## 一、安装SequoiaSQL

1、安装SequoiaSQL的run包（需要root用户权限，假设run包在/opt目录下）

cd /opt

chmod u+x SequoiaSQL-OLTP-2.8-x86\_64-enterprise-installer.run

./SequoiaSQL-OLTP-2.8-x86\_64-enterprise-installer.run --mode text

2、更改安装目录权限

chown –R sdbadmin:sdbadmin\_group SequoiaSQL

3、切换用户为sdbadmin

su - sdbadmin

4、添加环境变量：（建议写入sdbadmin用户的环境变量中）

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/opt/SequoiaSQL/lib

export PATH=$PATH:/opt/ SequoiaSQL/bin

5、在安装目录下创建SequoiaSQL的数据目录

cd SequoiaSQL

mkdir data

6、初始化数据目录（该操作只能执行一次）

initdb -D data/

## 二、部署SequoiaSQL

1、修改SequoiaSQL的配置文件，日志中增加打印信息、连接信息等

vi data/postgresql.conf

修改以下参数：

log\_connections=on

log\_disconnections=on

log\_line\_prefix=’%m %p %r’

exit\_on\_error=on

2、检查端口是否被占用（需要root权限）

SequoiaSQL默认端口为5432，检查该端口是否被占用

netstat –nap | grep 5432

如果端口被占用或者需要修改端口，则可以修改配置文件postgresql.conf

port=5432

3、启动SequoiaSQL服务进程（切换为sdbadmin用户后执行）

postgres -D data/ >> logfile 2>&1 &

4、检查SequoiaSQL是否启动成功（需要root权限）

netstat –nap | grep 5432

5、创建database

createdb foo

6、进入SequoiaSQL Shell环境

psql -p 5432 foo（端口号为5432时可以省略-p参数）

## 三、安装BenchmarkSQL工具

1、下载安装包

下载地址：<https://sourceforge.net/projects/benchmarksql/>

2、安装

安装配置：JDK1.7

解压安装包

unzip benchmarksql-5.0.zip

备份配置文件

cd /opt/benchmarksql-5.0/run/

cp props.pg props.pg.bak

修改配置文件（按测试条件进行修改）

vi prosps.pg

3、编译java源程序

使用ant工具编译benmarksql中的java程序（安装目录下有build.xml文件）

cd /opt/ benchmarksql-5.0

ant

## 四、使用BenchmarkSQL测试SequoiaSQL

1、创建测试库

首先进入SequoiaSQL Shell环境

psql postgres

创建用户

create user benchmarksql with superuser password ‘bench’

创建数据库

create database benchmarksql owner benchmarksql

退出SequoiaSQL Shell环境

\q

2、根据测试条件修改配置文件props.pg

cd /opt/benchmarksql-5.0/run/

vi props.pg

修改数据库连接、用户名、密码等参数

conn=jdbc:postgresql://localhost:5432/benchmarksql

user=benchmarksql

password=bench

3、创建测试表

./runSQL.sh pros.pg ./sql.common/tableCreates.sql

4、导入测试数据

./runLoader.sh props.pg

5、为测试表创建索引

./runSQL.sh props.pg ./sql.common/indexCreates.sql

6、测试

./runBenchmark.sh props.pg

改变测试条件只需要修改配置文件props.pg即可

7、生成测试报告（需要R语言软件的支持）

./generateReport resultDir

生成报告结果在resultDir目录中的report.html文件中

重复测试如果需要重新导入数据，可以先删除测试表

./runSQL.sh props.pg ./sql.common/tableDrops.sh

## 五、配置文件props.pg的各个参数含义

db=postgres // 数据库

driver=org.postgresql.Driver // 驱动

conn=jdbc:postgresql://localhost:5432/postgres // 数据库连接

user=benchmarksql // 用户名

password=PWbmsql // 密码

warehouses=1 // 仓库数

loadWorkers=4 // 导入数据线程数

terminals=1 // 事务线程数

//To run specified transactions per terminal- runMins must equal zero

runTxnsPerTerminal=10 // 每个线程上的事务数

//To run for specified minutes- runTxnsPerTerminal must equal zero

runMins=0 // 运行时间

//Number of total transactions per minute

limitTxnsPerMin=300 // 每分钟事务数

//Set to true to run in 4.x compatible mode. Set to false to use the

//entire configured database evenly.

terminalWarehouseFixed=true // 是否兼容4.x版本

// 各个事务占总事务的比例

//The following five values must add up to 100

//The default percentages of 45, 43, 4, 4 & 4 match the TPC-C spec

newOrderWeight=45 // 新订单事务

paymentWeight=43 // 支付事务

orderStatusWeight=4 // 订单状态查询事务

deliveryWeight=4 // 发货事务

stockLevelWeight=4 // 库存状态查询事务

// Directory name to create for collecting detailed result data.

