# 实验一 数据库定义与操作语言实验

16337341 朱志儒

## 实验1.1 数据库定义实验

### 实验目的：

理解和掌握数据库DDL语言，能够熟练的使用SQL DDL语句创建、修改和删除数据库、模式和基本表。

### 实验内容：

建立TPCH数据库模式，TPCH数据库模式由Part、Supplier、PartSupp、Customer、Nation、Region、Orders、Lineitem 8个基本表组成。

### 实验步骤：

1. **定义数据库**

采用中文字符集创建名为TPCH的数据库。

CREATE DATABASE TPCH;

1. **定义模式**

在数据库TPCH中创建名为Sales的模式。

CREATE SCHEMA Sales;

1. **定义基本表**

在TPCH数据库的Sales模式中创建8个基本表。

**地区表：**

    CREATE TABLE Region (

        regionkey INT PRIMARY KEY,

        name CHAR(250),

        comment VARCHAR(152));

**国家表：**

    CREATE TABLE Nation (

        nationkey INT PRIMARY KEY,

        name CHAR(250),

        regionkey INT REFERENCES Region(regionkey),

        comment VARCHAR(152));

**供应商基本表：**

    CREATE TABLE Supplier (

        suppkey INT PRIMARY KEY,

        name CHAR(250),

        address VARCHAR(400),

        nationkey INT REFERENCES Nation(nationkey),

        phone CHAR(150),

        acctbal REAL,

        comment VARCHAR(101));

**零件基本表：**

    CREATE TABLE Part (

        partkey INT PRIMARY KEY,

        name VARCHAR(550),

        mfgr CHAR(250),

        brand CHAR(100),

        type VARCHAR(250),

        size INT,

        container CHAR(100),

        retaiprice real,

        comment VARCHAR(230));

**零件供应联系表：**

    CREATE TABLE PartSupp (

        partkey INT REFERENCES Part(partkey),

        suppkey INT REFERENCES Supplier(suppkey),

        availqty INT,

        supplycost REAL,

        comment VARCHAR(199),

        PRIMARY KEY(partkey, suppkey));

**顾客表：**

    CREATE TABLE Customer (

        custkey INT PRIMARY KEY,

        name VARCHAR(250),

        address VARCHAR(400),

        nationkey INT REFERENCES Nation(nationkey),

        phone CHAR(150),

        acctbal REAL,

        mktsegment CHAR(100),

        comment VARCHAR(117));

**订单表：**

    CREATE TABLE Orders (

        orderkey INT PRIMARY KEY,

        custkey INT REFERENCES Customer(custkey),

        ordorstatus CHAR(1),

        totalprice REAL,

        orderdate DATE,

        orderpriority CHAR(150),

        cler CHAR(150),

        shippriority INT,

        comment VARCHAR(790));

