

《数据库系统原理》课程实验指导

《TPC-H Benchmark》数据库 数据建模



(版本：V 2.0)

2022 年 9 月

目 录

1. 数据处理测试基准 TPC Benchmark	4
1.1 背景.....	4
1.2 分类.....	4
1.3 TPC 国际组织及官网主页.....	4
1.4 数据下载地址.....	5
2. TPC-H Benchmark 测试及数据集.....	6
2.1 关系模式图	6
2.1.1 订单表 ORDERS	6
2.1.2 区域表 REGION.....	7
2.1.3 国家表 NATION	7
2.1.4 供应商表 SUPPLIER.....	7
2.1.5 零部件表 PART	8
2.1.6 零部件供应表 PARTSUPP.....	8
2.1.7 客户表 CUSTOMER	9
2.1.8 订单明细表 LINEITEM	9
2.2 数据对象 mapping cardinality/cardinality limit 及表间参照完整性约束.....	12
2.3 TPC-H 关系模式及其属性	13
2.3.1 订单表 ORDERS	13
2.3.2 供应商表 SUPPLIER.....	13
2.3.3 地区表 REGION.....	13
2.3.4 国家表.....	14
2.3.5 零部件表.....	14
2.3.6 零部件供应表.....	14
2.3.7 客户表.....	15
2.3.8 订单明细表.....	15
3. TPC-H Benchmark 数据集生成.....	16
3.1 基于脚本的数据生成	16
3.2 生成的关系表数据	16
3.3 数据生成脚本示例	16
4. 典型 TPC-H 测试语句	18
4.1 定价汇总报表查询	18
4.2 最低成本供应商查询	18
4.3 运输优先级查询	20
4.4 订单优先级查询	20
4.5 本地供应商收入量查询	21
4.6 预测收入变化查询	22
4.7 批量出货查询	22
4.8 全国市场份额查询	23
4.9 产品类型利润度量查询	24
4.10 退货报告查询	25
4.11 重要库存识别查询	26

4.12	运送方式和订单优先级查询	26
4.13	客户分布查询	27
4.14	促销效果查询	28
4.15	顶级供应商查询	28
4.16	零部件/供应商关系查询	29
4.17	小额订单收入查询	30
4.18	大批量客户查询	31
4.19	折扣收入查询	32
4.20	潜在零部件促销查询	33
4.21	不能按时交货供应商查询	34
4.22	全球销售机会查询	35
5.	TPC-H 数据库建库脚本	36
5.1	建表及数据导入步骤	36
5.2	订单表 ORDERS.....	36
5.3	区域表 REGION	36
5.4	国家表 NATION	36
5.5	供应商表 SUPPLIER.....	37
5.6	零部件表 PART.....	37
5.7	零部件供应表 PARTSUPP	37
5.8	客户表 CUSTOMER.....	37
5.9	订单明细表 LINEITEM.....	38
5.10	约束	38
6.	TPC-C 测试基准.....	40
6.1	实体-联系图	40
6.2	关系表	41
6.2.1	仓库表 WAREHOUSE	41
6.2.2	区域表 DISTRICT	41
6.2.3	客户表 CUSTOMER	42
6.2.4	历史记录表 HISTORY	43
6.2.5	新订单表 NEW-ORDER	43
6.2.6	订单表 ORDER	44
6.2.7	订单行表 ORDER-LINE	44
6.2.8	条目表 ITEM	45
6.2.9	库存表 STOCK.....	45
6.3	基准测试程序示例	46
6.3.1	基于嵌入式 SQL 的测试基准程序.....	46
6.3.2	A1 新订单事务 The New-Order Transaction	46
6.3.3	A2 The Payment Transaction	49

1. 数据处理测试基准 TPC Benchmark

1.1 背景

TPC (Transaction Processing Performance Council, 事务处理性能委员会) 是一个非盈利委员会, 专注于开发以数据为中心的基准标准, 并向行业传播客观、可验证的数据。TPC 针对不同的系统提出了不同的测试基准 Benchmark。这些测试基准针对不同的系统设计了对应的数据模型, 并设计了与系统在实际应用中对应的数据量单位, 然后将数据导入系统中, 按照不同的需求进行模拟操作, 根据对应的评价指标来评价系统的性能。测试基准包括测试数据、测试方法/程序等。

1.2 分类

TPC 测试基准包括如下几类:

1. 联机在线事务处理系统 (OLTP) 测试标准: TPC-C、TPC-E (最新)
2. 决策支持 (DS) 测试标准: TPC-H、TPC-DS(decision support)、TPC-DI(Data Integration)
3. 大数据测试标准: TPCx-HS(Hadoop, Spark)、TPCx-BB(Big Bench)
4. 服务器虚拟化 (VMS) 测试标准: TPC-VMS、TPCx-V
5. 物联网测试标准: TPCx-IoT(Internet of things)
6. 人工智能测试标准: TPCx-AI(Artificial intelligence)

1.3 TPC 国际组织及官网主页

英文全称: Transaction Processing Performance Council

中文全称: 事务处理性能委员会

官网主页: <https://www.tpc.org/default5.asp>

目前 TPC 组织的主要成员如下:



图 1 TPC 组织成员

1.4 数据下载地址

https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

2. TPC-H Benchmark 测试及数据集

TPC-H 是用于评估在线分析处理的基准程序和测试基准数据集，模拟了供应商和采购商之间的交易行为，其中包含针对 8 张表的 22 条分析型查询。

TPC-H 采用雪花模型，一共有 8 张表，其中 `nation`（国家）和 `region`（区域）两张表的数据量是固定的，其余 6 张表的数据量跟比例因子 `SF`（`Scale Factor`）相关，可以指定为 1、100、1000 等，分别代表 1GB、100GB、1000GB，根据指定的 `SF` 确定每张表的数据量。

例如，`tpclg` 的 `part` 表，其数据量=基础数据量 $200000 \times 1 = 200000$ （行），`tpch100g` 的 `part` 表，其数据量= $200000 \times 100 = 20000000$ （行）。

2.1 关系模式图

TPC-H 测试基准涵盖电子商务领域的 8 张关系表，包括：

2.1.1 订单表 **ORDERS**

订单表记录客户的订单信息，包含以下表项：

`o_orderkey` 订单 key

`o_custkey` 客户 key

`o_orderstatus` 订单状态

`o_totalprice` 订单总价

`o_orderdate` 下单日期

`o_orderpriority` 订单优先级

`o_clerk` 收银员

`o_shippriority` 发货优先级

`o_comment` 备注

主键为 `o_orderkey`，取值范围 $1 \sim SF \times 1500000$ ；

与表 `lineitem` 表间有外键/参照完整性关联。

2.1.2 区域表 REGION

区域表记录商品销售涉及的区域信息，由亚洲，美洲，非洲，欧洲，中东组成，包含以下表项：

r_regionkey 地区 key

r_name 地区名

r_comment 备注

主键为 r_regionkey，固定有 5 个地区。

2.1.3 国家表 NATION

国家表记录商品销售涉及的国家信息，包含以下表项：

n_nationkey 国家 key

n_name 国家名

n_regionkey 国家对应的区域 key

n_comment 备注

主键为 n_nationkey，有 25 个国家，这是固定值。

2.1.4 供应商表 SUPPLIER

供应商表记录商品销售的供应商信息，包含以下表项：

s_suppkey 供应商 key

s_name 供应商名字

s_address 供应商地址

s_nationkey 供应商国家 key

s_phone 供应商手机号

s_acctbal 供应商账户余额

s_comment 备注

主键为 s_suppkey，取值范围 1~ SF *10000；

与表 partsupp、customer、nation 间有外键/参照完整性关联。

2.1.5 零部件表 PART

零部件表记录商品销售的零件信息，包含以下表项：

p_partkey 零件 key

p_name 零件名称

p_mfgr 零件厂商

p_brand 零件品牌

p_type 零件类型

p_size 零件大小

p_container 零件包装

p_retailprice 零件零售价

p_comment 备注

主键为 p_partkey，取值范围：1~ SF *200000；

与 partsupp 表间有外键/参照完整性关联。

2.1.6 零部件供应表 PARTSUPP

零部件供应表记录商品销售中不同供应商供应的商品信息，包含以下表项：

ps_partkey 零件 key

ps_suppkey 零件供应商 key

ps_availqty 零件供应数量

ps_supplycost 零件供应成本

ps_comment 备注

主键为 ps_partkey、ps_suppkey;