// Comment this out to suppress.

resultDirectory=my\_result\_%tY-%tm-%td\_%tH%tM%tS // 结果保存路径

osCollectorScript=./misc/os\_collector\_linux.py // 收集结果脚本

osCollectorInterval=1 // 两次结果收集的间隔时间

//osCollectorSSHAddr=user@dbhost // 结果收集时连接的地址

osCollectorDevices=net\_eth0 blk\_sda // 结果收集的硬件，网卡磁盘

可用于测试条件的几个参数：

warehouses（仓库数）

loadWorkers（导入数据线程数）

terminals（线程数）

runTxnsPerTerminals（每个线程事务数） 这两个参数必须有一个为0

runMins（运行时间）

limitTxnsPerMin（每分钟事务数）

newOrderWeight（新订单事务比例）

paymentWeight（支付事务比例）

orderStatusWeight（订单状态查询事务比例） 五种事务比例总和为100

deliveryWeight（发货事务比例）

stockLevelWeight（库存状态查询事务比例）

BenchmarkSQL推荐配置：

每个仓库的数据量约为100MB，仓库数一般设置为内存的2-5倍。

查看内存大小: cat /proc/meminfo

如 192.168.31.61: 1500M warehouses:30-75

terminal 事务线程数，每个仓库最多10个线程，每个事务都可能在任意一个线程上执行，设置为CPU线程数的2-6倍。

查看CPU线程数：grep 'processor' /proc/cpuinfo | sort -u | wc -l

如 192.168.31.61: 4 terminals:8-24（ < warehouses\*10 ）

runMins 执行时间：至少运行12分钟

运行时间 = runMins 或terminals \* runTxnsPerTerminals / limitTxnsPerMin

TPCC的事务要求规范

|  |  |  |
| --- | --- | --- |
| 事务名 | 比例 | 响应时间 |
| New order | <= 45% | <= 5s |
| Payment | >= 43% | <= 5s |
| Order status | >= 4% | <= 5s |
| Delivery | >= 4% | <= 5s |
| Stock level | >= 4% | <= 20s |

## 六、HOW-TO-RUN.txt推荐运行测试的步骤

1、在postgresql shell中创建用户及数据库

2、编译BenchmarkSQL中的java源代码

3、备份并根据测试条件修改配置文件

cp props.pg my\_postgres.properties

vi my\_postgres.properties

4、测试前准备工作

tableCreates.sql 创建测试表

runLoader.sh 导入数据

indexCreates.sql 创建索引

./runDatabaseBuild.sh my\_postgres.properties foreignKeys.sql 创建外键

extraHistID.sql 为history表添加主键

buildFinish.sql 清理空间

vacuum analyze

5、测试

./runBenchmark.sh my\_postgres.properties

6、删除测试表（非必要）

再次测试如果需要重新导入数据，可以先删除测试表

. /runDatabaseDestroy.sh my\_postgres.properties tableDrops.sql

7、生成测试报告（R语言软件支持）

./ generateReport.sh resultDirectory

生成的测试报告文件为report.html

## 七、测试pg+sdb\_fdw（单主机）

测试pg+sdb\_fdw：关联SequoiaSQL与sequoiadb，将数据存储在sequoiadb数据库中，利用benchmarksql测试SequoiaSQL。

sequoiab数据库部署情况：单主机，集群模式，所有节点部署在单台主机上

1、SequoiaSQL与SequoiaDB建立关联

进入SequoiaSQL shell环境

psql postgres

加载SequoiaDB连接驱动

create extension sdb\_fdw

配置连接参数

create server sdb\_server foreign data wrapper sdb\_fdw options(address ‘127.0.0.1’,server ‘11810’,user ‘sdbadmin’,password ‘mypassword’)