**订单明细表：**

    CREATE TABLE Lineitem (

        orderkey INT REFERENCES Orders(orderkey),

        partkey INT REFERENCES Part(partkey),

        suppkey INT REFERENCES Supplier(suppkey),

        linenumber INT,

        quantity REAL,

        extendedprice REAL,

        discount REAL,

        tax REAL,

        returnflag CHAR(1),

        linestatus CHAR(1),

        shipdate DATE,

        commitdate DATE,

        receiptdate DATE,

        shipinstruct CHAR(250),

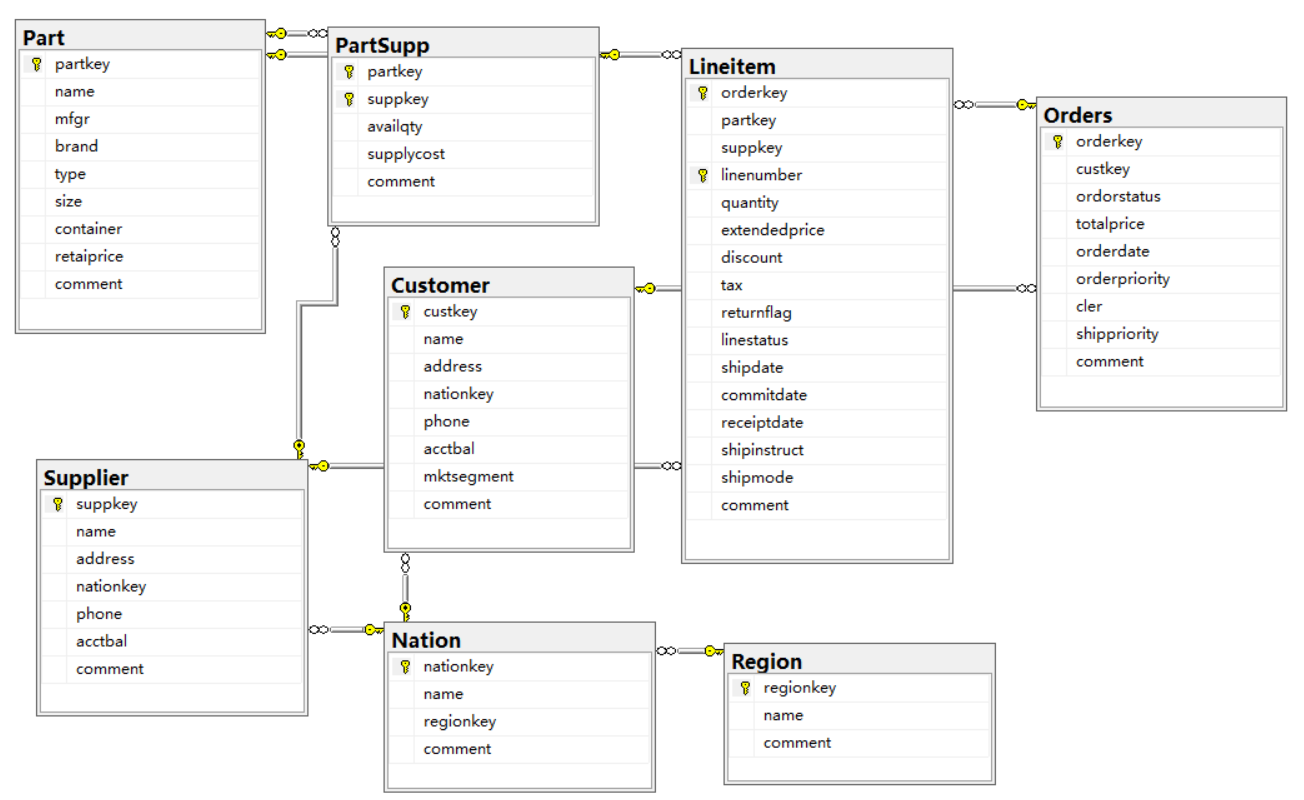
        shipmode CHAR(100),

        comment VARCHAR(440),

        PRIMARY KEY(orderkey, linenumber),

        FOREIGN KEY(partkey, suppkey) REFERENCES PartSupp(partkey, suppkey));

**数据库关系图：**



### 实验总结：

在本次实验中，我使用SQL语句创建、修改和删除数据库、模式和基本表，掌握了SQL语句常见语法错误的调试方法。通过这次实验，我理解并掌握了SQL DDL语句的语法，尤其是各种参数的具体含义和使用方法。在定义表格的时候注意主码和外码的确定和定义方式，注意每个变量类型和长度的定义，

## 实验1.2 数据库基本查询实验

### 实验目的

掌握SQL程序设计基本规范，熟练运用SQL语言实现数据基本查询，包括单表查询、分组统计查询和连接查询。

### 实验内容

针对TPCH数据库设计各种单表查询SQL语句、分组统计查询语句；设计单个表针对自身的连接查询，设计多个表的连接查询。理解和掌握SQL查询语句各个子句的特点和作用，按照SQL程序设计规范写出具体的SQL查询语句，并调试通过。

### 实验步骤

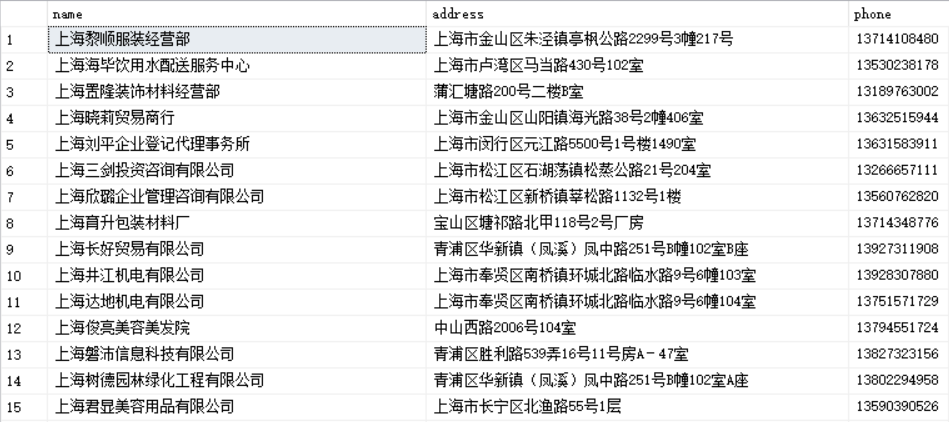
1. **单表查询（实现投影操作）**

查询供应商的名称、地址和联系电话。

SELECT name, address, phone

FROM Supplier;

结果：



1. **单表查询（实现选择操作）**

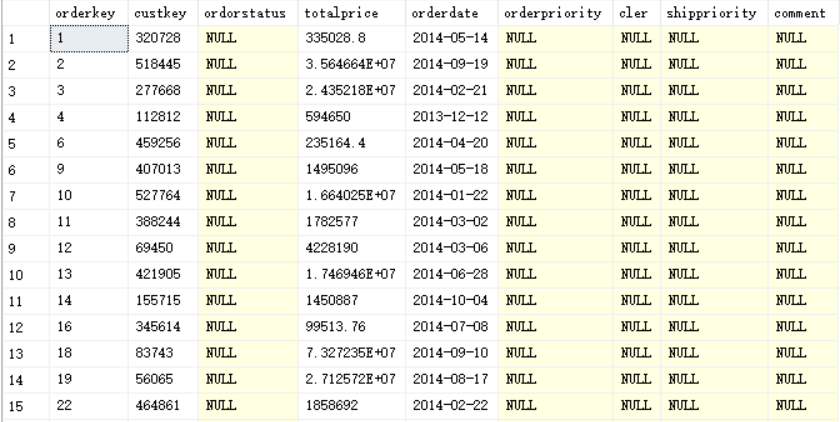
查询最近一周内提交的总价大于1000元的订单的编号、顾客编号等订单的所有信息。

SELECT \*

FROM Orders

WHERE DATEDIFF(d, CONVERT(date, GETDATE()), orderdate) < 7 AND totalprice > 1000;

结果：



1. **不带分组过滤条件的分组统计查询**

统计每个顾客的订购金额。

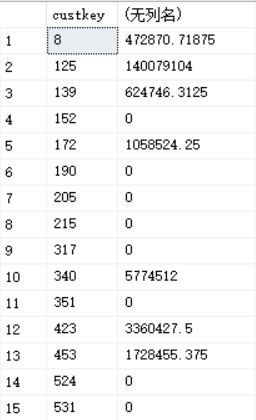
SELECT C.custkey, SUM(O.totalprice)

FROM Customer C, Orders O

WHERE C.custkey = O.custkey

GROUP BY C.custkey;