与表 part、supplier、lineitem 间有外键/参照完整性关联。

2.1.7 客户表 CUSTOMER

客户表记录商品销售中的客户信息，包含以下表项：

c_custkey 客户 key

c_name 客户姓名

c_address 客户地址

c_nationkey 客户国家 key

c_phone 客户电话

c_acctbal 客户账户余额

c_mktsegment 客户市场领域

c_comment 备注

主键为 c_custkey，取值范围 1~ SF *150000，与表 orders 间有外键/参照完整性关联。

2.1.8 订单明细表 LINEITEM

订单明细表记录商品销售中的订单明细，包含以下表项：

l_orderkey 订单 key

l_partkey 零件 key

l_suppkey 供应商 key

l_linenumber 流水号

l_quantity 数量

l_extendedprice 价格

l_discount 折扣
l_tax 税
l_returnflag 退货标志
l_linestatus 明细状态
l_shipdate 发货日期
l_commitdate 预计到达日期
l_receipdate 实际到达日期
l_shipinstruct 运单处理策略
l_shipmode 运送方式
l_comment 备注

lineitem 表示在售商品信息，主键为 l_orderkey, _linenumber，是数据量最大的一张表。

这些关系表根据主键、外键关联关系组成关系模式图，如下所示：

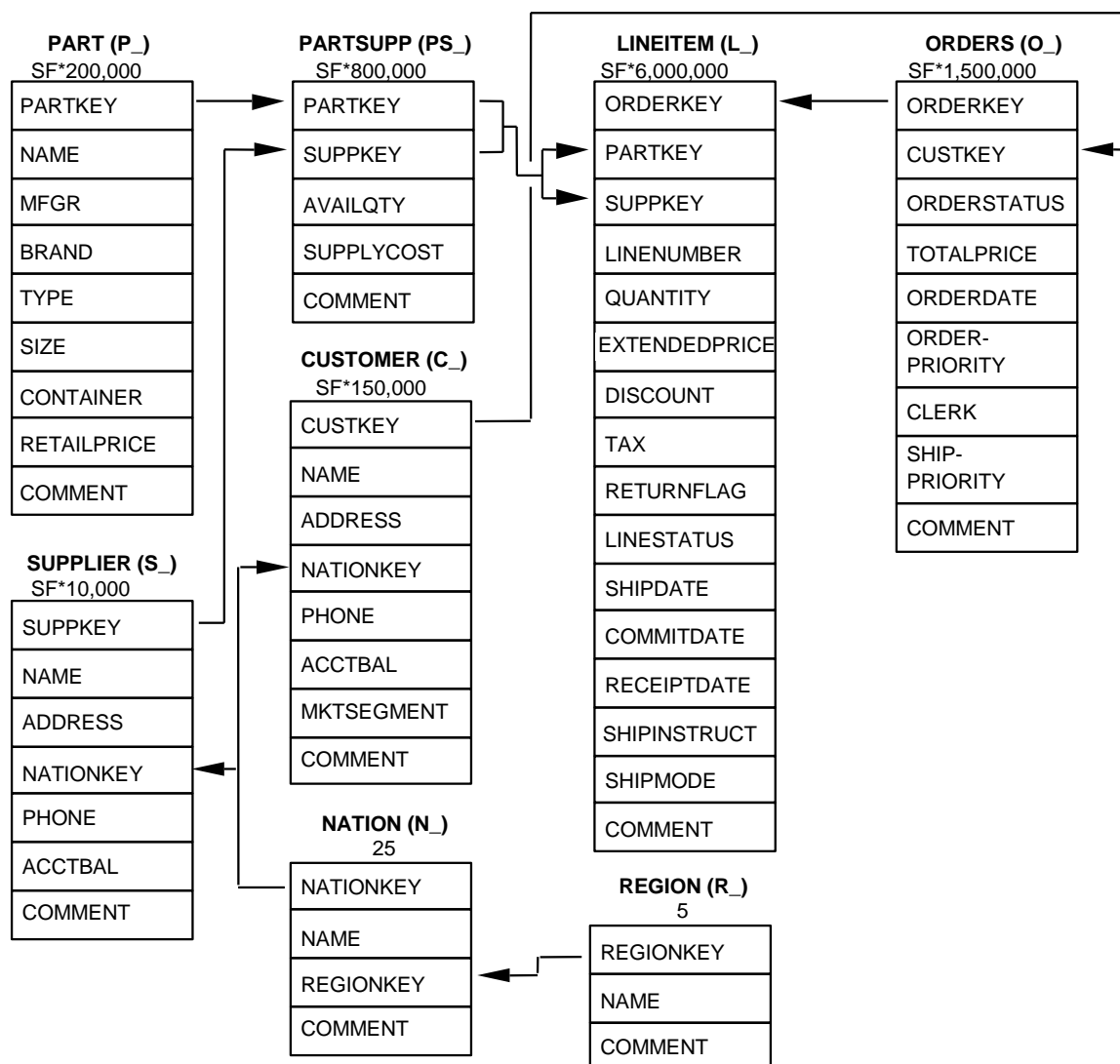


图2 TPC-H 关系模式图

2.2 数据对象 mapping cardinality/cardinality limit 及表间参照完整性约束

上述 8 张表代表了数据对象，这些数据对象间的定量联关系如下。

订单明细表 lineitem 的 orderkey 与 orders 表的 o_orderkey 是一一对应关系，订单中的每种商品都出现在 lineitem 表中，每笔订单(orderkey)有 1-7 (linenumber) 种商品，两张表数据量是 1: (1-7)。

orders 表的 o_custkey 信息都在 customer 表中，但不是一一对应关系，即订单上的所有消费者信息都在 customer 表中，但不是所有消费者都购买了商品，大约 2/3 的消费者有订单。

part 和 partsupp 中的 partkey 是一一对应关系，每个零件有 4 个供应商，两张表数据量是 1:4。

supplier 和 partsupp 中的 supkey 是一一对应关系，每个供应商供应 80 种零件，两张表数据量是 1:80。

lineitem 的 partkey 和 supkey 均出现在 partsupp 中。

每个 region 有 5 个 nation，每个 nation 中，supplier 和 customer 的比例大约为 1:15。

2.3 TPC-H 关系模式及其属性

2.3.1 订单表 ORDERS

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
O_ORDERKEY	订单 key	INTEGER	NOT NULL 主键
O_CUSTKEY	客户 key	INTEGER	NOT NULL 外键
O_ORDERSTATUS	订单状态	CHAR(1)	NOT NULL
O_TOTALPRICE	订单总价	DECIMAL(15,2)	NOT NULL
O_ORDERDATE	下单日期	DATE	NOT NULL
O_ORDERPRIORITY	订单优先级	CHAR(15)	NOT NULL
O_CLERK	收银员	CHAR(15)	NOT NULL
O_SHIPPRIORITY	发货优先级	INTEGER	NOT NULL
O_COMMENT	备注	VARCHAR(79)	NOT NULL

2.3.2 供应商表 SUPPLIER

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
S_SUPPKEY	供应商 key	INTEGER	NOT NULL 主键
S_NAME	供应商姓名	CHAR(25)	NOT NULL
S_ADDRESS	供应商地址	VARCHAR(40)	NOT NULL
S_NATIONKEY	供应商国家 key	INTEGER	NOT NULL 外键
S_PHONE	供应商手机号	CHAR(15)	NOT NULL
S_ACCTBAL	供应商账户余额	DECIMAL(15, 2)	NOT NULL
S_COMMENT	备注	VARCHAR(101)	NOT NULL

2.3.3 地区表 REGION

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
R_REGIONKEY	地区 key	INTEGER	NOT NULL 主键
R_NAME	地区名	CHAR(25)	NOT NULL
R_COMMENT	备注	VARCHAR(152)	

