YashanDB 快速上手

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

安装前准备

本章节将介绍个人版单机形态YashanDB服务端安装部署所需的前期准备，安装前请根据本文所述内容进行相关配置。

演示环境如下：

操作系统	CPU	内存
CentOS Linux	x86_64	32G

服务器准备

下表为个人开发环境最小配置，请根据如下配置自行调整软硬件配置。

项目	描述
操作系统	CentOS 7.6，kylin V10
CPU	X86_64，ARM64 2C
内存	4G
硬盘	50G
文件系统	ext4或xfs
网络	千兆以太网，支持TCP和UDP链接

创建安装用户

建议创建一个新用户安装YashanDB数据库。

1. 切换至root用户，并执行如下命令创建新用户yashan：

```
$ su root
# useradd yashan
```

2. 配置sudo免密。

   首先，请打开/etc/sudoers文件，通常情况下，即使root用户都无该文件的编辑权限，此时需要先对root授权。

```
# cd /etc
# ll sudoers
# chmod +w sudoers
# vi /etc/sudoers
```

   在文件的最后添加如下内容后保存退出：

```
yashan  ALL=(ALL)NOPASSWD:ALL
```

   最后，如该文件初始为只读，恢复其属性：

```
# chmod -w sudoers
```

3. 将yashan用户加入到YASDBA用户组。

```
# groupadd YASDBA
# usermod -a -G YASDBA yashan
```

4. 执行如下命令为用户yashan指定密码：

```
# passwd yashan

Changing password for user yashan.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

操作系统参数调整

下表为YashanDB数据库所需的资源限制值的最小要求，请根据下表所示将资源限制值调整为大于或等于最小要求的值。

资源项	描述	推荐值
open files	文件句柄	1048576
max user processes	最大用户线程数	1048576
max memory size	最大内存限制	unlimited
stack size	堆栈大小	8192

执行如下命令查看系统的所有资源限制值：

```
# ulimit -a

core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 127952
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1048576
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 65535
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

操作系统参数调整有如下两种方式，请根据自身需求选择其一进行配置：

- 配置参数临时生效

  执行如下命令使新配置的资源限制值临时生效，重启后无效：

  ```
  # ulimit -n 1048576
  # ulimit -u 1048576
  # ulimit -m unlimited
  # ulimit -s 8192
  ```

- 配置参数永久生效

  执行如下命令将参数写入`/etc/security/limits.conf`文件，重启后参数永久生效：

```
# echo "
 * soft nofile 1048576
 * hard nofile 1048576
 * soft nproc 1048576
 * hard nproc 1048576
 * soft rss unlimited
 * hard rss unlimited
 * soft stack 8192
 * hard stack 8192
" >> /etc/security/limits.conf
```

软件包准备

请联系我们的技术支持获取YashanDB数据库软件包，软件包名称示例：`yashandb-personal-xx.xx-linux-x86_64.tar.gz`。

YashanDB服务端安装

本章节将介绍Linux环境下单机形态的YashanDB数据库服务端安装部署方式，请确保已根据安装前准备章节进行相关配置，本文以`yashandb-personal-23.2.0.2-linux-x86_64.tar.gz`软件包为例进行阐述 。

如需安装其他部署形态的YashanDB数据库，请参考安装部署章节。

创建安装目录

1. 执行如下命令切换至yashan用户：

```
# su yashan
$ cd
```

2. 执行如下命令创建目录install：

```
$ mkdir install
```

获取yasboot安装工具

1. 执行如下命令进入目录install，此时用户所在路径为`/home/yashan/install`：

```
$ cd install
$ pwd
/home/yashan/install
```

2. 将软件包上传至install目录中。

3. 执行如下命令解压软件包，并查看解压后目录中所有文件，请将解压命令后的软件包名称更改为实际使用的软件包名称：

```
$ tar -zxf yashandb-personal-23.2.0.2-linux-x86_64.tar.gz
$ ll

total 238528
drwxrwxr-x 6 yashan yashan        70 Aug  8 01:29 admin
drwxrwxr-x 2 yashan yashan      4096 Aug  8 01:29 bin
drwxrwxr-x 2 yashan yashan       103 Aug  8 01:29 conf
drwxrwxr-x 4 yashan yashan        46 Aug  8 01:29 ext
-rw-rw-r-- 1 yashan yashan     11836 Aug  8 01:29 gitmoduleversion.dat
drwxrwxr-x 2 yashan yashan        79 Aug  8 01:29 include
drwxrwxr-x 3 yashan yashan        17 Aug  8 01:29 java
drwxr-xr-x 2 yashan yashan      4096 Aug  8 01:29 lib
drwxrwxr-x 3 yashan yashan        21 Aug  8 01:29 plug-in
drwxrwxr-x 2 yashan yashan       115 Aug  8 01:29 scripts
-rw-rw-r-- 1 yashan yashan 244227405 Aug  8 01:30 yashandb-personal-23.2.0.2-linux-x86_64.tar.gz
```

生成参数文件

1. 执行如下命令生成安装参数文件，ssh登录密码为创建yashan用户时指定的密码，请将`--ip`参数后面的值更换成安装服务端所在服务器的IP地址（使用127.0.0.1会导致客户端无法连接至服务端）：

```
$ ./bin/yasboot package se gen --cluster yashandb -u yashan -p ssh登录密码 --ip 192.168.1.2 --port 22 --install-path
/home/yashan/yasdb_home  --data-path /home/yashan/yasdb_data --begin-port 1688
 hostid    | group | node_type | node_name | listen_addr     | replication_addr | data_path
-------------------------------------------------------------------------------------------------------
 host0001 | dbg1  | db        | 1-1       | 192.168.1.2:1688 | 192.168.1.2:1689   | /home/yashan/yasdb_data
----------+-------+-----------+-----------+-----------------+------------------+-------------------------

Generate config success
```

执行安装

1. 执行如下命令安装YashanDB数据库，如实际安装数据库版本与示例中版本不同，请将`-i`参数后的软件包名称更改成实际名称：

```
$ ./bin/yasboot package install -t hosts.toml -i yashandb-personal-23.2.0.2-linux-x86_64.tar.gz
checking install package...
install version: yashandb 23.2.0.2
host0001 100% [=====================================================================]   27s
update host to yasom...
```

执行部署

1. 执行如下命令部署YashanDB数据库：

```
$ ./bin/yasboot cluster deploy -t yashandb.toml
 type | uuid             | name               | hostid | index    | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------------------
 task | 356b6a4a51ad600a | DeployYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 9
------+------------------+--------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

2. 执行如下命令配置环境变量：

```
$ cd /home/yashan/yasdb_home/yashandb/23.2.0.2/conf
# 如~/.bashrc中已存在YashanDB相关的环境变量，将其清除

$ cat yashandb.bashrc >> ~/.bashrc
$ source ~/.bashrc
```

3. 执行如下命令设置YashanDB数据库中sys用户的密码：

```
$ cd /home/yashan/yasdb_data/db-1-1/instance
$ mv yasdb.pwd yasdb1.pwd
$ yaspwd file=yasdb.pwd
Enter password for SYS:
```

4. 执行如下命令查看YashanDB数据库状态，如显示出数据库状态信息即为安装成功：

```
$ yasboot cluster status -c yashandb
 host_id  | node_type | nodeid | pid
--------------------------------------
 host0001 | db        | 1-1:1  | 8554
----------+-----------+--------+------
```

5. 执行如下命令连接数据库，请将`password`更改成设置的sys用户密码：

