**《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 3.0)

2023年9月

目 录

[1. 数据处理测试基准TPC Benchmark 4](#_Toc116424769)

[1.1 背景 4](#_Toc116424770)

[1.2分类 4](#_Toc116424771)

[1.3 TPC国际组织及官网主页 4](#_Toc116424772)

[1.4 数据下载地址 5](#_Toc116424773)

[2. TPC-H Benchmark测试及数据集 6](#_Toc116424774)

[2.1 关系模式图 6](#_Toc116424775)

[2.1.1 订单表ORDERS 6](#_Toc116424776)

[2.1.2 区域表REGION 7](#_Toc116424777)

[2.1.3 国家表 NATION 7](#_Toc116424778)

[2.1.4 供应商表SUPPLIER 7](#_Toc116424779)

[2.1.5 零部件表PART 8](#_Toc116424780)

[2.1.6 零部件供应表PARTSUPP 8](#_Toc116424781)

[2.1.7 客户表CUSTOMER 9](#_Toc116424782)

[2.1.8 订单明细表LINEITEM 9](#_Toc116424783)

[2.2 数据对象mapping cardinality/cardinality limit及表间参照完整性约束 12](#_Toc116424784)

[2.3 TPC-H关系模式及其属性 13](#_Toc116424785)

[2.3.1 订单表ORDERS 13](#_Toc116424786)

[2.3.2 供应商表SUPPLIER 13](#_Toc116424787)

[2.3.3 地区表REGION 13](#_Toc116424788)

[2.3.4 国家表 14](#_Toc116424789)

[2.3.5 零部件表 14](#_Toc116424790)

[2.3.6 零部件供应表 14](#_Toc116424791)

[2.3.7 客户表 15](#_Toc116424792)

[2.3.8 订单明细表 15](#_Toc116424793)

[3. TPC-H Benchmark数据集生成 16](#_Toc116424794)

[3.1 基于脚本的数据生成 16](#_Toc116424795)

[3.2 生成的关系表数据 16](#_Toc116424796)

[3.3 数据生成脚本示例 16](#_Toc116424797)

[4. 典型TPC-H测试语句 18](#_Toc116424798)

[4.1 定价汇总报表查询 18](#_Toc116424799)

[4.2 最低成本供应商查询 18](#_Toc116424800)

[4.3 运输优先级查询 20](#_Toc116424801)

[4.4 订单优先级查询 20](#_Toc116424802)

[4.5 本地供应商收入量查询 21](#_Toc116424803)

[4.6 预测收入变化查询 22](#_Toc116424804)

[4.7 批量出货查询 22](#_Toc116424805)

[4.8 全国市场份额查询 23](#_Toc116424806)

[4.9 产品类型利润度量查询 24](#_Toc116424807)

[4.10 退货报告查询 25](#_Toc116424808)

[4.11 重要库存识别查询 26](#_Toc116424809)

[4.12 运送方式和订单优先级查询 26](#_Toc116424810)

[4.13 客户分布查询 27](#_Toc116424811)

[4.14 促销效果查询 28](#_Toc116424812)

[4.15 顶级供应商查询 28](#_Toc116424813)

[4.16 零部件/供应商关系查询 29](#_Toc116424814)

[4.17 小额订单收入查询 30](#_Toc116424815)

[4.18 大批量客户查询 31](#_Toc116424816)

[4.19 折扣收入查询 32](#_Toc116424817)

[4.20 潜在零部件促销查询 33](#_Toc116424818)

[4.21 不能按时交货供应商查询 34](#_Toc116424819)

[4.22 全球销售机会查询 35](#_Toc116424820)

[5. TPC-H数据库建库脚本 36](#_Toc116424821)

[5.1 建表及数据导入步骤 36](#_Toc116424822)

[5.2 订单表ORDERS 36](#_Toc116424823)

[5.3 区域表REGION 36](#_Toc116424824)

[5.4 国家表NATION 36](#_Toc116424825)

[5.5 供应商表SUPPLIER 37](#_Toc116424826)

[5.6 零部件表PART 37](#_Toc116424827)

[5.7 零部件供应表PARTSUPP 37](#_Toc116424828)

[5.8 客户表CUSTOMER 37](#_Toc116424829)