结果：



1. **带分组过滤条件的分组统计查询**

查询订单平均金额超过1000元的顾客编号及其姓名。

SELECT C.custkey, MAX(C.name)

FROM Customer C, Orders O

WHERE C.custkey = O.custkey

GROUP BY C.custkey

HAVING AVG(O.totalprice) > 1000;

结果：



1. **单表自身连接查询**

查询与“光明耗材公司”在同一个国家的供应商编号、名称和地址信息。

SELECT F.suppkey, F.name, F.address

FROM Supplier F, Supplier S

WHERE F.nationkey = S.nationkey AND S.name = '光明耗材公司';

结果：



1. **两表连接查询（普通连接）**

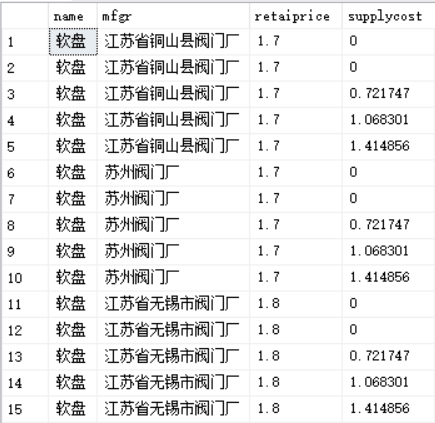
查询供应价格大于零售价格的零件名、制造商名、零售价格和供应价格。

SELECT P.name, P.mfgr, P.retaiprice, PS.supplycost

FROM Part P, PartSupp PS

WHERE P.retaiprice > PS.supplycost;

结果：



1. **两表连接查询（自然连接）**

查询供应价格大于零售价格的零件名、制造商名、零售价格和供应价格。

SELECT P.name, P.mfgr, P.retaiprice, PS.supplycost

FROM Part P, PartSupp PS

WHERE P.partkey = PS.partkey AND P.retaiprice > PS.supplycost

结果：



1. **三表连接查询**

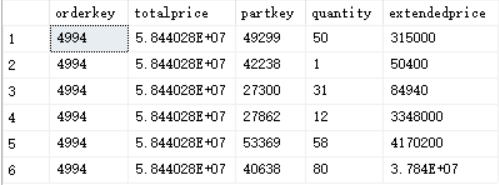
查询顾客“宁攸华”订购的订单编号、总价及其订购的零件编号、数量和明细价格。

SELECT O.orderkey, O.totalprice, L.partkey, L.quantity, L.extendedprice

FROM Customer C, Orders O, Lineitem L

WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND C.name = '宁攸华';

结果：



### 实验总结

只有正确数据库模式结构，才能正确设计数据库查询。连接查询是数据库SQL查询中最重要的查询，连接查询的设计要特别注意，不同的查询表达，其查询执行的性能会有很大差别。通过这次实验，我理解并掌握了SQL查询语句各个子句的特点和作用。

## 实验1.3 数据库高级查询实验

### 实验目的

掌握SQL嵌套查询和集合查询等各种高级查询的设计方法等。

### 实验内容

针对TPCH数据库，正确分析用户查询要求，设计各种嵌套查询和集合查询。

### 实验步骤

1. **IN嵌套查询**

查询订购了“北京蓝海办公”制造的“凌本LB125T－5B两轮摩托车”的顾客。

    SELECT custkey, name

    FROM Customer

    WHERE custkey IN (

        SELECT O.custkey

        FROM Orders O, Lineitem L, PartSupp PS, Part P

        WHERE O.orderkey = L.orderkey AND

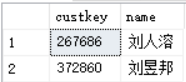
            L.partkey = PS.partkey AND

            L.suppkey = PS.suppkey AND

            PS.partkey = P.partkey AND

            P.mfgr = '北京蓝海办公' AND P.name = '凌本LB125T－5B两轮摩托车');

结果：



Lineitem表直接与Part表连接：

    SELECT custkey, name

    FROM Customer

    WHERE custkey IN (

        SELECT O.custkey

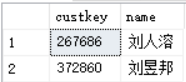
        FROM Orders O, Lineitem L, Part P

        WHERE O.orderkey = L.orderkey AND

            L.partkey = P.partkey AND

            P.mfgr = '北京蓝海办公' AND P.name = '凌本LB125T－5B两轮摩托车');

结果：



1. **单层EXISTS嵌套查询**

查询没有购买过“北京蓝海办公” 制造的“凌本LB125T－5B两轮摩托车”的顾客。

    SELECT custkey, name

    FROM Customer C

    WHERE NOT EXISTS (SELECT O.custkey

        FROM Orders O, Lineitem L, PartSupp PS, Part P

        WHERE C.custkey = O.custkey AND

            O.orderkey = L.orderkey AND

            L.partkey = PS.partkey AND

            L.suppkey = PS.suppkey AND

            PS.partkey = P.partkey AND

            P.mfgr = '北京蓝海办公' AND P.name = '凌本LB125T－5B两轮摩托车');

结果：



1. **双层EXISTS嵌套查询**

查询至少购买过顾客“张三”购买过的全部零件的顾客姓名。

    SELECT CA.name

    FROM Customer CA

    WHERE NOT EXISTS(

        SELECT \*

        FROM Customer CB, Orders OB, Lineitem LB

        WHERE CB.custkey = OB.custkey AND

            OB.orderkey = LB.orderkey AND

            CB.name = '张三' AND

            NOT EXISTS(

                SELECT \*

                FROM Orders OC, Lineitem LC

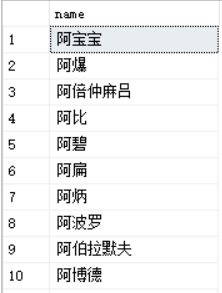
                WHERE CA.custkey = OC.custkey AND