如果没有配置数据库用户及密码，可以忽略user与password两个字段

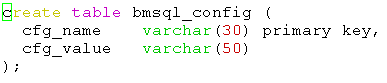
2、备份并修改benchmarksql的sql文件

备份并修改tableCreates.sql

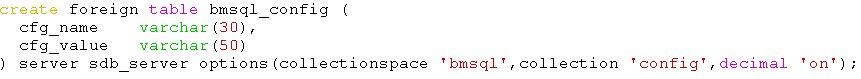
cp tableCreates.sql foreignTableCreates.sql

vi foreignTableCreates.sql

原文：



修改为：(其他表同样进行修改)



将创建普通表的sql语句修改为创建外表，外表对应的集合空间、集合在SequoiadDB中已存在。在SequoiaDB中创建对应的集合空间、集合写入js脚本中执行。

备份并修改tableDrops.sql

cp tableDrops.sql foreignTableDrops.sql

vi foreignTableDrops.sql

原文：



修改为：



将删除普通表的sql语句修改为删除外表

3、根据测试条件修改配置文件props.pg

cp props.pg props.pg.bak

vi props.pg

配置sequoiadb数据库所在主机及其网卡、磁盘

osCollectorSSHAddr=sdbadmin@192.168.31.61

osCollectorDevices=net\_eth0 blk\_sda

4、创建测试表

./runSQL.sh props.pg ./sql.common/foreignTableCreates.sql

5、导入数据

./runLoader.sh props.pg

6、创建索引

根据indexCreates.sql文件为sequoiadb中对应的集合创建索引



对应的sequoiadb shell语句为



为集合创建索引通过执行js脚本完成

7、测试

./runBenchmark.sh props.pg

8、生成测试报告（需要R语言软件的支持）

./generateReport resultDir

生成报告结果在resultDir目录中的report.html文件中

重复测试如果需要重新导入数据，可以先删除测试表

./runSQL.sh props.pg ./sql.common/foreignTableDrops.sh

## 八、测试pg+sdb\_fdw（多主机）

测试pg+sdb\_fdw：关联SequoiaSQL与sequoiadb，将数据存储在sequoiadb数据库中，利用benchmarksql测试SequoiaSQL的性能。

sequoiab数据库部署情况：多主机，集群模式，数据节点部署在多台主机上

1. 为了使benchmarksql支持测试多台主机的性能，修改代码
2. 备份并修改配置文件props.pg

cp props.pg props.pg

vi props.pg

配置osCollectorSSHAddr及osCollecctorDevices等参数

osCollectorSSHAddr= [sdbadmin@192.168.31.20,sdbadmin@192.168.31.21,sdbadmin@192.168.31.19](mailto:sdbadmin@192.168.31.20,sdbadmin@192.168.31.21,sdbadmin@192.168.31.19)

osCollectorDevices=

192.168.31.20:net\_eth0 blk\_sda,

192.168.31.21:net\_eth0 blk\_sda,

192.168.31.19:net\_eth0 blk\_sda blk\_sdc blk\_sdd

收集集群中各个主机的ip地址，网卡，磁盘信息等通过js脚本完成

1. 其他步骤同测试单主机时相同，关联pg与sequoiadb，修改sql文件，调用sql文件创建外表，创建索引，执行测试，生成测试报告，删除测试表等

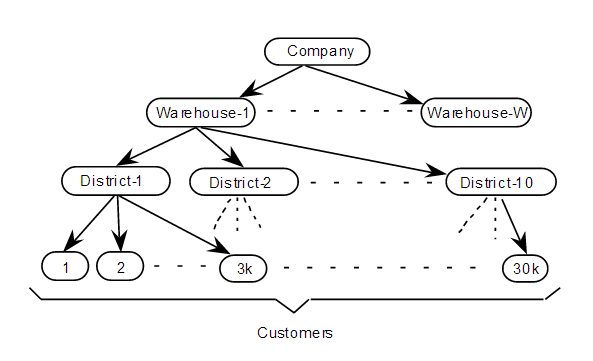
## 附录：

### 1、TPCC简介

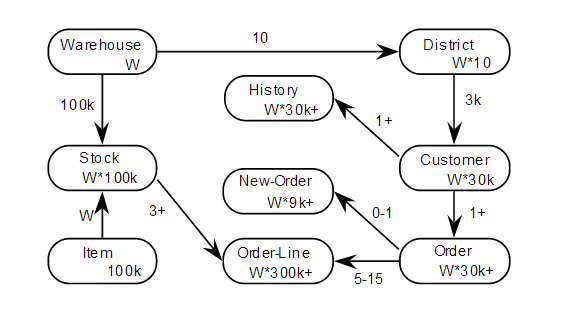
TPCC是专门针对联机交易处理系统（OLTP）的规范。这类系统也被称为业务处理系统，具有如下特点：