[5.9 订单明细表LINEITEM 38](#_Toc116424830)

[5.10 约束 38](#_Toc116424831)

[6. TPC-C测试基准 40](#_Toc116424832)

[6.1 实体-联系图 40](#_Toc116424833)

[6.2 关系表 41](#_Toc116424834)

[6.2.1 仓库表WAREHOUSE 41](#_Toc116424835)

[6.2.2 区域表DISTRICT 41](#_Toc116424836)

[6.2.3 客户表CUSTOMER 42](#_Toc116424837)

[6.2.4 历史记录表HISTORY 43](#_Toc116424838)

[6.2.5 新订单表NEW-ORDER 43](#_Toc116424839)

[6.2.6 订单表ORDER 44](#_Toc116424840)

[6.2.7 订单行表ORDER-LINE 44](#_Toc116424841)

[6.2.8 条目表ITEM 45](#_Toc116424842)

[6.2.9 库存表STOCK 45](#_Toc116424843)

[6.3 基准测试程序示例 46](#_Toc116424844)

[6.3.1 基于嵌入式SQL的测试基准程序 46](#_Toc116424845)

[6.3.2 A1新订单事务The New-Order Transaction 46](#_Toc116424846)

[6.3.3 A2 The Payment Transaction 49](#_Toc116424847)

# 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，然后通过将数据导入系统中，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

# 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，tpc1g的part表，其数据量＝基础数据量200000\*1=200000（行），tpch100g的 part 表，其数据量＝200000\*100=20000000（行）。

## 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：

### 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\_orderestatus 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~ SF \*1500000；

与表lineitem 表间有外键/参照完整性关联关联。

### 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。

### 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。

### 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。

### 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。

### 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。

### 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。

### 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

## 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，这些数据对象间的关定量联关系如下。

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suppkey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suppkey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。