                    OC.orderkey = LC.orderkey AND

                    LB.suppkey = LC.suppkey AND

                    LB.partkey = LC.partkey));

结果：



1. **FROM子句中的嵌套查询**

查询订单平均金额超过1万元的顾客中的中国籍顾客信息。

    SELECT C.\*

    FROM Customer C, (SELECT custkey

        FROM Orders

        GROUP BY custkey

        HAVING AVG(totalprice) > 10000) B, Nation N

    WHERE C.custkey = B.custkey AND C.nationkey = N.nationkey AND N.name = '中国';

结果：



1. **集合查询（交）**

查询顾客“曹操”和“阿波罗”都订购过的全部零件的信息。

SELECT P.\*

FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

PS.partkey = P.partkey AND C.name = '曹操'

INTERSECT

SELECT P.\*

FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

PS.partkey = P.partkey AND C.name = '阿波罗'

结果：



1. **集合查询（并）**

查询顾客“曹操”和“阿波罗”订购的全部零件的信息。

    SELECT P.\*

    FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

        PS.partkey = P.partkey AND C.name = '曹操'

    UNION

    SELECT P.\*

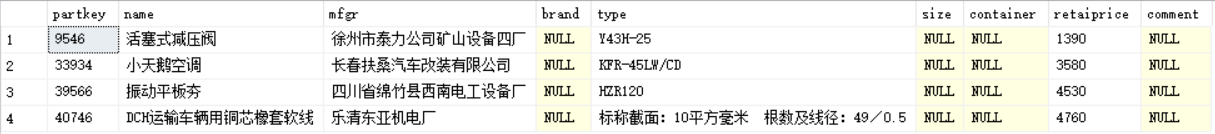
    FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

        PS.partkey = P.partkey AND C.name = '阿波罗'

结果：



1. **集合查询（差）**

查询顾客“曹操”订购过而“阿波罗”没有订购过的零件的信息。

    SELECT P.\*

    FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

        PS.partkey = P.partkey AND C.name = '曹操'

    EXCEPT

    SELECT P.\*

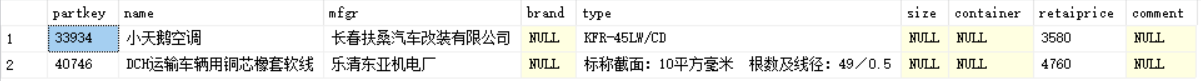
    FROM Customer C, Orders O, Lineitem L, PartSupp PS, Part P

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND

        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND

        PS.partkey = P.partkey AND C.name = '阿波罗'

结果：



### 实验总结

通过本次实验，我理解并掌握了SQL嵌套查询和集合查询等各种高级查询的设计方法。

## 实验1.4 数据更新实验

### 实验目的

熟悉数据库的数据更新操作，能够使用SQL语句对数据库进行数据的插入、修改、删除操作。

### 实验内容

针对TPCH数据库设计单元组插入、批量数据插入、修改数据和删除数据等SQL语句。理解和掌握INSERT、UPDATE和DELETE语法结构的各个组成成分，结合嵌套SQL子查询，分别设计几种不同形式的插入、修改和删除数据的语句，并调试成功。

### 实验步骤

1. **INSERT基本语句（插入全部列的数据）**

插入一条顾客记录，要求每列都给一个合理的值。

    INSERT INTO Customer

    VALUES(3799831, '张三', '北京市', 40, '010-51001199', 0.00, 'Northeast', 'VIP Customer');

结果：



1. **INSERT基本语句（插入部分列的数据）**

插入一条订单记录，给出必要的几个字段值。

    INSERT INTO Lineitem(orderkey, linenumber, partkey, suppkey, quantity, shipdate)

    VALUES(862, ROUND(RAND() \* 100, 0), 7531, 10911, 10, '2012-3-6');

结果：



1. **批量数据INSERT语句**

* 创建一个新的顾客表，把所有中国籍顾客插入到新的顾客表中。

    CREATE TABLE NewCustomer AS SELECT \* FROM Customer WITH NO DATA;

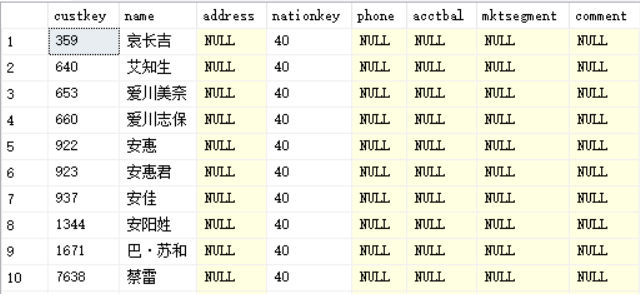
    INSERT INTO NewCustomer

    SELECT C.\*

    FROM Customer C, Nation N

    WHERE C.nationkey = N.nationkey AND N.name = '中国');

结果：



* 创建一个顾客购物统计表，记录每个顾客及其购物总数和总价等信息。

    CREATE TABLE ShoppingStat

    (custkey INT,

        quantity REAL,

        totalprice REAL);

    INSERT INTO ShoppingStat

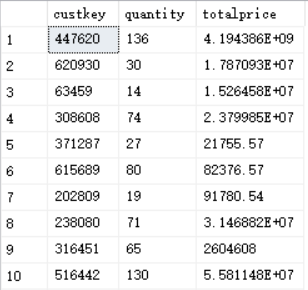
    SELECT C.custkey, SUM(L.quantity), SUM(O.totalprice)

    FROM Customer C, Orders O, Lineitem L

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey

    GROUP BY C.custkey;

结果：



* 倍增零件表的数据，多次重复执行，直到总记录数达到50万为止。

    INSERT INTO Part

    SELECT partkey + (SELECT COUNT(\*) FROM Part), name, mfgr, brand, type, size, container, retaiprice, comment