```
$ yasql sys/password
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL>
```

YashanDB客户端安装

本章节将介绍Linux环境下YashanDB数据库客户端安装方式，本文以`yashandb-client-23.2.0.2-linux-x86_64.tar.gz`软件包为例进行阐述，本文示例中YashanDB客户端安装于服务端所在服务器另一用户中。

YashanDB客户端中内置yasql工具，可连接至YashanDB服务端，通过SQL命令执行数据库操作。

创建安装用户

1. 切换至root用户，并执行如下命令创建新用户yasdb：

```
$ su root
# useradd yasdb
```

2. 执行如下命令为用户yasdb指定密码：

```
# passwd yasdb
Changing password for user yasdb.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

创建安装目录

1. 执行如下命令切换至yasdb用户：

```
# su yasdb
$ cd
```

2. 执行如下命令创建目录 yashandb_client ：

```
$ mkdir yashandb_client
```

解压安装包

1. 执行如下命令进入目录 yashandb_client，此时用户所在路径为`/home/yasdb/yashandb_client`：

```
$ cd yashandb_client
$ pwd
/home/yasdb/yashandb_client
```

2. 将软件包上传至 yashandb_client 目录中。

3. 执行如下命令解压软件包，并查看解压后目录中所有文件，解压命令后的软件包名称须更改为实际使用的软件包名称：

```
$ tar -zxf yashandb-client-23.2.0.2-linux-x86_64.tar.gz
$ ll

total 15548
drwxrwxr-x. 2 yasdb yasdb       19 Aug 13 10:29 bin
drwxrwxr-x. 2 yasdb yasdb       21 Aug 13 10:29 include
drwxrwxr-x. 2 yasdb yasdb     4096 Aug 13 10:29 lib
-rw-rw-r--. 1 yasdb yasdb 15915616 Aug 13 10:29 yashandb-client-23.2.0.2-linux-x86_64.tar.gz
```

配置环境变量

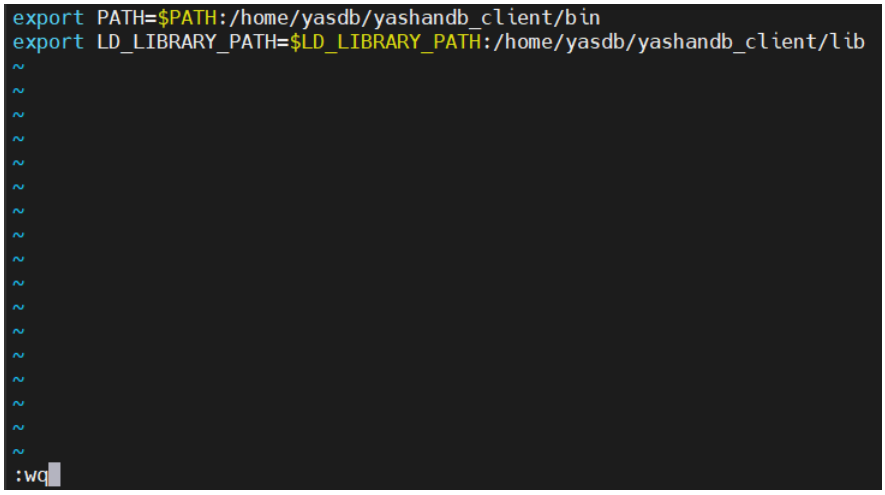
1. 执行如下命令使用vi编辑器打开文件`~/.bashrc`：

```
$ vi ~/.bashrc
```

2. 通过 `i` 键打开输入模式，并将如下内容输入至文件中：

```
export PATH=$PATH:/home/yasdb/yashandb_client/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/yasdb/yashandb_client/lib
```

3. 通过 `Esc` 键退出输入模式，然后输入 `:wq` 保存并关闭文件：



4. 执行如下命令生效环境变量：

```
$ source ~/.bashrc
```

连接YashanDB

执行如下命令连接至YashanDB服务端，请将 `password` 更改成设置的sys用户密码，`ip:port` 更换成服务端IP地址及端口号：

```
$ yasql sys/password@ip:port
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL>
```

服务端配置

如部署服务端时没有对监听地址参数进行修改，则服务端的监听地址默认为 `127.0.0.1:1688` ，即本地地址，此时仅支持同服务器不同用户下安装的YashanDB客户端进行连接，如位于不同服务器则无法连接至服务端，需要对服务端的监听地址进行修改。

1. 于数据库服务端执行如下SQL命令查看监听地址参数值：

```
show parameter LISTEN_ADDR;

NAME             VALUE
---------------- -------------------------
LISTEN_ADDR      127.0.0.1:1688
```

2. 执行如下SQL命令更改监听地址，请将其修改为服务端所在IP地址及端口号：

```
ALTER SYSTEM SET LISTEN_ADDR = ip:port SCOPE=SPFILE;
```

3. 执行如下命令重启数据库服务端：

```
SQL> SHUTDOWN;

EXIT

$ yasboot cluster restart -c yashandb
 type | uuid             | name                | hostid | index    | status  | return_code | progress | cost
------------------------------------------------------------------------------------------------------------
 task | 78d6449df62594b5 | ReStartYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 3
------+------------------+---------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS

$ yasql sys/password
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL>
```

4. 执行如下SQL命令重新查看监听地址参数值：

```
show parameter LISTEN_ADDR;

NAME             VALUE
---------------- -------------------------
LISTEN_ADDR      123.4.5.6:1688
```

实例启停

本章节将介绍YashanDB数据库的实例启停方式。数据库安装过程中将实例自动切换成OPEN阶段，并创建名为yashandb的数据库。

执行如下SQL命令查看当前实例状态及数据库名称：

```
SELECT status FROM V$INSTANCE;

STATUS
-----------------
OPEN

SELECT database_name FROM V$DATABASE;

DATABASE_NAME
----------------------------------------------------------------
yashandb
```

执行如下命令关闭YashanDB服务：

```
$ yasboot cluster stop -c yashandb
 type | uuid             | name             | hostid | index    | status  | return_code | progress | cost
------------------------------------------------------------------------------------------------------------
 task | 3b62bda48bad7fd1 | StopYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 1
------+------------------+------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

执行如下命令开启YashanDB服务，同时会将实例切换至OPEN阶段：

```
$ yasboot cluster start -c yashandb
 type | uuid             | name             | hostid | index    | status  | return_code | progress | cost
------------------------------------------------------------------------------------------------------------
 task | 3b62bda48bad7fd1 | StopYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 1
------+------------------+------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

执行如下命令重启YashanDB数据库，并将实例启动至OPEN阶段：

```
$ yasboot cluster restart -c yashandb
 type | uuid             | name                | hostid | index    | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------------------
 task | 78d6449df62594b5 | ReStartYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 3
------+------------------+---------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

执行如下命令重启YashanDB数据库，并将实例启动至NOMOUNT阶段：

```
$ yasboot cluster restart -c yashandb -m nomount
 type | uuid             | name                | hostid | index    | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------------------
 task | 78d6449df62594b5 | ReStartYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 3
------+------------------+---------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

执行如下命令重启YashanDB数据库，并将实例启动至MOUNT阶段：

```
$ yasboot cluster restart -c yashandb -m mount
 type | uuid             | name                | hostid | index    | status  | return_code | progress | cost
-------------------------------------------------------------------------------------------------------------
 task | 78d6449df62594b5 | ReStartYasdbCluster | -      | yashandb | SUCCESS | 0           | 100      | 3
```

```
------+-----------------+--------------------+--------+----------+---------+-------------+----------+------
task completed, status: SUCCESS
```

