节点出现故障：
  1. 重启MN节点，并手动在CN节点上执行如下语句：

```
DROP TABLESPACE IF EXISTS tablespace_name including contents AND datafiles;
```

- MN节点不存在Tablespace，CN/DN节点有Tablespace和Datafile或Databucket记录，部分CN/DN节点出现故障：
  1. 重启故障节点即可恢复。

ALTER TABLESPACE ADD(DATAFILE|TEMPFILE)

修改表空间数据文件时出现节点故障，具体情况和解决方案如下所示：

- 仅在MN节点（group_id为1）有Tablespace和Datafile或Databucket记录，DV$PUB_STAT视图没有异常Tablespace相关DDL记录，发现仅有MN节点出现故障：
  1. 需重启MN节点并手动在CN节点上执行如下语句：

```
ALTER TABLESPACE yashan DROP DATAFILE datafile_name;
```

  2. 查询 `DV$TABLESPACE` ， `DV$DATAFILE` 和 `DV$DATABUCKET` 视图，确认没有对应的Tablespace。
  3. 查询MN节点下的Datafile和Databucket文件夹，确认没有对应的Tablespace文件残留，若有则需手动清除该文件夹。
  4. 重启后，再次执行 `ALTER TABLESPACE ADD(DATAFILE|TEMPFILE)` 命令即可。
- MN节点，部分CN/DN节点有Tablespace和Datafile或Databucket记录，MN节点中DV$PUB_STAT视图有异常Tablespace相关DDL推送记录，发现有部分CN/DN节点出现故障：
  1. 重启故障节点，等待片刻后于MN节点中查询DV$PUB_STAT视图，如没有发现异常Tablespace相关的DDL记录，则表示恢复成功；如仍查询到异常DDL记录，可通过ERR_MSG字段查看失败原因：若显示DN资源不足，则需要重新合理分配资源。
- 所有节点都没有Tablespace和Datafile或Databucket记录，MN节点中DV$PUB_STAT视图没有异常Tablespace相关DDL记录，发现仅有MN节点出现故障：
  1. 重启MN节点，再次执行 `ALTER TABLESPACE ADD(DATAFILE|TEMPFILE)` 命令即可。

表空间集管理

用户表空间集管理准则

表空间集（TABLESPACE SET）是分布式数据库中一个逻辑存储单位，在分布式部署中用于存储分布表、建立在分布表上的索引、建立在分布表上的AC等。 用户创建分布表时可为其指定TABLESPACE SET，如果不指定则使用默认的TABLESPACE SET。

YashanDB默认表空间集

YashanDB内置了如下表空间集，其属性可以默认或者在建库时指定：

- USERS表空间集
- USERS_AIM表空间集

USERS表空间集管理

USERS表空间集是默认的用户表空间集。

基于分布表的索引/AC是不能指定表空间和表空间集，且索引/AC会默认存储到和表相同的表空间集。

修改USERS表空间集的最大拓展空间

示例（分布式部署）

```
ALTER TABLESPACE SET users MAXSIZE 22T;
```

修改USERS表空间集内部的数据文件每次自动扩展的大小

示例（分布式部署）

```
ALTER TABLESPACE SET users NEXT 100M;
```

修改USERS表空间集内部的数据文件的大小

示例（分布式部署）

```
ALTER TABLESPACE SET users resize 300M;
```

查看USERS表空间集中的文件信息

系统内置的表空间集将在DV$TABLESPACE中生成以'TSS_'开头的name记录。

示例（分布式部署）

```
SELECT group_id,group_node_id,id
FROM DV$TABLESPACE ts
WHERE ts.name LIKE 'TSS_%' AND ts.memory_mapped=false;

    GROUP_ID GROUP_NODE_ID           ID
------------ ------------- ------------
           3             1            7
           4             1            7
           5             1            7

SELECT group_id||'_'||group_node_id dn_node,
SPLIT(name,'/',-1) filename,
TS#,BYTES,RELATIVE_FNO,AUTO_EXTEND,NEXT_SIZE,MAX_SIZE
FROM DV$DATAFILE
WHERE TS#=7 AND group_id=3 AND group_node_id=1
ORDER BY 3,1,2;

DN_NODE   FILENAME                 TS#      BYTES RELATIVE_FNO AUTO_EXTEND   NEXT_SIZE       MAX_SIZE
--------- ------------------------ --- ---------- ------------ ----------- ----------- --------------
3-1       TSS_1800_CHUNK_0_FILE_0    7   68157440            0 ON            67108864   549755813888
3-1       TSS_1800_CHUNK_0_FILE_1    7    1048576            1 ON            67108864   549755813888
```

USERS_AIM表空间集管理

USERS_AIM表空间集是内置的memory mapped的表空间集，memory mapped类的表空间集的文件的所有页面都映射在内存中。

初始的USERS_AIM表空间集大小为建库参数中的mms_tablespace_set_size，用户可以在安装时根据资源需求配置该参数的值。

查看USERS_AIM表空间集中的文件信息

示例（分布式部署）

```
SELECT group_id||'_'||group_node_id dn_node,
SPLIT(name,'/',-1) filename,
TS#,BYTES,RELATIVE_FNO,AUTO_EXTEND,NEXT_SIZE,MAX_SIZE
FROM DV$DATAFILE
WHERE TS#=8
ORDER BY 3,1,2;

DN_NODE   FILENAME                  TS#      BYTES RELATIVE_FNO AUTO_EXTEND  NEXT_SIZE         MAX_SIZE
--------- ------------------------- ---- ---------- ------------ ----------- ---------- -----------------
3-1       TSS_1801_CHUNK_0_FILE_0      8  104857600            0 OFF                 0                 0
```

表管理

数据库开发人员在自己的权限范围内会建立各类数据库对象，如表、索引、视图、序列、同义词、触发器、存储过程等。通过将不同用户归属不同表空间，可以按业务类型将数据进行物理隔离，但对于同一物理空间内的对象，数据库管理员也需要建立一套资源规划、日常监控、告警处理等机制，保证系统的稳定和有效运行。这其中，表由于是业务数据的载体，也是其他所有对象的运行基础，对其的管理最为重要。

YashanDB提供一系列管理视图，供DBA查看和监控数据库中所有对象的相关信息：

- DBA_OBJECTS：所有的对象信息。
- DBA_TABLES/DBA_PART_TABLES/DBA_TAB_PARTITIONS/DBA_PART_KEY_COLUMNS/DBA_TAB_SUBPARTITIONS/DBA_SUBPART_KEY_COLUMNS：表及表分区，二级分区的各项信息。
- DBA_TRIGGERS：触发器相关信息。
- DBA_PROCEDURES：存储过程和自定义函数的各项信息。
- DBA_LOBS/DBA_LOB_PARTITIONS/DBA_LOB_SUBPARTITIONS：大对象相关信息。
- DBA_INDEXES/DBA_IND_COLUMNS/DBA_PART_INDEXES/DBA_IND_PARTITIONS/DBA_IND_SUBPARTITIONS：索引及索引分区，索引二级分区的各项信息。
- DBA_VIEWS：视图相关信息。
- DBA_SYNONYMS：同义词相关信息。
- DBA_SEQUENCES：序列器相关信息。

表的存储空间

存储空间规划

在搭建一套数据库时，管理员需要进行存储空间的设计，并将其分配给业务用户，数据库开发人员使用这套默认的存储空间创建表对象，可以让开发人员无需过多考虑底层存储约束，简化应用逻辑。

在使用CREATE TABLE创建表对象时，不使用TABLESPACE指定表空间，系统将默认为用户的所属表空间，表数据。

进入规划的存储空间内，其中临时表数据将进入系统临时表空间内。

除此之外，建议管理员还规划如下存储空间供开发人员指定：

- 为每一个用户表空间建立一个相应的索引表空间。
- 在需要时，为分区表的分区建立一个独立表空间。
- 为大对象数据建立一个或多个独立表空间。

需注意的是，对于LSC表，管理员需要预先为其规划和创建databucket表空间，用于存储SCOL稳态数据。

> **Note**： TAC表和LSC表可变数据都是按列组织，每列采用段页式结构存储。 即便插入一条数据，每列都会预分配特定数量（最大384个）的页面。 尤其在表的分区和列数很多的场景下，插入容易出现表空间不足的错误。处理方法可参考错误码YAS-02007。

空间优化

数据库管理员可以通过SQL语句管理表的存储空间，详细语法描述请参考开发手册CREATE TABLE、ALTER TABLE。

HEAP表

基于行存特征，对表的频繁更新、删除等操作可能导致产生碎片，表占用空间变得膨胀，此时采用shrink table功能可以重整HEAP表的存储结构，将表的数据空间进行收缩。

示例（单机HEAP表）

```
ALTER TABLE orders_info ENABLE ROW MOVEMENT;
--默认shrink后释放空出来的extent，供其他表使用
ALTER TABLE orders_info SHRINK SPACE;
--COMPACT表示不释放
ALTER TABLE orders_info SHRINK SPACE COMPACT;

--对某个分区空间进行shrink
ALTER TABLE orders_info MODIFY PARTITION p_orders_info_1 SHRINK SPACE COMPACT;
```

> **Note**：
> - 在执行shrink table之前，对表数据进行备份是更加保险的一种做法。
> - shrink table可以在线操作，不影响DML执行。

LSC表稳态数据

LSC表的稳态数据以slice文件存储，对这部分数据的空间优化包括：1）将离散的小的slice文件进行合并；2）将无序的slice文件进行排序。

YashanDB默认启动后台数据转换任务，自动对slice文件执行上述优化，优化期间会产生额外的资源消耗，用户可以根据实际情况对某张表关闭该操作：

示例（单机LSC表）

```
--关闭后，系统不再自动对orders_info表的稳态数据进行空间优化
ALTER TABLE orders_info DISABLE COMPACT;
```

并在需要时手工对表启动一次空间优化，提高查询性能：

示例（单机LSC表）

```
ALTER TABLE orders_info ENABLE COMPACT;
ALTER TABLE orders_info ALTER SLICE ALL COMPACT;
```

Note：
1.在资源允许的情况下，建议不要关闭COMPACT开关，由系统自动判断空间优化的时机并执行优化。
2.如必须采取手工执行COMPACT的方式，建议安排在闲时执行，避免影响其他的操作。

表的闪回

闪回恢复

若较短时间内发现由于操作不当等原因误删了表数据，可以使用闪回功能及时将数据恢复至指定时间点（无需还原备份），更多详情可查阅FLASHBACK。

闪回恢复需满足以下条件：

- 执行闪回恢复操作需使用具备DBA权限或FLASHBACK相关权限的用户。
- 在当前时间点至目标闪回时间点期间，表的结构未发生变化。
- 可闪回恢复的时间点由撤销保持期（UNDO_RETENTION）决定，建议将该参数设置为86400秒（24小时）或更长。
- 需闪回恢复的表对象为HEAP表。
- 通过时间戳或SCN闪回恢复前需手动开启表的ROW MOVEMENT，且闪回恢复后的数据可能会改变rowid，请确保应用程序不依赖于rowid才能执行闪回恢复。
- 对于被drop的表，需确保历史数据仍然存放在回收站中才能将该表（包括表结构和数据）闪回恢复。
  - drop前已开启回收站。
  - drop表时，未指定purge语句。
  - 当前时间点至目标闪回时间点期间，未对回收站执行purge。
  - drop表时，表空间未满。
- 对于被truncate的表，需确保历史数据仍然存放在回收站中才能将该表数据（包括truncate表或truncate分区）闪回恢复。若当前时间点与truncate时间点期间表格已插入新数据，执行闪回恢复后该类新数据将会被放入回收站中。
  - truncate前已开启回收站。
  - 当前时间点至目标闪回时间点期间，未对回收站执行purge。
  - truncate表时，表空间未满。

delete操作闪回

示例（HEAP表）

```
-- 开启finance_info的ROW MOVEMENT开关
ALTER TABLE finance_info ENABLE ROW MOVEMENT;

-- finance_info表中存在的一条记录
SELECT * FROM finance_info WHERE year='2021' AND month='02';
YEAR  MONTH BRANCH REVENUE_TOTAL  COST_TOTAL   FEE_TOTAL
----- ----- ------ ------------- ----------- -----------
2021  02    0101           37778       33000        6000

-- 获取当前时间戳
SELECT SYSTIMESTAMP res FROM dual;
RES
---------------------------------------------
2023-12-17 14:10:28.736908

-- 删除此条记录并提交
DELETE FROM finance_info WHERE year='2021' AND month='02';
COMMIT;
SELECT * FROM finance_info WHERE year='2021' AND month='02';
YEAR  MONTH BRANCH REVENUE_TOTAL  COST_TOTAL   FEE_TOTAL
----- ----- ------ ------------- ----------- -----------


-- 利用FLASHBACK闪回历史数据（通过时间戳闪回）
FLASHBACK TABLE finance_info TO TIMESTAMP  TIMESTAMP('2023-12-17 14:10:28.736908');
SELECT * FROM finance_info WHERE year='2021' AND month='02';
YEAR  MONTH BRANCH REVENUE_TOTAL  COST_TOTAL   FEE_TOTAL
----- ----- ------ ------------- ----------- -----------
2021  02    0101           37778       33000        6000

-- 获取最后一次修改时SCN（通过ROWSCN获取）
SELECT rowscn FROM finance_info WHERE year='2021' AND month='02';
              ROWSCN
--------------------
```

```
   408883147271815168

-- 再次删除记录
DELETE FROM finance_info WHERE year='2021' AND month='02';
COMMIT;
SELECT * FROM finance_info WHERE year='2021' AND month='02';
YEAR  MONTH BRANCH REVENUE_TOTAL  COST_TOTAL   FEE_TOTAL
----- ----- ------ ------------- ----------- -----------


-- 利用FLASHBACK闪回历史数据（通过SCN闪回）
FLASHBACK TABLE finance_info TO SCN 408883147271815168;
SELECT * FROM finance_info WHERE year='2021' AND month='02';
YEAR  MONTH BRANCH REVENUE_TOTAL  COST_TOTAL   FEE_TOTAL
----- ----- ------ ------------- ----------- -----------
2021  02    0101          37778       33000        6000
```

drop操作闪回

示例（HEAP表）

```
ALTER SYSTEM SET RECYCLEBIN_ENABLED=ON;

DROP TABLE finance_info;
DROP TABLE orders_info;
DROP TABLE sales_info;
DROP TABLE employees;

-- 1.查询回收站是否存在finance_info表
SELECT original_name,object_name FROM DBA_RECYCLEBIN
WHERE original_name IN ('FINANCE_INFO','ORDERS_INFO','SALES_INFO','EMPLOYEES');
ORIGINAL_NAME                  OBJECT_NAME
------------------------------ ------------------------
SALES_INFO                     BIN$2393
ORDERS_INFO                    BIN$2389
EMPLOYEES                      BIN$2385
FINANCE_INFO                   BIN$2408

-- 2.表结构及表数据闪回
FLASHBACK TABLE "BIN$2393" TO BEFORE DROP;
-- 或使用表的原始名称进行闪回
FLASHBACK TABLE finance_info TO BEFORE DROP;
-- 或通过RENAME TO命令指定表的新名称
FLASHBACK TABLE employees TO BEFORE DROP RENAME TO employees_recycle;

-- 3.验证相关对象是否保留系统生成的回收站名称，若名称未恢复，请使用ALTER INDEX语句手动修改
SELECT INDEX_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'FINANCE_INFO';
INDEX_NAME
--------------------------------
IDX_FINANCE_INFO_1
```

truncate操作闪回

示例（HEAP表）

```
ALTER SYSTEM SET RECYCLEBIN_ENABLED=ON;
TRUNCATE TABLE product;

-- 1.查询回收站是否存在product表
SELECT original_name,object_name FROM DBA_RECYCLEBIN WHERE original_name = 'PRODUCT';
ORIGINAL_NAME                  OBJECT_NAME
------------------------------ ------------------------
PRODUCT                        PRODUCT

-- 2.表数据闪回
FLASHBACK TABLE product TO BEFORE TRUNCATE;
```

```
-- 3.验证相关对象是否保留系统生成的回收站名称，若名称未恢复，请使用ALTER INDEX语句手动修改
SELECT INDEX_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'PRODUCT';
INDEX_NAME
-------------------------
SYS_C_133
```