    FROM Part;

结果：



1. **UPDATE语句（修改部分记录的部分列值）**

“上海三剑投资咨询有限公司”供应的所有零件的供应成本价下降10%。

    UPDATE PartSupp

    SET supplycost = supplycost \* 0.9

    WHERE suppkey = (SELECT suppkey

        FROM Supplier

        WHERE name = '上海三剑投资咨询有限公司');

结果：执行前：

执行后：



1. **UPDATE语句（利用一个表中的数据修改另一个表中的数据）**

利用Part表中的零售价格来修改Lineitem中的extendedprice，其中extendedprice = Part.retailprice \* quantity。

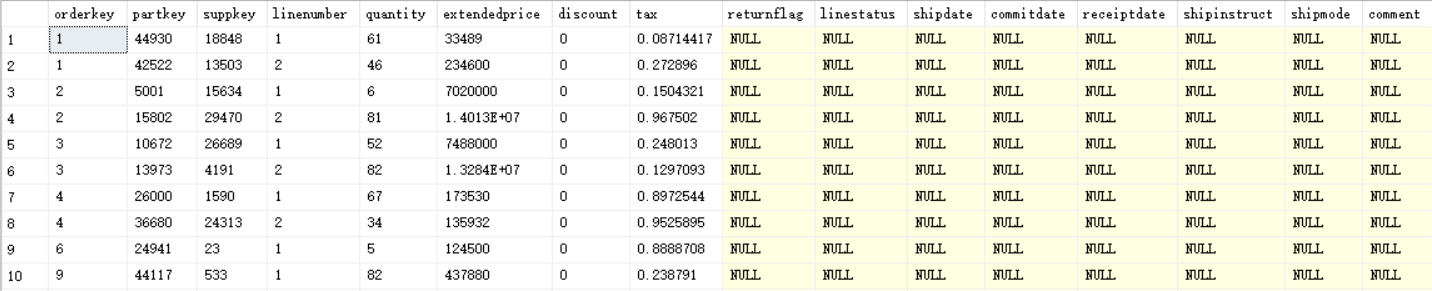
    UPDATE Lineitem

    SET extendedprice = Part.retaiprice \* Lineitem.quantity

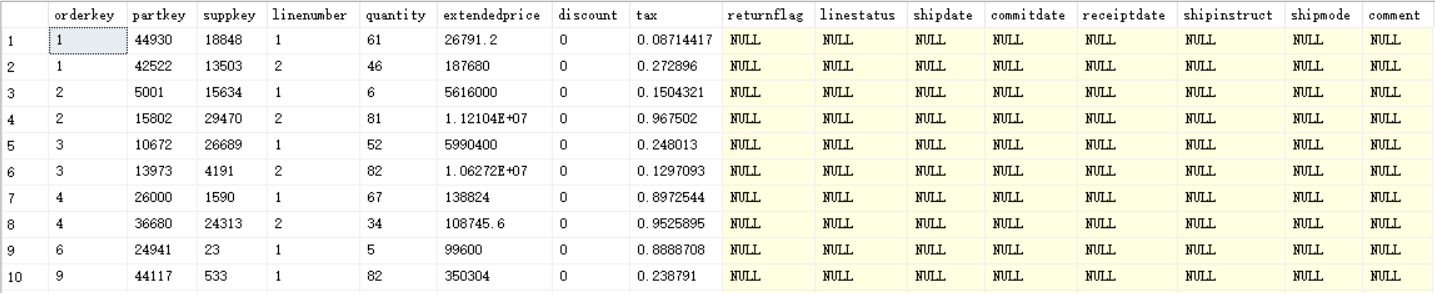
    FROM Part

    WHERE Part.partkey = Lineitem.partkey;

结果：执行前：



执行后：



1. **DELETE基本语句（删除给定条件的所有记录）**

删除顾客宁攸华的所有订单记录。

    DELETE FROM Lineitem

    WHERE orderkey IN (SELECT orderkey

        FROM Orders O, Customer C

        WHERE O.custkey = C.custkey AND C.name = '宁攸华');

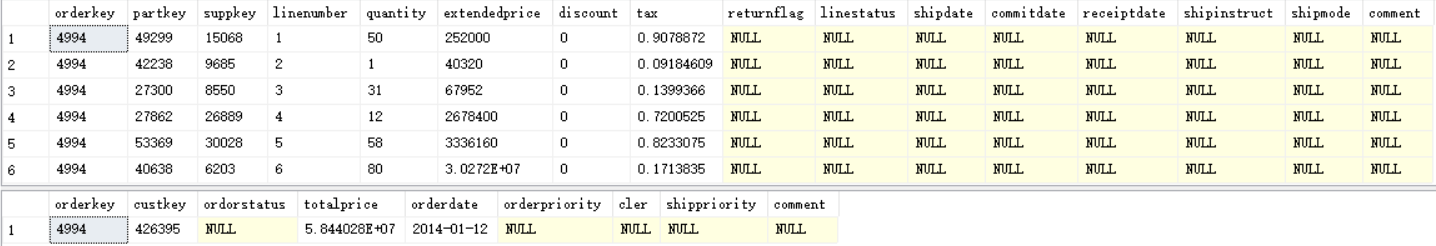
    DELETE FROM Orders

    WHERE custkey = (SELECT custkey

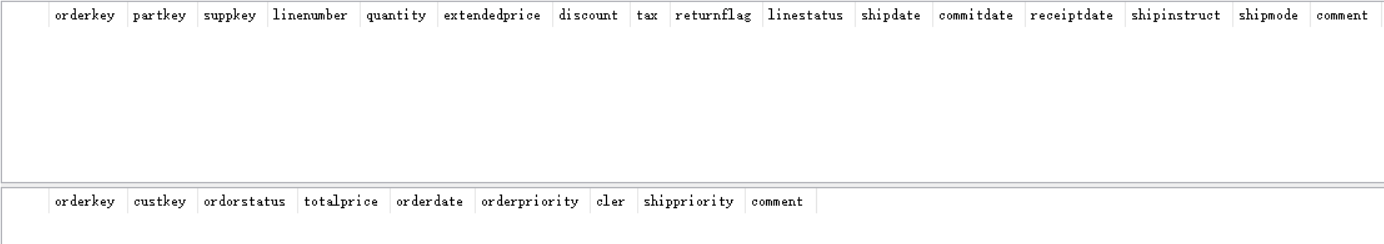
        FROM Customer

        WHERE name = '宁攸华');