> **Note**:
> 如需进行正常的数据库操作，请将实例切换至OPEN阶段。

环境变量

本章将展示数据库安装后的环境变量信息，具体以安装生成值为准。

如下为使用永久生效方法配置的环境变量信息：

```
export YASDB_HOME=/home/yashan/yasdb_home/yashandb/23.2.0.2
export PATH=${YASDB_HOME}/bin:$PATH
export LD_LIBRARY_PATH=${YASDB_HOME}/lib:$LD_LIBRARY_PATH
if command -v rlwrap >/dev/null 2>&1; then
  alias yasql="rlwrap yasql"
fi

export YASDB_DATA=/home/yashan/yasdb_data/db-1-1
```

YASDB_HOME

YASDB_HOME环境变量所指向路径为YashanDB的HOME目录，包含产品相关程序文件。

$YASDB_HOME/bin

$YASDB_HOME/bin目录包含YashanDB配套工具，将其加入PATH系统路径使其可以直接运行。

LD_LIBRARY_PATH

LD_LIBRARY_PATH环境变量所指向路径为YashanDB的LIB目录，包含产品运行所需库文件。

YASDB_DATA

YASDB_DATA环境变量所指向路径为YashanDB的DATA目录，包含产品相关数据文件。

用户操作

本章节将介绍YashanDB数据库用户相关的基本操作。

创建用户

执行如下SQL命令创建新用户yashan，并为其指定密码yashan：

```
CREATE USER yashan IDENTIFIED BY yashan;
```

创建角色

执行如下SQL命令创建新角色yashan_role：

```
CREATE ROLE yashan_role;
```

授权用户

执行如下SQL命令为用户yashan授予登录会话和创建资源的权限：

```
GRANT CONNECT TO yashan;
GRANT RESOURCE TO yashan;
```

切换用户

执行如下SQL命令切换至用户yashan：

```
conn yashan/yashan;

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

> **Note**:
> 切换对象须具有登录会话的权限方可进行切换操作。

修改密码

执行如下SQL命令将yashan用户的密码修改为yashandb：

```
ALTER USER yashan IDENTIFIED BY "yashandb";
```

表空间操作

本章节将介绍YashanDB数据库中表空间相关的基本语法和示例。

表空间是数据库的逻辑存储结构，所有数据库对象均存储于指定的表空间内。

创建表空间

执行 `CREATE TABLESPACE` 语句创建表空间：

```
CREATE TABLESPACE ts_yashan;
```

查看表空间

通过查询 `DBA_TABLESPACES` 视图查看当前数据库中存在的所有表空间：

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES;

TABLESPACE_NAME
----------------------------------------------------------------
SYSTEM
SYSAUX
TEMP
SWAP
USERS
UNDO
TS_YASHAN
```

修改表空间

执行 `ALTER TABLESPACE` 语句修改表空间的相关属性：

- 执行如下语句为ts_yashan表空间增加一个数据文件：

  ```
  ALTER TABLESPACE ts_yashan ADD DATAFILE;
  ```

- 执行如下语句收缩ts_yashan表空间大小：

  ```
  ALTER TABLESPACE ts_yashan SHRINK SPACE;
  ```

删除表空间

执行 `DROP TABLESPACE` 语句删除表空间：

```
DROP TABLESPACE ts_yashan;
```

表操作

本章节将介绍YashanDB数据库中表相关的基本语法和示例。

表是数据库用来存放数据的一个集合，一般与实体对象一一对应，如人员表、部门表、公司表等，一般由行和列这两个二维信息来组织表数据。

创建表

执行 `CREATE TABLE` 语句创建表：

- 执行如下语句创建表：

```
CREATE TABLE tb_yashan(c1 INT,C2 VARCHAR(10));
```

查看表

通过查询 `USER_TABLES` 视图查看当前用户中已存在的表名称：

```
SELECT TABLE_NAME,TABLE_TYPE FROM USER_TABLES;

TABLE_NAME                                                       TABLE_TYPE
---------------------------------------------------------------- ----------
TB_YASHAN                                                        HEAP
```

执行 `SELECT` 语句查看表中具体信息：

```
SELECT * FROM tb_yashan;

          C1 C2
------------ -------------
```

修改表

执行 `ALTER TABLE` 语句修改表的相关属性：

- 执行如下语句修改表名称：

```
ALTER TABLE tb_yashan RENAME TO tab_yashan;
```

- 执行如下语句在表中新增列字段：

```
ALTER TABLE tab_yashan ADD(c3 NUMBER);
```

- 执行如下语句修改表中列字段的数据类型：

```
ALTER TABLE tab_yashan MODIFY c3 FLOAT;
```

删除表

执行 `DROP TABLE` 语句删除表：

```
DROP TABLE tab_yashan;
```

索引操作

本章节将介绍YashanDB数据库中索引相关的基本语法和示例。

索引是一种物理的对数据库表中一列或多列的值进行排序的存储结构，它是某个表中一列或若干列值的集合，是指向表中物理标识这些值所在行的逻辑指针清单。

创建索引

执行 `CREATE INDEX` 语句创建索引：

```
CREATE TABLE tb_index(c1 INT,c2 INT);

CREATE INDEX inde1 ON tb_index(c1);
```

查看索引

通过查询 `USER_INDEXES` 视图查看当前用户的索引信息：

```
SELECT * FROM USER_INDEXES;

INDEX_NAME     INDEX_TYPE     TABLE_OWNER     TABLE_NAME      TABLE_TYPE       UNIQUENESS COMPRESSION PREFIX_LENGTH
TABLESPACE_NAME     INI_TRANS    MAX_TRANS        PCT_FREE LOGGING       BLEVEL           LEAF_BLOCKS         DISTINCT_KEYS
AVG_LEAF_BLOCKS_PER_KEY AVG_DATA_BLOCKS_PER_KEY STATUS                NUM_ROWS           SAMPLE_SIZE LAST_ANALYZED
PARTITIONED TEMPORARY GENERATED VISIBILITY    DATABASE_MAINTAINED CONSTRAINT_INDEX
-------------  -------------- --------------- --------------- ---------------- ---------- ----------- ------------- ---------
---------- ------------ ------------ ------------ ------- ------------ -------------------- -------------------- ----------
------------ --------------------- ---------- ------------------- -------------------- -------------------- -----------
--------- --------- ------------ ------------------ ---------------
INDE1          NORMAL         YASHAN          TB_INDEX        TABLE            N          DISABLED               0 USERS
2          255            8 Y
VALID                                                                 N           N         N         VISIBLE       N
N

SELECT INDEX_NAME FROM USER_INDEXES;

INDEX_NAME
-------------
INDE1
```

删除索引

执行 `DROP INDEX` 语句删除索引：

```
DROP INDEX inde1;
```

数据操作

本章节将介绍YashanDB数据库中表相关的基本语法和示例。

插入数据

通过执行 `INSERT` 语句往表中插入数据：

```
CREATE TABLE insert_tb(c1 INT,c2 CHAR(10));

INSERT INTO insert_tb VALUES(4,'hello');

INSERT INTO insert_tb VALUES(1,'world'),(2,'nihao'),(3,'shijie');

COMMIT;
```

删除数据

YashanDB中可选 `DELETE` 和 `TRUNCATE TABLE` 两种方式对表数据进行删除：

- 通过执行 `DELETE` 语句删除数据：
  - 执行如下语句删除 `insert_tb` 表中 `c1=1` 的行：

    ```
    DELETE FROM insert_tb WHERE c1=1;
    ```

  - 执行如下语句删除 `insert_tb` 表中所有行：

    ```
    DELETE insert_tb;
    ```

- 通过执行 `TRUNCATE TABLE` 语句一次性删除表中所有数据：
  - 执行如下语句删除 `insert_tb` 表中所有数据：

    ```
    TRUNCATE TABLE insert_tb;
    ```

更新数据

通过执行 `UPDATE` 语句更新表中数据：

- 执行如下语句将 `insert_tb` 表c1列字段中 `c1=1` 的数据更新为 `c1=5` ：

```
INSERT INTO insert_tb VALUES(1,'nihao'),(2,'hello'),(3,'shijie'),(4,'world');

SELECT * FROM insert_tb;

          C1 C2
------------ -------------
           1 nihao
           2 hello
           3 shijie
           4 world

UPDATE insert_tb SET c1=5 WHERE c1=1;

SELECT * FROM insert_tb;

          C1 C2
------------ -------------
           5 nihao
```

```
           2 hello
           3 shijie
           4 world
```

