四 数据库课程实验

表 3 列出数据库课程实验总体情况：共设置 11 个实验、26 个实验项目，其中必修 12 个实验项目，选修 14 个实验项目。本表还列出了各实验项目对应的章节内容，以便复习实验内容。各类实验项目可以自由裁剪组合。

表 3 数据库课程实验一览表

序号	实验名称	开设类别	难易程度	实验项目数（必修/选修）	实验学时（必修/选修）	对应《概论》章节
实验 1	数据库定义与操作语言	必修	适中	5/1	10/2	第 3 章
实验 2	安全性语言	必修	适中	1/1	2/2	第 4 章
实验 3	完整性语言	必修	适中	2/1	4/2	第 5 章
实验 4	触发器	必修	适中	1/0	2/0	第 5 章
实验 5	数据库设计	必修	适中	1/0	4/0	第 7 章
实验 6	存储过程	必修	适中	1/2	2/4	第 8 章
实验 7	数据库应用开发	选修	难	0/2	0/8	第 8 章
实验 8	数据库设计与应用开发大作业	必修	适中	1/0	8/0	第 3~8 章
实验 9	数据库监视与性能优化	选修	较难	0/3	0/12	第 9 章
实验 10	数据库恢复技术	选修	适中	0/3	0/6	第 10 章
实验 11	并发控制	选修	适中	0/1	0/2	第 11 章
合计				12/14	32/38	

说明：实验学时仅列出课堂学时，部分实验（如实验 8）还需学生利用较多的课外时间来完成。

实验 1 数据库定义与操作语言实验

数据库定义与操作语言实验包含 6 个实验项目(参见表 4),其中有 5 个必修实验项目,1 个选修实验项目。实验项目 1.1~1.5 为设计性实验,实验项目 1.6 为验证性。

表 4 "数据库定义与操作语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 1.1	数据库定义实验	必修	适中	2	第 3 章 3.3
实验 1.2	数据基本查询实验	必修	适中	2	第 3 章 3.4
实验 1.3	数据高级查询实验	必修	较难	2	第 3 章 3.4
实验 1.4	数据更新实验	必修	适中	2	第 3 章 3.5
实验 1.5	视图实验	必修	适中	2	第 3 章 3.7
实验 1.6	索引实验	选修	适中	2	第 3 章 3.3.3

实验 1.1 数据库定义实验

1. 实验目的

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

2. 实验内容和要求

理解和掌握 SQL DDL 语句的语法,特别是各种参数的具体含义和使用方法;使用 SQL 语句创建、修改和删除数据库、模式和基本表。掌握 SQL 语句常见语法错误的调试方法。

3. 实验重点和难点

实验重点:创建数据库、基本表。

实验难点:创建基本表时,为不同的列选择合适的数据类型,正确创建表级和列级完整性约束,如列值是否允许为空、主码和外码等。注意:数据完整性约束可以在创建基本表时定义,也可以先创建表然后定义完整性约束。由于完整性约束的限制,被引用的表要先创建。

4. 实验报告示例

实 验 报 告				
题目:数据库定义实验	姓名		日期	
本实验建立 TPC-H 数据库模式(如图 2 所示)。TPC-H 数据库模式由零件表(Part)、供应商表(Supplier)、零件供应商联系表(Partsupp)、顾客表(Customer)、国家表(Nation)、地区表(Region)、订单表(Orders)和订单明细表(Lineitem)8 个基本表组成。				

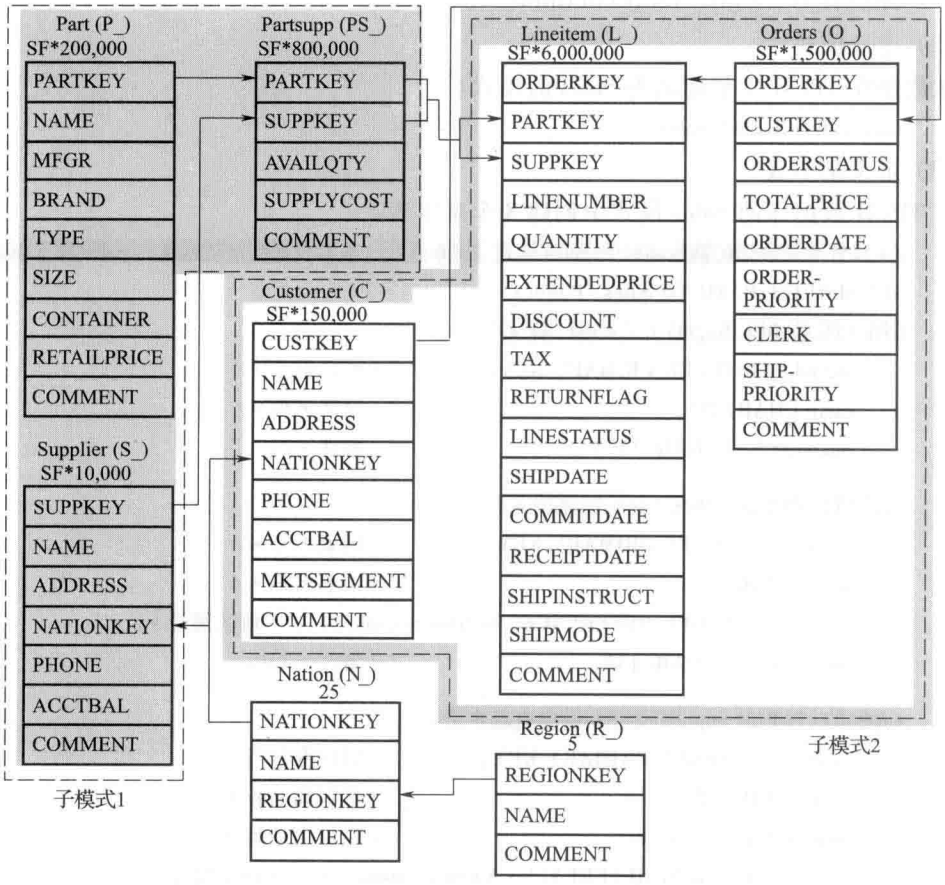


图 2　TPC-H 数据库模式图

TPC-H 数据库模式又可以分为以下两个子模式。

子模式 1:零件供应商子模式,包括 Part、Supplier 和 PartSupp 三个基本表,类似学生、课程和选课数据库模式,该子模式还可以增加 Nation 和 Region 两个表。该模式中 Part 和 Supplier 之间是多对多类型的联系。