结果：执行前：



执行后：



### 实验总结

正确地设计和执行数据更新语句，确保正确地录入数据和更新数据，才能保证查询出来的数据正确。当更新数据失败时，一个主要原因是更新数据时违反了完整性约束。

## 实验1.5 视图实验

### 实验目的

熟悉SQL语言有关视图的操作，能够熟练使用SQL语句来创建需要的视图，定义数据库外模式，并能使用所创建的视图实现数据管理。

### 实验内容

针对给定的数据库模式，以及相应的应用需求，创建视图和带WITH CHECK OPTION的视图，并验证视图WITH CHECK OPTION选项的有效性。理解和掌握视图消解执行原理，掌握可更新视图和不可更新视图的区别。

### 实验步骤

1. **创建视图（省略视图列名）**

创建一个“上海三剑投资咨询有限公司”供应商供应的零件视图V\_DLMU\_PartSupp1，要求列出供应零件的编号、零件名称、可用数量、零售价格、供应价格和备注的信息。

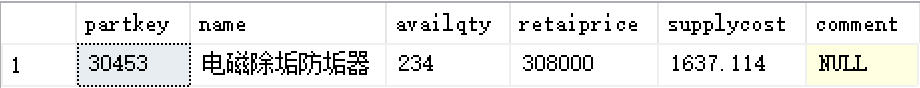
    CREATE VIEW V\_DLMU\_PARTSUPP1 AS

    SELECT P.partkey, P.name, PS.availqty, P.retaiprice, PS.supplycost, P.comment

    FROM Part P, PartSupp PS, Supplier S

    WHERE P.partkey = PS.partkey AND S.suppkey = PS.suppkey AND S.name = '上海三剑投资咨询有限公司';

结果：



1. **创建视图（不能省略列名的情况）**

创建一个视图V\_CustAvgOrder，按顾客统计平均每个订单的购买金额和零件数量，要求输出顾客编号、姓名，平均购买金额和平均购买零件数量。

    CREATE VIEW V\_CustAvgOrder(custkey, cname, avgprice, avgquantity) AS

    SELECT C.custkey, MAX(C.name), AVG(O.totalprice), AVG(L.quantity)

    FROM Customer C, Orders O, Lineitem L

    WHERE C.custkey = O.custkey AND L.orderkey = O.orderkey

    GROUP BY C.custkey;

结果：



1. **创建视图（WITH CHECK OPTION）**

使用WITH CHECK OPTION，创建一个“上海三剑投资咨询有限公司”供应商供应的零件视图V\_DLMU\_PartSupp2，要求列出供应零件的编号、可用数量和供应价格等信息。然后通过该视图分别增加、删除和修改一条“上海三剑投资咨询有限公司”零件供应记录，验证WITH CHECK OPTION是否起作用。

创建视图：

    CREATE VIEW V\_DLMU\_PartSupp2 AS

    SELECT partkey, suppkey, availqty, supplycost

    FROM PartSupp

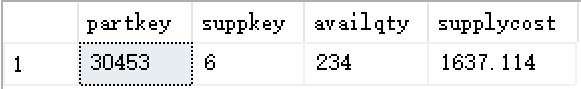
    WHERE suppkey = (SELECT suppkey

        FROM Supplier

        WHERE name = '上海三剑投资咨询有限公司')

    WITH CHECK OPTION;

结果：

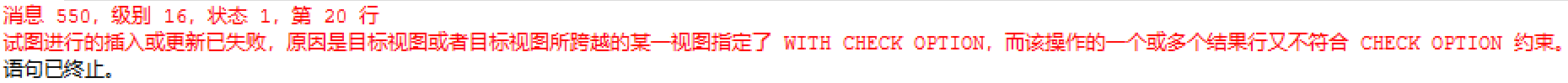


通过该视图进行添加操作：

    INSERT INTO V\_DLMU\_PartSupp2

    VALUES(58889, 5048, 704, 77760);

结果：



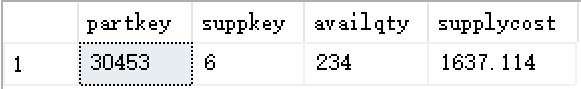
修改操作：

    UPDATE V\_DLMU\_PartSupp2

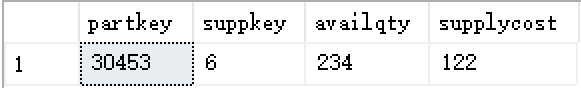
    SET supplycost = 122

    WHERE suppkey = 6;

执行前：



执行后：

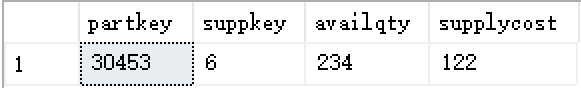


删除操作：

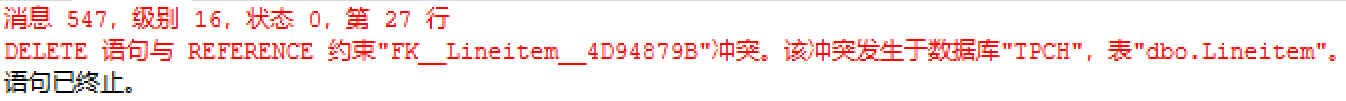
    DELETE FROM V\_DLMU\_PartSupp2

    WHERE suppkey = 6;