- 执行如下语句批量更新 `insert_tb` 表中的数据：

```
UPDATE insert_tb SET (c1,c2) = (7,'newvalue') WHERE c1=3;

SELECT * FROM  insert_tb;

          C1 C2
------------ -------------
           5 nihao
           2 hello
           7 newvalue
           4 world
```

查询数据

通过执行 `SELECT` 语句查询表数据：

- 执行如下语句查询 `insert_tb` 表中所有数据：

```
SELECT * FROM insert_tb;

          C1 C2
------------ -------------
           5 nihao
           2 hello
           7 newvalue
           4 world
```

- 执行如下语句按照 `insert_tb` 表中c1列字段的大小顺序进行排序查询：

```
SELECT * FROM insert_tb ORDER BY c1;

          C1 C2
------------ -------------
           2 hello
           4 world
           5 nihao
           7 newvalue
```

- 执行如下语句对 `insert_tb` 表进行条件查询：

```
SELECT C2 FROM insert_tb WHERE c1=5;

C2
-------------
nihao
```

事务操作

本章节将介绍YashanDB数据库中事务相关的基本语法和示例。

提交事务前，用户在事务过程做的任何修改只有自己能看到，其他用户无法看到，并可以通过回滚操作将数据恢复。

提交事务后，其他用户可看到修改后的数据，此时无法通过回滚操作将数据恢复。

提交事务

执行 `COMMIT` 语句提交事务：

```
CREATE TABLE COM_TB(c1 INT);

INSERT INTO COM_TB VALUES(1),(2),(3);

COMMIT;
```

回退事务

执行 `ROLLBACK` 语句回退事务：

- 执行如下语句于 `COM_TB` 表中添加新数据：

```
INSERT INTO COM_TB VALUES(6);

SELECT * FROM COM_TB;

          C1
------------
           1
           2
           3
           6
```

- 执行 `ROLLBACK` 语句将事务回退至修改前状态：

```
ROLLBACK;

SELECT * FROM COM_TB;

          C1
------------
           1
           2
           3
```

JDBC驱动应用示例

本章将介绍YashanDB的JDBC驱动的安装及基础的操作演示。

示例环境介绍

软件	版本
JDBC驱动	yashandb-jdbc-1.5.jar
JDK	1.8
JRE	1.8

安装前准备

1. 使用JDBC驱动前须先安装JDK，JDK版本兼容信息如下，请自行安装下述版本的环境：
   - JDK：1.8及以上
   - JRE：1.8及以上
2. 可通过执行 `java -version` 命令验证Java环境是否正常：

```
$ java -version
openjdk version "1.8.0_372"
OpenJDK Runtime Environment (build 1.8.0_372-b07)
OpenJDK 64-Bit Server VM (build 25.372-b07, mixed mode)
```

3. 请联系我们的技术支持获取JDBC驱动软件包，软件包名称示例： `yasdb-jdbc-版本号.jar` 。

JDBC驱动安装

1. 执行如下命令创建 `/home/yashan/JDBC` 目录，请注意区分大小写：

```
$ mkdir JDBC
$ cd JDBC
```

2. 将JDBC软件包上传至该目录中。
3. 配置JDBC驱动环境：
   1. 执行如下命令使用vi编辑器打开文件 `~/.bashrc` ：

```
$ vi ~/.bashrc
```

   2. 通过 `i` 键打开输入模式，并将如下内容输入至文件中，请将 `yasdb-jdbc-1.5.jar` 更换成实际软件包名称：

```
export CLASSPATH=/home/yashan/JDBC/yasdb-jdbc-1.5.jar:${CLASSPATH}
```

   3. 通过 `Esc` 键退出输入模式，然后输入 `:wq` 保存并关闭文件。
   4. 执行如下生效环境变量：

```
$ source ~/.bashrc
```

编写JAVA文件

1. 执行如下命令创建Java文件`Jdbcexample.java`：

```
$ vi Jdbcexample.java
```

2. 通过`i`键打开输入模式，并将如下内容输入至文件中：

```
package jdbc0;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.CallableStatement;

public class Jdbcexample {
    public static Connection getConnection(String username, String passwd) {
        String driver = "com.yashandb.jdbc.Driver";
        String sourceURL = "jdbc:yasdb://host:port/database_name";
        Connection conn = null;
        try {
            Class.forName(driver);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        try {
            conn = DriverManager.getConnection(sourceURL, username, passwd);
            System.out.println("Connection succeed!");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        return conn;
    }

    public static void createTable(Connection conn) {
        Statement stmt = null;
        try {
            stmt = conn.createStatement();
            stmt.execute("DROP TABLE IF EXISTS customer");
            stmt.execute("CREATE TABLE customer(id INTEGER, name VARCHAR(32))");
            System.out.println("create table customer succeed!");
            stmt.close();
        } catch (SQLException e) {
            if (stmt != null) {
                try {
                    stmt.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }

   public static void batchInsertData(Connection conn) {
        PreparedStatement pst = null;

        try {
            pst = conn.prepareStatement("INSERT INTO customer VALUES (?,?)");
            for (int i = 0; i < 3; i++) {
                pst.setInt(1, i);
                pst.setString(2, "sales" + i);
                pst.addBatch();
            }
            pst.executeBatch();
```

```
            System.out.println("insert table customer succeed!");
            pst.close();
        } catch (SQLException e) {
            if (pst != null) {
                try {
                    pst.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }

  public static int execJdbcexample(String ctrls) {
        Connection conn = getConnection(user, password);

        createTable(conn);

        batchInsertData(conn);

        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
            return 0;
        }
        return 1;
    }

    public static void main(String[] args) {
       int a = execJdbcexample("1");
    }
}
```

其中：

- `String sourceURL = "jdbc:yasdb://host:port/database_name";` 为数据库连接描述符，请将 `host:port` 更换成服务端所在IP地址及端口，`database_name` 更换成数据库名称。
- `Connection conn = getConnection(user, password);` 用于创建连接，请将 `user` 更换成用户名字符串，`password` 更换成该用户的密码字符串。

```
    public static Connection getConnection(String username, String passwd) {
        String driver = "com.yashandb.jdbc.Driver";
        String sourceURL = "jdbc:yasdb://127.0.0.1:1688/yashandb";
        Connection conn = null;
        try {
            Class.forName(driver);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        try {
            conn = DriverManager.getConnection(sourceURL, username, passwd);
            System.out.println("Connection succeed!");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        return conn;
    }
```

```
    public static int execJdbcexample(String ctrls) {
        Connection conn = getConnection("sys", "password");
```

```
        createTable(conn);

        batchInsertData(conn);

        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
            return 0;
        }
        return 1;
    }
```

3. 执行如下命令进行编译：

```
$ javac -d . Jdbcexample.java
```

4. 执行如下命令执行程序，如输出如下结果，则代表数据库连接成功：

```
$ java -Djdbc.drivers=com.yashandb.jdbc.Driver jdbc0.Jdbcexample
Connection succeed!
create table customer succeed!
insert table customer succeed!
```

csv数据快速导入

本章将对YashanDB内置数据导入工具yasldr进行介绍及提供基础示例。

yasldr是YashanDB提供的客户端导入工具，用于执行CSV格式的数据文件导入，每次导入单个数据文件至单个表中，有关yasldr工具的具体使用语法及限制可参考yasldr使用指导章节。