子模式 2:顾客订单子模式,包括 Customer、Orders 和 Lineitem 三个表,该子模式也可以增加 Nation 和 Region 两个表。该模式中各实体之间主要是一对多类型的联系。

后续各个实验项目的示例基于 TPC-H 数据模式,或者是基于上述两个子模式。

(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 ENCODING = 'GBK';
```

(2) 定义模式

在数据库 TPCH 中创建名为 Sales 的模式。

```
CREATE SCHEMA Sales;
```

(3) 定义基本表

在 TPCH 数据库的 Sales 模式中创建 8 个基本表。

```
/*设置当前会话的搜索路径为 Sales 模式、Public 模式,基本表就会自动创建在 Sales 模式下。*/
SET SEARCH_PATH TO Sales, Public;
CREATE TABLE Region ( /*地区表*/
    regionkey INTEGER PRIMARY KEY,          /*地区编号*/
    name CHAR(25),                          /*地区名称*/
    comment VARCHAR(152)                    /*备注*/) ;

CREATE TABLE Nation ( /*国家表*/
    nationkey INTEGER PRIMARY KEY,          /*国家编号*/
    name CHAR(25),                          /*国家名称*/
    regionkey INTEGER REFERENCES Region(regionkey), /*地区编号*/
    comment VARCHAR(152)                    /*备注*/) ;

CREATE TABLE Supplier ( /*供应商基本表*/
    suppkey  INTEGER PRIMARY KEY,           /*供应商编号*/
    name CHAR(25),                          /*供应商名称*/
    address VARCHAR(40),                    /*供应商地址*/
    nationkey INTEGER REFERENCES Nation(nationkey), /*国家编号*/
    phone CHAR(15),                         /*供应商电话*/
    acctbal REAL,                           /*供应商账户余额 account balance*/
    comment VARCHAR(101)                    /*备注*/) ;

CREATE TABLE Part ( /*零件基本表*/
    partkey INTEGER PRIMARY KEY,            /*零件编号*/
    name VARCHAR(55),                       /*零件名称*/
    mfgr CHAR(25),                          /*制造厂*/
    brand CHAR(10),                         /*品牌*/
    type VARCHAR(25),                       /*零件类型*/
    size INTEGER,                           /*尺寸*/
    container CHAR(10),                     /*包装*/
    retailprice REAL,                       /*零售价格*/
```

```
        comment VARCHAR(23)                          /*备注*/);

    CREATE TABLE PartSupp ( /*零件供应联系表*/
        partkey INTEGER REFERENCES Part(partkey),    /*零件编号*/
        suppkey INTEGER REFERENCES Supplier(suppkey), /*供应商编号*/
        availqty INTEGER,                            /*可用数量*/
        supplycost REAL,                             /*供应价格*/
        comment VARCHAR(199),                        /*备注*/
        PRIMARY KEY(partkey,suppkey)                 /*定义主码,表级约束*/);

    CREATE TABLE Customer ( /*顾客表*/
        custkey INTEGER PRIMARY KEY,                 /*顾客编号*/
        name VARCHAR(25),                            /*姓名*/
        address VARCHAR(40),                         /*地址*/
        nationkey INTEGER REFERENCES Nation(nationkey), /*国籍编号*/
        phone CHAR(15),                              /*电话*/
        acctbal REAL,                                /*账户余额*/
        mktsegment CHAR(10),                         /*市场分区*/
        comment VARCHAR(117)                         /*备注*/);

    CREATE TABLE Orders ( /*订单表*/
        orderkey INTEGER PRIMARY KEY,                /*订单编号*/
        custkey INTEGER REFERENCES Customer(custkey), /*顾客编号*/
        orderstatus CHAR(1),                         /*订单状态*/
        totalprice REAL,                             /*订单总金额*/
        orderdate DATE,                              /*订单日期*/
        orderpriority CHAR(15),                      /*订单优先级别*/
        clerk CHAR(15),                              /*记账员*/
        shippriority INTEGER,                        /*运输优先级别*/
        comment VARCHAR(79)                          /*备注*/);
/*其中 totalprice = SUM(Lineitem.extendedprice×(1-Lineitem.discount)×(1+ Lineitem.tax)) */

    CREATE TABLE Lineitem ( /*订单明细表 Lineitem*/
        orderkey INTEGER REFERENCES Orders(orderkey), /*订单编号*/
        partkey INTEGER REFERENCES Part(partkey),    /*零件编号*/
        suppkey INTEGER REFERENCES Supplier(suppkey), /*供应商编号*/
        linenumber INTEGER,                          /*订单明细编号*/
        quantity REAL,                               /*数量*/
```

```
    extendedprice REAL,                    /* 订单明细价格 */
    discount REAL,                         /* 折扣[0.00, 1.00] */
    tax REAL,                              /* 税率[0.00, 0.08] */
    returnflag CHAR(1),                    /* 退货标记 */
    linestatus CHAR(1),                    /* 订单明细状态 */
    shipdate DATE,                         /* 装运日期 */
    commitdate DATE,                       /* 委托日期 */
    receiptdate DATE,                      /* 签收日期 */
    shipinstruct CHAR(25),                 /* 装运说明,如 deliver in person */
    shipmode CHAR(10),                     /* 装运方式,如空运、陆运、铁运和海运等 */
    comment VARCHAR(44),                   /* 备注 */
    PRIMARY KEY(orderkey,linenumber),
    FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey) ) ;
    /* 其中订单明细价格 extendedprice = quantity × Part.retailprice */
```

实验总结：

① 初学者可以先定义零件供应商子模式，包括 Part、Supplier 和 PartSupp 三个基本表，类似学生、课程和选课数据库模式。

② 初学者可以先不定义实体完整性和参照完整性，待讲完有关概念后再在实验 3 中练习。

5. 思考题

（1）SQL 语法规定，双引号括定的符号串为对象名称，单引号括定的符号串为常量字符串，那么什么情况下需要用双引号来界定对象名呢？请实验验证。

（2）数据库对象的完整引用是“服务器名.数据库名.模式名.对象名”，但通常可以省略服务器名和数据库名，甚至模式名，直接用对象名访问对象即可。请设计相应的实验验证基本表及其列的访问方法。

实验 1.2 数据基本查询实验

1. 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。

2. 实验内容和要求

针对 TPC-H 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。

说明：简单地说，SQL 程序设计规范包含 SQL 关键字大写、表名、属性名、存储过程名等标识符大小写混合、SQL 程序书写缩进排列等编程规范。

3. 实验重点和难点

实验重点：分组统计查询、单表自身连接查询、多表连接查询。

实验难点：区分元组过滤条件和分组过滤条件；确定连接属性，正确设计连接条件。

4. 实验报告示例

实 验 报 告				
题目：数据基本查询实验	姓名		日期	

（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;
```