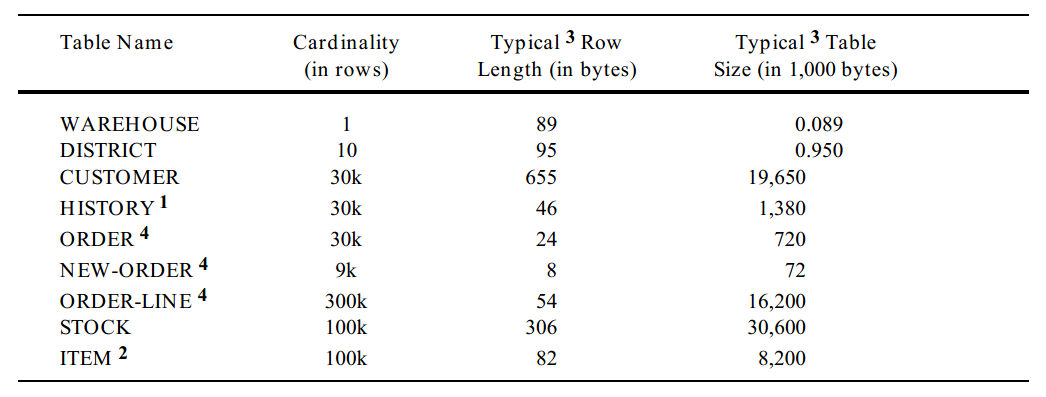
1. 多种事务处理并发执行
2. 在线与离线的事务执行模式
3. 多个在线会话终端
4. 适中的系统运行时间和应用程序运行时间
5. 大量的磁盘IO数据流
6. 强调事务的完整性要求（ACID）
7. 对于非一致的数据库分布，使用主键和从键进行访问
8. 数据库由许多大小不一、属性多样，而又互相关联的数据表组成
9. 存在较多数据访问和更新之间的资源争夺

TPCC的测试模型是一个大型的商品批发销售公司，它拥有若干个分布在不同区域的商品仓库，当业务扩展时，将添加新的仓库。每个仓库维护100000种商品的库存记录，负责为10个销售点供货；每个销售点为3000个客户提供服务；每个客户提交的订单中，平均每个订单有10项产品，所有订单中约1%的产品在其直接所属的仓库中没有存货，需要由其他仓库供货。



该系统的数据库由九张表组成，它们之间的关系如下图





该系统的事务主要有以下几种：

1. 新订单

对于任意一个客户端，从固定的仓库随机选取5-15件商品创建新订单，其中1%的订单由假想的用户操作失败而回滚

中量级、读写频繁、要求响应快

1. 支付

对于任意一个客户端，从固定的仓库随机选取一个区域及其用户，采用随机的金额支付一笔订单，并做相应的历史记录

轻量级、读写频繁、要求响应快

1. 发货

对于任意一个客户端，随机选取一个发货包，更新被处理订单的用户余额，并把订单从新订单中删除

1-10个批量、读写频率低、较宽松的响应时间

1. 订单状态查询

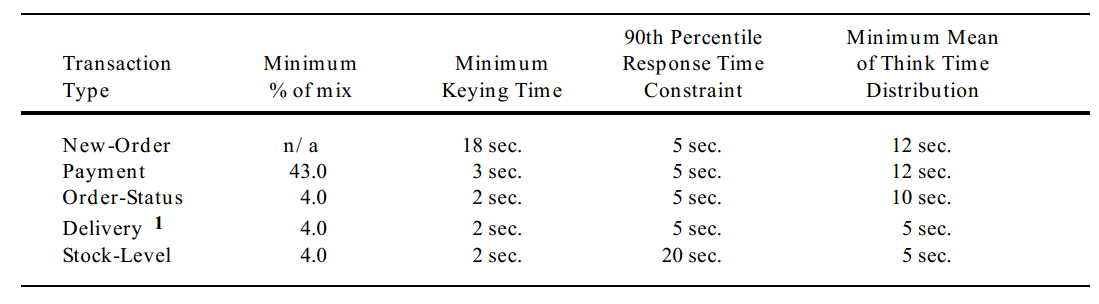
对于任意一个客户端，从固定的仓库随机选取一个区域及其用户，读取最后一条订单，显示订单内每件商品的状态