导入前准备

1. 准备导入数据文件：
    1. 于操作系统终端中执行如下命令在HOME目录下创建 `datafile` 文件：

    ```
    $ vi datafile
    ```

    2. 将如下内容写入 `datafile` 文件中：

    ```
    1|load|101
    2|load|201
    ```

    3. 通过 `Esc` 键退出输入模式，然后输入 `:wq` 保存并关闭文件。
2. 准备导入用户：
    1. 执行如下命令连接YashanDB数据库，请将 `password` 更改成设置的sys用户密码：

    ```
    $ yasql sys/password
    YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

    SQL>
    ```

    2. 执行如下SQL命令创建用户 `yasldr_user` ，并为其指定密码yasldr：

    ```
    CREATE USER yasldr_user IDENTIFIED BY yasldr;
    ```

    3. 执行如下SQL命令为 `yasldr_user` 用户授予登录会话和创建资源的权限：

    ```
    GRANT CONNECT TO yasldr_user;
    GRANT RESOURCE TO yasldr_user;
    ```

    4. 执行如下SQL命令切换至 `yasldr_user` 用户：

    ```
    conn yasldr_user/yasldr

    Connected to:
    YashanDB Server Personal Edition Release 23.2.0.2 x86_64 - X86 64bit Linux
    ```
3. 准备数据文件导入的目标表：
    1. 执行如下SQL命令创建表 `loadData` ：

    ```
    CREATE TABLE loadData (c1 INT,c2 CHAR(10),c3 INT);
    ```
4. 执行如下命令退出YashanDB数据库：

```
SQL> exit
$
```

数据导入

1. 于操作系统终端中执行如下命令将 `datafile` 文件中数据导入至 `yasldr_user` 用户的 `loadData` 表中：

```
$ yasldr yasldr_user/yasldr batch_size=4032 control_text="'LOAD DATA OPTIONS(DEGREE_OF_PARALLELISM=2) INFILE
'/home/yashan/datafile' FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '\"' APPEND INTO TABLE loadData(c1,c2,c3) '"

YashanDB Loader Release 23.2.0.2 x86_64 297f388
[YASLDR] execute succeeded
```

如上命令将 `datafile` 文件中数据根据 `|` 进行分隔，分别导入 `loadData` 表中的c1、c2和c3列字段中。

验证数据

1. 执行如下命令登录数据库：

```
$ yasql yasldr_user/yasldr
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL>
```

2. 执行如下SQL命令查看表 `loadData` 中数据：

```
SELECT * FROM loadData;

          C1 C2                    C3
------------ ------------- ------------
           1 load                  101
           2 load                  201
```

元数据和数据导入导出

本章将对YashanDB内置导入导出工具imp/exp进行介绍及提供基础示例，有关imp/exp工具的具体使用语法及限制可参考imp和exp章节。

exp工具是YashanDB的配套导出工具，提供元数据导出及CSV导出能力；imp为YashanDB的配套导入工具，提供元数据导入能力。

用户可通过使用exp工具将YashanDB数据库中的表结构、索引、约束等所有数据生成一个元数据文件，该元数据文件可通过配套的导入工具imp导入至同构的YashanDB数据库中。或者通过exp工具将指定表结构排列的数据导出至CSV文件中，并通过yasldr工具将该CSV文件导入至YashanDB数据库中。

导入前准备

1. 准备导入用户：

   1. 执行如下命令连接YashanDB数据库，请将`password`更改成设置的sys用户密码：

      ```
      $ yasql sys/password
      YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

      SQL>
      ```

   2. 执行如下SQL命令创建用户`import_user`，并为其指定密码import：

      ```
      CREATE USER import_user IDENTIFIED BY import;
      ```

   3. 执行如下SQL命令将DBA权限授权给`import_user`用户：

      ```
      GRANT DBA TO import_user;
      ```

   4. 执行如下SQL命令切换至`import_user`用户：

      ```
      conn import_user/import

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

   5. 执行如下SQL命令于`import_user`用户中创建表并插入数据：

      ```
      CREATE TABLE classmate_info(c1 INT,c2 CHAR(10));
      INSERT INTO classmate_info VALUES(1,'h'),(2,'a'),(3,'c');
      CREATE TABLE classmate_info1(c1 INT,c2 INT);
      INSERT INTO classmate_info1 VALUES(1,2),(3,4),(5,6);
      COMMIT;
      ```

导出数据

1. 执行如下SQL命令退出YashanDB数据库：

   ```
   SQL> exit
   $
   ```

2. 于操作系统终端中执行如下命令将`import_user`用户下所有元数据导出至`export.owner.export`文件中，请将`password`更改成设置的sys用户密码：

```
$ exp sys/password FILE=export.owner.export OWNER=import_user

YashanDB Export Release 23.2.0.2 x86_64 297f388
export terminated successfully
```

3. 执行如下命令查看导出的元数据文件：

```
$ ll

total 24
-rw-r-----  1 yashan yashan 16771 Aug 14 21:08 export.owner.export
drwxrwxr-x 12 yashan yashan   259 Aug  8 16:33 install
drwxr-----  3 yashan yashan    20 Aug  8 16:45 yasdb_data
drwxrwxr-x  3 yashan yashan    22 Aug  8 16:41 yasdb_home
```

导入数据

1. 执行如下命令连接YashanDB数据库：

```
$ yasql import_user/import
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL>
```

2. 执行如下SQL命令删除表classmate_info和classmate_info1：

```
DROP TABLE classmate_info;
DROP TABLE classmate_info1;
COMMIT;
```

3. 通过查询`USRE_TABLES`视图查看当前用户下所有表信息，此时`import_user`用户下不存在任何表：

```
SELECT table_name FROM USER_TABLES;

TABLE_NAME
----------------------------------------------------------------
```

4. 执行如下命令退出YashanDB数据库：

```
SQL> exit
$
```

5. 于数据库终端中执行如下命令将元数据文件导入至`import_user`用户，请将`password`更改成设置的sys用户密码：

```
$ imp sys/password FILE=export.owner.export FROMUSER=import_user

YashanDB Import Release 23.2.0.2 x86_64 297f388
import terminated successfully
```

验证数据

1. 登录数据库并查看用户表信息：

```
$ yasql import_user/import
YashanDB SQL Personal Edition Release 23.2.0.2 x86_64

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

SQL> SELECT table_name FROM USER_TABLES;

TABLE_NAME
----------------------------------------------------------------
classmate_info1
classmate_info
```

2. 执行如下SQL命令查看表classmate_info和classmate_info1中的数据：

```
SELECT * FROM classmate_info;

          C1 C2
------------ -------------
           1 h
           2 a
           3 c

SELECT * FROM classmate_info1;

          C1           C2
------------ ------------
           1            2
           3            4
           5            6
```

TPC-C测试

本章节将介绍在YashanDB单机数据库上运行基于BenchmarkSQL的TPC-C测试的具体操作及相关示例。

TPCC-C简介

TPC-C测试是针对业务处理系统的规范，该规范由TPC（Transaction Processing Performance Council）委员会制定，测试结果主要取决于流量指标和性价比指标，是最常用的在线事务处理（OLTP）测试基准。