（2）单表查询（实现选择操作）

查询最近一周内提交的总价大于 1 000 元的订单的编号、顾客编号等订单的所有信息。

```
SELECT *
FROM Sales.Orders      /* 模式名.表名，Sales 模式中的 Orders 表 */
WHERE CURRENT_DATE - orderdate < 7 AND totalprice > 1000;   /* 选择条件 */
```

（3）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)   /* 对每个组，作用聚集函数 SUM */
FROM Customer C, Orders O
WHERE C.custkey = O.custkey
GROUP BY C.custkey;   /* 按照 C.custkey 分组 */
```

（4）带分组过滤条件的分组统计查询

查询订单平均金额超过 1 000 元的顾客编号及其姓名。

```
SELECT C.custkey, MAX(C.name) /* 分组属性和聚集函数才能出现在 SELECT 子句 */
FROM Customer C, Orders O
WHERE C.custkey = O.custkey
GROUP BY C.custkey;   /* 按照 C.custkey 分组 */
HAVING AVG(O.totalprice) > 1000;   /* 按照条件对组进行过滤，只输出满足条件的组 */
```

(5) 单表自身连接查询

查询与"金仓集团"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   /* Supplier 表的自身连接 */
WHERE F.nationkey = S.nationkey AND S.name = '金仓集团';
```

(6) 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.retailoprice, PS.supplycost
FROM Part P, PartSupp PS   /* 两表连接,给表起个别名,简化表达 */
WHERE P.retailprice > PS.supplycost;   /* 限定条件 */
/* 说明:上述连接语句是从两个表的笛卡儿积中选出满足限定条件的元组,得到的结果可能
不是同一个商品的有关值,所以应改为下面的自然连接 */
```

(7) 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.retailoprice, PS.supplycost
FROM Part P, PartSupp PS   /* 两表连接,给表起个别名,简化表达 */
WHERE P.partkey = PS.partkey   /* 连接条件 */
        AND P.retailprice > PS.supplycost; /* 限定条件 */
```

(8) 三表连接查询

查询顾客"苏举库"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='苏举库';
```

实验总结:

① 正确理解数据库模式结构,才能正确设计数据库查询;

② 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。

5. 思考题

(1) 不在 GROUP BY 子句出现的属性,是否可以出现在 SELECT 子句中? 请举例并上机验证。

(2) 请举例说明分组统计查询中的 WHERE 和 HAVING 有何区别?

(3) 连接查询速度是影响关系数据库性能的关键因素。请讨论如何提高连接查询速度,

并进行实验验证。

实验 1.3　数据高级查询实验

1. 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。

2. 实验内容和要求

针对 TPC-H 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。

3. 实验重点和难点

实验重点：嵌套查询。

实验难点：相关子查询、多层 EXIST 嵌套查询。

4. 实验报告示例

实　验　报　告				
题目：数据高级查询实验	姓名		日期	

(1) IN 嵌套查询

查询订购了“海大”制造的“船舶模拟驾驶舱”的顾客。

```
SELECT   custkey, name
FROM   Customer
WHERE custkey IN (SELECT O.custkey
                  FROM Orders O, Lineitem L, PartSupp PS, Part P   /* 四表连接 */
                  WHERE   O.orderkey = L.orderkey AND
                          L.partkey = PS.partkey AND
                          L.suppkey = PS.suppkey AND
                          PS.partkey = P.partkey AND
                          P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
/* 试比较上述查询与下列查询有何区别？下列查询中 Lineitem 表直接与 Part 表连接。 */
/* 请见实验总结 */

SELECT   custkey, name
FROM   Customer
WHERE custkey IN (SELECT O.custkey
                  FROM Orders O, Lineitem L, Part P    /* 三表连接 */
                  WHERE   O.orderkey = L.orderkey AND
                          L.partkey = P.partkey AND
                          P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
```

(2) 单层 EXISTS 嵌套查询

查询没有购买过“海大”制造的“船舶模拟驾驶舱”的顾客。

```
SELECT  custkey, name
FROM  Customer C
WHERE NOT EXISTS(SELECT O.custkey /* 购买“海大”制造的“船舶模拟驾驶舱”的顾客 */
                 FROM  Orders O, Lineitem L, PartSupp PS, Part P
                 WHERE  C.custkey = O.custkey AND  /* 此条件为相关子查询条件 */
                        O.orderkey = L.orderkey AND
                        L.partkey = PS.partkey AND
                        L.suppkey = PS.suppkey AND
                        PS.partkey = P.partkey AND
                        P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
```

(3) 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  /* 查找 CA 客户,其不存在张三购买过而 CA 客户没有买过的零件 */
FROM Customer CA
WHERE NOT EXISTS
    (SELECT *    /* 张三购买过而 CA 客户没有买过的零件 */
     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 *   /* CA 客户与 CB 客户都购买过的零件 */
                      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 ) );
```

(4) 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      /* 子查询生成的临时派生表为 B 表 */
                 FROM  Orders
                 GROUP BY custkey
                 HAVING AVG(totalprice) > 10000) B, Nation N
```

