ALIBABA CLOUD

阿里云

云原生数据仓库 AnalyticDB PostgreSQL 版

最佳实践

文档版本：20220511

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.通过pg_stat_activity分析诊断正在执行的SQL

pg_stat_activity是云原生数据仓库AnalyticDB PostgreSQL版用来定位实例当前执行查询的系统视图，每行显示一个服务器进程同时详细描述与之关联的用户会话和查询，可以有效帮助用户分析排查当前运行的SQL任务以及异常问题。

注意事项

只有superuser用户或者是正在报告的进程的拥有者时，才可以使用pg_stat_activity视图。

pg_stat_activity视图的字段描述

字段	类型	描述
datid	oid	数据库OID。
datname	name	数据库名称。
procpid	integer	后端进程的进程ID。 **说明** 只有4.3版本支持procpid字段。
pid	integer	后端进程的进程ID。 **说明** 只有6.0版本支持pid字段。
sess_id	integer	会话ID。
usesysid	oid	用户OID。
usename	name	用户名。
current_query	text	当前正在执行的查询。默认情况下，查询文本最多显示1024个字符，超出部分会被截断，如需显示更多字符，可以通过参数track_activity_query_size配置。 **说明** 只有4.3版本支持current_query字段。
query	text	最近查询的文本。如果 `state` 为 `active` ，显示当前正在执行的查询。在其他状态下，显示上一个执行的查询。 默认情况下，查询文本最多显示1024个字符，超出部分会被截断，如需显示更多字符，可以通过参数track_activity_query_size配置。 **说明** 只有6.0版本支持query字段。

字段	类型	描述
waiting	boolean	如果当前SQL在锁等待，值为True，否则为False。
query_start	datetime	当前活动查询开始执行的时间。如果 `state` 不是 `active` ，显示上一个查询的开始时间。
backend_start	datetime	当前后端进程的开始时间。
backend_xid	xid	后端进程当前的事务ID。
backend_xmin	xid	后端的xmin范围。
client_addr	inet	客户端的IP地址。如果 `client_addr` 为空，表示客户端通过服务器上的Unix套接字连接，或者表示进程是内部进程（例如AUTOVACUUM）。
client_port	integer	客户端和后端通信的TCP端口号。如果使用Unix套接字，值为-1。
client_hostname	text	客户端主机名，通过 `client_addr` 的反向DNS查找报告。
application_name	text	客户端应用名。
xact_start	timestamptz	当前事务的启动时间。如果没有活动事务，值为空。如果当前查询是第一个事务，值与 `query_start` 的值相同。
waiting_reason	text	当前执行等待的原因，可能是等锁或者等待节点间数据的复制。
state	text	后端的当前状态，取值范围：active，idle，idle in transaction，idle in transaction (aborted)，fastpath function call，disabled。 **说明** 只有6.0版本支持state字段。
state_change	timestampz	上次 `state` 状态切换的时间。 **说明** 只有6.0版本支持state_change字段。
rsgid	oid	资源组OID。
rsgname	text	资源组名称。
rsgqueueduration	interval	对于排队查询，查询排队的总时间。

查看连接信息

通过下述SQL确认当前的连接用户和对应的连接机器。

```
SELECT datname,usename,client_addr,client_port FROM pg_stat_activity ;
```

```
datname   |  usename   |  client_addr   | client_port
----------+------------+----------------+-------------
postgres | joe        |  xx.xx.xx.xx   |       60621
postgres | gpmon      |  xx.xx.xx.xx   |       60312
(9 rows)
```

查看SQL运行信息

获取当前用户执行SQL信息：

6.0版本：

```
SELECT datname,usename,query FROM pg_stat_activity ;
```

```
 datname   | usename  |                        query
-----------+----------+-----------------------------------------------------------
 postgres | postgres | SELECT datname,usename,query FROM pg_stat_activity ;
 postgres | joe      |
(2 rows)
```

4.3版本：

```
SELECT datname,usename,current_query FROM pg_stat_activity ;
```

```
 datname   | usename  |                        current_query
-----------+----------+-----------------------------------------------------------
 postgres | postgres | SELECT datname,usename,current_query FROM pg_stat_activity ;
 postgres | joe      | <IDLE>
(2 rows)
```