2.3.4 国家表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
N_NATIONKEY	国家 key	INTEGER	NOT NULL 主键
N_NAME	国家名	CHAR(25)	NOT NULL
N_REGIONKEY	国家所在地区 key	INTEGER	NOT NULL 外键
N_COMMENT	备注	VARCHAR(152)	

2.3.5 零部件表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
P_PARTKEY	零件 key	INTEGER	NOT NULL 主键
P_NAME	零件名称	VARCHAR(55)	NOT NULL
P_MFGR	零件厂商	CHAR(25)	NOT NULL
P_BRAND	零件品牌	CHAR(10)	NOT NULL
P_TYPE	零件类型	VARCHAR(25)	NOT NULL
P_SIZE	零件大小	INTEGER	NOT NULL
P_CONTAINER	零件包装	CHAR(10)	NOT NULL
P_RETAILPRICE	零件零售价	DECIMAL(15,2)	NOT NULL
P_COMMENT	备注	VARCHAR(23)	NOT NULL

2.3.6 零部件供应表

属性名称	属性中文名称	数据类型	数据取值范围， 完整性/约束说明
PS_PARTKEY	零件 key	INTEGER	NOT NULL 主键 外键
PS_SUPPKEY	零件供应行 key	INTEGER	NOT NULL 主键 外键
PS_AVAILQTY	零件供应数量	INTEGER	NOT NULL
PS_SUPPLYCOST	零件供应成本	DECIMAL(15,2)	NOT NULL
PS_COMMENT	备注	VARCHAR(199)	NOT NULL

2.3.7 客户表

属性名称	属性中文名称	数据类型	数据取值范围, 完整性/约束说明
C_CUSTKEY	客户 key	INTEGER	NOT NULL 主键
C_NAME	客户姓名	VARCHAR(25)	NOT NULL 主键
C_ADDRESS	客户地址	VARCHAR(40)	NOT NULL
C_NATIONKEY	客户国家 key	INTEGER	NOT NULL 外键
C_PHONE	客户电话	CHAR(15)	NOT NULL
C_ACCTBAL	客户账户余额	DECIMAL(15, 2)	NOT NULL
C_MKTSEGMENT	客户市场领域	CHAR(10)	NOT NULL
C_COMMENT	备注	VARCHAR(117)	NOT NULL

2.3.8 订单明细表

属性名称	属性中文名称	数据类型	数据取值范围, 完整性/约束说明
L_ORDERKEY	订单 key	INTEGER	NOT NULL 主键 外键
L_PARTKEY	零件 key	INTEGER	NOT NULL 外键
L_SUPPKEY	供应商 key	INTEGER	NOT NULL 外键
L_LINENUMBER	流水号	INTEGER	NOT NULL 主键
L_QUANTITY	数量	DECIMAL(15,2)	NOT NULL
L_EXTENDEDPRICE	价格	DECIMAL(15,2)	NOT NULL
L_DISCOUNT	折扣	DECIMAL(15,2)	NOT NULL
L_TAX	税	DECIMAL(15,2)	NOT NULL
L_RETURNFLAG	退货标志	CHAR(1)	NOT NULL
L_LINESTATUS	明细状态	CHAR(1)	NOT NULL
L_SHIPDATE	发货日期	DATE	NOT NULL
L_COMMITDATE	预计到达日期	DATE	NOT NULL
L_RECEIPTDATE	实际到达日期	DATE	NOT NULL
L_SHIPINSTRUCT	运单处理策略	CHAR(25)	NOT NULL
L_SHIPMODE	运送方式	CHAR(10)	NOT NULL
L_COMMENT	备注	VARCHAR(44)	NOT NULL

3. TPC-H Benchmark 数据集生成

3.1 基于脚本的数据生成

数据由程序生成，通过 C 语言开发，可以通过 visual studio 生成或者配置好 gcc 环境后使用 make 命令生成，可以生成两个程序，dbgen 和 qgen，dbgen 用来生成数据库的数据，qgen 用来生成查询语句。dbgen 运行时需要将源代码目录中的 dists.dss 放到同一目录下。dbgen 通过-s 参数指定数据量为 1G 的倍率，-s 1 表示生成 1G 数据，-f 表示覆盖掉之前生成的数据文件。

3.2 生成的关系表数据

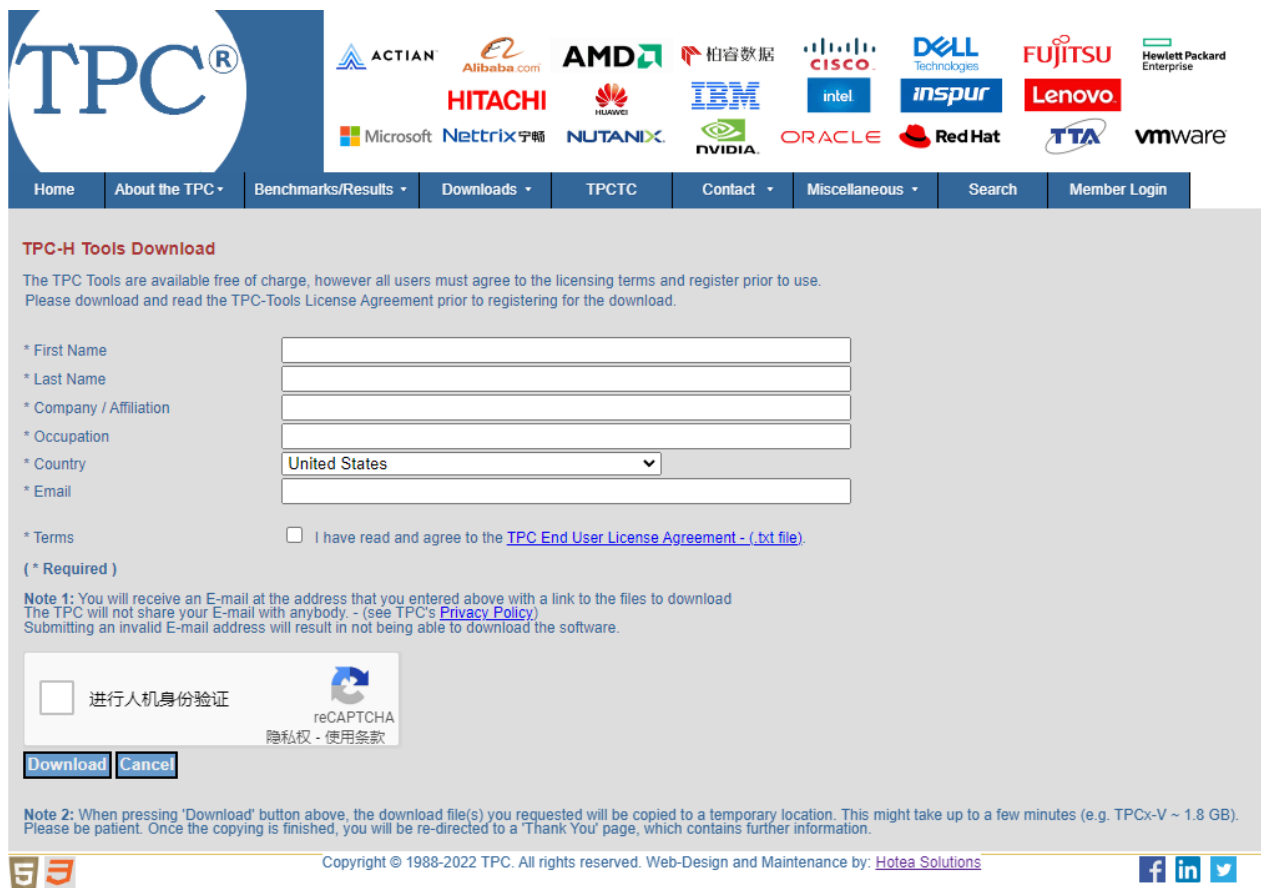
以 dbgen -s 0.2 生成 200M 数据为例。

关系模式	数据文件名称	数据文件中数据行数	
订单表 ORDERS	orders.txt	300000	
供应商表 SUPPLIER	supplier.txt	2000	
区域表 REGION	region.txt	5	
国家表 NATION	nation.txt	25	
零部件表 PART	part.txt	40000	
零 部 件 供 应 表 PARTSUPP	partsupp.txt	160000	
客户表 CUSTOMER	customer.txt	30000	
订 单 明 细 表 LINEITEM	lineitem.txt	1199969	

3.3 数据生成脚本示例

下载链接：

https://www.tpc.org/tpc_documents_current_versions/download_programs/tools-download-request5.asp?bm_type=TPC-H&bm_vers=3.0.1&mode=CURRENT-ONLY



TPC-H Tools Download

The TPC Tools are available free of charge, however all users must agree to the licensing terms and register prior to use. Please download and read the TPC-Tools License Agreement prior to registering for the download.