```
WHERE   C.custkey = B.custkey AND /*B表成为主查询的查询对象*/
        C.nationkey = N.nationkey AND   N.name = '中国';
```

(5) 集合查询(交)

查询顾客“张三”和“李四”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 = '张三';
INTERSECTION
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 = '李四';
```

(6) 集合查询(并)

查询顾客“张三”和“李四”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 = '张三';
UNION
SELECT P.*
FROM   Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey ANDO.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND
        PS.partkey = P.partkey AND C.name = '李四';
```

(7) 集合查询(差)

顾客“张三”订购过而“李四”没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 = '张三';
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 = '李四';
```

实验总结：

通过分析图 2 中 TPC-H 数据库模式可知，Lineitem 表是通过 Partsupp 表跟 Part 表联系的，所以，“(1) IN 嵌套查询”中第一个查询是正常的查询表达方式。而由于 partkey 是 Part 的主码，第二个查询也能得出相同的结果。因此，生成 Lineitem 记录时利用 PartSupp 表保证供应商和零件的一致性，而查询 Lineitem 时可以直接和 Part 相连接。同样，也可以直接和 Suppliers 相连接。

5. 思考题

(1) 试分析什么类型的查询可以用连接查询实现，什么类型的查询用只能用嵌套查询实现？

(2) 试分析不相关子查询和相关子查询的区别。

实验 1.4 数据更新实验

1. 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。

2. 实验内容和要求

针对 TPC-H 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。

3. 实验重点和难点

实验重点：插入、修改和删除数据的 SQL。

实验难点：与嵌套 SQL 子查询相结合的插入、修改和删除数据的 SQL 语句；利用一个表的数据来插入、修改和删除另外一个表的数据。

4. 实验报告示例

实 验 报 告				
题目:数据更新实验	姓名		日期	

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

(2) 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(RANDOM() * 100,0), 479, 1, 10, '2012-3-6');
/* RANDOM()函数为随机小数生成函数,ROUND()为四舍五入函数 */
```

(3) 批量数据 INSERT 语句

① 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;
/* WITH NO DATA 子句使得 SELECT 查询只生成一个结果模式,不查询出实际数据 */
INSERT INTO NewCustomer  /* 批量插入 SELECT 语句查询结果到 NewCustomer 表中 */
SELECT C.*
FROM Customer C, Nation N
WHERE C.nationkey = N.nationkey AND N.name = '中国';
```

② 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   INTEGER ,
    quantity REAL,
    totalprice   REAL);
INSERT INTO ShoppingStat
SELECT C.custkey,Sum(L.quantity), Sum(O.totalprice)   /* 对分组后的数据求总和 */
FROM Customer C, Orders O, Lineitem L
WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey
GROUP BY C.custkey
```

③ 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, retailprice, comment
    FROM Part;
```

说明：

① 该方法是迅速生成大量数据的一个有效方法，但其缺点是要求主码为数值类型，并且除了主码外其他属性上没有 Unique 约束，因此该方法生成的数据具有很大的重复性。

② 生成 50 万记录需要执行上述 INSERT 语句的次数，需要根据 Part 表的初始记录数人工计算。

③ 执行上述 INSERT 语句的速度会一次比一次慢，因此需要耐心等候。

(4) UPDATE 语句(修改部分记录的部分列值)

"金仓集团"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        /*找出要修改的那些记录*/
                 FROM Supplier
                 WHERE name = '金仓集团');
```

(5) UPDATE 语句(利用一个表中的数据修改另外一个表中的数据)

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 L
SET L.extendedprice = P.retailprice * L.quantity
FROM Part P
WHERE L.partkey = P.partkey
/*Lineitem 表也可以直接与 Part 表相连接，而不需通过 PartSupp 连接*/
```

(6) DELETE 基本语句(删除给定条件的所有记录)

删除顾客张三的所有订单记录。

```
DELETE FROM Lineitem    /*先删除张三的订单明细记录*/
WHERE orderkey IN (SELECT orderkey
                   FROM Orders O, Customer C
                   WHERE O.custkey = C.custkey AND C.name ='张三');
DELETE FROM  Orders   /*再删除张三的订单记录*/
WHERE custkey = (SELECT custkey
                 FROM Customer
                 WHERE name ='张三');
```

实验总结：

① 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。

② 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。

5. 思考题

（1）请分析数据库模式更新和数据更新 SQL 语句的异同。

（2）请分析数据库系统除了 INSERT、UPDATE 和 DELETE 等基本的数据更新语句之外，还有哪些可以用来更新数据库基本表数据的 SQL 语句？

实验 1.5　视图实验

1. 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。

2. 实验内容和要求

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。

3. 实验重点和难点

实验重点：创建视图。

实验难点：可更新的视图和不可更新的视图之区别，WITH CHECK OPTION 的验证。

4. 实验报告示例

实 验 报 告				
题目：视图实验	姓名		日期	

（1）创建视图（省略视图列名）

创建一个“海大汽配”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、供应价格和备注等信息。

```
CREATE VIEW V_DLMU_PARTSUPP1 AS /* 由 SELECT 子句目标列组成视图属性 */
SELECT P.partkey,P.name,PS.availqty,P.retailprice,PS.supplycost,P.comment
FROM   Part P,PartSupp PS,Supplier S
WHERE P.partkey=PS.partkey AND S.suppkey=PS.suppkey AND S.name='海大汽配';
```

(2) 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;
```

(3) 创建视图(WITH CHECK OPTION)

使用 WITH CHECK OPTION,创建一个"海大汽配"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"海大汽配"零件供应记录,验证 WITH CHECK OPTION 是否起作用。

```
CREATE VIEW V_DLMU_PartSupp2
AS
SELECT partkey, suppkey, availqty, supplycost
FROM PartSupp
WHERE suppkey = (SELECT suppkey
                 FROM Supplier
                 WHERE name = '海大汽配')
WITH CHECK OPTION;

INSERT INTO V_DLMU_PartSupp2
VALUES(58889,5048,704,77760);

UPDATE V_DLMU_PartSupp2
SET supplycost = 12
WHERE suppkey = 58889;