获取当前正在运行的SQL信息：

6.0版本：

```
SELECT datname,usename,query
   FROM pg_stat_activity
   WHERE state != 'idle' ;
```

4.3版本：

```
SELECT datname,usename,current_query
   FROM pg_stat_activity
   WHERE current_query != '<IDLE>' ;
```

查看耗时较长的查询

查看当前运行中的耗时较长的SQL语句：

6.0版本：

```
select current_timestamp - query_start as runtime, datname, usename, query
    from pg_stat_activity
    where state != 'idle'
    order by 1 desc;
```

4.3版本：

```
select current_timestamp - query_start as runtime, datname, usename, current_query
    from pg_stat_activity
    where current_query != '<IDLE>'
    order by 1 desc;
```

返回示例如下：

```
runtime         |    datname     | usename  |                                 current_query
----------------+---------------+---------+-----------------------------------------------
------------------------------
 00:00:34.248426 | tpch_1000x_col | postgres | select
                                           :         l_returnflag,
                                           :         l_linestatus,
                                           :         sum(l_quantity) as sum_qty,
                                           :         sum(l_extendedprice) as sum_base_pri
ce,
                                           :         sum(l_extendedprice * (1 - l_discoun
t)) as sum_disc_price,
                                           :         sum(l_extendedprice * (1 - l_discoun
t) * (1 + l_tax)) as sum_charge,
                                           :         avg(l_quantity) as avg_qty,
                                           :         avg(l_extendedprice) as avg_price,
                                           :         avg(l_discount) as avg_disc,
                                           :         count(*) as count_order
                                           : from
                                           :         public.lineitem
                                           : where
                                           :         l_shipdate <= date '1998-12-01' - in
terval '93' day
                                           : group by
                                           :         l_returnflag,
                                           :         l_linestatus
                                           : order by
                                           :         l_returnflag,
                                           :         l_linestatus;
 00:00:00        | postgres       | postgres | select
                                           :        current_timestamp - query_start as ru
ntime,
                                           :        datname,
                                           :        usename,
                                           :        current_query
                                           :     from pg_stat_activity
                                           :     where current_query != '<IDLE>'
                                           :     order by 1 desc;
(2 rows)
```

可以看到第一个查询耗时较久，已经运行了34s还没有结束。

异常SQL诊断及修复

如果一个SQL运行很长时间没有返回结果，需要检查该SQL还在运行中还是已经被Block:

6.0版本：

```
SELECT datname,usename,query
   FROM pg_stat_activity
   WHERE waiting;
```

4.3版本：

```
SELECT datname,usename,current_query
   FROM pg_stat_activity
   WHERE waiting;
```

需要注意的是上述返回结果只能获取当前因为Lock而被Block的SQL，无法获取因为其他原因被Block的SQL。绝大多数情况下SQL都是因为Lock而被Block，但也会存在其他情况，例如等待I/O、定时器等。如果上述SQL有返回结果，说明有SQL被Lock阻塞，需要进一步明确相互Block的SQL信息：

```
SELECT
        w.query as waiting_query,
        w.pid as w_pid,
        w.usename as w_user,
        l.query as locking_query,
        l.pid as l_pid,
        l.usename as l_user,
        t.schemaname || '.' || t.relname as tablename
    from pg_stat_activity w
    join pg_locks l1 on w.pid = l1.pid and not l1.granted
    join pg_locks l2 on l1.relation = l2.relation and l2.granted
    join pg_stat_activity l on l2.pid = l.pid
    join pg_stat_user_tables t on l1.relation = t.relid
    where w.waiting;
```

通过上述SQL的返回结果可以确认相互Block的SQL和对应的执行pid。在明确了SQL的阻塞信息后，可以通过Cancel或者Kill Query的方式进行恢复。通过Cancel取消一个正在运行的Query：

```
SELECT pg_cancel_backend(pid)
```

需要在一个运行Query的Session中执行，如果Session本身就是Idle的，执行不起作用。另外取消这个Query需要花费一定的时间来做清理和事务的回滚。使用pg_terminate_backend来清理Idle Session，也可以用来终止Query：

```
SELECT pg_terminate_backend(pid);
```