TPC-C测试中模拟了大型商品批发商的交易业务场景，该批发商总共生产销售100000种产品，拥有多个负责不同区域的商品仓库，每个仓库需要为10个分销商进行供货，且每个分销商需要为3000个客户进行服务，对数据库进行TPC-C测试时可根据服务器及数据库的配置对商品仓库的数量进行调整，从而模拟出不同的压力场景。

业务场景中主要包含如下5类事务：

- Stock-Level：用于表示分销商库存状态，库存较低时需要进行补货。
- New-Order：用于表示客户提交了一笔新订单，每笔订单平均包括10件产品。
- Payment：用于表示客户为订单支付费用，更新其账户余额。
- Delivery：用于表示根据用户提交的订单进行发货。
- Order-Status：用于表查询用户的最近交易状态。

TPC-C测试工具下载

- 请自行于BenchmarkSQL官网下载BenchmarkSQL5。
- 请确保服务器中已有JDK1.8及以上版本的JDK。

TPC-C测试准备

对YashanDB进行TPC-C测试前需对Benchmark SQL5进行配置，使之支持YashanDB数据库：

修改jTPCC.java文件

1. 在操作系统终端执行如下命令并输入密码，切换至root用户。

```
$ su root
Password:
```

2. 执行如下命令，进入tpc-c目录。

```
$ cd tpc-c
```

3. 执行如下命令，在vi编辑器中打开/home/yashan/tpc-c/benchmarksql-5.0/src/client/jTPCC.java文件，请注意区分大小写：

```
$ vi benchmarksql-5.0/src/client/jTPCC.java
```

4. 在文件中查找下图内容：

```
if (iDB.equals("firebird"))
    dbType = DB_FIREBIRD;
else if (iDB.equals("oracle"))
    dbType = DB_ORACLE;
else if (iDB.equals("postgres"))
    dbType = DB_POSTGRES;
else
{
    log.error("unknown database type '" + iDB + "'");
    return;
}
```

5. 按i进入编辑模式，在dbType = DB_POSTFRES后新增如下内容：

```
else if (iDB.equals("yashandb"))
    dbType = DB_YASHANDB;
```

```
if (iDB.equals("firebird"))
    dbType = DB_FIREBIRD;
else if (iDB.equals("oracle"))
    dbType = DB_ORACLE;
else if (iDB.equals("postgres"))
    dbType = DB_POSTGRES;
else if (iDB.equals("yashandb"))
    dbType = DB_YASHANDB;
else
{
    log.error("unknown database type '" + iDB + "'");
    return;
}
```

6. 修改完成后，按**Esc**，输入`:wq`保存并退出文件编辑。

修改jTPCCConfig.java文件

1. 执行如下命令，在vi编辑器中打开jTPCCConfig.java文件：

```
$ vi benchmarksql-5.0/src/client/jTPCCConfig.java
```

2. 在文件中查找下图内容：

```
public final static int     DB_UNKNOWN = 0,
                            DB_FIREBIRD = 1,
                            DB_ORACLE = 2,
                            DB_POSTGRES = 3;
```

3. 按i进入编辑模式，将该部分内容修改为如下内容：

```
public final static int DB_UNKNOWN = 0,
            DB_FIREBIRD = 1,
            DB_ORACLE = 2,
            DB_POSTGRES = 3,
            DB_YASHANDB = 4;
```

4. 修改完成后，按**Esc**，输入`:wq`保存并退出文件编辑。

编译源码

执行如下命令进入benchmarksql-5.0目录，并执行ant命令进行编译：

```
$ cd benchmarksql-5.0
$ ant
Buildfile: /home/yashan/tpc-c/benchmarksql-5.0/build.xml

init:

compile:
    [javac] Compiling 11 source files to /home/yashan/tpc-c/benchmarksql-5.0/build

dist:
    [mkdir] Created dir: /home/yashan/tpc-c/benchmarksql-5.0/dist
    [jar] Building jar: /home/yashan/tpc-c/benchmarksql-5.0/dist/BenchmarkSQL-5.0.jar

BUILD SUCCESSFUL
Total time: 1 second
```

> **Note**:
> 如此时返回`ant:command not found`，可通过执行`yum install ant`安装ant编译工具。

创建文件props.yashandb

1. 执行如下命令，进入/home/yashan/tpc-c/benchmarksql-5.0/run目录：

```
$ cd /home/yashan/tpc-c/benchmarksql-5.0/run
```

2. 执行如下命令，通过vi编辑器创建文件props.yashandb：

```
$ vi props.yashandb
```

3. 按i进入编辑模式，将如下内容新增至文件中（可根据实际环境进行修改）：

```
db=yashandb
driver=com.yashandb.jdbc.Driver
conn=jdbc:yasdb://localhost:1688/yashandb
user=sys
password=sys
warehouses=10
loadWorkers=2
terminals=10
runTxnsPerTerminal=0
runMins=5
limitTxnsPerMin=0
terminalWarehouseFixed=true
newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
osCollectorScript=./misc/os_collector_linux.py
osCollectorInterval=1
```

其中：

参数	含义
db	数据库，须与上述修改文件中添加的数据库名称相同
driver	驱动程序文件，须使用YashanDB的JDBC驱动
conn	连接描述符，格式为：`conn=jdbc:yasdb://ip:port/database_name`
user	数据库用户
password	数据库用户的密码
warehouses	用于指定测试中商品仓库的数量，通常测试场景为100~1000仓
loadWorkers	数据导入的并发数，通常设置为CPU线程总数的2~4倍
terminals	业务运行的并发数，通常设置为CPU线程总数的2~4倍
runTxnsPerTerminal	每个会话运行的固定事务数量，通常配置为0，以runMins限制测试时间
runMins	用于指定测试运行的时间，通常建议压测时间为10~30分钟
limitTxnsPerMin	每秒事务数上限，压测场景下通常设置为0
terminalWarehouseFixed	会话和仓库的绑定模式，通常设置为true

参数	含义
newOrderWeight paymentWeight orderStatusWeight deliveryWeight stockLevelWeight	五种事务的占比，所有值的和须为100 标准的业务比率配置为45:43:4:4:4
resultDirectory	指定存储测试结果的目录。%tY、%tm、%td、%tH、%tM和%tS是时间格式化参数
osCollectorScript	指定操作系统收集器的脚本路径。通常用于收集操作系统资源使用情况的统计信息，例如CPU使用率、内存使用情况、磁盘I/O等
osCollectorInterval	指定操作系统收集器脚本运行间隔（单位：秒）

4. 修改完成后，按**Esc**，输入`:wq`保存并退出文件编辑。

修改文件funcs.sh

1. 执行如下命令，在vi编辑器中打开`funcs.sh`文件：

```
$ vi /home/yashan/tpc-c/benchmarksql-5.0/run/funcs.sh
```

2. 按**i**进入编辑模式，将文档内容替换为如下内容：

```
# ----
# $1 is the properties file
# ----
PROPS=$1
if [ ! -f ${PROPS} ] ; then
    echo "${PROPS}: no such file" >&2
    exit 1
fi

# ----
# getProp()
#
#   Get a config value from the properties file.
# ----
function getProp()
{
    grep "^${1}=" ${PROPS} | sed -e "s/^${1}=//"
}

# ----
# getCP()
#
#   Determine the CLASSPATH based on the database system.
# ----
function setCP()
{
    case "$(getProp db)" in
        firebird)
            cp="../lib/firebird/*:../lib/*"
            ;;
        oracle)
            cp="../lib/oracle/*"
            if [ ! -z "${ORACLE_HOME}" -a -d ${ORACLE_HOME}/lib ] ; then
                cp="${cp}:${ORACLE_HOME}/lib/*"
            fi
            cp="${cp}:../lib/*"
            ;;
        postgres)
            cp="../lib/postgres/*:../lib/*"
            ;;
        yashandb)
```

```
            cp="../lib/yashandb/*:../lib/*"
            ;;
    esac
    myCP=".:${cp}:../dist/*"
    export myCP
}

# ----
# Make sure that the properties file does have db= and the value
# is a database, we support.
# ----
case "$(getProp db)" in
    firebird|oracle|postgres|yashandb)
        ;;
    "") echo "ERROR: missing db= config option in ${PROPS}" >&2
        exit 1
        ;;
    *)  echo "ERROR: unsupported database type 'db=$(getProp db)' in ${PROPS}" >&2
        exit 1
        ;;
esac
```