DELETE FROM V_DLMU_PartSupp2
WHERE suppkey = 58889;
```

(4) 可更新的视图(行列子集视图)

创建一个"海大汽配"供应商供应的零件视图 V_DLMU_PartSupp4,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"海大汽配"零件供应记录,验证该视图是否是可更新的,并比较上述"(3) 创建视图"实验任务与本任务结果有何异同。

```
CREATE VIEW V_DLMU_PartSupp3
AS
SELECT partkey, suppkey, availqty, supplycost
FROM PartSupp
WHERE suppkey = (SELECT suppkey
FROM Supplier
WHERE name = '海大汽配');

INSERT INTO V_DLMU_PartSupp3
VALUES(58889,5048,704,77760);

UPDATE V_DLMU_PartSupp3
SET supplycost = 12
WHERE suppkey = 58889;

DELETE FROM V_DLMU_PartSupp3
WHERE suppkey = 58889;
```

(5) 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);
```

(6) 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 选项删除视图 V_CustOrd，观察现象并检查 V_CustAvgOrder 是否存在，解释原因？

```
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;

CREATE VIEW V_CustAvgOrder(custkey,cname, avgqty, avgprice)
AS
SELECT custkey,MAX(cname),AVG(qty),AVG(extprice)
FROM V_CustOrd    /*在视图 V_CustOrd 上再创建视图*/
GROUP BY custkey;
```

```
DROP VIEW V_CustOrd RESTRICT;
DROP VIEW V_CustOrd CASCADE;
```

实验总结：

5. 思考题

（1）请分析视图和基本表在使用方面有哪些异同，并设计相应的例子加以验证。

（2）请具体分析修改基本表的结构对相应的视图会产生何种影响？

实验 1.6 索引实验

1. 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。

2. 实验内容和要求

针对给定的数据库模式和具体应用需求，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。

3. 实验重点和难点

实验重点：创建索引。

实验难点：设计 SQL 查询验证索引有效性。

4. 实验报告示例

实 验 报 告				
题目：索引实验	姓名		日期	

（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);
```

（2）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称长度的函数索引。

```
CREATE INDEX Idx_part_name_fun ON Part (LENGTH(name));
```

(3) 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 UNIQUE INDEX Idx_part_mfgr_brand ON Part (mfgr, brand);
```

(4) * 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。

```
CREATE UNIQUE INDEX Idx_part_mfgr ON Part (mfgr);
CLUSTER Idx_part_mfgr ON Part;
/* 有关聚簇索引的概念在《概论》第 7 章 7.5.2 小节"关系模式存取方法选择"中介绍 */
```

(5) 创建 Hash 索引

在零件表的名称字段上创建一个 Hash 索引。

```
CREATE INDEX Idx_part_name_hash ON Part USING HASH (name);
```

(6) 修改索引名称

修改零件表的名称字段上的索引名。

```
ALTER INDEX Idx_part_name_hash RENAME TO Idx_part_name_hash_new;
```

(7) 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';
```

(8) * 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)) RETURN INTEGER
AS  /* 自定义函数 TestIndex():输入参数为零件名称,返回 SQL 查询的执行时间 */
DECLARE
    begintime   TIMESTAMP;
    endtime   TIMESTAMP;
    durationtime INTEGER;
BEGIN
    SELECT CLOCK_TIMESTAMP() INTO begintime; /* 记录查询执行的开始时间 */
    PERFORM * FROM Part WHERE name =p_partname;
                                        /* 执行 SQL 查询,不保存查询结果 */
    SELECT CLOCK_TIMESTAMP() INTO endtime; /* 记录查询执行的结束时间 */
    SELECT DATEDIFF('ms', begintime , endtime) INTO durationtime;
    RETURN durationtime;          /* 计算并返回查询执行时间,时间单位为毫秒 ms */
END;

 /* 查看当零件表 Part 数据规模比较小,并且无索引时的执行时间 */
 SELECT TestIndex('零件名称');
```

```
INSERT INTO Part  /* 不断倍增零件表的数据,直到 50 万条记录 */
SELECT partkey + (SELECT COUNT( * ) FROM Part),
    name, mfgr, brand, type, size, container, retailprice, comment
FROM Part;

/* 查看当零件表 Part 数据规模比较大,但无索引时的执行时间 */
SELECT TestIndex('零件名称');

CREATE INDEX part_name ON Part(name); /* 在零件表的零件名称字段上创建索引 */

/* 查看零件表 Part 数据规模比较大,有索引时的执行时间 */
SELECT TestIndex( );
```

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。

实验总结:

5. 思考题

在一个表的多个字段上创建的复合索引,与在相应的每个字段上创建的多个简单索引有何异同? 请设计相应的例子加以验证。

实验 2 安全性语言实验

安全性实验包含两个实验项目(参见表 5),其中 1 个为必修,1 个为选修。自主存取控制实验为设计性实验项目,审计实验为验证性实验项目。本节目前没有设置强制存取控制和数据加密等方面的实验项目。

表 5 "安全性语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 2.1	自主存取控制实验	必修	适中	2	第 4 章 4.2
实验 2.2	审计实验	选修	适中	2	第 4 章 4.4

实验 2.1　自主存取控制实验

1. 实验目的

掌握自主存取控制权限的定义和维护方法。

2. 实验内容和要求

定义用户、角色，分配权限给用户、角色，回收权限，以相应的用户名登录数据库验证权限分配是否正确。选择一个应用场景，使用自主存取控制机制设计权限分配。可以采用两种方案。

方案一：采用 SYSTEM 超级用户登录数据库，完成所有权限分配工作，然后用相应用户名登录数据库以验证权限分配正确性；

方案二：采用 SYSTEM 用户登录数据库创建三个部门经理用户，并分配相应的权限，然后分别用三个经理用户名登录数据库，创建相应部门的 USER、ROLE，并分配相应权限。