* First Name

* Last Name

* Company / Affiliation

* Occupation


* Country

* Email

* Terms ☐ I have read and agree to the [TPC End User License Agreement - \(.txt file\)](#).

(* Required)

Note 1: You will receive an E-mail at the address that you entered above with a link to the files to download. The TPC will not share your E-mail with anybody. - (see TPC's [Privacy Policy](#))
Submitting an invalid E-mail address will result in not being able to download the software.

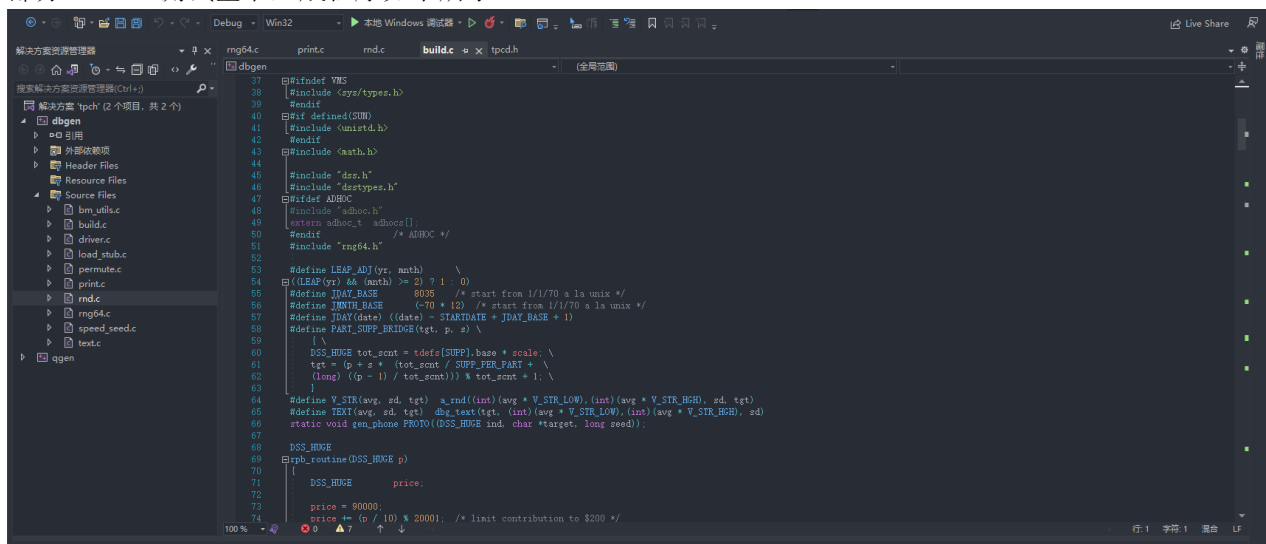
☐ 进行人机身份验证  隐私权 - 使用条款

Note 2: When pressing 'Download' button above, the download file(s) you requested will be copied to a temporary location. This might take up to a few minutes (e.g. TPCx-V ~ 1.8 GB). Please be patient. Once the copying is finished, you will be re-directed to a 'Thank You' page, which contains further information.

Copyright © 1988-2022 TPC. All rights reserved. Web-Design and Maintenance by: [Hotea Solutions](#)

图3 TPC-H 测试基准程序下载页面

部分 TPC-H 测试基准生成程序如下所示：



```

37 #ifndef VMS
38 #include <sys/types.h>
39 #endif
40 #if defined(SUN)
41 #include <unistd.h>
42 #endif
43 #include <math.h>
44
45 #include "dss.h"
46 #include "dss/types.h"
47 #ifndef ARBOC
48 #include "arbo.h"
49 extern adhoc_t adhoc[];
50 #endif /* ARBOC */
51 #include "rng64.h"
52
53 #define LEAF_ADJ(yr, month) \
54 ((LEAF_Gr) && (month) >= 2) ? 1 : 0)
55 #define JDAY_BASE 8035 /* start from 1/1/70 a la unix */
56 #define JMONTH_BASE (-70 * 12) /* start from 1/1/70 a la unix */
57 #define JDAY(date) ((date) - STARTDATE + JDAY_BASE + 1)
58 #define PART_SUPP_BRIDGE(tgt, p, s) \
59 { \
60     DSS_HUGE tot_cnt = tdefs(SUPP).base * scale; \
61     tgt = (p * s * (tot_cnt / SUPP_PER_PART + \
62         (long) ((p - 1) / tot_cnt))) * tot_cnt + 1; \
63 }
64 #define V_STR(avg, sd, tgt) a_snd((int)(avg * V_STR_LOW), (int)(avg * V_STR_HIGH), sd, tgt)
65 #define TBI(avg, sd, tgt) dbg_test(tgt, (int)(avg * V_STR_LOW), (int)(avg * V_STR_HIGH), sd)
66 static void gen_phone FROM(DSS_HUGE ind, char *target, long seed);
67
68 DSS_HUGE
69 gen_routine(DSS_HUGE p)
70 {
71     DSS_HUGE price;
72     price = 90000;
73     price += (p / 10) * 2000; /* limit contribution to $200 */
74 }

```

图4 TPC-H 测试基准生成程序（部分）示例

4. 典型 TPC-H 测试语句

4.1 定价汇总报表查询

查询已经开票，发货和退货三个类别订单的业务总量，根据给定日期，返回退货标志，订单状态，总价格，总折扣价格，总折扣价格加税，平均数量，平均价格，平均折扣，订单数量，结果按照退货标志和订单状态升序。

```
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 '2021-12-01' - interval '90' day (3)
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

4.2 最低成本供应商查询

这个查询用来确定每个地区该选择哪个供货商，在给定区域中，针对给定类型和大小的零件，找到能够以最低价格供应的供应商。如果该地区的多个供应商以相同（最低）价格提供所需的零件类型和尺寸，则列出来具有 100 个最高帐户余额的供应商的零件。返回供应商的帐户余额、名称和国家，零件 key 和制造商，供应商的地址、电话号码和备注信息，结果按照账户余额降序，国家名，供应商名，地区 key 升序。

```
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 = 15
    and p_type like '%BRASS'
    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
```

4.3 运输优先级查询

根据给定的日期，查找出 10 个未发货的价格最高的订单，返回订单 key，总潜在收入，下单日期，和运输优先级，按照潜在收入降序，下单日期升序，如果超过 10 个，只返回 10 个。

```
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'BUILDING'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '2018-03-15'
    and l_shipdate > date '2018-03-15'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
LIMIT 10
```

4.4 订单优先级查询

这个查询用来确定订单优先级系统的工作是否良好，并给出客户满意度。根据下单日期，返回至少有一个订单明细在预计送达时间之后收到的订单优先级和订单数量，并根据订单优先级升序。

```
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '2016-07-01'
    and o_orderdate < date '2016-07-01' + 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;
```

4.5 本地供应商收入量查询

列出给定地区每个国家的订单交易产生的收入量，返回国家名字和收入量，并根据收入量降序。

```
select
  n_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue
from
  customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and l_suppkey = s_suppkey
  and c_nationkey = s_nationkey
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'ASIA'
  and o_orderdate >= date '2017-01-01'
  and o_orderdate < date '2017-01-01' + interval '1' year
group by
  n_name
order by
  revenue desc;
```

4.6 预测收入变化查询

根据给定发货时间、折扣，数量，返回在发货时间一年内，折扣增减 0.01，数量小于给定数量的订单明细的总折扣。