闪回查询

表数据被UPDATE或DELETE更改后，可以通过闪回查询功能追溯到近期的历史数据，更多详情可查阅SELECT的flashback_query_clause。

可闪回查询的时间点由撤销保持期（UNDO_RETENTION）决定，建议将该参数设置为86400秒（24小时）或更长。

闪回查询不适用于分布式部署。

示例（单机、共享集群部署）

```
--area表中存在的一条记录
SELECT * FROM area WHERE area_no='03';
AREA_NO AREA_NAME          DHQ
------- ------------------ ------------
03      华南               Guangzhou

--获取当前时间
SELECT SYSTIMESTAMP res FROM dual;
RES
---------------------------------------------------------------
2023-12-17 14:14:08.498126

--删除此条记录并提交
DELETE FROM area WHERE area_no='03';
COMMIT;
SELECT * FROM area WHERE area_no='03';
AREA_NO AREA_NAME          DHQ
------- ------------------ ------------


--利用flashback可查询到快照历史数据
SELECT * FROM area AS OF TIMESTAMP TIMESTAMP('2023-12-17 14:14:08.498126')
WHERE area_no='03';
AREA_NO AREA_NAME          DHQ
------- ------------------ ------------
03      华南               Guangzhou
```

表的紧急恢复

对于无法通过闪回进行快速恢复的表，可使用YashanDB的物理备份恢复功能和导入、导出功能进行紧急恢复。

要恢复意外删除的表，请执行以下操作：

1.备份现有数据库的所有数据文件，以防止在此过程的剩余步骤中出错。

2.将数据库备份还原到备库环境。至少应恢复以下内容：

- SYSTEM和SYSAUX表空间
- 包含还原或回退段的表空间
- 包含要检索的数据的独立表空间

3.使用还原的备份控制文件对该备份执行不完全恢复，直到删除表之前。

4.从数据库的临时还原版本中导出丢失的数据。

5.将丢失的表导入生产数据库。

6.回收备库环境。

日志管理

日志管理章节所述范围为运维相关的日志管理，不包括与数据相关的redo/归档日志，对于redo/归档日志的管理将在文件管理章节描述。

日志分类

YashanDB的运维类日志分类如下：

- **运行日志run log**：运行日志记录了数据库各服务运行产生的轨迹信息、调试信息、状态变迁、未产生影响的潜在问题和直接的错误信息。
- **告警日志alert log**：当数据库发生死锁、表空间满、网络断连、坏块等错误时，会处于“带病运行”的状态，运维人员应该及时知道这些信息并作出处理。通过数据库在告警日志中发出告警（上报告警事件），运维人员定时检查告警日志，可以防患于未然，时刻了解和维护系统的运行状况。YashanDB包含的所有告警事件信息见参考手册告警事件。
- **监听日志listen log**：当数据库遭遇连接风暴等问题时，需要查看连接来源定位原因，监听日志会向运维人员提供连接状态和频率等信息。
- **慢日志slow log**：慢日志全称为慢查询日志（Slow Query Log），主要用于记录在数据库中执行时间超过指定时间的SQL语句，可选择输出到日志文件或系统表。通过慢查询日志，可以查找出哪些语句的执行效率低，以便进行优化。
- **代理程序日志yex_server.log**：代理程序日志记录了外置UDF代理程序服务运行产生的轨迹信息、调试信息、状态变迁、未产生影响的潜在问题和直接的错误信息。

下表列示了上述日志对应的文件信息：

日志文件	日志内容	说明
$YASDB_HOME/log/集群名/node-nodeid/run/run.log	运行日志	1.安装时默认按此路径及名称创建 2.可通过RUN_LOG_FILE_PATH参数修改文件路径，若路径不存在则会自动创建，需保证用户对该路径拥有相应权限 3.文件名称不可修改
$YASDB_DATA/log/alert/alert.log	告警日志	1.安装时默认按此路径及名称创建 2.文件路径不可修改 3.文件名称不可修改
$YASDB_DATA/log/listener/listener.log	监听日志	1.安装时默认按此路径及名称创建 2.文件路径不可修改 3.文件名称不可修改
$YASDB_HOME/log/集群名/node-nodeid/slow/slow.log	慢查询日志	1.安装时默认按此路径及名称创建 2.可通过SLOW_LOG_FILE_PATH参数修改文件路径，若路径不存在则会自动创建，需保证用户对该路径拥有相应权限 3.可通过SLOW_LOG_FILE_NAME参数修改文件名称 4.输出方式为FILE时记入此文件，为TABLE时记入系统表
$YASDB_DATA/log/external/server/yex_server.log	代理程序日志	1.安装时默认按此路径及名称创建 2.文件路径不可修改 3.文件名称不可修改

日志归档

下表列示了YashanDB支持日志归档的情况：

日志类型	支持情况
运行日志	支持
告警日志	不支持
监听日志	不支持
慢日志	不支持
代理程序日志	支持

> **Caution：**
> 对于不支持归档的日志，数据库管理员应该对其设置定期清理策略，避免占满磁盘空间。

运行日志归档

当运行日志文件达到RUN_LOG_FILE_SIZE（默认为20M）后，将进行归档并创建一个新的日志文件，归档的日志文件命名为原日志文件名+时间后缀，如：run/run-20211213151402.log。

代理程序日志归档

当代理程序日志文件达到16M后，将进行归档并创建一个新的日志文件，归档的日志文件命名为原日志文件名+时间后缀，如：external/server/yex_server20211213151402.log。

日志格式

运行日志文件内容格式

格式为：日期+时间+[级别]+[模块]+日志信息。如下：

```
2021-12-13 14:51:23.342 [INFO] [TABLESPACE] succeed to create tablespace SYSTEM
2021-12-13 14:53:18.559 [WARN] [CKPT] checkpoint not complete
2021-12-13 14:53:22.483 [DEBUG] [HA] send redo from point 0-5-12840-18360, size 278528, standby 1, next send point 0-5-12908-18373
```

日志级别

YashanDB运行日志支持不同级别的日志管理，并根据配置打印不同级别的日志：

级别	描述
OFF	关闭日志
FATAL	致命错误产生的日志
ERROR	一般错误产生的日志
WARN	告警类错误产生的日志
INFO	正常运行日志（默认日志级别）
DEBUG	调试日志级别
TRACE	追踪日志
ALL	所有日志

从上到下，日志等级依次增高，高日志级别包含低日志级别。

告警日志文件内容格式

格式为：TimeStamp | Session ID | Event Name | Object | Action | content |Event Escription。各字段说明见下表：

字段	类型	说明
TimeStamp	timestamp	产生告警的时间
Session ID	uint16	产生告警的会话ID
Event Name	text	告警类型名称
Object	uint16	产生告警的对象ID * Object和Event Name一起构成告警的唯一标识符 * 有些告警并不需要Object信息，则该值为0

字段	类型	说明
Action	int	告警状态： * 0：产生告警 * 1：告警消除
Content	text	对该告警的其他附带信息
Description	text	供用户进一步了解告警的详细信息

监听日志文件内容格式

日志格式示例如下：

```
2023-08-02 09:47:26.143 protocal = TCP  ip = fc00:7::126 port = 14526 user = SYS status = SUCCESS
2023-08-02 09:47:26.151 protocal = TCP  ip = fc00:7::126 port = 14528 user = SYS status = SUCCESS
2023-08-02 09:50:42.005 protocal = TCP  ip = fc00:7::126 port = 14650 user = SYS status = SUCCESS
2023-08-02 09:51:34.354 protocal = TCP  ip = fc00:7::126 port = 44058 user = REGRESS status = ERROR
2023-08-02 09:54:32.526 protocal = TCP  ip = fc00:7::126 port = 14656 user = SYS status = SUCCESS
2023-08-02 09:56:32.126 protocal = IPC  user = SYS status = SUCCESS
```

格式为：

TimeStamp | Protocal | Ip | Port | User | Status

字段	说明
TimeStamp	产生监听记录的时间
Protocal	链路类型 * TCP：TCP连接 * IPC： IPC连接
Ip	客户端IP
Port	客户端PORT
User	连接用户
Status	连接状态 * SUCCESS：连接成功 * ERROR：连接失败

慢日志内容格式

慢日志文件格式

示例如下：

```
# TIME: 2022-11-18 20:51:30.374
# USER_HOST: SYS@127.0.0.1
# DB_NAME: test_base1
# COST_EXECUTE_TIME: 0.000119
# COST_OPTIMIZE_TIME: 0.000040
# ROWS_SENT: 1
# SQL_ID: 3jd5aq4g9t7gf
SQL: select * from v$parameter where name = 'SLOW_LOG_SQL_MAX_LEN'
```

格式为：

字段	说明
TIME	打印该条慢SQL信息的时间

字段	说明
USER_HOST	执行SQL时的连接信息
DB_NAME	使用的数据库名称
COST_EXECUTE_TIME	SQL执行花费的时间，单位为秒
COST_OPTIMIZE_TIME	SQL优化计划阶段花费的时间，单位为秒
ROWS_SENT	返回的行数
SQL_ID	SQL ID

慢日志系统表格式

慢日志系统表名为SLOW_LOG$，表结构定义如下：

列名	说明
DATABASE_NAME	数据库名称
USER_NAME	用户名
START_TIME	日志记录时的时间，非执行开始时间
USER_HOST	连接IP
QUERY_TIME	执行时间
ROWS_SENT	查询返回的行数
SQL_ID	SQL的ID值
SQL_TEXT	SQL语句，超过2000会截断

运行日志管理

运行日志通过日志级别控制记录开关和记录策略，级别为OFF时不记录运行日志。

设置日志文件保留个数

通过设置RUN_LOG_FILE_COUNT参数修改运行日志的保留个数，当达到该参数设置的值（默认10）时，系统会自动删除时间靠前的日志。

日志文件保留个数修改指令：

```
ALTER SYSTEM SET RUN_LOG_FILE_COUNT=<保留个数> SCOPE=SPFILE;
```

该参数需重启数据库才能生效。

可进入日志文件存放位置观察文件的保留数量：

```
$ ls -l $YASDB_DATA/log/run
```

设置日志文件最大大小

通过设置 RUN_LOG_FILE_SIZE 参数修改运行日志的最大大小，当达到该参数设置的值（默认20M）时，该日志文件会自动归档。

日志文件最大大小修改指令：

```
ALTER SYSTEM SET RUN_LOG_FILE_SIZE=<日志文件大小> SCOPE=SPFILE;
```

该参数需重启数据库才能生效。

设置运行日志的存放路径

通过设置 RUN_LOG_FILE_PATH参数修改运行日志的存放路径。

运行日志的存放路径修改指令：

```
ALTER SYSTEM SET RUN_LOG_FILE_PATH='<path>' SCOPE=SPFILE;
```

该参数需重启数据库才能生效。

> **Warn**：
> 为运行日志设置的新路径必须已存在且对其拥有写权限，否则数据库将启动失败。

运行日志级别管理

运行日志级别的排序为：

ERROR < WARN < INFO < DEBUG < TRACE < ALL

从左到右，日志等级依次增高，高日志级别包含低日志级别。

查看当前日志级别的指令如下：

```
show parameter RUN_LOG_LEVEL;
```

ERROR级别

数据库产生了可处理的关键错误时，将输出ERROR日志。这些错误一般无法返回给客户端，例如后台线程的产生的错误。

将日志级别设置为ERROR的指令如下：

```
ALTER SYSTEM SET RUN_LOG_LEVEL=ERROR;
```

当YashanDB丢失表空间数据文件时，run.log打印情况如下：（本文以单机部署为例，具体日志以实际为准）

```
2022-06-17 19:20:56.981 20098 [ERROR] [MONITOR]
/data/yashan/yasdb_data/db-1-1/dbfiles/temp metadata changed, for example, permissions, timestamps, extended attributes,
and user/group ID, etc
2022-06-17 19:21:55.839 20308 [ERROR][errno=00313]: open file /data/yashan/yasdb_data/db-1-1/dbfiles/temp failed, errno 2,
error message "No such file or directory"
```

INFO级别

YashanDB默认使用的日志级别为INFO。

INFO级别记录数据库正常运行中发生的关键事件，主要包括：

- 数据库状态变更：例如启动、关闭、升主降备。
- 数据库关键资源的变更：例如表空间、用户增加删除等。
- 数据库资源变更：例如线程的启动和停止等。
- 数据库关键活动：例如重启恢复、残留事务等。
- 数据库参数变化：修改系统配置。

将日志级别设置为INFO的指令如下：

```
ALTER SYSTEM SET RUN_LOG_LEVEL=INFO;
```

示例

```
//数据库启动时，run.log打印情况如下：
2022-06-17 19:29:18.341 21320 [INFO] [DB] start database
2022-06-17 19:29:18.570 21320 [INFO] [DB] database mount
2022-06-17 19:29:18.570 21331 [INFO] [MONITOR] patrol process start
2022-06-17 19:29:18.599 21320 [INFO] [RCY] replay start, rcy begin point asn 1, block id 4650, lfn 1290. rcy least lfn 1289.
flush point asn 1, block id 4650, lfn 1290. scn 317965525569531904
2022-06-17 19:29:18.602 21320 [INFO] [RCY] stop to replay, point 0-1-4650-1290
2022-06-17 19:29:18.603 21320 [INFO] [RCY] replay end, rcy replay point asn 1, block id 4650, lfn 1290. current scn
317965525569531904
2022-06-17 19:29:18.603 21320 [INFO] [RCY] flush point 0-1-4650-1290, receive point 0-1-4650-1290
2022-06-17 19:29:18.700 21320 [INFO] [DB] database open
2022-06-17 19:29:18.701 21320 [INFO] [DATABASE] database start up,phase 2
2022-06-17 19:29:18.702 21338 [INFO] [MMS PRELOAD] start to preload memory mapped tablespaces
2022-06-17 19:29:18.702 21342 [INFO] pre-process TABXFMR$ successful
2022-06-17 19:29:18.704 21338 [INFO] [MMS PRELOAD] end to preload memory mapped tablespaces

//数据库关闭时，run.log打印情况如下：
2022-06-17 19:28:45.612 20747 [INFO] [DATABASE] shutdown database,mode: 2
2022-06-17 19:28:47.043 20747 [INFO] [HA] stop redo sender
2022-06-17 19:28:47.043 20747 [INFO] [HA] stop redo receiver
2022-06-17 19:28:47.088 20763 [INFO] [MONITOR] patrol process exited
```

DEBUG级别

DEBUG级别用于打印追溯问题时可能用到的关键信息。

将日志级别设置为DEBUG的指令如下：

```
ALTER SYSTEM SET RUN_LOG_LEVEL=DEBUG;
```

示例

```
//当YashanDB执行shutdown abort强制关闭数据库时，run.log打印如下：
2022-06-17 19:33:58.961 21916 [DEBUG][errno=02094]: current session has been killed or canceled
2022-06-17 19:33:58.961 21916 [DEBUG][errno=02094]: promise catch unhandled error: current session has been killed or
canceled
2022-06-17 19:34:15.683 21984 [DEBUG][yasdb][col_executor::executor]remove temp dir/home/yashan/yashandb/yasdb_data/tmp

//当数据文件的物理文件重命名后，启动数据库，run.log打印如下：
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][col_executor::executor]Local execute, end local resource
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][col_executor::executor]Local execute, end column query context
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][col_executor::executor]Local execute, prepare column query context
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][anchor::plsql::statement]Statement is cloned, check it is safe
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][anchor::plsql::statement]Statement is cloned, check it is safe
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][anchor::plsql::statement]Statement is cloned, check it is safe
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][col_executor::executor]Local execute, end local resource
2022-06-17 20:26:13.475 28790 [DEBUG][MMON][col_executor::executor]Local execute, end column query context
```

ALL级别

ALL级别记录所有日志信息，是最高级别的日志级别模式。

将日志级别设置为ALL的指令如下：

```
ALTER SYSTEM SET RUN_LOG_LEVEL=ALL;
```

告警日志管理

告警日志不设开关控制，恒为打开状态，即一直保持记录文件。记录文件大小没有设置上限，当alert.log文件过大时需要手动清理告警日志。

常见告警示例

表空间满

space full：当表空间已满，且申请不到页面时，记录告警日志。

当某个表空间已全部使用完时，告警日志将产生报错信息。

示例

```
//session 24在2022-06-20 00:20:46发现表空间TBS_TPCC空间使用率已达到100%，触发SpaceFull告警事件
//53表示表空间的space id，0表示该告警未消除。tablespace TBS_TPCC is full为提示信息。
2022-06-20 00:20:46.514|22|SpaceFull|53|0||tablespace TBS_TPCC is full
2022-06-20 00:20:46.520|24|SpaceFull|53|0||tablespace TBS_TPCC is full
```