该用户的连接会断开。尽量避免在正在运行Query的进程pid上执行。需要注意的是文中提到操作需要用户有superuser的权限。

2.导入中特殊符号处理

云数据库AnalyticDB for PostgreSQL支持多种数据导入方法，数据迁移及同步方案综述。

在通过OSS高速并行导入和通过COPY数据导入过程中，经常因为存在特殊字符导致导入失败。本文介绍了预先处理导入数据中的特殊字符的方法，从而消除特殊字符带来的问题。

OSS高速并行导入

在数据导入过程中，一般是将文件的每行作为一个元组，通过在每行中规定分隔符来分割每一列的数据。下文首先介绍分隔符的使用方法和约束，然后介绍在每列中遇到特殊符号的处理方法。

分隔符

在创建OSS外部表语法中，您可以通过在FORMAT子句后面指定DELIMITER分隔符，如下：

```
FORMAT 'TEXT' (DELIMITER ',')
```

- 如果 `FORMAT 'TEXT'` ，则 `DELIMITER` 缺省值为 `'\t'` 。
- 如果 `FORMAT 'CSV'` ，则 `DELIMITER` 缺省值为 `','` 。

您也可以自定义DELIMITER，但是创建外部表语法中自定义的DELIMITER必须满足以下约束：

- 必须是一个ASCII字符，不允许是汉字或者2个以及以上ASCII字符。
- 不支持 `'\n'` 和 `'\r'` 。
- 支持除 `'\n'` 和 `'\r'` 之外的其他转义字符，使用时前面加E或者e。
- 支持前面不加E的转义字符 `'\t'` 。
- 如果是TEXT模式，可以设置DELIMITER为OFF，支持单列外部表。

为了能够正常读取数据，您提供的OSS文件内容必须严格遵守设置的DELIMITER。

数据中的特殊符号

在数据导入过程中，出现特殊符号的场景可以分为以下几种：

- 列中存在和DELIMITER相同的字符。
  - 如果您使用TEXT模式，则需要在每个DELIMITER字符前加ESCAPE符。ESCAPE符可以在创建外表时使用以下命令指定，缺省值为反斜杠（\）。

    ```
    FORMAT 'TEXT' (ESCAPE '\' )
    ```

  - 如果您使用的是CSV模式，则需要在每个DELIMITER字符前加双引号（”）。
- 列中存在中文。OSS外表支持中文数据，但是为了保证显示正确，您需要在创建外表时设置如下编码格式：

  ```
  ENCODING 'UTF8'
  ```

- 列中存在null。您可以设置null对应的匹配字符，在导入数据时将对应的字符匹配为null。CSV模式下缺省值为不带引号的空值，TEXT模式下缺省值为\N。以下命令将空格作为null的匹配字符，如果该列为空格，则在使用OSS文件导入的数据中该列值为null。

  ```
  FORMAT 'text' (null ' ' )
  ```

- 列中存在转义字符。您可以在转义字符前增加ESCAPE符。ESCAPE符在创建外表时指定，CSV模式缺省值为双引号（”），TEXT模式缺省值为反斜杠（\）。
  - 您可以自定义ESCAPE为单个字符。例如，以下命令将ESCAPE设置为反斜杠：

    ```
    FORMAT 'csv' (ESCAPE '\' )
    ```

  - 您也可以设置ESCAPE为OFF，避免所有字符被自动转义。
- 列中存在单引号或者双引号。
  - 如果您使用TEXT模式，需要在单引号或者双引号前面增加ESCAPE符，默认为反斜杠（\）。
  - 如果您使用CSV模式，需要在单引号或者双引号前面增加ESCAPE符，默认为双引号（”），同时在该列前后加双引号（”），将整列括起来。

COPY数据导入

您在使用\COPY语句导入数据时，分隔符的使用方法和OSS高速并行导入时的使用方法一样，而对数据中出现特殊符号的处理方法也和OSS高速并行导入相类似。不同的是COPY语句和 `CREATE EXTERNAL TABLE` 语句用法略有不同，COPY语句详细用法见使用COPY命令导入数据。