```
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '2017-01-01'
    and l_shipdate < date '2017-01-01' + interval '1' year
    and l_discount between .06 - 0.01 and .06 + 0.01
    and l_quantity < 24;
```

4.7 批量出货查询

给定时间和供应商所在的国家 and 客户所在的国家，查找出在该段时间内两个国家出货的总收入，返回供应国家名，客户国家名，年份，和出货的收入，结果按照供应国家名，客户国家名，年份升序。

```
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 = 'FRANCE' and n2.n_name = 'GERMANY')
            or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
        )
        and l_shipdate between date '2018-01-01' and date '2019-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;

```

4.8 全国市场份额查询

给定国家，地区，周期和零件 key，查找周期内的每年该国在该地区收入所占的份额，结果返回年份和周周期比，按照年份升序。

```

select
    o_year,
    sum(case
        when nation = 'BRAZIL' 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 = 'AMERICA'
        and s_nationkey = n2.n_nationkey
        and o_orderdate between date '2018-01-01' and date '2019-12-31'
        and p_type = 'ECONOMY ANODIZED STEEL'
    ) as all_nations
group by
    o_year
order by
    o_year;
```

4.9 产品类型利润度量查询

给出一个零件 key，查询该零件在不同国家，不同年份的利润，返回国家，年份和利润，按照国家升序，年份降序。

```
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 '%green%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
```

4.10 退货报告查询

根据给定的零件 key，查找出收入损失最多的前 20 个客户，结果返回客户 key，客户名字，收入损失，客户账户余额，客户所在国家，客户地址，客户电话，备注，查询结果按照收入损失降序排序。

```
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 '2016-10-01'
    and o_orderdate < date '2016-10-01' + 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 10
```

4.11 重要库存识别查询

给出国家和百分比，查找出供应商库存里超过这个国家总库存价值百分比的所有库存，结果返回零件 key，库存价值，按照价值降序排序。

```
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 = 'GERMANY'  
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 = 'GERMANY'  
        )  
order by  
    value desc;
```

4.12 运送方式和订单优先级查询

这条查询语句用来确定选择便宜的货运模式是否会导致消费者更多在承诺日期之后收到货物，对订单优先级条目产生影响。给定日期和两种货运方式，在给定日期一年内收货并且属于两种货运模式的订单明细中，选出比承诺日期晚送达的，统计两种货运模式下，订单优先级为”1-URGENT”和”2-HIGH”的数量与不为这两种的数量，比较区别。结果返回货运模式，不同订单优先级数量。

```
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 ('MAIL', 'SHIP')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '2017-01-01'
    and l_receiptdate < date '2017-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;
```

4.13 客户分布查询

根据客户订单数量确定客户分布，包括没有下过订单的客户，统计不同订单数量的客户数量，并且要在备注中检查订单不属于几种特殊类别的订单，结果返回订单数量和客户数量，按照客户数量和订单数量降序。

```
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 '%special%requests%'
group by
    c_custkey
) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
```

4.14 促销效果查询

给定日期，查询这个日期后一个月内的收入有多大比例来自促销零件，结果返回促销零件收入所占的百分比。

```
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 '2018-09-01'
    and l_shipdate < date '2018-09-01' + interval '1' month;
```

4.15 顶级供应商查询

查询给定时间内收入最多的供应商，结果返回供应商 key，名称，地址，电话，和总收入，按照供应商 key 升序。

```
create view revenue0 (supplier_no, total_revenue) as
select
    l_suppkey,
    sum(l_extendedprice * (1 - l_discount))
from
    lineitem
```

```
where
    l_shipdate >= date '2019-01-01'
    and l_shipdate < date '2019-01-01' + 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;
```

4.16 零部件/供应商关系查询

这个查询可以超出有多少供应商能够按照给定条件供应，且没有被客户投诉过，给定零件品牌，类型，大小，结果返回零件品牌，类型，大小和供应商数量，按照供应商数量降序，品牌，类型，大小升序。

```
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#45'
and p_type not like 'MEDIUM POLISHED%'
and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
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;
```

4.17 小额订单收入查询

给定零件品牌和包装查询比平均供货量百分之 20 还小的小额订单，结果返回这些订单收入在七年内的平均值。

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
```

```
);
```

4.18 大批量客户查询

给定数量，查询购买数量比指定数量大的客户信息，结果返回客户名字，客户 key，订单 key，下单日期，订单总价，购买数量，结果按照购买数量降序，下单日期升序，返回前 100 个。

```
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) > 300
    )
    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;
```

4.19 折扣收入查询

查询一些空运或者人工运输的订单的折扣收入，零件按照给定的组合进行选择，结果返回总折扣收入。

```
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#12'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 1 and l_quantity <= 1 + 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#23'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 10 and l_quantity <= 10 + 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#34'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
```


4.20 潜在零部件促销查询

给定国家和开始时间，查询在一年时间内，该国对给定条件的零件有超过百分之 50 供应量的供应商，结果返回供应商名字和地址，按照供应商名字升序。

```
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 'forest%'
            )
        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 '2017-01-01'
                and l_shipdate < date '2017-01-01' + interval '1' year
        )
    )
    and s_nationkey = n_nationkey
    and n_name = 'CANADA'
order by
    s_name;
```

4.21 不能按时交货供应商查询

给定国家，查找出具有多个供货商的订单中唯一一个没有按时交货的供货商，返回供货商名字和没有交货的订单数量，按照订单数量降序，名字升序。

```
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 = 'SAUDI ARABIA'
group by
    s_name
order by
    numwait desc,
    s_name;
```

4.22 全球销售机会查询

查询指定国家代码内，对应的客户没有下七年的订单，但是账户余额大于平均账户余额的客户，结果返回国家代码，客户数量，总账户余额，按照国家代码升序。

```
select
    centrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substring(c_phone from 1 for 2) as centrycode,
            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
    centrycode
order by
    centrycode;
```

5. TPC-H 数据库建库脚本

5.1 建表及数据导入步骤

建议：

通过下述步骤在数据库中建立 TPC-H 中的 8 张关系表，并向表中导入数据，构建 TPC-H 数据库：

- (1) 使用 5.1-5.8 给出的 8 条 create table 建表语句，建立不带完整性约束的关系表
- (2) 使用数据库管理系统提供的数据导入机制，将 customer.txt 等数据文件中的数据导入这 8 张表。

具体数据导入方法参见“实验指导书-03-1 openGauss 数据库建表及数据导入-22-v1.0”。

- (3) 使用 5.9 中的 SQL 语句，在这 8 张表上添加约束

5.2 订单表 ORDERS

```
CREATE TABLE ORDERS ( O_ORDERKEY      INTEGER NOT NULL,
                        O_CUSTKEY       INTEGER NOT NULL,
                        O_ORDERSTATUS   CHAR(1) 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);
```

5.3 区域表 REGION

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

5.4 国家表 NATION

```
CREATE TABLE NATION ( N_NATIONKEY  INTEGER NOT NULL,
                        N_NAME       CHAR(25) NOT NULL,
                        N_REGIONKEY  INTEGER NOT NULL,
```

```
N_COMMENT    VARCHAR(152));
```

5.5 供应商表 SUPPLIER

```
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);
```

5.6 零部件表 PART

```
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 );
```

5.7 零部件供应表 PARTSUPP

```
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 );
```

5.8 客户表 CUSTOMER

```
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);
```

5.9 订单明细表 LINEITEM

```
CREATE TABLE LINEITEM ( L_ORDERKEY    INTEGER 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(1) NOT NULL,  
                        L_LINESTATUS   CHAR(1) 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);
```