执行前：



执行后：



从上面的实验可以看出WITH CHECK OPTION在进行INSERT操作时起作用了。

1. **可更新的视图（行列子集视图）**

创建一个“上海三剑投资咨询有限公司”供应商供应的零件视图V\_DLMU\_PartSupp3，要求列出供应零件的编号、可用数量和供应价格等信息。然后通过该视图分别增加、删除和修改一条“上海三剑投资咨询有限公司”零件供应记录，验证该视图是否可更新的，并比较上述“3）创建视图”实验任务与本任务结果有何异同。

创建视图：

    CREATE VIEW V\_DLMU\_PartSupp3 AS

    SELECT partkey, suppkey, availqty, supplycost

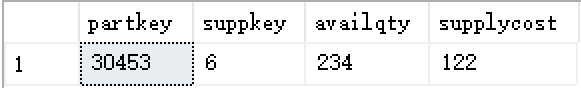
    FROM PartSupp

    WHERE suppkey = (SELECT suppkey

        FROM Supplier

        WHERE name = '上海三剑投资咨询有限公司');

结果：

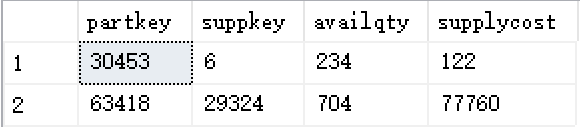


增加操作：

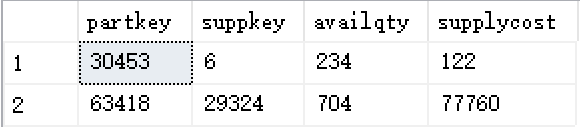
    INSERT INTO V\_DLMU\_PartSupp3

    VALUES(63418, 29324, 704, 77760);

通过视图看不到INSERT的结果：



而新插入的项可以查询到：



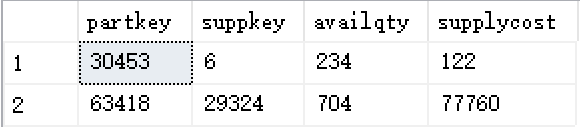
修改操作：

    UPDATE V\_DLMU\_PartSupp3

    SET supplycost = 12

    WHERE suppkey = 6;

执行前：



执行后：



删除操作：

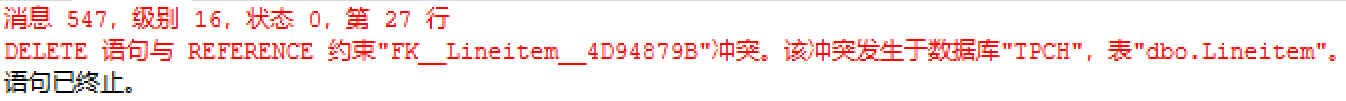
    DELETE FROM V\_DLMU\_PartSupp3

    WHERE suppkey = 6;

执行前：



执行后：



实验显示该视图可以更新，与3）相比，在没有WITH CHECK OPTION的情况下，可以进行INSERT操作，UPDATE和DELETE操作与3）相比没有区别。

1. **不可更新的视图**

2）中创建的视图是可更新的吗？通过SQL更新语句加以验证，并说明原因。

    INSERT INTO V\_CustAvgOrder

    VALUES(100000, NULL, 20, 2000);

结果：



实验显示2）中创建的视图不可更新，因为所创建的视图对其属性值进行了计算的其他形式上的改变，而对视图的更改最终表现为对表的更改而表中不存在视图的某一属性，或属性的性质不相同，则无法更改，这是一种视图机制。

1. **删除视图（RESTRICT/CASCADE）**

创建顾客订购零件明细视图V\_CustOrd，要求列出顾客编号、姓名、购买零件数、金额，然后在该视图的基础上，再创建2）的视图V\_CustAvgOrder，然后使用RESTRICT选项删除视图V\_CustOrd，观察现象并解释原因。利用CASCADE选项删除V\_CustOrd，观察现象并检查V\_CustAvgOrder是否存在，解释原因？

创建视图V\_CustOrd：

    CREATE VIEW V\_CustOrd(custkey, cname, qty, extprice) AS

    SELECT C.custkey, C.name, L.quantity, L.extendedprice

    FROM Customer C, Orders O, Lineitem L

    WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey;

结果：



创建视图V\_CustAvgOrder：

    CREATE VIEW V\_CustAvgOrder(custkey, cname, avgqty, avgprice) AS

    SELECT custkey, MAX(cname), AVG(qty), AVG(extprice)

    FROM V\_CustOrd

    GROUP BY custkey;

结果：



使用RESTRICT选项删除视图V\_CustOrd：

    DROP VIEW V\_CustOrd RESTRICT;

结果：



利用CASCADE选项删除V\_CustOrd：

    DROP VIEW V\_CustOrd CASCADE;

结果：



实验显示SQL Server2017不支持级联删除，不过可以通过先删除视图V\_CustAvgOrder，再删除视图V\_CustOrd达到级联删除的效果。

### 实验总结

通过本次实验，我了解到WITH CHECK OPTION的作用：在视图上的修改都要符合视图定义时的SELECT语句所指定的限制条件，如果不符合限制条件则不能执行操作。

对于视图的更新我有了更深刻的了解：一般行列子集视图是可以更新的，因为它只是去掉了基本表的某些行或者列，并且保留了主码，对视图的更新通过视图消解可以转化为对基本表的更新。