下面的实验报告示例采用了实验方案一。验证权限分配之前，请备份好数据库；针对不同用户所具有的权限，分别设计相应的 SQL 语句加以验证。

3. 实验重点和难点

实验重点：定义角色，分配权限和回收权限。

实验难点：实验方案二实现权限的再分配和回收。

4. 实验报告示例

实 验 报 告				
题目：自主存取控制实验	姓名		日期	

设有一个企业，包括采购、销售和客户管理等三个部门，采购部门经理 David，采购员 Jeffery；销售部门经理 Tom，销售员 Jane；客户管理部门经理 Kathy，职员 Mike。该企业一个信息系统覆盖采购、销售和客户管理等三个部门的业务，其数据库模式为 TPC-H 数据模式。针对此应用场景，使用自主存取控制机制设计一个具体的权限分配方案。

（1）创建用户

① 为采购、销售和客户管理等三个部门的经理创建用户标识，要求具有创建用户或角色的权利。

```
CREATE USER David WITH CREATEROLE PASSWORD '123456';
CREATE USER Tom WITH CREATEROLE PASSWORD '123456';
CREATE USER Kathy WITH CREATEROLE PASSWORD '123456';
/*注意：CREATE USER 语句不是 SQL 标准，因此不同的 RDBMS 语法和内容相差甚远。
这里采用的是 Kingbase 创建用户语句*/
```

② 为采购、销售和客户管理等三个部门的职员创建用户标识和用户口令。

```
CREATE USER Jeffery WITH PASSWORD '123456';
CREATE USER Jane WITH PASSWORD '123456';
CREATE USER Mike WITH PASSWORD '123456';
```

(2) 创建角色并分配权限

① 为各个部门分别创建一个查询角色,并分配相应的查询权限。

```
CREATE ROLE PurchaseQueryRole;
GRANT SELECT ON TABLE Part TO PurchaseQueryRole;
GRANT SELECT ON TABLE Supplier TO PurchaseQueryRole;
GRANT SELECT ON TABLE PartSupp TO PurchaseQueryRole;

CREATE ROLE SaleQueryRole;
GRANT SELECT ON TABLE Order TO SaleQueryRole;
GRANT SELECT ON TABLE LineItem TO SaleQueryRole;

CREATE ROLE CustomerQueryRole;
GRANT SELECT ON TABLE Customer TO CustomerQueryRole;
GRANT SELECT ON TABLE Nation TO CustomerQueryRole;
GRANT SELECT ON TABLE Region TO CustomerQueryRole;
```

② 为各个部门分别创建一个职员角色,对本部门信息具有查看、插入权限。

```
CREATE ROLE PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE Part TO PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE Supplier TO PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE PartSupp TO PurchaseEmployeeRole;

CREATE ROLE SaleEmployeeRole;
GRANT SELECT,INSERT ON TABLE Order TO SaleEmployeeRole;
GRANT SELECT,INSERT ON TABLE LineItem TO SaleEmployeeRole;

CREATE ROLE CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Customer TO CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Nation TO CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Region TO CustomerEmployeeRole;
```

③ 为各部门创建一个经理角色,相应角色对本部门的信息具有完全控制权限,对其他部门的信息具有查询权。经理有权给本部门职员分配权限。

```
CREATE ROLE PurchaseManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Part TO PurchaseManagerRole;
```

```
GRANT ALL ON TABLE Supplier TO PurchaseManagerRole;
GRANT ALL ON TABLE PartSupp TO PurchaseManagerRole;
GRANT SaleQueryRole TO PurchaseManagerRole;
GRANT CustomerQueryRole TO PurchaseManagerRole;

CREATE ROLE SaleManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Order TO SaleManagerRole;
GRANT ALL ON TABLE LineItem TO SaleManagerRole;
GRANT PurchaseQueryRole TO SaleManagerRole;
GRANT CustomerQueryRole TO SaleManagerRole;

CREATE ROLE CustomerManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Customer TO CustomerManagerRole;
GRANT ALL ON TABLE Nation TO CustomerManagerRole;
GRANT ALL ON TABLE Region TO CustomerManagerRole;
GRANT PurchaseQueryRole TO CustomerManagerRole;
GRANT SaleQueryRole TO CustomerManagerRole;
```

(3) 给用户分配权限

① 给各部门经理分配权限。

```
GRANT PurchaseManagerRole TO David WITH ADMIN OPTION;
GRANT SaleManagerRole TO Tom WITH ADMIN OPTION;
GRANT CustomerManagerRole TO Kathy WITH ADMIN OPTION;
```

② 给各部门职员分配权限。

```
GRANT PurchaseEmployeeRole TO Jeffery;
GRANT SaleEmployeeRole TO Jane;
GRANT CustomerEmployeeRole TO Mike;
```

(4) 回收角色或用户权限

① 收回客户经理角色的销售信息查看权限。

```
REVOKE SaleQueryRole FROM CustomerManagerRole;
```

② 回收 MIKE 的客户部门职员权限。

```
REVOKE CustomerEmployeeRole FROM Mike;
```

(5) 验证权限分配正确性

① 以 David 用户名登录数据库,验证采购部门经理的权限

```
SELECT * FROM Part;
DELETE * FROM Order;
```

② 回收 MIKE 的客户部门职员权限

```
SELECT * FROM Customer;
SELECT * FROM Part;
```

实验总结：

在进行权限分配之后，针对不同用户所具有的权限，设计并执行若干 SQL 语句，验证权限分配是否有效。

5. 思考题

（1）请分析 WITH CHECK OPTION、WITH GRANT OPTION 和 WITH ADMIN OPTION 有何区别与联系。

（2）请结合上述实验示例分析使用角色进行权限分配有何优缺点。

实验 2.2　审计实验

1. 实验目的

掌握数据库审计的设置和管理方法，以便监控数据库操作，维护数据库安全。

2. 实验内容和要求

打开数据库审计开关。以具有审计权限的用户登录数据库，设置审计权限，然后以普通用户登录数据库，执行相应的数据操纵 SQL 语句，验证相应审计设置是否生效，最后再以具有审计权限的用户登录数据库，查看是否存在相应的审计信息。