5.10 约束

```
ALTER TABLE REGION  
ADD PRIMARY KEY (R_REGIONKEY);
```

```
ALTER TABLE NATION  
ADD PRIMARY KEY (N_NATIONKEY);
```

```
ALTER TABLE NATION  
ADD FOREIGN KEY (N_REGIONKEY) references REGION;
```

```
ALTER TABLE PART  
ADD PRIMARY KEY (P_PARTKEY);
```

```
ALTER TABLE SUPPLIER  
ADD PRIMARY KEY (S_SUPPKEY);
```

```
ALTER TABLE SUPPLIER
ADD FOREIGN KEY (S_NATIONKEY) references NATION;
```

```
ALTER TABLE PARTSUPP
ADD PRIMARY KEY (PS_PARTKEY,PS_SUPPKEY);
```

```
ALTER TABLE CUSTOMER
ADD PRIMARY KEY (C_CUSTKEY);
```

```
ALTER TABLE CUSTOMER
ADD FOREIGN KEY (C_NATIONKEY) references NATION;
```

```
ALTER TABLE LINEITEM
ADD PRIMARY KEY (L_ORDERKEY,L_LINENUMBER);
```

```
ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);
```

```
ALTER TABLE PARTSUPP
ADD FOREIGN KEY (PS_SUPPKEY) references SUPPLIER;
```

```
ALTER TABLE PARTSUPP
ADD FOREIGN KEY (PS_PARTKEY) references PART;
```

```
ALTER TABLE ORDERS
ADD FOREIGN KEY (O_CUSTKEY) references CUSTOMER;
```

```
ALTER TABLE LINEITEM
ADD FOREIGN KEY (L_ORDERKEY) references ORDERS;
```

```
ALTER TABLE LINEITEM
ADD FOREIGN KEY (L_PARTKEY,L_SUPPKEY) references PARTSUPP;
```

6. TPC-C 测试基准

6.1 实体-联系图

TPC-C 数据库记录了 Warehouse、Customer 等九个数据对象（实体）及其关联关系，由九个单独的关系表和索引表组成。数据库的实体-联系图如下所示。

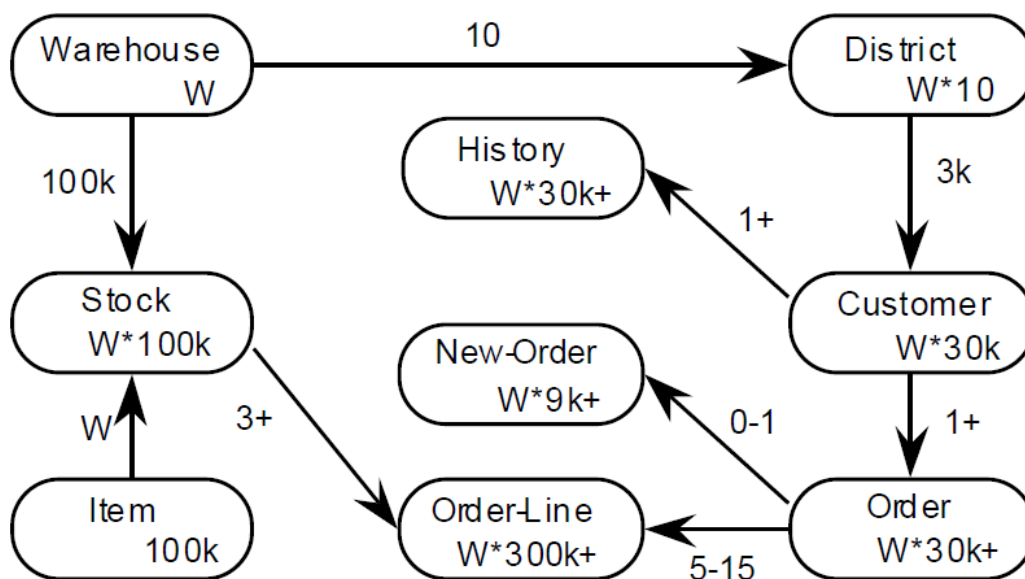


图5 TPC-C 实体-联系图

说明：

- 图中显示的数字说明了数据库填充要求
- 实体块中的数字代表表的基数（行数）。这些数字由仓库数 W 分解，以说明数据库扩展。
- 联系箭头旁边的数字代表联系的基数。
- 加号 (+) 用于关系或表的基数之后，以说明在添加或删除行时，该数字在测量间隔内的初始数据库总体中会发生微小变化。

6.2 关系表

与上述实体-联系图对应的 9 张关系表如下。

6.2.1 仓库表 WAREHOUSE

属性名称	字段定义	注释, 完整性/约束说明
W_ID	2*W unique IDs	W Warehouses are populated 主键
W_NAME	variable text, size 10	
W_STREET_1	variable text, size 20	
W_STREET_2	variable text, size 20	
W_CITY	variable text, size 20	
W_STATE	fixed text, size 2	
W_ZIP	fixed text, size 9	
W_TAX	signed numeric(4,4)	Sales tax
W_YID	signed numeric(12,2)	Year to date balance

6.2.2 区域表 DISTRICT

属性名称	字段定义	注释, 完整性/约束说明
D_ID	20 unique IDs	10 are populated per warehouse 主键
D_W_ID	2*W unique IDs	主键/外键, 参考 W_ID
D_NAME	variable text, size 10	
D_STREET_1	variable text, size 20	
D_STREET_2	variable text, size 20	
D_CITY	variable text, size 20	

D_STATE	fixed text, size 2	
D_ZIP	fixed text, size 9	
D_TAX	signed numeric(4,4)	Sales tax
D_YTD	signed numeric(12,2)	Year to date balance
D_NEXT_O_ID	10,000,000 unique IDS	Next available Order number

6.2.3 客户表 CUSTOMER

属性名称	字段定义	注释, 完整性/约束说明
C_ID	96,000 unique IDs	3,000 are populated per district 主键
C_D_ID	20 unique IDs	主键/外键, 参考 D_ID
C_W_ID	2*W unique IDs	主键/外键。参考 D_W_ID
C_FIRST	variable text, size 16	
C_MIDDLE	fixed text, size 2	
C_LAST	variable text, size 16	
C_STREET_1	variable text, size 20	
C_STREET_2	variable text, size 20	
C_CITY	variable text, size 20	Sales tax
C_STATE	fixed text, size 2	Year to date balance
C_ZIP	fixed text, size 9	Next available Order number
C_PHONE	fixed text, size 16	
C_SINCE	date and time	
C_CREDIT	fixed text, size 2	"GC"=good, "BC"=bad
C_CREDIT_LIM	signed numeric(12, 2)	

C_DISCOUNT	signed numeric(4, 4)	
C_BALANCE	signed numeric(12, 2)	
C_YTD_PAYMENT	signed numeric(12, 2)	
C_PAYMENT_CNT	numeric(4)	
C_DELIVERY_CNT	numeric(4)	
C_DATA	variable text, size 500	Miscellaneous information

6.2.4 历史记录表 HISTORY

属性名称	字段定义	注释, 完整性/约束说明
H_C_ID	96,000 unique IDs	外键, 参考 C_ID
H_C_D_ID	20 unique IDs	外键, 参考 C_D_ID
H_C_W_ID	2*W unique IDs	外键, 参考 C_W_ID
H_D_ID	20 unique IDs	外键, 参考 D_ID
H_W_ID	2*W unique IDs	外键, 参考 D_W_ID
H_DATE	date and time	
H_AMOUNT	signed numeric(6, 2)	
H_DATA	variable text, size 24	Miscellaneous information

6.2.5 新订单表 NEW-ORDER

属性名称	字段定义	注释, 完整性/约束说明
NO_O_ID	10,000,000 unique IDs	主键/外键, 参考 O_ID
NO_C_ID	20 unique IDs	主键/外键, 参考 O_D_ID
NO_W_ID	2*W unique IDs	主键/外键, 参考 O_W_ID

6.2.6 订单表 ORDER

属性名称	字段定义	注释, 完整性/约束说明
O_ID	10,000,000 unique IDs	主键
O_D_ID	20 unique IDs	主键/外键, 参考 C_D_ID
O_W_ID	2*W unique IDs	主键/外键, 参考 C_W_ID
O_C_ID	96,000 unique IDs	外键, 参考 C_ID
O_ENTRY_D	date and time	
O_CARRIER_ID	10 unique IDs, or null	
O_OL_CNT	numeric(2)	Count of Order-Lines
O_ALL_LOCAL	numeric(1)	