连接数满

session exhausted：当连接数满时，上报连接数满告警。当可用连接数达到30%及以上，消除告警。

当前会话连接数超过MAX_SESSIONS参数设置的最大值时，运行日志将产生报错信息。

示例

```
2022-06-19 22:45:46.542 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
2022-06-19 22:45:46.554 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
2022-06-19 22:45:46.554 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
2022-06-19 22:45:46.554 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
2022-06-19 22:45:46.554 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
2022-06-19 22:45:46.554 5367 [ERROR] alloc session failed, error code:6013, error message: too many sql handlers
```

此时客户端无法连接数据库，并提示：

```
YAS-00406 connection is closed
YASQL-00007 invalid username/password, logon denied
```

主备连接断开

standby disconnect：当主库发现备库断连，记录告警日志。当主库发现备库重新连接上，记录消除告警的日志。

主库与备库断连时，alert.log打印情况如下：

```
//2021-10-29 19:41:05.841，在session 18 中发生当前HA架构中主库发现备库断开连接，主库向备库传输日志失败，触发了StandbyDisconnect告警事件。
//1表示备库 id，0表示该告警产生未消除。redo sender failed to connect为提示信息。
2021-10-29 19:41:05.841|18|StandbyDisconnect|1|0||redo sender failed to connect with standby
```

主库与备库断连修复时，alert.log打印情况如下：

```
//在2021-10-29 19:41:23.877主库和备库重新连接，断连告警修复。
//第一个1表示备库id，第二个1表示该告警产生已消除。
2021-10-29 19:41:23.877|18|StandbyDisconnect|1|1|
```

发生死锁

dead lock：当用户操作不当导致出现数据库出现死锁情况，存储引擎能够检测出死锁问题，通过中止构成死锁环中的某个会话的等待，从而解除死锁状态。

当以下3种死锁发生时候，数据库会选择死锁环中的一个session返回error，生成对应的死锁trace log，并记录告警日志，告警日志中指示trace log文件路径：

- 事务死锁：事务修改相同的数据产生并发等待，可能导致死锁。
- Xslot死锁：页面内事务槽位（Xslot）资源不足时，会产生事务等待，可能导致死锁。
- 表锁死锁：事务之间对表加共享锁与排他锁时，会产生事务等待，可能导致死锁。

例如，当发生事务相互更新导致死锁时，alert.log打印情况为：

```
//session 19发现了事务死锁，触发了DeadLock告警事件。
//该死锁的id号为21，0表示该告警的状态为未消除，lockType代表产生锁的类型，found xact dead lock为提示信息。
2022-06-23 18:44:17.191|19|DeadLock|21|0|lockType=2|found xact dead lock，more info in xxx.trc（xxx.trc为具体的死锁trace log路径）
```

监听日志管理

监听日志启动和关闭

通过设置LSNR_LOG参数来控制监听日志记录的启动和关闭，默认开启监听。

```
--查看当前监听是否打开
SHOW PARAMETER LSNR_LOG;
NAME             VALUE
---------------- ----------
LSNR_LOG         ON

--关闭监听
ALTER SYSTEM SET LSNR_LOG= OFF SCOPE=SPFILE;

--开启监听
ALTER SYSTEM SET LSNR_LOG= ON SCOPE=SPFILE;
```

该参数需重启数据库才能生效。

日志归档

监听日志记录文件大小没有设置上限，当listener.log文件过大时需要手动清理监听日志。

监听项

自监听开启后，所有的连接信息均会被记录在listener.log文件中，包括连接成功和连接失败的记录。

远程连接监听

YashanDB远程连接遵循TCP/IP协议，监听日志中会记录时间戳、TCP协议、IP、端口号、用户名和连接状态信息，如下所示：

```
2022-11-02 17:58:47.357 protocal = TCP  ip = 127.0.0.1 port = 60607 user = SALES status = SUCCESS
2022-11-02 17:59:59.414 protocal = TCP  ip = 127.0.0.1 port = 60693 user = SALES status = ERROR
```

UDS本地连接监听

YashanDB的UDS本地连接遵循IPC协议，监听日志中会记录时间戳、IPC协议、用户名和连接状态信息，如下所示：

```
2022-11-02 18:02:06.839 protocal = IPC  user = SYS status = SUCCESS
2022-11-02 18:50:34.438 protocal = IPC  user = SYS status = ERROR
```

慢日志管理

慢日志启动和关闭

通过设置ENABLE_SLOW_LOG参数来控制慢日志记录的启动和关闭，默认关闭。

```
--查看当前慢日志是否打开
SHOW PARAMETER ENABLE_SLOW_LOG;
NAME                VALUE
------------------- ----------
ENABLE_SLOW_LOG     FALSE

--关闭慢日志
ALTER SYSTEM SET ENABLE_SLOW_LOG = FALSE;

--开启慢日志
ALTER SYSTEM SET ENABLE_SLOW_LOG = TRUE;
```

设置慢日志时间阈值

通过设置SLOW_LOG_TIME_THRESHOLD参数来控制慢日志中的时间阈值，执行时间大于时间阈值的SQL会被认为慢SQL而记录到慢日志中。时间阈值的单位为毫秒。

```
--查看当前慢日志时间阈值
SHOW PARAMETER SLOW_LOG_TIME_THRESHOLD
NAME                        VALUE
--------------------------- ----------
SLOW_LOG_TIME_THRESHOLD     1000

--设置时间阈值
ALTER SYSTEM SET SLOW_LOG_TIME_THRESHOLD = 2000;
```

设置慢日志记录的SQL最大长度

通过设置SLOW_LOG_SQL_MAX_LEN参数来控制慢日志中的记录的SQL语句的最大长度，超过该值的SQL会被截断记录。

```
--查看当前慢日志记录的SQL最大长度
SHOW PARAMETER SLOW_LOG_SQL_MAX_LEN
NAME                        VALUE
--------------------------- ----------
SLOW_LOG_SQL_MAX_LEN        2000

--设置慢日志记录的SQL最大长度
ALTER SYSTEM SET SLOW_LOG_SQL_MAX_LEN = 1000;
```

设置慢日志文件名

通过设置SLOW_LOG_FILE_NAME参数来控制慢日志文件名。

```
--查看当前慢日志文件名
SHOW PARAMETER SLOW_LOG_FILE_NAME
NAME                       VALUE
-------------------------- ----------
SLOW_LOG_FILE_NAME         slow.log

--设置慢日志文件名
ALTER SYSTEM SET SLOW_LOG_FILE_NAME = myslow.LOG SCOPE = SPFILE;
```

设置慢日志文件的存储路径

通过设置SLOW_LOG_FILE_PATH参数来控制慢日志文件的存储路径。

```
--查看当前慢日志文件的存储路径
SHOW PARAMETER SLOW_LOG_FILE_PATH
NAME                    VALUE
----------------------- ------------
SLOW_LOG_FILE_PATH      ?/slow/log

--设置慢日志文件的存储路径
ALTER SYSTEM SET SLOW_LOG_FILE_PATH = '/home/user/log/slow/' SCOPE = SPFILE;
```

设置慢日志记录的输出方式

通过设置SLOW_LOG_OUTPUT参数来控制慢日志记录的输出方式。

目前支持TABLE和FILE类型：

TABLE：慢日志输出到SLOW_LOG$系统表内。

FILE：慢日志输出到慢日志文件内。

```
--查看当前慢日志记录的输出方式
SHOW PARAMETER SLOW_LOG_OUTPUT
NAME                      VALUE
----------------       ------------
SLOW_LOG_OUTPUT           FILE

--设置慢日志记录的输出方式
ALTER SYSTEM SET SLOW_LOG_OUTPUT = 'FILE';
```

日志归档

慢日志记录文件大小没有设置上限，当slow.log文件或SLOW_LOG$表过大时需要手动清理。

> **Note**：
> 1.慢日志采用异步输出方式，即执行完不会立即输出，而是放到日志队列里，通过后台线程分批次输出。
> 2.日志队列最大长度256，线程每200ms输出64条记录。
> 3.基于上述原理，在掉电情况下可能会丢失日志；如果时间阈值设置过小，致使队列满的情况下亦会丢失日志。
> 4.主备环境中，备库将SLOW_LOG_OUTPUT设置为FILE时，慢日志输出到文件，将SLOW_LOG_OUTPUT设置为TABLE不会报错，但不会生效，日志也不会继续输出到文件。备库SLOW_LOG$的内容是同步的主库的SLOW_LOG$。
> 5.根据上条规则，不建议将备库的SLOW_LOG_OUTPUT设为TABLE；如一个TABLE模式的主库被切换为备库，请注意其慢日志将无法输出，需修改SLOW_LOG_OUTPUT参数为FILE使慢日志能输出到文件。

日志收集

日志收集

日志收集命令主要是一键收集节点的日志，把远端服务器上的节点的日志统一打包发送到本地服务器。

日志收集命令仅适用于通过yasboot安装的数据库，否则需要先对该数据库进行托管。

示例

```
./bin/yasboot cluster log -c yashandb
start:  2023-07-28 00:00:00
end  :  2023-07-28 20:11:19
 type  | uuid             | name        | hostid   | index               | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------------------------
 task  | 05fc032f6cc54f69 | LogCollect  | -        | tt                  | SUCCESS | 0           | 100      | 4
-------+------------------+-------------+----------+---------------------+---------+-------------+----------+------
 child | 74413dd89a9de397 | LogCompress | host0001 | host0001            | SUCCESS | 0           | 100      | 1
       +------------------+-------------+----------+---------------------+---------+-------------+----------+------
       | bd728249e323192e | LogDownload | host0001 | host0001            | SUCCESS | 0           | 100      | 1
       +------------------+-------------+----------+---------------------+---------+-------------+----------+------
       | d08b47e74604478a | LogCombine  | -        | tt-05fc032f6cc54f69 | SUCCESS | 0           | 100      | 1
-------+------------------+-------------+----------+---------------------+---------+-------------+----------+------
task completed, status: SUCCESS
download log to /var/lib/jenkins/anchorbase/work/bin
downloading...
download file block size: 2097152
```

命令执行完成后，打包的日志文件默认存放在当前目录下，可以使用-o参数来控制下载目录。 具体参数以及含义可查看yasboot。

资源和计划管理

- 资源管理主要是指对CPU或会话等资源进行合理分配与调节：
  - CPU资源管理适用于单机部署（非级联备）和分布式部署。
  - 会话管理适用于分布式部署。
- 计划管理是指通过yasboot工具定时执行SQL任务，例如DBMS_STATS相关计划可定时管理优化器的统计信息，计划与产品部署形态间的适用范围取决于实际需执行的SQL任务。
- 为提高安全性，执行部分功能时YashanDB会自发启动yex_server沙箱进程独立加载相应的功能模块。
  - 调用外置存储过程（外置自定义函数等）时。
  - 对dblink远端表做INSERT、DELETE、UPDATE以及SELECT操作时。

CPU资源管理

YashanDB CPU资源管理用于保证数据库在稳定运行的前提下，保障核心用户的使用和紧急任务的运行及最大限度提高CPU整体资源的利用率。

本功能适用于单机部署（非级联备）和分布式部署。

CPU资源管理功能按节点生效，例如配置某个用户的CPU资源上限为20%则表示该用户对环境中每个节点的CPU可使用上限为20%。CPU资源上限可能存在波动，上限+3%以内均属正常情况。

核心概念

资源使用组

资源使用组由资源使用者组成，同一资源使用组内的资源使用者共用一份资源计划指令。

系统包含如下默认资源使用组：

- SYS_GROUP：包括系统用户sys和系统进程。
- DEFAULT_CONSUMER_GROUP：不存在资源映射关系的资源使用者默认划分至该组。

创建会话时，YashanDB会根据用户的资源映射关系自动将其划分至相应资源使用组中。数据库管理员可以通过修改用户的资源映射关系调整该用户的新会话所属资源使用组。

资源映射

资源映射是指将资源使用者加入至资源使用组中，资源使用者以用户（USER）为维度。

资源使用组与资源使用者为一对多关系，即一个资源使用者只能加入一个资源使用组但一个资源使用组可以包含多个资源使用者。

资源计划指令

资源计划指令是对环境中每个节点物理资源（CPU）的分配计划，通过关联资源使用组为资源使用者分配资源。

资源计划指令与资源使用组为一对多关系，即一个资源使用组只能指定一个资源计划指令但一个资源计划指令可以应用于多个资源使用组。

使用资源管理

Step1：配置CPU资源管理功能

场景一：暂未安装YashanDB

1. 在YashanDB单机部署或YashanDB分布式部署的 `Step1：生成部署文件` 中，增加 `--create-cgroup` 选项，用于创建cgroup目录。

2. 执行如下命令，部署具备CPU资源管理功能的数据库。

```
# 单机
$ yasboot package se gen --cluster yashandb -u yashan -p password --ip 192.168.1.2 --port 22 --install-path
/data/yashan/yasdb_home  --data-path /data/yashan/yasdb_data --begin-port 1688 --create-cgroup --sudo-username yashan --
sudo-password password
# 分布式
$ yasboot package de gen --cluster yashandb -u yashan -p password --ip 192.168.1.2 --port 22 --install-path
/data/yashan/yasdb_home  --data-path /data/yashan/yasdb_data --begin-port 1688 --create-cgroup --sudo-username yashan --
sudo-password password
```

部署完成后，会自动在/etc/rc.local文件中增加一行命令，以便在数据库启动后启用资源管理功能。

场景二：已安装YashanDB

通过yasboot工具的 `host cgroup create` 命令创建cgroup目录，详细参数说明请查阅yasboot host。

```
# 为所有主库创建资源管理cgroup目录
$ yasboot host cgroup create --cluster yashandb --sudo-username yashan --sudo-password ssh密码
```

Step2：创建DBA用户

如下语句仅为示例，具体操作请查阅创建用户。

```
-- 创建名为RMUSER的用户，并为其赋予DBA权限
CREATE USER RMUSER IDENTIFIED BY RMUSER DEFAULT TABLESPACE SYSTEM;
GRANT DBA TO RMUSER;
```

Step3：创建资源使用组

如下语句仅为示例，具体参数请查阅DBMS_RESOURCE_MANAGER中CREATE_CONSUMER_GROUP相关介绍。

```
-- 创建名为RESMANGROUP的资源使用组，只有SYS用户才有权限执行
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP('RESMANGROUP','');
END;
/
```

Step4：创建用户映射关系

如下语句仅为示例，具体参数请查阅DBMS_RESOURCE_MANAGER中SET_CONSUMER_GROUP_MAPPING相关介绍。

```
-- 将用户RMUSER映射到RESMANGROUP资源使用组，只有SYS用户才有权限执行
BEGIN
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING('USER','RMUSER','RESMANGROUP');
END;
/
```

Step5：创建资源计划指令

如下语句仅为示例，具体参数请查阅DBMS_RESOURCE_MANAGER中CREATE_PLAN_DIRECTIVE相关介绍。

```
-- 创建名为RESMAN的资源计划指令：RESMANGROUP资源使用组的CPU共享资源为11%、CPU上限为50%，只有SYS用户才有权限执行
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE('RESMAN','RESMANGROUP',11,50);
END;
/
```

Step6：启用资源管理功能

1. 通过yasboot工具修改 `RSRC_MODE` 参数开启资源管理功能，重启数据库使之生效。

```
$ yasboot cluster config set -c yashandb -k RSRC_MODE -v CPU -d
$ yasboot cluster restart -c yashandb
# 可通过如下命令查看RSRC_MODE参数于所有节点中修改结果
$ yasboot cluster config show -c yashandb -q RSRC_MODE
```

> **Note**:
> `RSRC_MODE` 参数只允许使用 `yasboot cluster config set` 命令修改。

2. 重启后连接数据库，检查资源管理功能是否成功开启，`RSRC_MODE` 参数值为CPU表示已成功开启。

```
show parameter RSRC_MODE;

NAME                                                                 VALUE
-------------------------------------------------------------------- ----------------------------------------------------------------
------
RSRC_MODE                                                            CPU
```