3.TPC-H

AnalyticDB PostgreSQL 6.0版在支持ACID和分布式事务的同时，提供了优秀的大数据MPP分析性能。本文讲述如何运行TPC-H进行测试。

TPC-H简介

以下文字描述引用自TPC Benchmark™ H (TPC-H)规范：

“TPC-H是一个决策支持基准，由一套面向业务的临时查询和并发数据修改组成。选择的查询和填充数据库的数据具有广泛的行业相关性。该基准测试说明了决策支持系统可以检查大量数据，执行高度复杂的查询，并解答关键的业务问题。”

详情请参见TPCH Specification。

> **说明**
>
> 本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

前提条件

- 已注册阿里云账号。
- 已创建用于测试的AnalyticDB PostgreSQL实例。创建实例具体操作，请参见创建实例。

  本文测试的AnalyticDB PostgreSQL实例规格如下：
  - 引擎版本：6.0标准版
  - 节点规格（segment）：2C16G
  - 节点数量（segment）：32
  - 存储磁盘类型：ESSD云盘
  - 节点存储容量（segment）：200 GB
- 已创建用于测试的ECS实例。创建实例具体操作，请参见创建实例。

  本文测试的ECS实例规格如下：
  - 实例规格：ecs.g6e.4xlarge
  - 操作系统：CentOS 7.x
  - 系统盘：磁盘类型为ESSD云盘、容量为40 GiB、性能级别为PL1。
  - 数据盘：磁盘类型为ESSD云盘、容量为2048 GiB、性能级别为PL3。

    > **说明**
    >
    > ECS实例创建完成后需要挂载数据库，具体操作，请参见分区格式化数据盘（Linux）。
- 已开通OSS并创建存储空间。
- 已将ECS实例的IP地址添加到AnalyticDB PostgreSQL实例的白名单中，操作方式，请参见设置白名单。
- 已在ECS上安装psql，安装方式，请参见psql。

生成测试数据

1. 登录ECS实例，登录方式，请参见连接ECS实例。
2. 在ECS上执行以下命令，下载TPC-H dbgen代码到数据盘并编译。

   本文中数据盘的路径为 `/mnt` 。

```
wget https://github.com/electrum/tpch-dbgen/archive/refs/heads/master.zip
yum install -y unzip zip
unzip master.zip
cd tpch-dbgen-master/
echo "#define EOL_HANDLING 1" >> config.h # 消除生成数据末尾的'|'
make
./dbgen --help
```

3. 在ECS上执行如下命令，生成1 TB测试数据集。建议使用分片文件，分片数量与AnalyticDB PostgreSQL版实例节点数量一致，例如本文中示例AnalyticDB PostgreSQL实例有32个计算节点，以下示例中将创建32个分片文件。

```
for((i=1;i<=32;i++));
do
    ./dbgen -s 1000 -S $i -C 32 -f &
done
```

> **说明**
>
> - 数据量的大小对查询速度有直接的影响，TPC-H中使用SF描述数据量，1SF对应1 GB单位。1000SF，即1 TB。1SF对应的数据量只是8个表的总数据量不包括索引等空间占用，准备数据时需预留更多空间。
> - 后台运行dbgen程序时间较长，您可以通过 `ps- fHU $USER | grep dbgen` 命令查看进度，确保dbgen程序运行完成。