6.2.7 订单行表 ORDER-LINE

属性名称	字段定义	注释, 完整性/约束说明
OL_O_ID	10,000,000 unique IDs	主键/外键, 参考 O_ID
OL_D_ID	20 unique IDs	主键/外键, 参考 O_D_ID
OL_W_ID	2*W unique IDs	主键/外键, 参考 O_W_ID
OL_NUMBER	15 unique IDs	主键
OL_I_ID	200,000 unique IDs	外键, 参考 S_I_ID
OL_SUPPLY_W_ID	2*W unique IDs	外键, 参考 S_W_ID
OL_DELIVERY_D	date and time, or null	
OL_QUANTITY	numeric(2)	
OL_AMOUNT	signed numeric(6, 2)	
OL_DIST_INFO	fixed text, size 24	

6.2.8 条目表 ITEM

属性名称	字段定义	注释, 完整性/约束说明
I_ID	200,000 unique IDs	100,000 items are populated 主键
I_M_ID	200,000 unique IDs	Image ID associated to Item
I_NAME	variable text, size 24	
I_PRICE	numeric(5, 2)	
I_DATA	variable text, size 50	Brand information

6.2.9 库存表 STOCK

属性名称	字段定义	注释, 完整性/约束说明
S_I_ID	200,000 unique IDs	100,000 populated per warehouse 主键/外键, 参考 I_ID
S_W_ID	2*W unique IDs	主键/外键, 参考 W_ID
S_QUANTITY	signed numeric(4)	
S_DIST_01	fixed text, size 24	
S_DIST_02	fixed text, size 24	
S_DIST_03	fixed text, size 24	
S_DIST_04	fixed text, size 24	
S_DIST_05	fixed text, size 24	
S_DIST_06	fixed text, size 24	
S_DIST_07	fixed text, size 24	
S_DIST_08	fixed text, size 24	
S_DIST_09	fixed text, size 24	
S_DIST_10	fixed text, size 24	
S_YTD	numeric(8)	
S_ORDER_CNT	numeric(4)	
S_REMOTE_CNT	numeric(4)	
S_DATA	variable text, size 50	Make information

6.3 基准测试程序示例

6.3.1 基于嵌入式 SQL 的测试基准程序

TPC-C 采用嵌入式 SQL 方式，定义了一组面向联机在线事务处理（OLTP）的测试程序，每个测试程序中包含多条测试用的 SQL 语句，并组织成一个完整的测试事务。

各个测试事务的详细代码及说明的 pdf 文件可从此处下载（TPC-C）：

https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

下面给出 2 个典型 TPC-C 测试事务的代码及说明。

6.3.2 A1 新订单事务 The New-Order Transaction

新订单事务包括通过单个数据库事务输入完整的订单。它代表了一种中等等级的读写事务，具有高执行频率和严格的响应时间要求，以满足在线用户的需求。此事务是工作负载的支柱。它旨在对系统施加可变负载，以反映生产环境中常见的在线数据库活动。

新订单事务过程说明：

对于给定的仓库编号（W_ID）、地区编号（D_W_ID、D_ID）、客户编号（C_W_ID、C_D_ID、C_ID）、物品计数（ol_cnt）以及给定的一组物品（OL_I_ID）、供应仓库（OL_SUPPLY_W_ID）和数量（OL_QUANTITY）：

- 1) 选择 WAREHOUSE 表中与 W_ID 匹配的行，并检索仓库税率 W_TAX。同时选择 CUSTOMER 表中匹配 C_W_ID、C_D_ID 和 C_ID 的行，并检索客户贴现率 C_DISCOUNT、客户姓氏 C_LAST 和客户信用状态 C_CREDIT。
- 2) 选择 DISTRICT 表中 D_W_ID 和 D_ID 匹配的行，检索地区税率 D_TAX，检索该地区的下一个可用订单号 D_NEXT_O_ID 并加一。
- 3) 在 NEW-ORDER 表和 ORDER 表中都插入了一个新行，以反映新订单的创建。O_CARRIER_ID 设置为空值。如果订单仅包括主订单行，则 O_ALL_LOCAL 设置为 1，否则 O_ALL_LOCAL 设置为 0。计算项目数 O_OL_CNT 以匹配 ol_cnt。
- 4) 对于订单上的每个 O_OL_CNT 项目：
 - 选择 ITEM 表中具有匹配 I_ID（等于 OL_I_ID）的行，并检索项目价格 I_PRICE、项目名称 I_NAME 和 I_DATA。如果 I_ID 有一个未使用的值，则发出“未找到”条件，导致数据库事务回滚。
 - 选择 STOCK 表中匹配 S_I_ID（等于 OL_I_ID）和 S_W_ID（等于 OL_SUPPLY_W_ID）的行。检索库存数量 S_QUANTITY，S_DIST_xx（其中 xx 代表区号），S_DATA。如果 S_QUANTITY 的检索值超过 OL_QUANTITY 10 或更多，则 S_QUANTITY 减少 OL_QUANTITY；否则 S_QUANTITY 更新为 (S_QUANTITY - OL_QUANTITY)+91。S_YTD 增加 OL_QUANTITY 并且 S_ORDER_CNT 增加 1。如果订单行是远程的，则 S_REMOTE_CNT 增加 1。
 - 订单中商品的金额 (OL_AMOUNT) 计算为：

OL_QUANTITY * I_PRICE

- 检查 I_DATA 和 S_DATA 中的字符串。如果它们都包含字符串 “ORIGINAL”，则该项目的品牌通用字段设置为 “B”，否则，品牌通用字段设置为 “G”。

- 将新行插入到 ORDER-LINE 表中以反映订单上的项目。 OL_DELIVERY_D 设置为空值，OL_NUMBER 设置为在所有具有相同 OL_O_ID 值的 ORDER-LINE 行中的唯一值，OL_DIST_INFO 设置为 S_DIST_xx 的内容，其中 xx 代表区号 (OL_D_ID)

5) 完整订单的总金额计算如下：

总和 (OL_AMOUNT) * (1 - C_DISCOUNT) * (1 + W_TAX + D_TAX)

- 数据库事务被提交，除非它由于最后一个项目编号的未使用值而被回滚。
- 输出数据被传送到终端。

代码：

```
int neword()
{
EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

gettimestamp(datetime);

EXEC SQL SELECT c_discount, c_last, c_credit, w_tax
      INTO :c_discount, :c_last, :c_credit, :w_tax
      FROM customer, warehouse
      WHERE w_id = :w_id AND c_w_id = w_id AND
            c_d_id = :d_id AND c_id = :c_id;

EXEC SQL SELECT d_next_o_id, d_tax INTO :d_next_o_id, :d_tax
      FROM district
      WHERE d_id = :d_id AND d_w_id = :w_id;

EXEC SQL UPDATE district SET d_next_o_id = :d_next_o_id + 1
      WHERE d_id = :d_id AND d_w_id = :w_id;

o_id=d_next_o_id;

EXEC SQL INSERT INTO ORDERS (o_id, o_d_id, o_w_id, o_c_id,
      o_entry_d, o_ol_cnt, o_all_local)
      VALUES (:o_id, :d_id, :w_id, :c_id,
      :datetime, :o_ol_cnt, :o_all_local);

EXEC SQL INSERT INTO NEW_ORDER (no_o_id, no_d_id, no_w_id)
      VALUES (:o_id, :d_id, :w_id);
```

```
for (ol_number=1; ol_number<=o_ol_cnt; ol_number++)
{
    ol_supply_w_id=atol(supware[ol_number-1]);
    if (ol_supply_w_id != w_id) o_all_local=0;
    ol_i_id=atol(itemid[ol_number-1]);
    ol_quantity=atol(qty[ol_number-1]);

EXEC SQL WHENEVER NOT FOUND GOTO invaliditem;

EXEC SQL SELECT i_price, i_name , i_data
    INTO :i_price, :i_name, :i_data
    FROM item
    WHERE i_id = :ol_i_id;

price[ol_number-1] = i_price;
strncpy(iname[ol_number-1],i_name,24);

EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;

EXEC SQL SELECT s_quantity, s_data,
    s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05
    s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
    INTO :s_quantity, :s_data,
    :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05
    :s_dist_06, :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10
    FROM stock
    WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id;

pick_dist_info(ol_dist_info, ol_w_id); // pick correct s_dist_xx
stock[ol_number-1] = s_quantity;

if ( (strstr(i_data,"original") != NULL) &&
    (strstr(s_data,"original") != NULL) )
    bg[ol_number-1] = 'B';
else
    bg[ol_number-1] = 'G';

if (s_quantity > ol_quantity)
    s_quantity = s_quantity - ol_quantity;
else
    s_quantity = s_quantity - ol_quantity + 91;

EXEC SQL UPDATE stock SET s_quantity = :s_quantity
    WHERE s_i_id = :ol_i_id
```



```

        AND s_w_id = :ol_supply_w_id;

ol_amount = ol_quantity * i_price * (1+w_tax+d_tax) * (1-c_discount);
amt[ol_number-1]=ol_amount;
total += ol_amount;

EXEC SQL INSERT
    INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number,
                    ol_i_id, ol_supply_w_id,
                    ol_quantity, ol_amount, ol_dist_info)
    VALUES (:o_id, :d_id, :w_id, :ol_number,
            :ol_i_id, :ol_supply_w_id,
            :ol_quantity, :ol_amount, :ol_dist_info);
} /*End Order Lines*/

EXEC SQL COMMIT WORK;
return(0);

invaliditem:
    EXEC SQL ROLLBACK WORK;
    printf("Item number is not valid");
    return(0);

sqlerr:
    error();
}