通过上述示例成功开启CPU资源管理功能后，使用RMUSER用户连接数据库时会受到资源计划RESMAN约束。

修改映射关系

如需更改某个用户映射的资源使用组，有如下两种方式，下述语句仅为示例，具体参数请查阅DBMS_RESOURCE_MANAGER相关介绍。

- 先删除原有的映射关系，再将该用户映射到新的资源使用组。

```
-- 创建新资源使用组NEWGROUP
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP('NEWGROUP','');
END;
/

-- 删除RMUSER用户原有的映射关系
BEGIN
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP_MAPPING('USER','RMUSER');
END;
/

-- 将RMUSER映射至NEWGROUP组中
BEGIN
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING('USER','RMUSER','NEWGROUP');
END;
/
```

- 先将用户映射到DEFAULT_CONSUMER_GROUP组中，再将该用户映射到新的资源使用组。

```
-- 创建新资源使用组NEWGROUP2
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP('NEWGROUP2','');
END;
/

-- 将RMUSER映射至DEFAULT_CONSUMER_GROUP组中
BEGIN
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING('USER','RMUSER','DEFAULT_CONSUMER_GROUP');
END;
/

-- 将RMUSER映射至NEWGROUP2组中
BEGIN
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING('USER','RMUSER','NEWGROUP2');
END;
/
```

修改映射关系后，用户新建的会话会归属于新的资源使用组，但已在使用中的会话会继续归属于原有资源使用组直至结束会话。

关闭资源管理

如需删除某个资源使用组，需先删除该资源使用组相关的映射关系和资源计划指令，否则返回错误。

1. 删除映射关系。

> **Note**：
> - 删除用户时，该用户的映射关系会被删除。
> - 删除某个用户的映射关系时，该用户会被自动映射至DEFAULT_CONSUMER_GROUP组中。
> - 无法删除系统用户SYS的映射关系。

```
BEGIN
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP_MAPPING('USER','RMUSER');
END;
/
```

2. 删除资源计划指令。

```
BEGIN
DBMS_RESOURCE_MANAGER.DELETE_PLAN_DIRECTIVE('RESMAN','RESMANGROUP');
END;
/
```

3. 删除资源使用组。

   如果资源使用组仍有其它资源使用者，将返回错误。

```
BEGIN
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP('RESMANGROUP');
END;
/
```

4. 通过yasboot工具修改`RSRC_MODE`参数关闭资源管理功能并重启数据库，待数据库重启后生效。

```
$ yasboot cluster config set -c yashandb -k RSRC_MODE -v NONE -d
$ yasboot cluster restart -c yashandb
# 可通过如下命令查看RSRC_MODE参数于所有节点中的修改结果
$ yasboot cluster config show -c yashandb -q RSRC_MODE
```

5. 删除资源管理的重启自动生效的命令。

   在YashanDB卸载时，需要使用`-t`参数，以便删除资源管理功能开机自启动的功能。

```
$ yasboot package uninstall --cluster yashandb -t hosts.toml
```

会话管理

YashanDB会话管理在分布式环境下需要协调各节点间的会话资源，用以隔离用户对于DN上相关会话资源管理。主要在于会话管理采用线程池的方式处理来自客户端连接的任务，并且同时限制任务实际在数据库系统中的并发数量，以供数据库更加高效的运行。

查看相关配置参数

MAX_REACTOR_CHANNELS表示负责监听处于等待状态的网络连接的后台线程，MAX_REACTOR_CHANNELS = 0表示不启用REACTOR模式，每个客户端连接分配独立线程处理进行处理，MAX_REACTOR_CHANNELS >= 1表示启用REACTOR模式，客户端连接采用共享线程池处理，共享线程池线程数量请参考参数MAX_WORKERS。当并发量较大时，建议适当调大MAX_REACTOR_CHANNELS。

MAX_REACTOR_CHANNELS在分布式部署中的默认安装值为1，若MAX_WORKERS >= MAX_SESSIONS，对应的共享线程池线程数量MAX_WORKERS的值取两者中较小的值。

MAX_REACTOR_CHANNELS在单机、集群部署中的默认安装值为0，若MAX_WORKERS >= MAX_SESSIONS，即使配置了该参数>=1，也不会起作用。

MAX_SESSIONS表示最大可申请的会话数，包含一定数量的后台会话，要求单个MN，DN上的配置=各个CN上配置之和。

MAX_WORKERS表示线程池的数量，即实际数据库中并行的最大任务数，要求单个MN，DN上的配置=各个CN上配置之和。

> **Note**:
> 注意：处于事务中的客户端请求会一直占用共享线程，若超过共享线程数量的事务不提交，会阻塞其他连接的任务处理，若发生类似情况，请使用保留连接进行应急处理。

示例

```
show parameter MAX_REACTOR_CHANNELS;

show parameter MAX_SESSIONS;

show parameter MAX_WORKERS;
```

查看相关视图信息

DV$SESSION包含分布式各个节点上会话的相关信息，可通过GLOBAL_SESSION_ID字段建立各个节点间会话之间的联系。

DV$DATA_CONNECTION包含各个CN上会话的相关信息，主要是CN和各个已建立连接的节点之间的关系。

示例（分布式部署）

```
SELECT * FROM DV$SESSION;

SELECT * FROM DV$DATA_CONNECTION;
```

计划管理

YashanDB计划管理是指通过yasboot工具定时执行SQL任务，具体的命令参数介绍请查阅yasboot job命令。

通过yasboot调用DBMS_STATS高级包可定时管理优化器的统计信息。

1. 生成job配置文件。

```
$ ./bin/yasboot job config gen -c yashandb --job-name test --user username --password password --sql "exec
DBMS_STATS.GATHER_DATABASE_STATS('GATHER AUTO', 1, 2,  'FOR ALL COLUMNS SIZE AUTO', 'AUTO', TRUE, FALSE);" -ce "30 0 * * *"
|key                   |value
|----------------------+------
|cluster               |yashandb
|job_name              |test
|sql                   |exec DBMS_STATS.GATHER_DATABASE_STATS('GATHER AUTO', 1, 2,  'FOR ALL COLUMNS SIZE AUTO', 'AUTO',
TRUE, FALSE);
|--------------------- |--------
|cron_expression       |30 0 * * *

Generate config success
```

成功执行上述命令后，系统会生成job_*集群名*_*job名*.toml配置文件（根据上述示例命令，文件名为job_yashandb_test.toml），允许手动修改该配置文件。

```
cluster = "yashandb"
job_name = "test"
username = "username"
password = "password"
sql = "exec DBMS_STATS.GATHER_DATABASE_STATS('GATHER AUTO', 1, 2,  'FOR ALL COLUMNS SIZE AUTO', 'AUTO', TRUE, FALSE);"

[time_config]
cron_expression = "30 0 * * *"
```

2. 添加job。

```
$ ./bin/yasboot job add -t job_yashandb_test.toml
```

3. 将job应用到数据库节点，本文以1-1节点为例。

```
$ ./bin/yasboot job apply -c yashandb --job-name test -n 1-1
```

4. 查看计划执行状态。

达到设定的计划时间时会自动触发job，自动执行配置的SQL命令或SQL文件。job执行完成后，可以通过job show命令查看执行情况。

```
$ ./bin/yasboot job show -c yashandb -n 1-1 --job-name test
id | job_id                           | job_name | node_id | status  | start_time          | completion_time     | hostid
| failed_reason
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
2  | 65a8a0d8954d05c0fef16ef3021bf34a | test     | 1-1     | failed  | 2024-01-18 11:58:28 | 2024-01-18 11:58:28 | host0001
| node 1-1 exec job sql failed, stdout:
[1:1]YAS-04231 keyword expected
----+----------------------------------+----------+---------+---------+---------------------+---------------------+--------
--+-------------------------------------------------------------------------
1  | 65a8a0d8954d05c0fef16ef3021bf34a | test     | 1-1     | success | 2024-01-18 11:55:00 | 2024-01-18 11:55:00 | host0001
| -
----+----------------------------------+----------+---------+---------+---------------------+---------------------+--------
--+-------------------------------------------------------------------------
```

yex_server沙箱进程管理

yex_server沙箱进程是由yasdb进程在特定场景中自发启动的守护进程，YashanDB将相关功能模块独立加载到沙箱进程上，利用进程隔离思想和进程间通信技术，提高相关功能执行安全性。

使用场景

- 为提高安全性，执行部分功能时YashanDB会自发启动yex_server沙箱进程独立加载相应的功能模块。
  - 调用外置存储过程（外置自定义函数等）时。
  - 对dblink远端表做INSERT、DELETE、UPDATE以及SELECT操作时。

调整内存池大小

yex_server沙箱进程驱动默认的内存池大小为32M，取值范围/格式：[32M,1T]，该配置由yex_server.ini配置文件中的YDBC_BUFFER_SIZE参数控制。YashanDB安装后，yex_server.ini文件不会自动生成，您可以根据实际需求创建文件并调整该参数值。

1. 查询$YASDB_DATA/external/server路径下是否存在yex_server.ini文件，如不存在则创建：

```
$ echo $YASDB_DATA
/data/yashan/yasdb_data/db-1-1 # 本文以/data/yashan/yasdb_data/db-1-1为例

$ cd /data/yashan/yasdb_data/db-1-1/external/server
$ ll
$ vi yex_server.ini
```

2. 在yex_server.ini文件中新增或修改如下配置：

```
YDBC_BUFFER_SIZE = 64M
```

3. 保存并退出编辑。

4. 重启实例后配置生效。

```
$ yasboot cluster restart -c yashandb
```

备份恢复

使用备份恢复的常见场景如下：

- 数据保护：系统崩溃、磁盘损坏、应用错误、人为误操作等因素可能导致数据丢失，对数据库进行定期备份，并在需要时恢复，可以规避数据丢失对业务造成的影响。
- 数据迁移：因某种需求，数据库中的数据需要迁移至其他主库，此时可以借助备份恢复功能实现数据的迁移。

作为业务系统的数据底座，数据库必须要保证其数据的安全性，而对于数据库管理员，制定一份合适的备份策略则至关重要。

传统的备份方法分为冷备（离线备份，数据库停止服务）和热备（在线备份，数据库正常运行），本章将着重介绍YashanDB的热备能力，数据库管理员在使用时可以参考这些文档，并根据自身情况作适当调整，确保企业的数据安全。

YashanDB支持的备份恢复能力主要包括如下内容：

- **全量备份**：对某一时间点上的所有数据（分布式部署中对所有主节点）进行完全复制，不依赖之前的备份集。一个全量备份集可以恢复出所有数据，恢复时间最短且操作最方便。
- **增量备份**：首次执行基线备份（level 0），后续每次只需备份增量数据（level 1），备份效率高，节省磁盘空间。恢复时需要从基线备份集开始依次恢复所有增量备份。
- **归档备份**：对当前正常运行的数据库产生的归档日志文件进行完全复制，可以灵活指定备份的归档文件范围（包括SEQUENCE、SCN、TIME），生成属于该范围归档文件的独立备份集。
- **全量恢复**：从备份集执行恢复后，若在该备份集后生成的归档日志和在线日志连续且完整，YashanDB将应用所有日志，恢复数据库至连续日志序号的最新时间点；若该备份集后生成的归档日志无保留，则只能恢复备份集时间点数据。对于分布式部署，全量恢复指所有的节点都能恢复到备份时间点，并且在恢复后open所有节点。
- **归档恢复**：指定已存在的归档日志备份集并指定要恢复的归档范围（包括SEQUENCE、SCN、TIME）即可恢复至连接数据库。
- **指定时间点恢复**：从备份集执行恢复后，若在该备份集后生成的归档日志和在线日志连续且完整，指定让YashanDB继续恢复数据库至任意时间点。
- 通过**并行度**设置提高备份和恢复的效率。
- **备份压缩**：当需要备份的数据量较大、磁盘空间紧张或存在其他数据压缩需求时，可以执行压缩备份，提高磁盘利用率。
- **备份加密**：若对数据备份内容有较高安全性要求，在备份时可指定不同的加密算法对备份数据集加密，保证备份数据集的安全性。

> **Note**：
> - 备份命令仅SYS超级用户或拥有SYSDBA、SYSBACKUP权限的用户才可执行。恢复命令仅SYS用户才可执行。
> - 在执行备份恢复操作前，建议先检查当前主节点上的HA_ELECTION_LEADER_LEASE_ENABLED参数值，若为TRUE，表明自动降备功能已开启，需先关闭此功能再进行备份恢复。待备份恢复操作结束后，可根据实际需求决定是否再次开启自动降备功能。开启后，主节点在未收到多数派备节点心跳响应时会主动降为备节点。
> - 归档备份需保证正常注册在数据库中的日志文件可用，否则会导致备份失败。
> - 如果数据库中存在本地临时表空间或本地SWAP表空间，备份恢复后不会直接创建本地临时表空间或本地SWAP表空间的文件，待数据库触发到open状态时才会创建相应文件。

SQL命令备份恢复

SQL命令方式的备份恢复操作适用于单机/共享集群部署的数据库。

操作示例

以下为对单机部署的数据库执行备份恢复的模拟场景：

1.通过yasql连接数据库，将数据库切换到归档模式，归档模式必须在数据库MOUNT状态下才能开启。

```
$ yasql sales/sales

SQL> ALTER DATABASE ARCHIVELOG;
```

2.模拟业务场景，在数据库中创建表并插入数据。

```
CREATE TABLE backuptable (b1 INT,b2 INT);

INSERT INTO backuptable VALUES (2,3);

COMMIT;
```

3.执行数据全量备份。

```
BACKUP DATABASE FULL FORMAT '/YashanDB/backup/full_20211212181530';
```

> **Caution**：
> 确保“/YashanDB/backup”目录存在且YashanDB安装用户具有读写权限，如果指定的数据备份目录已存在，则会触发 YAS-00318 错误。

4.使用具有DBA权限的用户查询DBA_BACKUP_SET视图，检查备份详情。

```
SELECT * FROM DBA_BACKUP_SET;

      RECID# START_TIME                      COMPLETION_TIME                TYPE          INCREMENT_LEVEL INCREMENT_ID#
PATH                                                          TAG
TRUNC_LSN COMPRESS_LEVEL           INPUT_BYTES          OUTPUT_BYTES RCY_BEGIN_ASN         RCY_BEGIN_LFN  RCY_END_ASN
RCY_END_LFN COMPRESS_ALGO ENCRYPT_ALGO
----------- ------------------------------ ------------------------------ ------------- --------------- ------------- --
---------------------------------------------------------- ------------------------------------------------------------
-- -------------------- -------------- -------------------- -------------------- ------------- -------------------- -----
------- -------------------- ------------- ------------
           1 2023-06-19                     2023-06-19                     FULL                       0             0
/YashanDB/backup/full_20211212181530                    bak_2023061902151684
53767 NONE                       516325376            516325376            8                51469           8
51471 NONE          NONE
```

5.检查生成的备份集物理文件。

```
$ cd /YashanDB/backup/full_20211212181530
$ ls -rlt
total 504232
-rw-r-----. 1 yashan yashan  28729344 Jun 19 02:15 ctrl_0_0_0.bak
-rw-r-----. 1 yashan yashan  67108864 Jun 19 02:15 data_0_0_0.bak
-rw-r-----. 1 yashan yashan  67108864 Jun 19 02:15 data_1_0_0.bak
-rw-r-----. 1 yashan yashan      8192 Jun 19 02:15 data_3_0_0.bak
-rw-r-----. 1 yashan yashan      8192 Jun 19 02:15 data_2_0_0.bak
-rw-r-----. 1 yashan yashan  67108864 Jun 19 02:15 data_4_0_0.bak
-rw-r-----. 1 yashan yashan 134217728 Jun 19 02:15 data_5_0_0.bak
-rw-r-----. 1 yashan yashan 134217728 Jun 19 02:15 data_5_0_1.bak
```

```
-rw-r-----. 1 yashan yashan   1040384 Jun 19 02:15 arch_0_8_0.bak
drwx------. 2 yashan yashan         6 Jun 19 02:15 bucket_4_0_0.bak
-rw-r-----. 1 yashan yashan  16777216 Jun 19 02:15 backup_profile
-rw-r-----. 1 yashan yashan      5120 Jun 19 02:15 backup_filelist
```