中量级、只读频率低、要求响应快

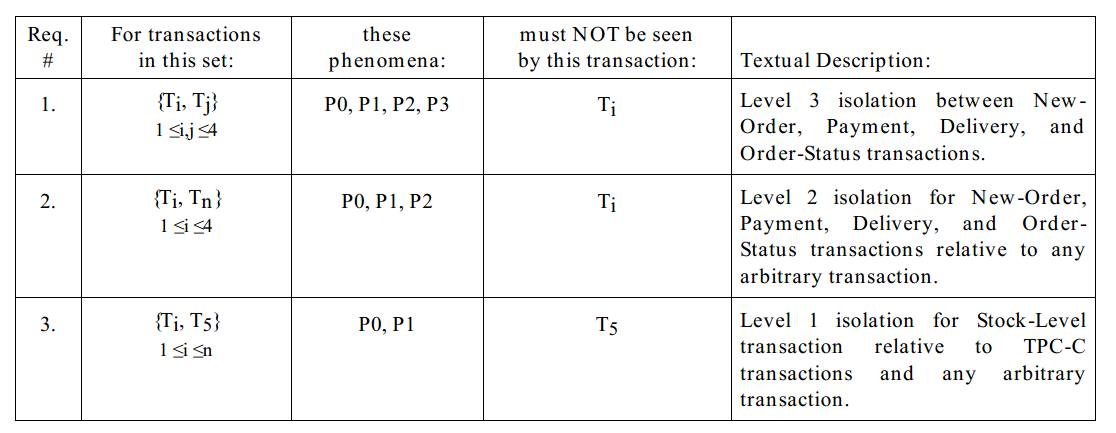
1. 库存状态查询

对于任意一个客户端，从固定的仓库和区域选取最后20条订单，查看订单中所有商品的库存，计算并显示库存数量低于随机生成阈值的商品数量

重量级、只读频率低、较宽松的响应时间



事务之间的隔离性要求：



T1 新订单事务 T2 支付事务 T3 发货事务

T4 订单状态查询事务 T5 库存状态查询事务 Tn 其他任意事务

P0 脏写：一个事务写入的数据被其他事务修改

事务1读取数据记录并修改，事务2修改同一条数据记录或者删除该记录并已提交，当事务1再次读取时是已经被事务2修改后的数据，或者发现该数据已被删除

P1 脏读：一个事务读取到了另一个事务没有提交的数据

事务1修改数据记录，此时事务2在事务1尚未提交之前去读取数据，称为脏读，读取的数据为脏数据。若事务1失败回滚，则事务2读取的数据是一个未提交的数据，不存在于数据库中

P2 不可重复读：同一事务中，两次读取同一数据，得到内容不同

事务1读取数据，事务2修改同一条数据记录或者删除该记录并已提交，当事务1再次读取时是已经被事务2修改后的数据，或者发现该数据已被删除

P3 幻读：同一事务中，两次使用同样方式读取，记录数不同

事务1读取一个数据记录集合（多条数据记录），事务2在这些数据记录中插入或者删除某些数据，当事务1再次读取时，获得的记录数与上一次有差别。

事务隔离的五种级别：

1. transaction\_none 不使用事务
2. transaction\_read\_uncommitted 允许脏读
3. transaction\_read\_committed 防止脏读
4. transaction\_repeatable\_read 防止脏读和不可重复读
5. transaction\_serializable 防止脏读、不可重复读和幻读，事务串行化

TPCC的测试主要有两个指标：

1. 流量指标tpmC

描述系统在执行支付、发货、订单状态查询和库存状态查询的同时，每分钟可以执行的新订单事务数。流量指标越大说明系统的联机事务处理能力越高

1. 性价比

测试系统的整体价格与流量指标的比值，在流量指标相同时，价格越低，性价比越高。