创建测试表

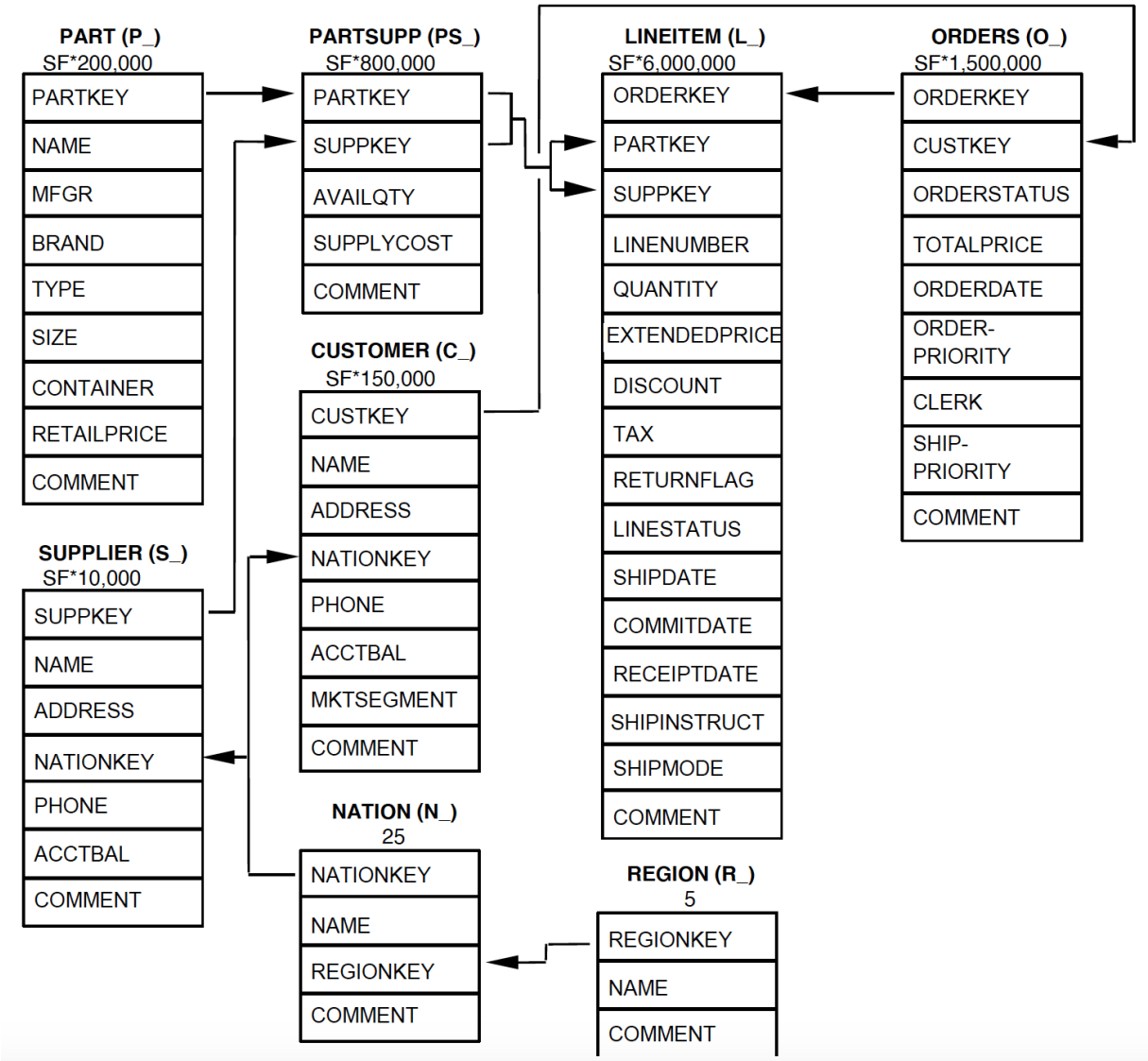
1. 使用psql连接AnalyticDB PostgreSQL，连接方式，请参见psql。
2. 执行如下语句创建用于测试的8张表。

```
CREATE TABLE NATION (
    N_NATIONKEY  INTEGER NOT NULL,
    N_NAME       CHAR(25) NOT NULL,
    N_REGIONKEY  INTEGER NOT NULL,
    N_COMMENT    VARCHAR(152)
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED Replicated
;

CREATE TABLE REGION (
    R_REGIONKEY  INTEGER NOT NULL,
    R_NAME       CHAR(25) NOT NULL,
    R_COMMENT    VARCHAR(152)
```

```
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED Replicated
;

CREATE TABLE PART (
    P_PARTKEY     INTEGER NOT NULL,
    P_NAME        VARCHAR(55) NOT NULL,
    P_MFGR        CHAR(25) NOT NULL,
    P_BRAND       CHAR(10) NOT NULL,
    P_TYPE        VARCHAR(25) NOT NULL,
    P_SIZE        INTEGER NOT NULL,
    P_CONTAINER   CHAR(10) NOT NULL,
    P_RETAILPRICE DECIMAL(15,2) NOT NULL,
    P_COMMENT     VARCHAR(23) NOT NULL
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (P_PARTKEY)
;

CREATE TABLE SUPPLIER (
    S_SUPPKEY     INTEGER NOT NULL,
    S_NAME        CHAR(25) NOT NULL,
    S_ADDRESS     VARCHAR(40) NOT NULL,
    S_NATIONKEY   INTEGER NOT NULL,
    S_PHONE       CHAR(15) NOT NULL,
    S_ACCTBAL     DECIMAL(15,2) NOT NULL,
    S_COMMENT     VARCHAR(101) NOT NULL
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (S_SUPPKEY)
;

CREATE TABLE PARTSUPP (
    PS_PARTKEY     INTEGER NOT NULL,
    PS_SUPPKEY     INTEGER NOT NULL,
    PS_AVAILQTY    INTEGER NOT NULL,
    PS_SUPPLYCOST  DECIMAL(15,2)  NOT NULL,
    PS_COMMENT     VARCHAR(199) NOT NULL
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (PS_PARTKEY)
;