其中：

- ctrl_*文件为备份的控制文件。
- data_*文件为备份的数据文件。
- arch_*文件为备份的归档文件。
- redo_*文件为备份的在线日志文件，仅在备库执行备份生成的备份集中存在 。
- 当业务中存在LSC表时，bucket_4_0_0.bak/下存放LSC表的可变数据文件。
- backup_profile文件为备份集元数据文件，用于备份集和数据库版本的校验，备份集恢复关键信息等。
- backup_filelist文件用于校验备份集数据的完整性正确性。

6.模拟灾难场景，删除数据库文件。

> **Warn**：
> 以下操作只能在测试环境上执行，请勿直接在生产环境上测试。

```
#$YASDB_DATA/dbfiles 为数据库文件路径
$ cd $YASDB_DATA/dbfiles
$ rm -rf ./*
$ ls -rlt
total 0
```

7.重启数据库到NOMOUNT状态，执行恢复，然后检查新增数据是否恢复。

```
SHUTDOWN IMMEDIATE;
exit;

$ yasboot cluster start -c yashandb -m nomount

$ yasql sales/sales

SQL> RESTORE DATABASE FROM '/YashanDB/backup/full_20211212181530';

RECOVER DATABASE;

ALTER DATABASE OPEN;

SELECT * FROM backuptable;

          B1           B2
------------ ------------
           2            3
```

归档文件备份恢复

归档日志备份恢复的SQL命令操作方式适用于单机/共享集群部署的数据库。

注意事项

（一）使用本方式执行备份时，要求数据库运行于OPEN状态且归档模式开启。

（二）使用本方式执行恢复时，要求当前数据库实例处于执行完RESTORE且未执行RECOVER的MOUNT状态。

操作示例

以下为对单机部署的数据库执行归档备份恢复的模拟场景：

备份示例

1.通过yasql连接数据库，将数据库切换到归档模式，归档模式必须在数据库MOUNT状态下才能开启。

```
$ yasql username/password

SQL> ALTER DATABASE ARCHIVELOG;
```

2.模拟业务场景，在数据库中创建表并插入数据，并执行多次切换LOGFILE生成多个归档。

```
CREATE TABLE backuptable (b1 INT,b2 INT);

INSERT INTO backuptable VALUES (2,3);

COMMIT;

-- 执行多次
ALTER SYSTEM SWITCH LOGFILE;
```

```
-- 查看已生成归档
SELECT SEQUENCE#,FIRST_CHANGE#,NEXT_CHANGE# FROM V$ARCHIVED_LOG;
```

3.指定归档的SEQUENCE区间备份。

```
SELECT SEQUENCE#,FIRST_CHANGE#,NEXT_CHANGE# FROM V$ARCHIVED_LOG;

   SEQUENCE#         FIRST_CHANGE#          NEXT_CHANGE#
------------ -------------------- --------------------
           1    525940019749924864    525940036244176896
           2    525940036244176896    525940261064138752
           3    525940261064138752    525940261064138752
           4    525940261064138752    525940261064138752
           5    525940261064138752    525940400603951104


-- 执行备份
BACKUP ARCHIVELOG SEQUENCE BETWEEN  2 AND 5  FORMAT '/YashanDB/backup/SEQUENCE_2_5';
```

> **Note**：
> 因为上述示例中在切日志期间并未做任何业务，所以部分归档的SCN起始是一样的，实际生产环境中表现正常。

4.使用具有DBA权限的用户查询DBA_ARCHIVE_BACKUPSET视图，检查备份详情。

```
SELECT * FROM DBA_ARCHIVE_BACKUPSET;

               RECID# INSTANCE_NUMBER# START_TIME                                    COMPLETION_TIME
TYPE          PATH                                                                  TAG
COMPRESS_ALGO COMPRESS_LEVEL ENCRYPT_ALGO           INPUT_BYTES          OUTPUT_BYTES SEQUENCE_BEGIN# SEQUENCE_END#
MIN_FIRST_CHANGE#      MAX_NEXT_CHANGE# RESTORE_TIME
-------------------- --------------- ------------------------------------------------------------ --------------------------------
----------- ------------ --------------- ------------------------------------------------------------ --------------------------------
------------------------------ ------------- -------------- ------------ -------------------- -------------------- --------
------- ------------- -------------------- -------------------- ------------------------------------------------
                   1               1 2024-01-26                                    2024-01-26
ARCHIVE       /YashanDB/backup/SEQUENCE_2_5                                         bak_2024012611380123
NONE          NONE          NONE                    17362944             17362944               2             5
525940036244176896    525940400603951104
```

5.检查生成的备份集物理文件。

```
$ cd /YashanDB/backup/SEQUENCE_2_5
$ ls -rlt
total 16960
-rw-r----- 1 zhangxt zhangxt    12288 Jan 26 11:38 arch0_0_3_0.bak
-rw-r----- 1 zhangxt zhangxt   544768 Jan 26 11:38 arch0_0_2_0.bak
-rw-r----- 1 zhangxt zhangxt    12288 Jan 26 11:38 arch0_0_4_0.bak
-rw-r----- 1 zhangxt zhangxt    16384 Jan 26 11:38 arch0_0_5_0.bak
-rw-r----- 1 zhangxt zhangxt 16777216 Jan 26 11:38 backup_profile
-rw-r----- 1 zhangxt zhangxt     2560 Jan 26 11:38 backup_filelist
```

其中：

- arch*为独立的归档文件，arch{instanceId}{*resetId*}{asn}_{secId}，分别表示节点的ID，resetlogs的id，归档文件的序列号ASN，归档文件的分片序列号。
- backup_profile文件为备份集元数据文件，用于备份集和数据库版本的校验，备份集恢复关键信息等。
- backup_filelist文件用于校验备份集数据的完整性正确性。

恢复示例

1.数据库只可以在DB执行RESTORE之后且未执行RECOVER之前执行归档恢复，然后检查归档文件是否恢复到指定目录。

```
$ cd /YASDATA/archive
$ ll -rlt
total 23296
-rw-r----- 1 yashan yashan    36864 Nov 29 09:41 arch_0_1.ARC
-rw-r----- 1 yashan yashan 23781376 Nov 29 09:46 arch_0_2.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 09:47 arch_0_3.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 09:47 arch_0_4.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 09:47 arch_0_5.ARC

-- 模拟归档文件丢失，请勿在生产环境上测试
$ rm arch_0_3.ARC arch_0_5.ARC

$ ll -rlt
total 23272
-rw-r----- 1 yashan yashan    36864 Nov 29 09:41 arch_0_1.ARC
-rw-r----- 1 yashan yashan 23781376 Nov 29 09:46 arch_0_2.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 09:47 arch_0_4.ARC

-- 恢复指定SEQUENCE的范围为2-4，即只恢复该区间内的归档
SQL> RESTORE ARCHIVELOG SEQUENCE BETWEEN 2 AND 4 FROM SEARCHDIR '/YashanDB/backup';


-- 检查文件仅恢复了SEQUENCE为3的归档文件。
$ ll -rlt
total 23284
-rw-r----- 1 yashan yashan    36864 Nov 29 09:41 arch_0_1.ARC
-rw-r----- 1 yashan yashan 23781376 Nov 29 09:46 arch_0_2.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 09:47 arch_0_4.ARC
-rw-r----- 1 yashan yashan    12288 Nov 29 10:17 arch_0_3.ARC
```

> **Note**：
> 除指定SEQUENCE区间外，也可指定为时间、SCN区间，详见BACKUP ARCHIVELOG。

备份

操作说明

- 使用本方式执行备份时，要求数据库运行于OPEN状态且归档模式开启。
- 备份命令仅SYS超级用户或拥有SYSDBA、SYSBACKUP权限的用户才可执行，备份语法详细说明请参考开发手册BACKUP DATABASE。
- 若数据库在只读模式（主备模式下，备库进行备份）下备份，需要指定FORCE关键字。
- 备份会消耗一定的系统和数据库资源，请根据业务负载酌情选择备份窗口。
- 跟踪$YASDB_DATA/log/run/run.log运行日志，可查看备份恢复过程的详细日志。
- 可通过 `select * from V$BACKUP_PROGRESS` 语句实时查看备份进度。
- 增量备份的多个备份集可采用不同的压缩算法。
- 增量备份的多个备份集必须采用一致的加密策略（都加密或都不加密，都加密时密码相同），但可以采用不同的加密算法。
- 共享集群部署中，同一时间只能在一个实例上执行备份操作，且备份期间如果任一实例宕机，该集群的备份操作将备中断。

备份

全量备份

示例（单机）

```
BACKUP DATABASE FULL FORMAT '/data/backup/full_20211209191000' TAG 'yas_full_backup' PARALLELISM 3;
```

示例（共享集群）

```
BACKUP DATABASE FULL FORMAT '+DG0/backup/yfs_full_20211209191000' TAG 'yfs_yas_full_backup' PARALLELISM 3;
```

本示例中，full关键字表明当前为全量备份，format关键字指定生成备份集的文件名称，tag关键字指定备份集的标识为"yas_full_backup"，parallelism关键字指定备份任务以3个线程的并行度执行。

其中，备份共享集群部署的数据库时，format关键字指定的内容决定了备份集放于实例所在服务器磁盘，或者放于共享存储。当指定的内容为YFS路径时，表示将备份至共享存储；为普通磁盘绝对路径时，表示备份至服务器磁盘。

LEVEL 0增量备份

示例（单机）

```
BACKUP DATABASE INCREMENTAL LEVEL 0  FORMAT '/data/backup/incr_0_20211209193516';
```

示例（共享集群）

```
BACKUP DATABASE INCREMENTAL LEVEL 0 FORMAT '+DG0/backup/incr_0_20211209193516';
```

本示例中，incremental level 0 关键字表明当前为零级增量备份，format关键字指定生成备份集的文件名称。

增量备份有LEVEL 0和LEVEL 1两个级别，其中LEVEL 0作为后续所有LEVEL 1的上层基线，本质上也是一种全量备份，但是在备份概要文件中添加了与全量备份区分的物理标识。

其中，备份共享集群部署的数据库时，format关键字可以指定为普通磁盘路径或者YFS路径。

LEVEL 1增量备份

示例（单机）

```
BACKUP DATABASE INCREMENTAL LEVEL 1 FORMAT '/data/backup/incr_1_20211209193740';
```

示例（共享集群）

```
BACKUP DATABASE INCREMENTAL LEVEL 1 FORMAT '+DG0/backup/incr_1_20211209193740';
```

本示例中，incremental level 1 关键字表明当前为一级增量备份，format关键字指定生成备份集的文件名称。

增量备份只有LEVEL 0和LEVEL 1两个级别，其中LEVEL 1只备份自上次执行增量备份以后系统产生的增量数据（以上示例），相较于全量备份，增量备份数据量不大，能节省磁盘空间，恢复所需时间短。

YashanDB支持1000次连续LEVEL 1增量备份，考虑到备份集的可维护性与存储资源使用，不建议连续多次LEVEL 1增量备份。

其中，备份共享集群部署的数据库时，format关键字可以指定为普通磁盘路径或者YFS路径，但需注意的是，如果某次增量备份想与之前的备份指定不同的备份集存储（例如在level 0指定了YFS路径存储至共享存储，而level 1希望指定普通磁盘路径存储至实例所在磁盘），必须保证其与之前所有次备份的操作均是在同一个实例上发起，否则备份操作将失败。

基于tag的增量备份

LEVEL 0

示例（单机）

```
BACKUP DATABASE INCREMENTAL LEVEL 0  FORMAT '/data/backup/base_incr_0_20211209193516' TAG 'base_incr_0' INDEPEND ;
```

本示例中，incremental level 0 关键字表明当前为零级增量备份，format关键字指定生成备份集的文件名称为"/YashanDB/backup/base_incr_0_20211209193516"，INDEPEND关键字指定为独立的增量备份链，后续基于tag的增量备份必须为独立的增量备份链，指定的tag关键字是该备份集的唯一标识，后续可指定该tag作为基线做增量备份。

LEVEL 1

示例（单机）

```
BACKUP DATABASE INCREMENTAL LEVEL 1  FORMAT '/data/backup/base_incr_1_20211209193516' TAG 'base_incr_1' BASE ON 'base_incr_0'
;
```

本示例中，incremental level 1 关键字表明当前为一级增量备份，format关键字指定生成备份集的文件名称为"/YashanDB/backup/base_incr_1_20211209193516"，指定的tag关键字是该备份集的唯一标识，后续可指定该tag作为基线做增量备份。BASE ON 可以指定独立增量备份集的tag作为基线，在该基线的基础上做增量备份。实际表现与普通增量备份一致，仅仅是通过指定基线tag来生成不同的增量备份链。

> **Note**：
> 若指定的基线为level 0，其备份集生成时必须指定INDEPEND关键字。 若指定的基线为level 1备份集，该备份集也必须是指定BASE ON关键字生成的。 如果源库执行过RESTORE DATABASE 重新建库，必须重建基于tag增量备份的基线，即重新生成新的增量备份链。

备份压缩

示例（单机）

```
BACKUP DATABASE COMPRESSION ALGORITHM  ZSTD LOW FORMAT '/data/backup/full_compress' TAG 'yas_full_compress';
```

示例（共享集群）

```
BACKUP DATABASE COMPRESSION ALGORITHM  ZSTD LOW FORMAT '+DG0/backup/full_compress' TAG 'yas_full_compress';
```

本示例中，COMPRESSION关键字表明对源数据进行压缩备份，ALGORITHM关键字指定压缩算法，压缩算法可选ZSTD或者LZ4。LOW表示该种算法的压缩级别，有LOW、MEDIUM、HIGH 三种可选项。压缩也可以单独指定COMPRESSION参数，其余参数可以省略，即会使用默认的压缩算法ZSTD和压缩级别LOW。

备份加密

示例（单机）

```
BACKUP DATABASE ENCRYPTION AES256 IDENTIFIED BY yas2022 FORMAT '/data/backup/full_encryption' TAG 'yas_full_encryption';
```

示例（共享集群）

```
BACKUP DATABASE ENCRYPTION AES256 IDENTIFIED BY yas2022 FORMAT '+DG0/backup/full_encryption' TAG 'yas_full_encryption';
```

本示例中，ENCRYPTION关键字表明对源数据进行加密备份，AES256关键字表示加密算法，缺省值为AES128加密算法。其中加密算法可选AES128、AES192、AES256和SM4。IDENTIFIED BY关键字为指定加密备份集使用的密码。

常见问题

- 备份必须运行在归档模式下，否则会触发YAS-02079报错。
- 第一次增量备份必须为LEVEL 0备份，否则会触发YAS-02507错误。
- 如指定了备份文件保存目录，必须保证该目录为空，否则会触发YAS-00318错误。
- 确保备份文件保存目录的磁盘空间充足，否则会触发YAS-00301错误。
- 备份并行度取值范围为[0-8]，否则会触发YAS-04204错误。
- 超过1000次连续LEVEL 1增量备份，会触发YAS-02508错误。
- 主备在最大保护模式下，如果备库断连，主库事务无法提交，继续在主库执行备份会偶现备份卡住问题。将主库事务切换为最大可用模式即可继续执行备份。
- 备份不允许和表空间、redo文件等数据文件的增删、resize操作并发，需要等待任一操作完成之后才可执行。

恢复

操作说明

- 使用本方式执行恢复时，要求当前数据库实例运行于NOMOUNT状态。
- 共享集群部署中，只能在主集群上执行恢复操作（即其角色必须为MASTER_ROLE，可查询视图V$INSTANCE的INSTANCE_ROLE字段可获得集群中每个实例的角色属性）。
- 恢复命令仅SYS用户才可执行，恢复语法详细说明请参考开发手册RESTORE DATABASE和RECOVER DATABASE。

恢复

全量恢复

示例

```
-- 默认并发度为2，共享集群可指定YFS路径。
RESTORE DATABASE FROM '/YashanDB/backup/incr_1_20211209193740';

-- 指定并发度为6。
RESTORE DATABASE FROM '/YashanDB/backup/incr_1_20211209193740' PARALLELISM 6;

-- 成功恢复备份集后，若归档日志和在线日志连续且完整，YashanDB将自动应用所有日志，即全量恢复。
RECOVER DATABASE;

-- 打开数据库
ALTER DATABASE OPEN;
```