3. 修改完成后，按**Esc**，输入 `:wq` 保存并退出文件编辑。

添加yashandb java connector驱动

执行此步骤前须确保已安装YashanDB JDBC驱动，本例中驱动包所在路径为/home/yashan/yashandb_jdbc/。

执行如下命令添加驱动：

```
$ mkdir -p /home/yashan/tpc-c/benchmarksql-5.0/lib/yashandb/
$ cp /home/yashan/yashandb_jdbc/yashandb-jdbc-1.5-SNAPSHOT.jar /home/yashan/tpc-c/benchmarksql-5.0/lib/yashandb/
```

修改runDatabaseBuild.sh文件

1. 执行如下命令，在vi编辑器中打开runDatabaseBuild.sh文件：

```
$ vi /home/yashan/tpc-c/benchmarksql-5.0/run/runDatabaseBuild.sh
```

2. 在文件中查找下图内容：

```
AFTER_LOAD="indexCreates foreignKeys extraHistID buildFinish"
```

3. 按**i**进入编辑模式，将该部分内容修改为如下内容：

```
AFTER_LOAD="indexCreates foreignKeys buildFinish"
```

4. 修改完成后，按**Esc**，输入 `:wq` 保存并退出文件编辑。

TPC-C测试运行

部署YashanDB数据库

最佳性能数据会因为测试环境的CPU、内存、IO、网络条件差异而不同，为了让YashanDB在测试环境上达到最佳的TPC-C性能表现，需要根据测试环境的配置进行性能调优，数据库性能调优的详细信息请查阅YashanDB性能调优文档。

TPC-C测试调优主要分为参数配置调优和建库配置调优：

数据库参数配置调优 在TPC-C测试场景下主要关注缓存大小与分区、IO参数等性能参数的配置。

以1000仓、256并发的测试场景为例，推荐配置以下性能调优参数：

```
# Data buffer用于数据块的缓存，其大小会影响数据访问的缓存命中率。建议将规划内存的80%配置为数据缓存区，如果缓存区的大小大于总数据大小可取得最佳性能。
DATA_BUFFER_SIZE=200G
# Data buffer的分区数，在大并发的测试场景下，将Data buffer分区可以降低缓存区的锁冲突。
_DATA_BUFFER_PARTS=8
# VM Buffer用于保存例如order/group by等数据运算的中间结果，当VM空间不足时会产生内存与SWAP表空间的换入换入，影响数据库性能表现。
# 因此需要配置合理的VM Buffer大小以避免换入换入的产生。通过视图V$VM中的SWAPPED_OUT_BLOCKS字段可以获取SWAP的次数，当为0时可以取到最佳性能。
VM_BUFFER_SIZE=25G
# VM Buffer的分区数，在大并发的测试场景下，将VM buffer可以降低缓存区的锁冲突。
VM_BUFFER_PARTS=8
# 全局大页内存区大小，在大并发的测试场景下，需要提高配置以避免资源不足。
LARGE_POOL_SIZE=1G
# 全局执行内存区大小，在大并发的测试场景下，需要提高配置以避免资源不足。
WORK_AREA_POOL_SIZE=2G
# undo数据的保留时间，用于一致性读或者数据闪回。对于类似TPC-C小事务的场景下，降低undo数据的保留时间可以提高undo分配效率，提升数据库性能。
UNDO_RETENTION=15
# 会话级CURSOR的数量，在大并发的测试场景下提高配置可以消除全局CURSOR的竞争。
_SESSION_RESERVED_CURSORS=64
# 增量Checkpoint的时间间隔，后台脏块刷盘会与redo刷盘产生IO争用，在DATA和redo同盘部署的场景下，降低Checkpoint频率可以提升数据库性能。
CHECKPOINT_TIMEOUT=900
# 指定触发checkpoint的从恢复点到当前redo日志刷盘点的redo大小间隔，通常配置为redo文件总大小的一半。
CHECKPOINT_INTERVAL=10G
# 共享内存池的大小，提高配置以避免SQL缓存或者元数据缓存的频繁失效。
SHARE_POOL_SIZE=2G
```

数据库配置参数的详细说明请查阅配置参数。

数据库建库配置调优 创建数据库时主要关注以下方面：

- redo文件的大小：redo文件配置太小，在压测场景下容易出现redo追尾的情况，严重影响数据库性能。通过V$SYSTEM_EVENT视图中的“checkpoint completed”的等待事件判断是否出现redo追尾的情况。
- DATA文件的大小：预占数据空间可以避免数据库运行过程中空间动态扩展对性能的影响，因此为了最佳性能表现，需要初始化足够大的表空间。

如果存储条件允许，请将redo文件与数据文件分盘部署，以减少两者间的IO争用，以1000仓、256并发的测试场景为例，推荐的建库语句如下：

```
CREATE DATABASE tpcc LOGFILE(
'/data1/redo1' size 20G BLOCKSIZE 512,
'/data1/redo2' size 20G BLOCKSIZE 512,
'/data1/redo3' size 20G BLOCKSIZE 512,
'/data1/redo4' size 20G BLOCKSIZE 512,
'/data1/redo5' size 20G BLOCKSIZE 512,
'/data1/redo6' size 20G BLOCKSIZE 512,
'/data1/redo7' size 20G BLOCKSIZE 512,
'/data1/redo8' size 20G BLOCKSIZE 512,
'/data1/redo9' size 20G BLOCKSIZE 512,
'/data1/redo10' size 20G BLOCKSIZE 512)
UNDO TABLESPACE DATAFILE '/data2/undo' size 10G
SWAP TABLESPACE TEMPFILE '/data2/swap' size 10G
SYSTEM TABLESPACE DATAFILE '/data2/system' size 5G
SYSAUX TABLESPACE DATAFILE '/data2/sysaux' size 5G
DEFAULT TABLESPACE DATAFILE '/data2/users' size 300G;
```

数据库配置参数和建库配置调优的详细介绍请查阅数据库配置调优。

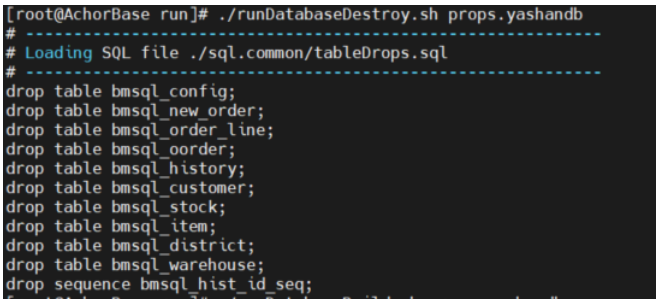
清理TPC-C数据

在/home/yashan/tpc-c/benchmarksql-5.0/run目录中，执行如下命令进行数据清理：

```
$ ./runDatabaseDestroy.sh props.yashandb

# ------------------------------------------------------------
# Loading SQL file ./sql.common/tableDrops.sql
# ------------------------------------------------------------
drop table bmsql_config;
drop table bmsql_new_order;
drop table bmsql_order_line;
```

```
drop table bmsql_oorder;
drop table bmsql_history;
drop table bmsql_customer;
drop table bmsql_stock;
drop table bmsql_item;
drop table bmsql_district;
drop table bmsql_warehouse;
drop sequence bmsql_hist_id_seq;
```