CREATE TABLE CUSTOMER (
    C_CUSTKEY     INTEGER NOT NULL,
    C_NAME        VARCHAR(25) NOT NULL,
    C_ADDRESS     VARCHAR(40) NOT NULL,
    C_NATIONKEY   INTEGER NOT NULL,
    C_PHONE       CHAR(15) NOT NULL,
    C_ACCTBAL     DECIMAL(15,2)   NOT NULL,
    C_MKTSEGMENT  CHAR(10) NOT NULL,
    C_COMMENT     VARCHAR(117) NOT NULL
)
```

```
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (C_CUSTKEY)
;

CREATE TABLE ORDERS (
    O_ORDERKEY       BIGINT NOT NULL,
    O_CUSTKEY        INTEGER NOT NULL,
    O_ORDERSTATUS    "char" NOT NULL,
    O_TOTALPRICE     DECIMAL(15,2) NOT NULL,
    O_ORDERDATE      DATE NOT NULL,
    O_ORDERPRIORITY  CHAR(15) NOT NULL,
    O_CLERK          CHAR(15) NOT NULL,
    O_SHIPPRIORITY   INTEGER NOT NULL,
    O_COMMENT        VARCHAR(79) NOT NULL
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (O_ORDERKEY)
ORDER BY(O_ORDERDATE)
;

CREATE TABLE LINEITEM (
    L_ORDERKEY    BIGINT NOT NULL,
    L_PARTKEY     INTEGER NOT NULL,
    L_SUPPKEY     INTEGER NOT NULL,
    L_LINENUMBER  INTEGER NOT NULL,
    L_QUANTITY    DECIMAL(15,2) NOT NULL,
    L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
    L_DISCOUNT    DECIMAL(15,2) NOT NULL,
    L_TAX         DECIMAL(15,2) NOT NULL,
    L_RETURNFLAG  "char" NOT NULL,
    L_LINESTATUS  "char" NOT NULL,
    L_SHIPDATE    DATE NOT NULL,
    L_COMMITDATE  DATE NOT NULL,
    L_RECEIPTDATE DATE NOT NULL,
    L_SHIPINSTRUCT CHAR(25) NOT NULL,
    L_SHIPMODE     CHAR(10) NOT NULL,
    L_COMMENT      VARCHAR(44) NOT NULL
)
WITH (APPENDONLY=TRUE, ORIENTATION=COLUMN, COMPRESSTYPE=LZ4, COMPRESSLEVEL=9)
DISTRIBUTED BY (L_ORDERKEY)
ORDER BY(L_SHIPDATE)
;
```