全量恢复属于完全恢复，成功后可以直接打开当前数据库，对于共享集群需要依次启动其他备集群（查看共享集群启停）。

恢复备份集的并发度默认设置为2，当备份集数据量大或希望提升恢复效率时，可增大并发度，并发度范围为`[1-8]`。

如果备份集是加密生成的备份集，在恢复时还需要指定解密密码。

示例

```
-- 对加密的备份集恢复指定DECRYPTION关键字，并且密码与加密时一致
RESTORE DATABASE DECRYPTION yas2022 FROM '/YashanDB/backup/full_encryption';
```

基于tag增量备份集的连续恢复

在执行增量备份集恢复时需注意如下事项：

- 假如零级增量备份指定tag为incr_0，普通一级增量备份指定tag为incr_1_1，incr_1_2...（按顺序发起备份）：
  - 当指定tag为incr_1_2的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份以及tag为incr_1_1的普通一级增量备份备份文件保存目录存在且完整，否则将会触发 YAS-00313 错误。
- 假如零级增量备份指定tag为incr_0，累积一级增量备份指定tag为incr_1_1，incr_1_2...（按顺序发起备份）：
  - 当指定tag为incr_1_2的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份备份文件保存目录存在且完整，否则将会触发 YAS-00313 错误。
- 假如零级增量备份指定tag为incr_0，累积一级增量备份指定tag为incr_1_1，普通一级增量备份指定tag为incr_1_2，累积一级增量备份指定tag为incr_1_3，普通一级增量备份指定tag为incr_1_4...（按顺序发起备份）：
  - 当指定tag为incr_1_4的一级增量备份恢复时，必须保证tag为incr_0、incr_1_3的增量备份备份文件保存目录存在且完整，否则将会触发 YAS-00313 错误。
  - 当指定tag为incr_1_3的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份备份文件保存目录存在且完整，否则将会触发 YAS-00313 错误。

> **Note**：
> 相比较原始的备份集恢复，基于tag的增量备份集恢复新增两个语法关键字：INCREMENTAL和NOREDO，INCREMENTAL可单独出现，NOREDO必须和INCREMENTAL成对出现。
> - INCREMENTAL：指定该关键字，恢复完成之后数据库处于nomount状态，若要拉起数据库，需要先将数据库拉起至mount状态，再执行recover、open。

- NOREDO：指定该关键字，数据库处于不完整恢复，恢复完成之后数据库处于nomount状态，不恢复备份集中的redo和归档文件，该数据库不可拉起正常使用，若在最后一次恢复增量备份集后要拉起数据库正常使用，不可使用该参数。 如果连续增量备份集中包含slice文件，连续restore必须严格按照备份生成顺序restore，若未遵守该规则，可能导致slice文件丢失。 如果restore失败使用同一个备份集重复restore，可能会因为已存在slice文件导致restore失败，这种情况下只能通过手动清除对应slice文件后继续执行restore。

示例

基于tag的增量备份集恢复首次必须恢复LEVEL 0的备份集，指定NOREDO参数可加快恢复速度。

```
RESTORE DATABASE INCREMENTAL NOREDO FROM '/YashanDB/backup/incr_0_20211209193740';
```

Note：
在此基线恢复成功的基础上，可连续恢复同一数据库的增量备份集。

```
RESTORE DATABASE INCREMENTAL FROM '/YashanDB/backup/incr_1_20211209193740';
```

Note：
最后一次恢复时不可指定NOREDO参数。

```
ALTER DATABASE MOUNT;

RECOVER DATABASE;

ALTER DATABASE OPEN;
```

Note：
执行RECOVER DATABASE完成恢复后不可再执行增量备份集恢复。

指定时间点恢复

本功能只适用于单机部署的数据库恢复。

示例

```
RESTORE DATABASE FROM '/YashanDB/backup/incr_0_20211209193516';

--成功恢复备份集后，指定日志序号时间点进行恢复
RECOVER DATABASE UNTIL TIME TO_DATE('2021-12-09 19:35:55','yyyy-mm-dd hh24:mi:ss');

--可按SCN进行指定时间点恢复，"25847702023708672O"为用户在数据库发生故障前记录下的SCN值
RECOVER DATABASE until SCN 258477020237086720;

--重置redo时间线并打开数据库
ALTER DATABASE OPEN RESETLOGS;
```

本示例执行指定时间点恢复，属于不完全恢复，成功后需重置redo时间线。

until time的时间格式可以为：

- `'2021-12-09 19:35:55'`：此格式需要与当前会话的DATE_FORMAT参数所指定格式匹配。
- `TO_DATE('2021-12-09 19:35:55','yyyy-mm-dd hh24:mi:ss')`：使用TO_DATE函数转换。
- `TO_TIMESTAMP('2021-12-09 19:35:55','yyyy-mm-dd hh24:mi:ss')`：使用TO_TIMESTAMP函数转换。

SCN值可通过动态视图获取：

```
SELECT CURRENT_SCN FROM V$DATABASE;
```

Note：
当LSC表在备份集后生成了不可变数据文件时，在以下场景可能导致恢复后该LSC表存在数据不一致现象：

> - 数据库原来的不可变数据文件全部存在，但是使用老的时间点恢复，此时该LSC表会多出部分不可变数据文件。
> - 数据库原来的不可变数据文件被清理，使用基于时间点的恢复，LSC表可能丢失存储在不可变数据文件中的数据。

使用恢复场景

场景1：数据库无法启动

可能发生原因：

- 控制文件损坏或者丢失
- 系统关键数据文件损坏或者丢失
- 在线日志文件损坏或者丢失

恢复方式： 建议采取全量恢复。

场景2：表误操作

可能发生原因：

- delete或者update误操作导致表数据被删除或者修改
- truncate导致表数据被清空
- drop导致表被删除

恢复方式： 1.若能及时发现误操作导致数据丢失，建议采取闪回的方式恢复表数据（参考UNDO_RETENTION/UNDO_FORCE_RETENTION具体含义）。

- delete或者update误操作，若undo被覆盖，则恢复不一定成功。闪回指令为：

```
FLASHBACK TABLE {table_name} TO SCN {SCN};
FLASHBACK TABLE {table_name} TO TIMESTAMP {TIMESTAMP};
```

- truncate误操作，需确保回收站开启，回收站被覆盖，则恢复不一定成功。闪回指令为：

```
FLASHBACK TABLE TO BEFORE TRUNCATE;
```

- drop误操作，需确保回收站开启，回收站被覆盖，则恢复不一定成功。闪回指令为：

```
FLASHBACK TABLE TO BEFORE DROP
```

2.若无法使用闪回的方式恢复，建议采取指定时间点恢复，恢复到表误操作之前的时间点。

场景3：主备模式下，主库崩溃时备库自动提升为主库，搭建新备库

可能发生原因：

- 主库操作系统崩溃
- 数据库关键文件被误删除
- 网络原因导致无法连接至主库

恢复方式： 若数据量大，建议采用全量恢复的方式重建备库；否则可以采用build database方式重建备库。

场景4：数据迁移

可能发生原因：

- 主库系统崩溃，需要进行异机恢复
- 数据库克隆

恢复方式： 全量恢复和指定时间点恢复都能完成数据库恢复，需按照具体要求选择恢复方式。

常见问题

- 恢复到与备份集不同的数据库文件或日志文件路径（例如HA架构中的主备复制）时，需要先设置DB_FILE_NAME_CONVERT和REDO_FILE_NAME_CONVERT两个配置参数指定文件路径转换。
- 单机部署中，若$YASDB_DATA/dbfiles目录下存在数据库文件，执行恢复将触发YAS-00311错误；共享集群部署中则要求+DG0目录（安装时默认的第一个磁盘组名称，调整为其他值时此处相应修改）下不存在数据库文件，否则将触发YAS-00311错误。

yasrman工具备份恢复

yasrman工具方式的备份恢复操作适用于所有架构，可用于从远程工具侧发起备份恢复。

注意事项

（一）使用本方式执行数据库备份时，要求：

1. 数据库集群运行于OPEN状态，所有节点在线（如果有部分备节点异常，只要不影响对应的主节点事务执行，则不影响备份）。
2. 所有节点均开启归档模式。
3. yasrman工具对应的catalog文件已经创建。
4. 备份命令仅SYS超级用户或拥有SYSDBA、SYSBACKUP权限的用户才可执行。恢复命令仅SYS用户才可执行。

（二）使用本方式执行数据库恢复时，要求：

1. 所有节点为NOMOUNT状态。
2. 所有节点的残留文件已经清理。
3. 数据库集群的部署状态和备份时保持一致，如节点个数，节点监听地址，路径等信息没有发生变化。

（三）使用本方式执行归档日志恢复时，要求：

- 数据库执行完RESTORE命令且未执行RECOVER的MOUNT状态。

（四）在执行增量备份集恢复时需注意如下事项：

- 假如零级增量备份指定tag为incr_0，普通一级增量备份指定tag为incr_1_1，incr_1_2...（按顺序发起备份）：
  - 当指定tag为incr_1_2的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份以及tag为incr_1_1的普通一级增量备份备份文件保存目录存在且完整，否则将触发 YAS-00313 错误。
- 假如零级增量备份指定tag为incr_0，累积一级增量备份指定tag为incr_1_1，incr_1_2...（按顺序发起备份）：
  - 当指定tag为incr_1_2的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份备份文件保存目录存在且完整，否则将触发 YAS-00313 错误。
- 假如零级增量备份指定tag为incr_0，累积一级增量备份指定tag为incr_1_1，普通一级增量备份指定tag为incr_1_2，累积一级增量备份指定tag为incr_1_3，普通一级增量备份指定tag为incr_1_4...（按顺序发起备份）：
  - 当指定tag为incr_1_4的一级增量备份恢复时，必须保证tag为incr_0、incr_1_3的增量备份备份文件保存目录存在且完整，否则将触发 YAS-00313 错误。
  - 当指定tag为incr_1_3的一级增量备份恢复时，必须保证tag为incr_0的零级增量备份备份文件保存目录存在且完整，否则将触发 YAS-00313 错误。
- HA模式下，由于每个复制组的备份集只保存在主节点，所以同一个复制组内，需要增量备份集及其依赖的备份集，都在同一个节点上。

操作示例

见工具手册yasrman。

备份

全量备份

示例

```
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP DATABASE TAG 'full_backup' FULL FORMAT 'full_001' PARALLELISM 3 DEST SERVER"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP CLUSTER TAG 'full_backup' FULL FORMAT 'full_001' PARALLELISM 3"
-D /home/yashan/catalog
```

对本示例中出现信息解释如下：

- "sys/sys@127.0.0.1:1688"是连接分布式部署中的CN节点或单机部署中的主库的用户名、密码和地址。
- full关键字表明当前为全量备份。
- format关键字指定生成备份集的文件名称为"$YASDB_DATA/backup/full_001"。由于分布式备份是对所有节点的备份，每个节点（除了备节点）都会生成各自的备份集，因此format建议用相对路径，使备份集保存在各节点的"$YASDB_DATA/backup"目录下；如使用绝对路径需保证所有节点不在同一个服务器，否则可能会报文件已存在的错误。
- tag关键字指定备份集的标识为"full_backup"。
- parallelism关键字指定备份任务以3个线程的并行度执行。
- -D指定catalog所在文件路径。
- DEST关键字指定备份集备份在服务器端还是工具端，server即为指定备份在服务器端，client指定备份在工具端，缺省为server。分布式备份不可指定为client。

增量备份

LEVEL 0 增量备份

示例

```
##单机
$ yasrman sys/password@127.0.0.1:1688
-c "BACKUP DATABASE TAG 'incr_0_backup' INCREMENTAL LEVEL 0 FORMAT 'incr_0_002'"
-D /home/yashan/catalog

##分布式
$ yasrman sys/password@127.0.0.1:1688
-c "BACKUP CLUSTER TAG 'incr_0_backup' INCREMENTAL LEVEL 0 FORMAT 'incr_0_002'"
-D /home/yashan/catalog
```

本示例中，incremental level 0 关键字表明当前为零级增量备份，format关键字指定生成备份集的文件名称为"$YASDB_DATA/backup/incr_0_002"。

增量备份有LEVEL 0和LEVEL 1两个级别，其中LEVEL 0作为后续所有LEVEL 1的上层基线，本质上也是一种全量备份，但是在备份概要文件中添加了与全量备份区分的物理标识。

LEVEL 1 增量备份

示例

```
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP DATABASE TAG 'incr_1_backup' INCREMENTAL LEVEL 1 FORMAT 'incr_1_002'"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
```

```
-c "BACKUP CLUSTER TAG 'incr_1_backup' INCREMENTAL LEVEL 1 FORMAT 'incr_1_002'"
-D /home/yashan/catalog
```

本示例中，incremental level 1 关键字表明当前为一级增量备份，format关键字指定生成备份集的文件名称为"$YASDB_DATA/backup/incr_1_002"。

增量备份只有LEVEL 0和LEVEL 1两个级别，其中LEVEL 1只备份自上次执行增量备份以后系统产生的增量数据（以上示例），相较于全量备份，增量备份数据量不大，能节省磁盘空间，恢复所需时间短。

YashanDB支持1000次连续LEVEL 1增量备份，考虑到备份集的可维护性与存储资源使用，不建议连续多次LEVEL 1增量备份。

在零级增量备份指定了DEST备份方向之后，后续依赖于该零级增量备份的一级增量备份都需指定和零级增量备份一致的备份方向，否则会导致备份集无法正常恢复。

备份压缩

示例

```
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP DATABASE TAG 'full_compress' COMPRESSION ALGORITHM ZSTD LOW"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP CLUSTER TAG 'full_compress' COMPRESSION ALGORITHM ZSTD LOW"
-D /home/yashan/catalog
```

本示例中，COMPRESSION关键字表明对源数据进行压缩备份，ALGORITHM关键字指定压缩算法，压缩算法可选ZSTD或者LZ4。LOW表示该种算法的压缩级别，有LOW、MEDIUM、HIGH 三种可选项。压缩也可以单独指定COMPRESSION参数，其余参数可以省略，即会使用默认的压缩算法ZSTD和压缩级别LOW。 FORMAT可以省略，默认会在"$YASDB_DATA/backup"下生成默认的备份集名称。

备份加密

示例

```
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP DATABASE TAG 'full_encryption' ENCRYPTION AES256 IDENTIFIED BY yas2022"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
-c "BACKUP CLUSTER TAG 'full_encryption' ENCRYPTION AES256 IDENTIFIED BY yas2022"
-D /home/yashan/catalog
```

本示例中，ENCRYPTION关键字表明对源数据进行加密备份，AES256关键字表示加密算法，缺省值为AES128加密算法。其中加密算法可选AES128、AES192、AES256和SM4。IDENTIFIED BY关键字为指定加密备份集使用的密码。

备份查询

```
# 显示指定TAG的备份集信息
$ yasrman sys/sys@127.0.0.1:1688
-c "LIST BACKUP TAG 'full_encryption'"
-D /home/yashan/catalog

# 显示所有的备份集信息
$ yasrman sys/sys@127.0.0.1:1688
-c "LIST BACKUP"
-D /home/yashan/catalog
```

本示例中，将打印备份集的元信息，包括各节点的 地址、TAG名称以及备份集路径等 。

备份删除

```
$ yasrman sys/sys@127.0.0.1:1688
-c "DELETE BACKUPSET TAG 'full_encryption'"
-D /home/yashan/catalog
```

本示例中，将删除TAG为"full_encryption"的备份集，前提是目标主库为OPEN状态。

说明

- 分布式备份，只能在主库上执行备份，备份集会生成在主库节点，**如果发生了主备切换，建议重新执行LEVEL 0的增量备份**，否则恢复增量备份集的时候，会报找不到备份集的错误。
- 备份语法详细说明请参考工具手册yasrman。
- 备份会消耗一定的系统和数据库资源，请根据业务负载酌情选择备份窗口。
- 跟踪各个节点的$YASDB_DATA/log/run/run.log运行日志，可查看该节点备份恢复过程的详细日志。
- 增量备份的多个备份集可采用不同的压缩算法。
- 增量备份的多个备份集必须采用一致的加密策略（都加密或都不加密，都加密时密码相同），但可以采用不同的加密算法。
- 删除备份集如果中途失败，某些可能会残留备份数据，需要在分布式集群状态OPEN的状态下，重新执行删除。
- 分布式生成备份集之后，若后续要使用该备份集做恢复，必须保证待恢复分布式集群和备份前的节点部署状态完全一致。该备份集生成时分布式集群的节点状态可使用YASRMAN工具LIST命令查看，参考yasrman。