我还了解了RESTRICT和CASCADE的作用：RESTRICT是如果子表中有匹配的记录,则不允许对父表对应候选键进行DELETE操作，CASCADE是在父表上DELETE记录时，同步DELETE掉子表的匹配记录。

## 实验1.6 索引实验

### 实验目的

掌握索引设计原则和技巧，能够创建合适的索引以提高数据库查询、统计分析效率。

### 实验内容

针对给定的数据库模式和具体应用要求，创建唯一索引、函数索引、复合索引等；修改索引；删除索引。设计相应的SQL查询验证索引有效性。学习利用EXPLAIN命令分析SQL查询是否使用了所创建的索引，并能够分析其原因，执行SQL查询并估算索引提高查询效率的百分比。要求实验数据集达到10万条记录以上的数据量，以便验证索引效果。

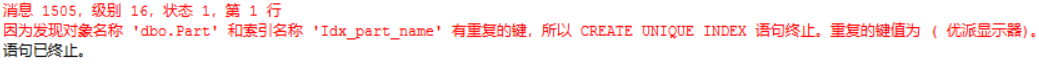
### 实验步骤

1. **创建唯一索引**

在零件表的零件名称字段上创建唯一索引。

    CREATE UNIQUE INDEX Idx\_part\_name ON Part(name);

结果：



创建唯一索引失败，因为Part表中的name列上存在重复的键。

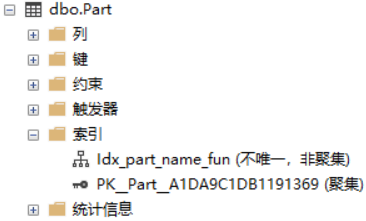
1. **创建函数索引（对某个属性的函数创建索引，称为函数索引）**

在零件表的零件名称字段上创建一个零件名词长度的函数索引（由于SQL Server不支持函数索引，所以添加长度列，并在该列上创建索引）。

ALTER TABLE Part ADD name\_length AS LEN(name) PERSISTED

CREATE INDEX Idx\_part\_name\_fun ON Part(name\_length);

结果：



1. **创建复合索引（对两个及两个以上的属性创建索引，称为复合索引）**

在零件表的制造商和品牌两个字段上创建一个复合索引。

    CREATE INDEX Idx\_part\_mfgr\_brand ON Part(mfgr, brand);

结果：



1. **\*创建聚簇索引**

在零件表的制造商字段上创建一个聚簇索引（SQL Server自动将主键设为聚簇索引，所以新创建表Part2的同时创建聚簇索引）。

CREATE TABLE Part2(

partkey INT,

name VARCHAR(550),

mfgr CHAR(250),

brand CHAR(100),

type VARCHAR(250),

size INT,

container CHAR(100),

retailprice REAL,

comment VARCHAR(230)

);

CREATE CLUSTERED INDEX Idx\_part\_mfgr ON Part2(mfgr);

结果：



1. **创建Hash索引**

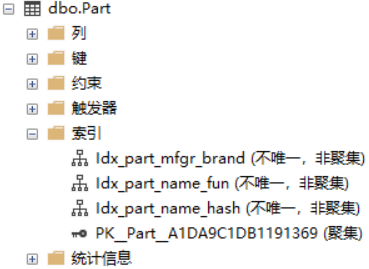
在零件表的名称字段上创建一个Hash索引。

ALTER TABLE Part ADD hash\_name

AS CAST(HASHBYTES('MD2', name) AS UNIQUEIDENTIFIER) PERSISTED;

CREATE INDEX Idx\_part\_name\_hash ON part(hash\_name);

结果：

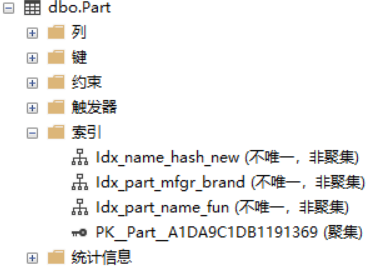


1. **修改索引名称**

修改零件表的名称字段上的索引名。

EXEC sp\_rename N'dbo.Part.Idx\_part\_name\_hash', N'Idx\_name\_hash\_new', N'INDEX';

结果：



1. **分析某个SQL查询语句执行时是否使用了索引**

EXPLAIN SELECT \* FROM Part WHERE name = '零件';

结果显示该SQL查询语句执行时没有使用索引。

1. **\*验证索引效率**

创建一个函数TestIndex，自动计算SQL查询执行的时间。

CREATE FUNCTION TestIndex (@p\_partname CHAR(55)) RETURNS INT

AS

BEGIN

DECLARE @begintime DATETIME;

DECLARE @endtime DATETIME;

DECLARE @durationtime INT;

DECLARE @TMP INT;

SET @begintime = CURRENT\_TIMESTAMP;

SELECT @TMP = SUM(partkey)

FROM Part

WHERE name = @p\_partname;

SET @endtime = CURRENT\_TIMESTAMP;

SET @durationtime = DATEDIFF(ms, @begintime, @endtime);

RETURN @durationtime;

END;

无索引时的执行时间：

SELECT TPCH.dbo.TestIndex('中科院电子所微波公司');

结果：



有索引时的执行时间：

CREATE INDEX part\_name ON Part(name);

SELECT TPCH.dbo.TestIndex('中科院电子所微波公司');

结果：



比较上述各次执行时间，可计算出索引提高查询效率的百分比为96%。

### 实验总结

在使用SQL Server创建聚簇索引时发现，如果在建表的时候建立了主键，没有建立聚簇索引，那么系统默认将主键设置为聚簇索引，因此想要在有主键的表上实现建立聚簇索引，只能先删除主键，这样做并不太好，所以创建新表Part2的同时创建聚簇索引。

通过这次实验，我发现创建索引可以加快查询的速度。