8张表的逻辑关系如下图所示。



（图片来源于：TPC Benchmark H Standard Specification）

导入数据

以下内容为您介绍如何导入数据到AnalyticDB PostgreSQL数据库中。

1. 在AnalyticDB PostgreSQL数据库中使用COPY命令导入nation和region两张小表，命令如下：

```
\copy nation from '/mnt/tpch-dbgen-master/nation.tbl' DELIMITER '|';
\copy region from '/mnt/tpch-dbgen-master/region.tbl' DELIMITER '|';
```

> **说明**
>
> 上述示例中的 `/mnt/tpch-dbgen-master` 为示例路径，测试过程中请替换成 `nation.tbl` 和 `region.tbl` 文件的实际路径。

2. 将其余的6张大表通过ossutil上传到OSS。更多关于ossutil的使用方法，请参见概述。

> **说明**
>
> 由于\COPY命令需要通过Master节点进行串行数据写入处理，无法实现并行写入大批量数据，因此需要通过OSS进行数据导入。

i. 在ECS上下载ossutil，命令如下：

```
wget http://gosspublic.alicdn.com/ossutil/1.7.3/ossutil64
```

ii. 修改文件的执行权限，命令如下：

```
chmod 755 ossutil64
```

iii. 执行如下语句，依次将其余6张表的 `.tbl` 文件通过ossutil上传到OSS。

```
ls <table_name>.tbl* | while read line;
do
~/ossutil64 -e <EndPoint> -i <AccessKey ID> -k <Access Key Secret> cp $line oss://<
oss bucket>/<目录>/ &
done
```

3. 待所有表上传到OSS后，在AnalyticDB PostgreSQL数据库中执行如下语句将数据从OSS导入表。具体信息，请参见使用COPY/UNLOAD导入/导出数据到OSS。

```
COPY customer
FROM 'oss://<oss bucket>/<目录>/customer.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
FORMAT AS text
"delimiter" '|'
"null" ''
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;

COPY lineitem
FROM 'oss://<oss bucket>/<目录>/lineitem.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
FORMAT AS text
"delimiter" '|'
"null" ''
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;

-- lineitem表定义了排序列，数据导入完成后可对数据进行聚簇排序
sort lineitem;

COPY orders
FROM 'oss://<oss bucket>/<目录>/orders.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
```

```
FORMAT AS text
"delimiter" '|'
"null" ''
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;

-- orders表定义了排序列，数据导入完成后可对数据进行聚簇排序
sort orders;

COPY part
FROM 'oss://<oss bucket>/<目录>/part.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
FORMAT AS text
"delimiter" '|'
"null" ''
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;

COPY supplier
FROM 'oss://<oss bucket>/<目录>/supplier.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
FORMAT AS text
"delimiter" '|'
"null" ''
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;

COPY partsupp
FROM 'oss://<oss bucket>/<目录>/partsupp.tbl'
ACCESS_KEY_ID '<AccessKey ID>'
SECRET_ACCESS_KEY '<Access Key Secret>'
FORMAT AS text
"delimiter" '|'
ENDPOINT '<EndPoint>'
FDW 'oss_fdw'
;
```

执行查询

您可以通过Shell脚本进行测试，也可以通过psql等客户端工具逐条执行查询SQL，以下内容将分别介绍两种执行查询的方法。

Shell脚本

1. 下载tpch_query .tar.gz文件并解压到 `~/tpch_query` 目录。
2. 创建一个名为 `query.sh` 的Shell脚本，用于执行全部查询，并记录每条耗时和总耗时。

   Shell脚本内容如下：

```
#!/bin/bash

total_cost=0

for i in {1..22}
do
        echo "begin run Q${i}, tpch_query/q$i.sql , `date`"
        begin_time=`date +%s.%N`
        ./psql ${实例连接地址} -p ${端口号} -U ${数据库用户} -f ~/tpch_query/q${i}.sql > ~/
log/log_q${i}.out
        rc=$?
        end_time=`date +%s.%N`
        cost=`echo "$end_time-$begin_time"|bc`
        total_cost=`echo "$total_cost+$cost"|bc`
        if [ $rc -ne 0 ] ; then
             printf "run Q%s fail, cost: %.2f, totalCost: %.2f, `date`\n" $i $cost $to
tal_cost
         else
             printf "run Q%s succ, cost: %.2f, totalCost: %.2f, `date`\n" $i $cost $to
tal_cost
         fi
done
```