常见问题

- 备份必须运行在归档模式下，否则会触发 YAS-02079 报错。
- 第一次增量备份必须为LEVEL 0备份，否则会触发 YAS-02507 错误。
- 如指定了备份文件保存目录，必须保证该目录为空，否则会触发 YAS-00318 错误。
- 确保备份文件保存目录的磁盘空间充足，否则会触发 YAS-00301 错误。
- 备份并行度只能在`[0-8]`范围内选择，否则会触发 YAS-04204 错误。
- 超过1000次连续LEVEL 1增量备份，会触发 YAS-02508 错误。
- 备份不允许和表空间、redo文件等数据文件的增删、resize操作并发，需要等待任一操作完成之后才可执行。

恢复

分布式环境清理

在对分布式集群执行恢复前，请使用yasboot工具清理环境并重启分布式节点到NOMOUNT状态。

示例

```
$ yasboot cluster clean -c yashan --restore --with-arch --force
```

全量恢复

示例

```
# 以默认2个并行度执行恢复
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "RESTORE DATABASE FROM TAG 'incr_2'"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
-c "RESTORE CLUSTER FROM TAG 'incr_2'"
-D /home/yashan/catalog
```

本示例执行全量恢复，属于完全恢复，其中redo回放和备库build等操作都集成在该命令内部。成功后CN/DN/MN所有节点都将处于OPEN状态。相比使用SQL命令的恢复操作，使用yasrman工具少执行了recover和open的语句操作。

恢复备份集的并发度默认设置为2，当备份集数据量大或希望提升恢复效率时，可酌情加大并发度，该值最大可设置为8。

如果备份集是加密生成的备份集，在恢复时还需要指定解密密码。

```
##单机
$ yasrman sys/sys@127.0.0.1:1688
-c "RESTORE DATABASE DECRYPTION yas2022 FROM TAG 'full_encryption'"
-D /home/yashan/catalog

##分布式
$ yasrman sys/sys@127.0.0.1:1688
-c "RESTORE CLUSTER DECRYPTION yas2022 FROM TAG 'full_encryption'"
-D /home/yashan/catalog
```

本示例中，在对加密的备份集恢复时需要指定DECRYPTION关键字，并指定与加密时一致的密码，才能成功解密和恢复。

说明

- 备份语法详细说明请参考工具手册yasrman。
- 分布式部署模式下，需要保证待恢复的集群部署状态与备份前节点部署状态一致，包括节点类型、节点监听IP、节点个数、节点的部署路径。该备份集生成时分布式集群的节点状态可使用YASRMAN工具LIST命令查看，参考yasrman。

常见问题

- 若$YASDB_DATA/dbfiles目录下存在数据库文件，执行恢复将触发YAS-00311错误。
- 恢复任务的并行度值必须在`[1-8]`范围内指定，否则将触发YAS-04204错误。
- 恢复时指定的catalog文件，必须和备份时指定的为同一个文件，否则会触发YAS-02519错误。

故障诊断

YashanDB提供故障诊断架构，用于收集和管理诊断数据，从而诊断和解决数据库的问题。

故障诊断架构有助于预防、检测、诊断和解决问题。发生严重错误时，将为其分配一个事件编号，并立即捕获该错误的诊断数据并使用该编号进行标记，然后将诊断数据存储在自动诊断存储库中，以后可以根据事件编号检索并进行分析。

故障诊断架构的目标如下：

- 首次故障诊断
- 故障预防
- 检测到故障后报错，防止数据进一步损坏
- 减少故障诊断的时间
- 减少故障解决的时间

实现这些目标的关键技术如下：

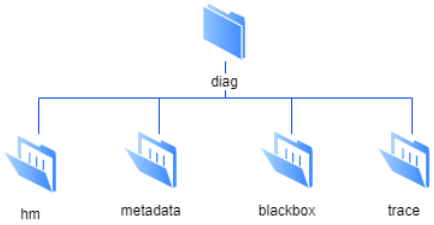
- **自动捕获数据——黑匣子**：进程出现故障宕机前，收集进程运行堆栈等信息，存储在自动诊断存储库中。这种主动诊断数据类似于飞机“黑匣子”飞行记录仪收集的数据。
- **健康检查**：检测到严重错误后，数据库会触发运行一个或多个健康检查，以对严重错误执行更深入的分析，健康检查的结果存储到自动诊断存储库中。单个健康检查会检查数据块损坏、redo坏块等。DBA可以手动调用这些健康检查，定期检查或者是根据需要调用。
- **事件警报**：对于严重的错误，会在第一时间收集诊断数据，分配事件编号标识，存储在自动诊断存储库中，以便问题的追踪和解决。
- **故障检测及响应**：当数据库检测到错误时，会采取一定的手段限制损坏或故障中断。例如：归档磁盘空间不足时，记录告警日志，数据库被设置为故障状态，避免用户执行业务卡住时无法感知错误。DBA需要及时的释放空间，恢复数据库状态。
- **巡检**：实时监控数据库的一些组件，检测到严重错误时，立即上报或者是自动修复，及时修复避免引起更严重的错误。例如：数据文件监控等。
- **手动捕获数据——dump**：允许用户手动执行dump命令，将系统内部结构信息转储到trace文件中，分析这些信息以便判断故障问题。

故障诊断架构

故障诊断架构由多个组件构成，包括自动诊断存储库、运行日志、告警日志等。其中运行日志、告警日志参考日志管理中说明。

自动诊断存储库

自动诊断存储库是基于文件的存储库，用于存储数据库的诊断数据。它的目录结构如下（默认放在YASDB_DATA目录下，可设置参数进行配置）：



其中，主目录中包含了诊断数据的子目录，配置参数DIAGNOSTIC_DEST用于标识主目录的位置。

子目录名称	内容
hm	存放健康检查的报告
metadata	存放自动诊断存储库的元数据文件（主要有incident、problem等）
blackbox	存放黑匣子的诊断数据
trace	存放手动dump的数据或后台自动生成的trace日志

YashanDB中，自动诊断存储库功能默认是打开的，如想关闭该功能可以配置参数DIAG_ADR_ENABLED，并重新启动数据库。

```
ALTER SYSTEM SET DIAG_ADR_ENABLED = FALSE SCOPE = SPFILE;
```

> **Note**： 目录更新后，需要拷贝其子目录及文件，如果没有拷贝文件，数据库将自动创建新的自动诊断存储库文件！ 子目录blackbox不需手动创建，在数据库进程宕机前将自动创建该路径及其中文件。

故障诊断概念

故障、问题和事件

为便于诊断和解决问题，YashanDB引入了三个概念：故障、问题和事件。

故障

故障是YashanDB运行过程中可能出现的一些严重错误，由数据库内部定义。可以通过视图V$DIAG_FAULT查看相关故障的信息。

问题

问题是数据库运行过程中真实发生的故障，相同的故障可能会记录多个问题，相同故障的问题是通过问题键值进行区分的。可以通过视图V$DIAG_PROBLEM查看数据库运行过程中出现的问题。

事件

事件是单次出现的问题，数据库运行过程中，不同的会话在不同的时间可能产生相同的问题，数据库会创建多个事件，每个事件都有唯一标识的ID用于区分，也会记录当前会话的ID。可以通过视图V$DIAG_INCIDENT查看数据库运行过程中出现的严重错误，和获取更多的诊断数据，例如创建时间、会话ID、故障详细描述等。

```
--  该视图里定义当前YashanDB的所有故障信息
SELECT * FROM V$DIAG_FAULT;

TYPE              CODE DESCRIPTION
--------- ------------ ----------------------------------------------------------------
YASF               101 cannot allocate memory
YASF               102 too many open files
YASF               103 no space left on device
YASF               104 failed to open file
YASF               105 failed to create file
YASF               106 failed to write file
YASF               107 file metadata changed, for example, permissions, timestamps, extended attributes, and user/group ID,
etc
YASF               108 file is missing
YASF               109 file is moved
YASF               110 file system is unmounted
YASF               111 failed to read file
YASF               112 failed to extend file
YASF               201 the data file is corrupted
YASF               202 the redo log file is corrupted
YASF               203 the ctrl files are corrupted
YASF               204 data block versions are mismatching
YASF               205 the database is aborted
YASF               206 the database needs to be repaired
YASF               207 deadlock detected
YASF               208 inconsistent database startup
YASF               209 watcher has an exception
YASF               210 the user limit on the total number of watches was reached
YASF               211 synchronization standby database destinations have failed
```

当问题（严重错误）多次发生时，系统将为每次问题创建一个事件。 这些事件将在加上时间戳后，存储在自动诊断存储库中。每个事件都由事件ID标识，该ID在自动诊断存储库中是唯一的。

发生一个事件时，数据库将执行如下操作：

- 收集首次失败的诊断数据。
- 使用事件ID标记事件。
- 将事件dump到为该事件创建的自动诊断存储库子目录中。

事件防洪

一个问题可能会在短时间内产生数十个或数百个事件，这将导致系统中生成过多的诊断数据，占用 自动诊断存储库中的太多空间，并可能减慢诊断和解决问题的速度。由于这些原因，故障诊断机制将在达到洪水阈值后控制事件的生成。

受到洪水控制的事件只记录告警日志，不再dump事件。洪水控制事件提供了一种通知用户正在发生严重错误的方法，但不会存在诊断数据过载的现象。

事件洪水控制的阈值水平是预先确定的，无法更改。它们定义如下：

- 在一小时内同一问题发生5个事件后，此问题的后续事件将受到洪水控制。该问题的事件的正常（非洪水控制）记录将在下一个小时再次开始。
- 在一天内同一问题发生25个事件后，此问题的后续事件将受到洪水控制。该问题的事件的正常记录将在第二天再次开始。
- 在一小时内发生同一问题的50个事件，或者在一天内发生同一问题的250个事件后，此问题的后续事件永远不再记录到自动诊断存储库中。该情况下，数据库将向告警日志中写入一条消息，指示不会再记录其他事件。

> **Note**： 事件洪水控制后不会再记录事件，但是会记录事件的防洪累计次数。 作为DBA，应时常排查是否有新的事件产生，及时定位解决问题。 事件和问题诊断数据的存储上限为1000000条，如果个数超过该上限，新的诊断数据会覆盖旧的诊断数据。

自动捕获诊断数据——“黑匣子”

YashanDB提供一种机制，在数据库进程出现故障宕机前，收集进程运行堆栈等信息，将诊断数据dump到自动诊断存储库中，这种主动诊断数据类似于飞机“黑匣子”飞行记录仪收集的数据。

> **Note**： 当收集到 “黑匣子” 诊断数据，请打包诊断数据并及时联系我们的技术支持。

健康检查

YashanDB提供健康检查框架，用于数据库运行诊断检查。

健康检查也称为检查器，检查数据库的各个组件，如检测文件损坏、数据块损坏、redo日志损坏等，并生成一份报告，记录发现的错误以及错误带来的影响。

可以通过如下两种方式运行健康检查：

- **反应式**——故障诊断架构自动运行健康检查以响应严重错误。
- **手动**——使用内置高级包手动运行健康检查。如有需要，可以通过定义JOB定期运行健康检查。

健康检查执行的结果将存储在自动诊断存储库中。 健康检查必须在数据库为OPEN模式或者MOUNT模式下运行。

健康检查项

数据库结构完整性检查

此检查验证数据库文件的完整性，在检查到数据库文件（控制文件、redo文件、数据文件、归档文件）不可访问、损坏或不一致时报告失败。

数据块完整性检查

此检查可检测磁盘映像块损坏，例如checksum失败、头尾不匹配以及块内的逻辑不一致。对于页面损坏，通常可以通过备库或者是页面修复工具来修复。

单个数据文件检查

此检查可检测数据文件可访问性，例如数据文件所有的块损坏，包含有效的块可未使用的块等。

redo完整性检查

此检查可检测数据库的一致性和所有的redo文件的头部信息，避免数据库重启后无法恢复到一致性点，数据库不可用。

单个redo文件检查

此检查可检测单个redo文件的可访问性和redo块损坏。

归档日志文件检查

此检查检测单个归档日志文件的可访问性和内容损坏。

死锁检测检查

此检查项检测用户行为上的事务死锁，在检测出死锁后会在trace目录下（按seesion id的trace文件）报告出具体的死锁环信息，即Wait-For-Graph（WFG）。

手动运行健康检查

用户可以手动调用DBMS_HM.RUN_CHECK程序运行健康检查，该程序需提供运行状况检查名称和运行的名称（自定义），如下所示：

```
BEGIN
   DBMS_HM.RUN_CHECK('DB Structure Integrity Check', 'my_run', NULL);
END;
/
```

若要获取运行状况检查名称的列表，请运行以下查询：

```
SELECT name FROM V$HM_CHECK;
```

输出的内容如下：

```
NAME
----------------------------------------------------------------
```

```
DB Structure Integrity Check
Data Block Integrity Check
Single Datafile Check
Redo Integrity Check
Redo File Check
Archived Log Check
```

大多数健康检查需要输入参数，可以通过视图 V$HM_CHECK_PARAM 查看参数名称和说明。有些参数是必须的，有些参数是可选的。可选参数输入NULL 则表示使用默认值。 以下查询显示所有健康检查的参数信息：

```
SELECT c.name check_name, p.name parameter_name, p.description
FROM V$HM_CHECK_PARAM p, V$HM_CHECK c
WHERE p.check_id = c.id ORDER BY c.name;
```

输出的内容如下：

```
CHECK_NAME                     PARAMETER_NAME         DESCRIPTION
------------------------------ ---------------------- --------------------------
Archived Log Check             ARC_SEQ_NUM            Archive sequence number
Data Block Integrity Check     BLC_BL_NUM             Block number
Data Block Integrity Check     BLC_DF_NUM             Data file number
Redo File Check                RF_NUM                 Redo file number
Single Datafile Check          DF_NUM                 Data file number
```

输入参数的名称和值在参数中传递是成对的，以分号`;`分隔。下面示例演示如何将文件ID和块ID作为参数传递数据块完整性检查：

```
BEGIN
  DBMS_HM.RUN_CHECK('Data Block Integrity Check','my_run','BLC_DF_NUM=1;BLC_BL_NUM=234');
END;
/
```

> **Note**： 详细使用信息请参考开发手册DBMS_HM的定义及参数说明。 健康检查的数据存储的上限是1000000条记录，如果超过该上线，新的诊断数据将覆盖掉最旧的诊断数据。

查看健康检查报告

健康检查运行的结果和其他信息存储在自动诊断存储库中，但不会立即生成报告。 如果需要查看报告，用户可以通过执行DBMS_HM包生成，如果该报告尚不存在，系统首先从自动诊断存储库中的检查器运行数据生成，并以TEXT格式文件存储在自动诊断存储库主目录的HM子目录下，然后向用户输出显示。如果报告文件已存在，则系统仅执行输出显示。

DBMS_HM.GET_RUN_REPORT程序用于查看健康检查的报告，默认格式为文本。示例如下：

```
SELECT DBMS_HM.GET_RUN_REPORT('HM_RUN_13') FROM DUAL;

DBMS_HM.GET_RUN_REPO
------------------------------------------------------------
 Run Name                    : hm_run_13
 Run Id                      : 13
 Check Name                  : Data Block Integrity Check
 Mode                        : MANUAL
 status                      : COMPLETED
 Start Time                  : 2022-07-22 10:21:40
 End Time                    : 2022-07-22 10:21:40
 Error Encountered           : 0
 Source Incident Id          : 0
 Number of Incidents Created : 0

Input Parameters for the Run
 BLC_DF_NUM=6
 BLC_BL_NUM=132

Run Findings And Recommendations
 Finding
```

```
Finding Name  : block corruption
Finding ID    : 12
Message       : block 132 in datafile 6: '/data/yashan/dbdata/dbfiles/hm_test' is corrupted
Message       : object might be unavailable
```