```

6.3.3 A2 The Payment Transaction

支付事务更新客户的余额，并将支付反映在地区和仓库销售统计上。它代表了一种轻量级的读写事务，具有高执行频率和严格的响应时间要求，以满足在线用户的需求。此外，此事务包括对 CUSTOMER 表的非主键访问。

支付事务过程说明：

对于给定的仓库编号 (W_ID)、地区编号 (D_W_ID、D_ID)、客户编号 (C_W_ID、C_D_ID、C_ID) 或客户姓氏 (C_W_ID、C_D_ID、C_LAST) 和付款金额 (H_AMOUNT)：

- 1) WAREHOUSE 表中具有匹配 W_ID 的行被选中。检索 W_NAME、W_STREET_1、W_STREET_2、W_CITY、W_STATE 和 W_ZIP，并将仓库年初至今余额 W_YTD 增加 H_AMOUNT。
- 2) 选择 DISTRICT 表中 D_W_ID 和 D_ID 匹配的行。检索 D_NAME、D_STREET_1、D_STREET_2、D_CITY、D_STATE 和 D_ZIP，并且 D_YTD（该地区的年初至今余额）增加 H_AMOUNT。
- 3) 根据客户姓氏选择客户：

CUSTOMER 表中所有匹配 C_W_ID、C_D_ID 和 C_LAST 的行都按照 C_FIRST 升序选择。设 n 为选定的行数。C_ID、C_FIRST、C_MIDDLE、C_STREET_1、C_STREET_2、C_CITY、C_STATE、C_ZIP、C_PHONE、C_SINCE、C_CREDIT、C_CREDIT_LIM、C_DISCOUNT 和 C_BALANCE 从排序集中位置 (n/2 向上舍入到下一个整数) 处的行检索从 CUSTOMER 表中选择的行数。

C_BALANCE 减少 H_AMOUNT。C_YTD_PAYMENT 增加了 H_AMOUNT。C_PAYMENT_CNT 增加 1。

- 4) 如果 C_CREDIT 的值等于“BC”，则还从所选客户检索 C_DATA，并在 C_DATA 字段的左侧插入以下历史信息：C_ID、C_D_ID、C_W_ID、D_ID、W_ID 和 H_AMOUNT。将 C_DATA 的现有内容向右移动相等数量的字节，并丢弃移出 C_DATA 字段右侧的字节。C_DATA 字段的内容不超过 500 个字符。使用新的 C_DATA 字段更新选定的客户。如果 C_DATA 实现为两个字段，则必须将它们视为一个字段并对其进行操作。
- 5) H_DATA 是通过连接由 4 个空格分隔的 W_NAME 和 D_NAME 构建的。新行插入到 HISTORY 表中，H_C_ID=C_ID, H_C_D_ID=C_D_ID, H_C_W_ID=C_W_ID、H_D_ID=D_ID 和 H_W_ID=W_ID。
 - 数据库事务已提交。
 - 输出数据被传送到终端。

代码:

```
int payment()
{
    EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
    EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

    gettimestamp(datetime);
    EXEC SQL UPDATE warehouse SET w_ytd = w_ytd + :h_amount
        WHERE w_id=:w_id;

    EXEC SQL SELECT w_street_1, w_street_2, w_city, w_state, w_zip, w_name
        INTO :w_street_1, :w_street_2, :w_city, :w_state, :w_zip, :w_name
        FROM warehouse
        WHERE w_id=:w_id;

    EXEC SQL UPDATE district SET d_ytd = d_ytd + :h_amount
        WHERE d_w_id=:w_id AND d_id=:d_id;

    EXEC SQL SELECT d_street_1, d_street_2, d_city, d_state, d_zip, d_name
        INTO :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :d_name
        FROM district
        WHERE d_w_id=:w_id AND d_id=:d_id;

    if (byname)
    {
        EXEC SQL SELECT count(c_id) INTO :namecnt
            FROM customer
            WHERE c_last=:c_last AND c_d_id=:c_d_id AND c_w_id=:c_w_id;

        EXEC SQL DECLARE c_byname CURSOR FOR
            SELECT c_first, c_middle, c_id,
```

```

        c_street_1, c_street_2, c_city, c_state, c_zip,
        c_phone, c_credit, c_credit_lim,
        c_discount, c_balance, c_since
    FROM customer
    WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_last=:c_last
    ORDER BY c_first;

EXEC SQL OPEN c_byname;

if (namecnt%2) namecnt++; // Locate midpoint customer;
for (n=0; n<namecnt/2; n++)
{
    EXEC SQL FETCH c_byname
        INTO :c_first, :c_middle, :c_id,
            :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
            :c_phone, :c_credit, :c_credit_lim,
            :c_discount, :c_balance, :c_since;
}
EXEC SQL CLOSE c_byname;
}
else
{
    EXEC SQL SELECT c_first, c_middle, c_last,
        c_street_1, c_street_2, c_city, c_state, c_zip,
        c_phone, c_credit, c_credit_lim,
        c_discount, c_balance, c_since
    INTO :c_first, :c_middle, :c_last,
        :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
        :c_phone, :c_credit, :c_credit_lim,
        :c_discount, :c_balance, :c_since
    FROM customer
    WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;
}
c_balance += h_amount;
c_credit[2]='\0';
if (strstr(c_credit, "BC"))
{
    EXEC SQL SELECT c_data INTO :c_data
        FROM customer
        WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;

    sprintf(c_new_data, "| %4d %2d %4d %2d %4d $%7.2f %12c %24c",
        c_id, c_d_id, c_w_id, d_id, w_id, h_amount
        h_date, h_data);
}

```

```
strncat(c_new_data,c_data,500-strlen(c_new_data));

EXEC SQL UPDATE customer
    SET c_balance = :c_balance, c_data = :c_new_data
    WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND
        c_id = :c_id;
}
else
{
    EXEC SQL UPDATE customer SET c_balance = :c_balance
        WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND
            c_id = :c_id;
}
strncpy(h_data,w_name,10);
h_data[10]='\0';
strncat(h_data,d_name,10);
h_data[20]=' ';
h_data[21]=' ';
h_data[22]=' ';
h_data[23]=' ';

EXEC SQL INSERT INTO history (h_c_d_id, h_c_w_id, h_c_id, h_d_id,
    h_w_id, h_date, h_amount, h_data)
    VALUES (:c_d_id, :c_w_id, :c_id, :d_id,
        :w_id, :datetime, :h_amount, :h_data);
EXEC SQL COMMIT WORK;
return(0);

sqlerr:
    error();
}
```