3. 在ECS上运行 `query.sh` 脚本。

```
nohup bash ~/query.sh > /tmp/tpch.log &
```

4. 执行如下命令查看结果。

```
cat /tmp/tpch.log
```

客户端工具

连接AnalyticDB PostgreSQL数据库后，逐条执行以下语句进行查询，并对比查询结果。

```
--创建向量化计算引擎Laser插件
create extension if not exists laser;

-- Q1
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
```

```
where
    l_shipdate <= date '1998-12-01' - interval '93 day'
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;

-- Q2
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 23
    and p_type like '%STEEL'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'EUROPE'
    )
order by
    s_acctbal desc,
    n_name,
    s_name,
```

```
    p_partkey
limit 100;

-- Q3
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'MACHINERY'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '1995-03-24'
    and l_shipdate > date '1995-03-24'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate
limit 10;

-- Q4
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '1996-08-01'
    and o_orderdate < date '1996-08-01' + interval '3' month
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
```

```
-- Q6
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1994-01-01'
    and l_shipdate < date '1994-01-01' + interval '1' year
    and l_discount between 0.06 - 0.01 and 0.06 + 0.01
    and l_quantity < 24;

-- Q7
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            extract(year from l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'JORDAN' and n2.n_name = 'INDONESIA')
                or (n1.n_name = 'INDONESIA' and n2.n_name = 'JORDAN')
            )
            and l_shipdate between date '1995-01-01' and date '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation
```

```
    case_nation,
    l_year;

-- Q8
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    o_year,
    sum(case
        when nation = 'INDONESIA' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'ASIA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between date '1995-01-01' and date '1996-12-31'
            and p_type = 'STANDARD BRUSHED BRASS'
    ) as all_nations
group by
    o_year
order by
    o_year;

-- Q9
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
```

```
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%chartreuse%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;

-- Q10
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= date '1994-08-01'
    and o_orderdate < date '1994-08-01' + interval '3' month
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
```

```
    n_name,
    c_address,
    c_comment
order by
    revenue desc
limit 20;

-- Q11
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'INDONESIA'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) * 0.0001000000
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'INDONESIA'
        )
order by
    value desc;

-- Q12
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
```

```
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('REG AIR', 'TRUCK')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '1994-01-01'
    and l_receiptdate < date '1994-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;

-- Q13
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%pending%requests%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;

-- Q14
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
```

```
    and l_shipdate >= date '1994-11-01'
    and l_shipdate < date '1994-11-01' + interval '1' month;

-- Q15
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
create view revenue0 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= date '1997-10-01'
        and l_shipdate < date '1997-10-01' + interval '3' month
    group by
        l_suppkey;
select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue0
    )
order by
    s_suppkey;
drop view revenue0;

-- Q16
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#44'
    and p_type not like 'SMALL BURNISHED%'
    and p_size in (36, 27, 34, 45, 11, 6, 25, 16)
    and ps_suppkey not in (
```

```
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;

-- Q17
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#42'
    and p_container = 'JUMBO PACK'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );

-- Q18
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
```

```
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 312
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate
limit 100;

-- Q19
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#43'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 5 and l_quantity <= 5 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 12 and l_quantity <= 12 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#11'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
```

```
        and l_quantity >= 24 and l_quantity <= 24 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );

-- Q20
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'magenta%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= date '1996-01-01'
                    and l_shipdate < date '1996-01-01' + interval '1' year
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'RUSSIA'
order by
    s_name;

-- Q21
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
    s_name,
    count(*) as numwait
from
```

```
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'MOZAMBIQUE'
group by
    s_name
order by
    numwait desc,
    s_name
limit 100;

-- Q22
-- 开启向量加速引擎，并设置开关变量为on
set laser.enable = on;
select
        cntrycode,
        count(*) as numcust,
        sum(c_acctbal) as totacctbal
from
        (
                select
                        substring(c_phone from 1 for 2) as cntrycode,
                        c_acctbal
                from
                        customer
                where
                        substring(c_phone from 1 for 2) in
                                ('13', '31', '23', '29', '30', '18', '17')
```

```
                        and c_acctbal > (
                                select
                                        avg(c_acctbal)
                                from
                                        customer
                                where
                                        c_acctbal > 0.00
                                        and substring(c_phone from 1 for 2) in
                                                ('13', '31', '23', '29', '30', '18', '17')
                        )
                        and not exists (
                                select
                                        *
                                from
                                        orders
                                where
                                        o_custkey = c_custkey
                        )
        ) as custsale
group by
        cntrycode
order by
        cntrycode;
```