Note： 详细使用信息请参考开发手册DBMS_HM的定义及参数说明。 如果报告信息太长，系统仅输出显示部分信息，完整信息需从报告文件中获得。

健康检查视图

用户可以直接查询创建报告的自动诊断存储库数据，查看运行特定的健康检查的结果，而不是通过健康检查报告。 此数据可以通过视图V$HM_RUN和视图V$HM_FINDING获得。 下面的示例查询视图V$HM_RUN以确定检查器运行的历史记录：

```
SELECT run_id, name, check_name, run_mode FROM V$HM_RUN;

 RUN_ID NAME            CHECK_NAME                          RUN_MODE
------- --------------- ----------------------------------- ---------
      1 hm_run_1        Single Datafile Check               REACTIVE
      2 hm_run_2        Single Datafile Check               REACTIVE
      3 hm_run_3        DB Structure Integrity Check        REACTIVE
      4 hm_run_4        DB Structure Integrity Check        MANUAL
        .
        .
        .
     39 hm_run_39       Redo Integrity Check                REACTIVE
     40 hm_run_40       DB Structure Integrity Check        REACTIVE
     41 hm_run_41       Data Block Integrity Check          MANUAL
     42 hm_run_42       Single Datafile Check               MANUAL
     43 hm_run_43       DB Structure Integrity Check        REACTIVE
     44 hm_run_44       Single Datafile Check               REACTIVE
     45 hm_run_45       Redo Integrity Check                MANUAL
     46 hm_run_46       Redo File Check                     MANUAL
     47 hm_run_47       Archived Log Check                  MANUAL
     48 hm_run_48       Redo File Check                     REACTIVE
```

下面示例查询视图V$HM_FINDING，获取run_id=44（反应式单个数据文件检查）的结果详细信息：

```
SELECT finding_id, description FROM V$HM_FINDING WHERE run_id = 44;

  FINDING_ID DESCRIPTION
------------ ----------------------------------------------------------------
          16 block 1432 in datafile 6: '/data/yashan/dbdata/dbfiles/hm_test' is corrupted
          17 block 1549 in datafile 6: '/data/yashan/dbdata/dbfiles/hm_test' is corrupted
```

巡检

巡检在YashanDB中为一个单独的后台线程，该线程类似于巡逻小队，不断地监控数据库的运行状况。当发生严重错误时，收集诊断数据存储在自动诊断存储库中，并且触发相应的修复手段或者限制损坏及中断。 巡检主要包含如下内容：

- 监控数据库文件
- 发生严重错误时触发健康检查
- 监控同步备库（最大保护模式）

文件监控

YashanDB的后台文件都存储着重要的信息，部分文件丢失可能导致数据库无法正常使用。此外，用户不可以手动改动数据库的文件，如数据文件被改动可能造成各种影响，系统需要对此及时响应避免造成更大的损坏及中断，共享集群模式下该功能禁用。

文件监控的范围

控制文件、数据文件、redo文件，如果开启双写也将监控双写文件。

文件监控的异常操作

- 文件元数据的变动：文件权限、文件用户及用户组以及时间戳等。
- 文件被删除：文件直接被删除，例如：rm命令等。
- 文件名被修改或者文件被移动：文件名称被修改或者修改文件的路径，例如：mv命令等。
- 文件系统被卸载：文件所在的磁盘被卸载或者时磁盘损坏无法装载。

异常响应

检测到数据库文件存在异常时，触发事件警报，收集事件诊断数据，并且触发反应式健康检查，将诊断数据存储在自动诊断存储库中。

故障处理

出现数据文件、redo文件被删除或者被移动等操作时，系统将数据库的状态置为ABNORMAL，数据库处于只读状态，用户执行业务报错，等待DBA介入修复故障。

下面以数据文件hm_test丢失时，文件监控的诊断数据为例：

```
-- 事件警报，通过V$DIAG_INCIDENT查看，输出信息如下

SELECT incident_id, session_id, error_number, error_argument, error_comments FROM V$DIAG_INCIDENT;
 INCIDENT_ID   SESSION_ID ERROR_NUMBER ERROR_ARGUMENT    ERROR_COMMENTS
------------ ------------ ------------ ---------------- ----------------------------------------------------------------
          11           12          107 data file        [MONITOR] /data/yashan/dbdata/dbfiles/hm_test does not exist and may
have been deleted
          12           12          108 data file        [MONITOR] /data/yashan/dbdata/dbfiles/hm_test was deleted

-- 反应式健康检查，输出信息如下
SELECT run_id, name, check_name, run_mode FROM V$HM_RUN;
 RUN_ID NAME          CHECK_NAME                          RUN_MODE
------- ------------ ----------------------------------- ---------
     51 hm_run_51     DB Structure Integrity Check          REACTIVE

SELECT finding_id, description FROM V$HM_FINDING WHERE run_id = 51;
  FINDING_ID DESCRIPTION
------------ ----------------------------------------------------------------
          21 datafile /data/yashan/dbdata/dbfiles/hm_test is missing
```

数据文件丢失，数据库状态变为ABNORMAL，用户无法执行业务。

```
SELECT status FROM V$DATABASE;
STATUS
--------------------------------
ABNORMAL

-- 执行业务报错
```

```
CREATE TABLE student_no(ID INT);
YAS-06023 database is set to read-only because of database abnormal

-- 查询正常
SELECT * FROM DUAL;
DUMMY
-----
X
```

> **Note**： 数据库的状态变为ABNORMAL时，DBA需要及时处理，避免用户业务一直等待。
> 数据文件丢失，数据库有可能异常退出，需要DBA及时响应。

数据文件丢失，DBA可以考虑是否offline该文件或者对该文件所在表空间进行离线修复，消除故障状态以使用户继续执行业务，也可以使用其他的处理策略。 以 offline 该文件所在表空间为例进行修复：

```
-- 消除故障状态
ALTER DATABASE CONVERT TO NORMAL;

SELECT status FROM V$DATABASE;
STATUS
---------------------------------
NORMAL

-- offline 该文件所在表空间hm_test
ALTER TABLESPACE hm_test OFFLINE IMMEDIATE;

-- 用户继续执行业务
CREATE TABLE student_no(ID INT);
```

> **Note**： 当出现ABNORMAL时，请及时查阅告警日志或者是事件视图V$DIAG_INCIDENT查看数据库ABNORMAL的诊断数据。
> 出现ABNORMAL时，需要先修复故障再消除故障，但是offline表空间数据DDL 操作，是不可以执行的，所以需要先消除故障状态，再进行offline 操作。
> 使用offline操作时，应采用IMMEDIATE选项，避免数据库出现异常退出的状况。
> 建议将数据库重启到mount阶段后，再执行offline数据文件操作，open阶段执行该操作存在风险，故障有可能会扩散。

健康检查——反应式

反应式健康检查主要是用来排查潜在的故障。当YashanDB运行过程中出现某些故障时，会触发相应的健康检查，尽早地检测出潜在的故障，及时处理避免造成损坏。

以读取页面失败为例：

```
-- 查询数据发现页面 (block 132) 损坏
SELECT * FROM HM_TEST;
YAS-02147 the block 6-0-132 is corrupted

-- 发现页面损坏，巡检线程会触发健康检查 (Single Datafile Check) 检查该页面所在的整个文件
-- 触发反应式健康检查，输出信息如下
SELECT run_id, name, check_name, run_mode, status FROM V$HM_RUN;
RUN_ID NAME        CHECK_NAME              RUN_MODE  STATUS
------ ----------- ----------------------- --------- -----------
    61 hm_run_61   Single Datafile Check   REACTIVE  COMPLETED

-- 查看健康检查的具体诊断数据
SELECT DBMS_HM.GET_RUN_REPORT('HM_RUN_61') FROM DUAL;
DBMS_HM.GET_RUN_REPO
------------------------------------------------------------------
 Run Name                    : hm_run_61
 Run Id                      : 61
 Check Name                  : Single Datafile Check
 Mode                        : REACTIVE
 status                      : COMPLETED
 Start Time                  : 2022-07-22 11:51:16
 End Time                    : 2022-07-22 11:51:16
 Error Encountered           : 0
```

```
 Source Incident Id          : 0
 Number of Incidents Created : 0

Input Parameters for the Run
 DF_NUM=6

Run Findings And Recommendations
 Finding
 Finding Name  : block corruption
 Finding ID    : 1
 Message       : block 132 in datafile 6: '/data/yashan/dbdata/dbfiles/hm_test' is corrupted
 Message       : object might be unavailable
 Finding
 Finding Name  : block corruption
 Finding ID    : 2
 Message       : block 152 in datafile 6: '/data/yashan/dbdata/dbfiles/hm_test' is corrupted
 Message       : object might be unavailable
```

从以上示例可以发现：当执行业务发现页面损坏时，会触发单个文件的健康检查，不仅检测到页面（block 132）损坏，还发现了潜在的页面（block 152）损坏。

监控同步备库（最大保护模式）

YashanDB的高可用架构支持三种保护模式，最大保护模式为其中一种，在最大保护模式下，同步备库与主库断连或者同步备库不可用时，主库用户的业务提交会卡住。

巡检线程检测到同步备库长时间不可用或者是断连状态时，会将数据库的状态设置为ABNORMAL，后续的业务将不再执行，如果正在提交或者是已经提交的业务会一直卡住，等待DBA介入修复故障。

处理该故障的推荐方案如下：

- 查看故障备库是否可以重新启动或者修复，如果可以，请尽快启动或者修复。
- 根据需要，可以考虑修改配置参数 QUORUM_SYNC_STANDBYS 和 REQUIRED_SYNC_STANDBYS。
- 根据需要，可以考虑修改主库的保护模式。

出现该故障的处理流程：

1.执行业务报错，数据库状态为 ABNORMAL。

```
SELECT status FROM V$DATABASE;
STATUS
----------------------------------
ABNORMAL
```

2.查阅视图V$DIAG_INCIDENT或告警日志。

```
SELECT incident_id, session_id, error_number, error_comments FROM V$DIAG_INCIDENT;

INCIDENT_ID   SESSION_ID ERROR_NUMBER ERROR_COMMENTS
----------- ------------ ------------ --------------------------------------------------------------------------
         22           12          211 synchronization standby database destinations have failed, database is set to read-only
```

3.查看视图V$ARCHIVE_DEST_STATUS。

```
-- 存在一个断连备库
SELECT dest_id, connection, status, database_mode FROM V$ARCHIVE_DEST_STATUS;
DEST_ID CONNECTION        STATUS           DATABASE_MODE
------- ----------------- ---------------- -----------------
      2 CONNECTED         NORMAL           OPEN
      3 DISCONNECTED      UNKOWN           UNKOWN
```

4.选取修复方案（以重新启动备库为例）。

```
SELECT dest_id, connection, status, database_mode FROM V$ARCHIVE_DEST_STATUS;
DEST_ID CONNECTION        STATUS           DATABASE_MODE
------- ----------------- ---------------- -----------------
      2 CONNECTED         NORMAL           OPEN
      3 CONNECTED         NORMAL           OPEN
```

5.DBA 消除故障状态，提交卡住的用户业务提交成功，用户可以继续执行业务。

```
ALTER DATABASE CONVERT TO NORMAL;

SELECT status FROM V$DATABASE;
STATUS
--------------------------------
NORMAL
```

Note：

有关最大保护模式的详细信息请查阅高可用相关的文档。

同步备库信息可以查看配置参数 QUORUM_SYNC_STANDBYS 和 REQUIRED_SYNC_STANDBYS 获取更多的信息。

如果故障未修复，直接执行 ALTER DATABASE CONVERT TO NORMAL 消除故障状态的话，数据库会重新将数据库设置为ABNORMAL状态，再次记录告警日志和收集诊断数据。

有些故障修复之后数据库会自动消除ABNORMAL 状态。例如：

- 归档空间不足，数据库状态被置为ABNORMAL，DBA 释放磁盘空间，数据库会自动消除ABNORMAL状态，用户可以继续执行业务。
- 最大保护模式下，同步备库长时间异常，数据库状态被置为 ABNORMAL， DBA 重启同步备库或者是其他的修复方案，数据库检测到修复完成，会自动消除ABNORMAL 状态，无需手动消除，用户可以继续执行业务。

dump

YashanDB提供dump命令，让用户手动将系统内部结构信息转储到trace文件中，这些信息可被用于进行故障问题的跟踪和分析。

dump命令

dump命令通过SQL语句实现，对其详细的语法描述请参考开发手册ALTER SYSTEM的dump_clause子句。

YashanDB允许将如下内部信息dump到trace文件中：

- **private redo in memory**

内存中的私有日志数据，对应的dump命令为：

```
ALTER SYSTEM DUMP PRIVATE LOG;
```

- **logfile blocks**

磁盘中的redo日志数据，对应的dump命令为：

```
ALTER SYSTEM DUMP LOGFILE 'redo1';
```

- **datafile blocks**

磁盘中的数据文件数据，对应的dump命令为：

```
ALTER SYSTEM DUMP DATAFILE 6;  --DUMP整个数据文件

ALTER SYSTEM DUMP DATAFILE 6 BLOCK 0; --DUMP一个页面

ALTER SYSTEM DUMP DATAFILE 6 MINBLOCK 128 MAXBLOCK 137;  --DUMP一批页面
```

- **stack of this session**

某个会话的堆栈数据，对应的dump命令为：

```
ALTER SYSTEM DUMP SESSION 20 BACKTRACE;
```

> **Note**：
> - dump操作转储的是某一时间点的内部数据，多次dump可以跟踪变化。
> - dump操作涉及磁盘写，运行时间势必受数据量和IO影响。

trace文件

trace文件与会话一一对应，在会话首次执行dump时创建，之后每一次dump操作在文件里生成一段数据。

文件路径

trace文件为自动诊断存储库的一部分，默认存放在{YASDB_DATA}/diag/trace中，文件名称为{dbname}*yas*{sid}.trc。

文件内容

每一次dump操作对应文件中的一段内容，每一段内容由标题和数据组成：

- 标题：包含dump时间、dump类型、dump数据大小等信息。
- 数据：根据不同的dump类型，获取到的系统内部结构数据。

> **Note**：
> - 由于会话ID可以重复使用，当相同ID的新会话第一次dump，而该ID对应trace文件存在时，不会创建新文件，而是继续追加写入。

- 死锁发生时，后台线程若检测到死锁，数据库会选择死锁环中的一个session返回error，并且dump对应的死锁信息到该session对应的trace文件，死锁信息不可手动通过SQL语句实现，在检测到死锁时自动触发。
- trace文件不会自动清理，管理员需关注其占用空间，判断不再使用的trace文件手动清理。
- trace文件里的数据为YashanDB运行时产生的内部结构数据，如在故障跟踪时希望获得对这些数据的解析，请咨询我们的技术支持。

故障状态

YashanDB检测到异常故障时，防止扩散影响，会将数据库的状态置为ABNORMAL，数据库处于故障只读状态，可以查询，不能执行写的业务。

当数据库为ABNORMAL状态时，可以查看V$DIAG_INCIDENT视图或告警日志明确故障原因。

```
--故障发生时，数据库状态为ABNORMAL
SELECT STATUS FROM V$DATABASE;

STATUS
---------------------------------
ABNORMAL

1 row fetched.

--故障修复后，执行如下语句消除ABNORMAL状态
ALTER DATABASE CONVERT TO NORMAL;


SELECT STATUS FROM V$DATABASE;

STATUS
---------------------------------
NORMAL
```

下表列示数据库状态被置为ABNORMAL的常见故障及修复建议。

故障类型	说明	修复建议
归档日志磁盘空间不足	归档空间不足会导致业务卡住，设置为ABNORMAL，避免大量业务连接再进来，等待DBA释放空间	故障修复后，手动消除状态或等数据库检测到修复后会自动清除 * 增加磁盘空间或释放空间 *归档清理，删除无用的归档，参考归档管理
最大保护模式，同步备库异常	最大保护模式下，同步备库长时间异常会导致主库的业务提交会卡住	故障修复后，手动修复或等数据库检测到修复后会清除故障 * 排查原因，修复同步备 * 修改同步备配置 * 调整保护模式
数据文件或redo文件被手动操作，（例如：rm或mv等）	数据文件被异常操作，可能会导致数据库不可用	故障修复后，需要手动消除。
数据库发生Fatal错误	数据库可能因为资源、IO等原因出现故障，导致数据库无法运行，具体的错误原因查看V$DIAG_INCIDENT视图或运行日志明确故障原因。	该故障状态不可消除，数据库已经不能再继续运行了，只能shutdown abort，修复故障后重启数据库。

> **Note**： 有些故障未修复，直接消除ABNORMAL状态，数据库会检测到还是存在故障，会重新置为ABNORMAL状态，重新上报事件。例如：归档磁盘空间不足、最大保护模式下同步备异常等。
> 数据文件被手动操作，手动消除ABNORMAL状态后，不会再重新置为ABNORMAL，所以需要保证故障被修复。