## TPC-H关系模式及其属性

### 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 |

### 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 |

### 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) |  |

### 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) |  |

### 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 |

### 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 |

### 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 |

### 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 |

# TPC-H Benchmark数据集生成

## 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。

## 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 |  |

## 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>



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：

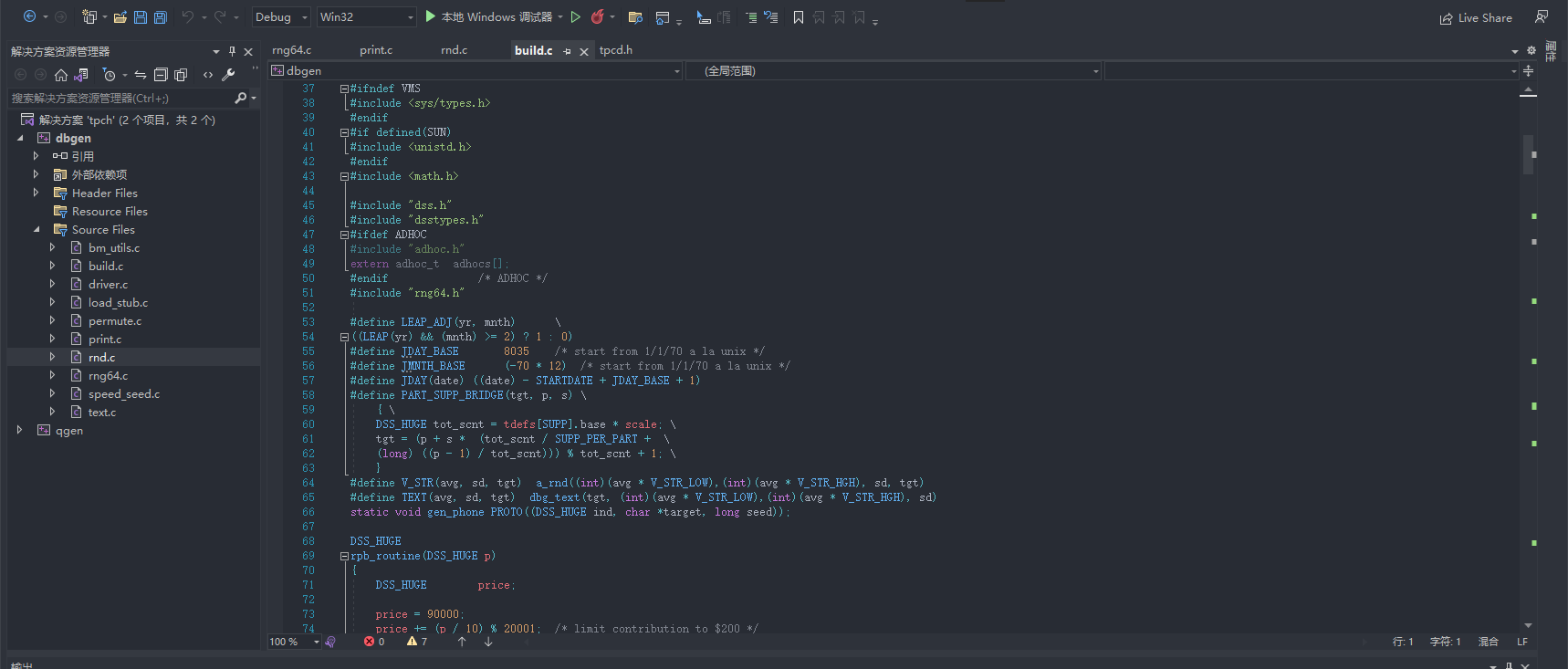


图4 TPC-H测试基准生成程序（部分）示例

# 典型TPC-H测试语句

## 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;

## 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。 返回供应商的账户余额、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

## 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

## 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;

## 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;

## 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;

## 批量出货查询

给定时间和供应商所在的国家和客户所在的国家，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;

## 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;

## 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;

## 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

## 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;

## 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;

## 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;

## 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;

## 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;

## 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;

## 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

);

## 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;

## 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'

);

## 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;

## 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;

## 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

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;

# TPC-H数据库建库脚本

## 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**

## 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);

## 区域表REGION

CREATE TABLE REGION ( R\_REGIONKEY INTEGER NOT NULL,

R\_NAME CHAR(25) NOT NULL,

R\_COMMENT VARCHAR(152));

## 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));

## 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);

## 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 );

## 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 );

## 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);

## 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);

## 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;

# TPC-C测试基准

## 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。

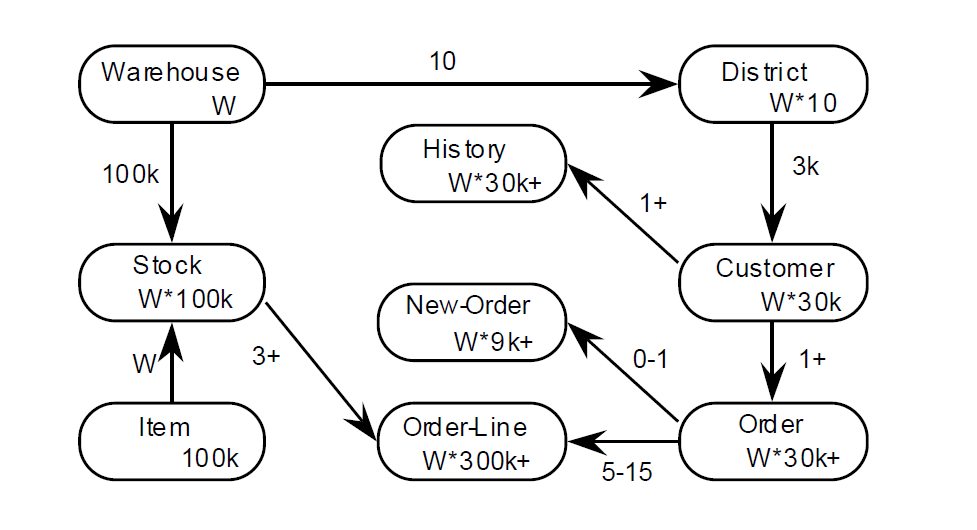


图5 TPC-C实体-联系图

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。

## 关系表

与上述实体-联系图对应的9张关系表如下。

### 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 |

### 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 |

### 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 |

### 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 |

### 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 |

### 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) |  |

### 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 |  |

### 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 |

### 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 |

## 基准测试程序示例

### 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。

### 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）

1. 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();

}

### 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。

1. 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。
2. 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();

}