TPC-C数据装载

目录中执行如下命令进行数据导入：

```
$ ./runDatabaseBuild.sh props.yashandb

# ------------------------------------------------------------
# Loading SQL file ./sql.common/tableCreates.sql
# ------------------------------------------------------------
create table bmsql_config (
cfg_name    varchar(30) primary key,
cfg_value   varchar(50)
);
create table bmsql_warehouse (
w_id        integer   not null,
w_ytd       decimal(12,2),
w_tax       decimal(4,4),
w_name      varchar(10),
w_street_1  varchar(20),
w_street_2  varchar(20),
w_city      varchar(20),
w_state     char(2),
w_zip       char(9)
);
```

```
[root@AchorBase run]# ./runDatabaseBuild.sh props.yashandb
# ------------------------------------------------------------
# Loading SQL file ./sql.common/tableCreates.sql
# ------------------------------------------------------------
create table bmsql_config (
cfg_name    varchar(30) primary key,
cfg_value   varchar(50)
);
create table bmsql_warehouse (
w_id        integer   not null,
w_ytd       decimal(12,2),
w_tax       decimal(4,4),
w_name      varchar(10),
w_street_1  varchar(20),
w_street_2  varchar(20),
w_city      varchar(20),
w_state     char(2),
w_zip       char(9)
);
create table bmsql_district (
d_w_id       integer       not null,
d_id         integer       not null,
d_ytd        decimal(12,2),
d_tax        decimal(4,4),
d_next_o_id  integer,
d_name       varchar(10),
d_street_1   varchar(20),
d_street_2   varchar(20),
d_city       varchar(20),
d_state      char(2),
d_zip        char(9)
);
create table bmsql_customer (
c_w_id         integer        not null,
c_d_id         integer        not null,
c_id           integer        not null,
c_discount     decimal(4,4),
c_credit       char(2),
c_last         varchar(16),
c_first        varchar(16),
c_credit_lim   decimal(12,2),
c_balance      decimal(12,2),
c_ytd_payment  decimal(12,2),
c_payment_cnt  integer,
c_delivery_cnt integer,
c_street_1     varchar(20),
c_street_2     varchar(20),
c_city         varchar(20),
c_state        char(2),
c_zip          char(9),
c_phone        char(16),
c_since        timestamp,
c_middle       char(2),
c_data         varchar(500)
);
create sequence bmsql_hist_id_seq;
create table bmsql_history (
hist_id  integer,
h_c_id   integer,
h_c_d_id integer,
h_c_w_id integer,
h_d_id   integer,
h_w_id   integer,
h_date   timestamp,
h_amount decimal(6,2),
h_data   varchar(24)
);
create table bmsql_new_order (
no_w_id  integer   not null,
no_d_id  integer   not null,
no_o_id  integer   not null
);
create table bmsql_oorder (
o_w_id       integer      not null,
o_d_id       integer      not null,
o_id         integer      not null,
o_c_id       integer,
```

运行TPC-C测试

执行如下语句进行TPC-C测试：

```
$ ./runBenchmark.sh props.yashandb

20:50:04,561 [main] INFO   jTPCC : Term-00,
20:50:04,563 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
20:50:04,563 [main] INFO   jTPCC : Term-00,      BenchmarkSQL v5.0
20:50:04,563 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
20:50:04,563 [main] INFO   jTPCC : Term-00,  (c) 2003, Raul Barbosa
20:50:04,563 [main] INFO   jTPCC : Term-00,  (c) 2004-2016, Denis Lussier
```

```
20:50:04,565 [main] INFO   jTPCC : Term-00,  (c) 2016, Jan Wieck
20:50:04,565 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
```

```
[root@AchorBase run]# ./runBenchmark.sh props.yashandb
20:50:04,561 [main] INFO   jTPCC : Term-00,
20:50:04,563 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
20:50:04,563 [main] INFO   jTPCC : Term-00,      BenchmarkSQL v5.0
20:50:04,563 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
20:50:04,563 [main] INFO   jTPCC : Term-00,  (c) 2003, Raul Barbosa
20:50:04,563 [main] INFO   jTPCC : Term-00,  (c) 2004-2016, Denis Lussier
20:50:04,565 [main] INFO   jTPCC : Term-00,  (c) 2016, Jan Wieck
20:50:04,565 [main] INFO   jTPCC : Term-00, +-------------------------------------------------------------+
20:50:04,565 [main] INFO   jTPCC : Term-00,
20:50:04,565 [main] INFO   jTPCC : Term-00, db=yashandb
20:50:04,565 [main] INFO   jTPCC : Term-00, driver=com.yashandb.jdbc.Driver
20:50:04,565 [main] INFO   jTPCC : Term-00, conn=jdbc:yasdb://localhost:1688/yashandb
20:50:04,565 [main] INFO   jTPCC : Term-00, user=sys
20:50:04,565 [main] INFO   jTPCC : Term-00,
20:50:04,565 [main] INFO   jTPCC : Term-00, warehouses=100
20:50:04,565 [main] INFO   jTPCC : Term-00, terminals=10
20:50:04,567 [main] INFO   jTPCC : Term-00, runMins=5
20:50:04,567 [main] INFO   jTPCC : Term-00, limitTxnsPerMin=0
20:50:04,567 [main] INFO   jTPCC : Term-00, terminalWarehouseFixed=true
20:50:04,567 [main] INFO   jTPCC : Term-00,
20:50:04,567 [main] INFO   jTPCC : Term-00, newOrderWeight=45
20:50:04,567 [main] INFO   jTPCC : Term-00, paymentWeight=43
20:50:04,567 [main] INFO   jTPCC : Term-00, orderStatusWeight=4
20:50:04,567 [main] INFO   jTPCC : Term-00, deliveryWeight=4
20:50:04,567 [main] INFO   jTPCC : Term-00, stockLevelWeight=4
20:50:04,567 [main] INFO   jTPCC : Term-00,
20:50:04,567 [main] INFO   jTPCC : Term-00, resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
20:50:04,567 [main] INFO   jTPCC : Term-00, osCollectorScript=./misc/os_collector_linux.py
20:50:04,567 [main] INFO   jTPCC : Term-00,
20:50:04,576 [main] INFO   jTPCC : Term-00, copied props.yashandb to my_result_2023-08-14_205004/run.properties
20:50:04,576 [main] INFO   jTPCC : Term-00, created my_result_2023-08-14_205004/data/runInfo.csv for runID 2
20:50:04,577 [main] INFO   jTPCC : Term-00, writing per transaction results to my_result_2023-08-14_205004/data/result.csv
20:50:04,577 [main] INFO   jTPCC : Term-00, osCollectorScript=./misc/os_collector_linux.py
20:50:04,578 [main] INFO   jTPCC : Term-00, osCollectorInterval=1
20:50:04,578 [main] INFO   jTPCC : Term-00, osCollectorSSHAddr=null
20:50:04,578 [main] INFO   jTPCC : Term-00, osCollectorDevices=null
20:50:04,631 [main] INFO   jTPCC : Term-00,
20:50:04,760 [main] INFO   jTPCC : Term-00, C value for C_LAST during load: 58
20:50:04,761 [main] INFO   jTPCC : Term-00, C value for C_LAST this run:    169
20:50:04,761 [main] INFO   jTPCC : Term-00,
```

测试结果中的tpmC值表示每分钟内系统处理的新订单个数，即系统最大吞吐量。tpmC值常作为性能指标，值越高表示数据库性能越好。