3. 实验重点和难点

实验重点：数据库对象级审计，数据库语句级审计。

实验难点：合理地设置各种审计信息。一方面，为了保护系统重要的敏感数据，需要系统地设置各种审计信息，不能留有漏洞，以便随时监督系统使用情况，一旦出现问题，也便于追查；另一方面，审计信息设置过多，会严重影响数据库的使用性能，因此需要合理设置。

4. 实验报告示例

实 验 报 告				
题目：审计实验	姓名		日期	

（1）审计开关

① 显示当前审计开关状态。

```
SHOW AUDIT_TRAIL;
```

② 打开审计开关。

```
SET AUDIT_TRAIL TO ON;
```

(2) 数据库操作审计

① 对客户信息表上的删除操作设置审计。

```
AUDIT DELETE ON Sales.Customer BY ACCESS;
/* BY ACCESS 审计方式表示系统对每个设置的审计操作都要进行记录;BY SESSION 审计方式表示对于每次会话中涉及的同类审计操作,系统只记录最早的一次。 */
```

② 以普通用户登录,执行 SQL 语句。

```
DELETE Sales.Customer WHERE custkey = 1011;
```

③ 查看数据库对象审计信息,验证审计设置是否生效。

```
SELECT * FROM SYS_AUDIT_OBJECT;
```

(3) 语句级审计

① 对表定义的更改语句 ALTER 设置审计。

```
AUDIT ALTER TABLE BY ACCESS;
```

② 查看数据库所有语句级审计设置,验证审计设置是否生效。

```
SELECT * FROM SYS_STMT_AUDIT_OPTS;
```

③ 以普通用户登录,执行 SQL 语句,验证审计设置是否生效。

```
ALTER TABLE Customer ADD COLUMN tt INT;
```

④ 查看所有审计信息。

```
SELECT * FROM SYS_AUDIT_TRAIL;
```

实验总结:

① 在 KingbaseES 系统中,只有审计管理员(SAO)才能进行对象级、语句级、系统权限级等审计权限设置。通过系统视图 SYS_STMT_AUDIT_OPTS 查看所有数据库中语句级审计设置,通过系统视图 SYS_AUDIT_OBJECT 查看数据库对象的审计设置。

② 通过系统视图 SYS_AUDIT_STATEMENT 可以查询所有关于 GRANT、REVOKE、AUDIT、NOAUDIT, 以及 ALTER SYSTEM 语句的审计结果;通过系统视图 SYS_AUDIT_OBJECT 可以查询数据库中所有对象的审计结果;通过系统视图 SYS_AUDIT_TRAIL 可以查看所有审计记录。

③ 审计语句不是标准 SQL 语句,所以不同的系统语句格式和语法不尽相同。

5. 思考题

试着设计一个例子,分析数据库审计对数据库性能的影响情况。

实验 3 完整性语言实验

完整性语言实验包含三个实验项目(参见表 6),其中 2 个必修项目,1 个选修项目。该实验的各个实验项目均为验证性实验项目。

表 6 "完整性语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 3.1	实体完整性实验	必修	适中	2	第 5 章 5.1
实验 3.2	参照完整性实验	必修	适中	2	第 5 章 5.2
实验 3.3	用户自定义完整性试验	选修	适中	2	第 5 章 5.3

实验 3.1 实体完整性实验

1. 实验目的

掌握实体完整性的定义和维护方法。

2. 实验内容和要求

定义实体完整性,删除实体完整性。能够写出两种方式定义实体完整性的 SQL 语句:创建表时定义实体完整性、创建表后定义实体完整性。设计 SQL 语句验证完整性约束是否起作用。

3. 实验重点和难点

实验重点:创建表时定义实体完整性。

实验难点:有多个候选码时实体完整性的定义。

4. 实验报告示例

实 验 报 告				
题目:实体完整性实验	姓名		日期	

(1) 创建表时定义实体完整性(列级实体完整性)

定义供应商表的实体完整性。

```
CREATE TABLE Supplier(
    suppkey INTEGER CONSTRAINT PK_supplier PRIMARY KEY,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
```

```
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101) );
```

(2) 创建表时定义实体完整性(表级实体完整性)

定义供应商表的实体完整性。

```
CREATE TABLE Supplier (
    suppkey INTEGER,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101),
    CONSTRAINT PK_supplier PRIMARY KEY(suppkey));
```

(3) 创建表后定义实体完整性

定义供应商表。

```
CREATE TABLE Supplier (
    suppkey INTEGER,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101) );

ALTER TABLE Supplier     /*再修改供应商表,增加实体完整性*/
ADD CONSTRAINT PK_Supplier PRIMARY KEY(suppkey);
```

(4) 定义实体完整性(主码由多个属性组成)

定义供应关系表的实体完整性。

```
CREATE TABLE PartSupp (
    partkey INTEGER,
    suppkey INTEGER,
    availqty INTEGER,
    supplycost REAL,
    comment VARCHAR(199),
    PRIMARY KEY(partkey,suppkey)) ;
```

```
    /* 主码由多个属性组成,实体完整性必须定义在表级 */
```

(5) 有多个候选码时定义实体完整性

定义国家表的实体完整性,其中 nationkey 和 name 都是候选码,选择 nationkey 作为主码,name 上定义唯一性约束。

```
CREATE TABLE nation (
    nationkey INTEGERCONSTRAINT PK_nation PRIMARY KEY,
    name CHAR(25)UNIQUE,
    regionkey INTEGER,
    comment VARCHAR(152) );
```

(6) 删除实体完整性

删除国家实体的主码。

```
ALTER TABLE nation DROP CONSTRAINT PK_nation;
```

(7) 增加两条相同记录,验证实体完整性是否起作用

```
/* 插入两条主码相同的记录就会违反实体完整性约束 */
INSERT INTO Supplier (suppkey,name,address,nationkey,phone,acctbal,comment )
        VALUES (11,'test1','test1',101,'12345678',0.0,'test1');
INSERT INTO Supplier (suppkey,name,address,nationkey,phone,acctbal,comment )
        VALUES (11,'test2','test2',102,'23456789',0.0,'test2');
```

实验总结:

5. 思考题

(1) 所有列级完整性约束都可以改写为表级完整性约束,而表级完整性约束不一定能改写成列级完整性约束。请举例说明。

(2) 什么情况下会违反实体完整性约束,DBMS 将做何种违约处理? 请用实验验证。

实验 3.2 参照完整性实验

1. 实验目的

掌握参照完整性的定义和维护方法。

2. 实验内容和要求

定义参照完整性,定义参照完整性的违约处理,删除参照完整性。写出两种方式定义参照完整性的 SQL 语句:创建表时定义参照完整性、创建表后定义参照完整性。

3. 实验重点和难点

实验重点：创建表时定义参照完整性。

实验难点：参照完整性的违约处理定义。

4. 实验报告示例

实 验 报 告				
题目：参照完整性实验	姓名		日期	

（1）创建表时定义参照完整性

先定义地区表的实体完整性，再定义国家表上的参照完整性。

```
CREATE TABLE region(
  regionkey INTEGER PRIMARY KEY,
  name CHAR(25),
  comment VARCHAR(152));

CREATE TABLE nation(
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER REFERENCES Region(regionkey),   /*列级参照完整性*/
  comment VARCHAR(152));
```

或者：

```
CREATE TABLE nation(
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER,
  comment VARCHAR(152),
  CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(region-
  key));      /*表级参照完整性*/
```

（2）创建表后定义参照完整性

定义国家表的参照完整性。

```
CREATE TABLE nation (
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER,
  comment VARCHAR(152));

ALTER TABLE Nation
```

```
    ADD CONSTRAINT FK_Nation_regionkey
    FOREIGN KEY(regionkey) REFERENCES Region(regionkey );
```

(3) 定义参照完整性(外码由多个属性组成)

定义订单项目表的参照完整性。

```
CREATE TABLE PartSupp (
  partkey INTEGER,
  suppkey INTEGER,
  availqty INTEGER,
  supplycost REAL,
  comment VARCHAR(199),
  PRIMARY KEY(partkey,suppkey)) ;

CREATE TABLE Lineitem (
  orderkey INTEGER REFERENCES Orders(orderkey),
  partkey INTEGER REFERENCES Part(partkey),
  suppkey INTEGER REFERENCES Supplier(suppkey),
  linenumber INTEGER,
  quantity REAL,
  extendedprice REAL,
  discount REAL,
  tax REAL,
  returnflag CHAR(1),
  linestatus CHAR(1),
  shipdate DATE,
  commitdate DATE,
  receiptdate DATE,
  shipinstruct CHAR(25),
  shipmode CHAR(10),
  comment VARCHAR(44),
  PRIMARY KEY(orderkey,linenumber),
  FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey));
```

(4) 定义参照完整性的违约处理

定义国家表的参照完整性，当删除或修改被参照表记录时,设置参照表中相应记录的值为空值。

```
CREATE TABLE nation (
  nationkey INTEGER PRIMARY KEY,
```

```
    name CHAR(25),
    regionkey INTEGER,
    comment VARCHAR(152),
    CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(regionkey)
    ON DELETE SET NULL ON UPDATE SET NULL );
```

(5) 删除参照完整性

删除国家表的外码

```
ALTER TABLE nation DROP CONSTRAINT FK_Nation_regionkey;
```

(6) 插入一条国家记录,验证参照完整性是否起作用

```
/* 插入一条国家记录,如果'1001'号地区记录不存在,违反参照完整性约束。 */
INSERT INTONation(nationkey,name,regionkey,comment )
VALUES (1001,'nation1',1001,'comment1');
```

实验总结:

5. 思考题

对于自引用表,例如课程表(课程号、课程名、先修课程号,学分)中的先修课程号引用该表的课程号,请完成如下任务:

(1) 写出课程表上的实体完整性和参照完整性。

(2) 在考虑实体完整性约束的情况下,试举出几种录入课程数据的方法。

实验 3.3　用户自定义完整性实验

1. 实验目的

掌握用户自定义完整性的定义和维护方法。

2. 实验内容和要求

针对具体应用语义,选择 NULL/NOT NULL、DEFAULT、UNIQUE、CHECK 等,定义属性上的约束条件。

3. 实验重点和难点

实验重点:NULL/NOT NULL, DEFAULT。

实验难点:CHECK。

4. 实验报告示例

实 验 报 告				
题目:自定义完整性实验	姓名		日期	

(1) 定义属性 NULL/NOT NULL 约束

定义地区表各属性的 NULL/NOT NULL 属性。

```
CREATE TABLE region (
    regionkey INTEGER NOT NULL PRIMARY KEY,
    name CHAR(25) NOT NULL,
    comment VARCHAR(152)NULL ) ;
```

(2) 定义属性 DEFAULT 约束

定义国家表的 regionkey 的缺省属性值为 0 值,表示其他地区。

```
CREATE TABLE nation (
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25),
    regionkey INTEGER DEFAULT 0,
    comment VARCHAR(152),
    CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(re-
    gionkey));
```

(3) 定义属性 UNIQUE 约束

定义国家表的名称属性必须唯一的完整性约束。

```
CREATE TABLE nation (
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25) UNIQUE,
    regionkey INTEGER,
    comment VARCHAR(152));
```

(4) 使用 CHECK

使用 CHECK 定义订单项目表中某些属性应该满足的约束。

```
CREATE TABLE Lineitem (
    orderkey INTEGER REFERENCES Orders(orderkey),
    partkey INTEGER REFERENCES Part(partkey),
    suppkey INTEGER REFERENCES Supplier(suppkey),
    linenumber INTEGER,
    quantity REAL,
    extendedprice REAL,
```

```
    discount REAL,
    tax REAL,
    returnflag CHAR(1),
    linestatus CHAR(1),
    shipdate DATE,
    commitdate DATE,
    receiptdate DATE,
    shipinstruct CHAR(25),
    shipmode CHAR(10),
    comment VARCHAR(44),
    PRIMARY KEY(orderkey,linenumber),
    FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey),
    CHECK(shipdate < receiptdate ),          /*装运日期<签收日期*/
    CHECK(returnflag IN ('A','R','N') ) );    /*退货标记为A或R或N*/
```

(5) 修改 Lineitem 的一条记录验证是否违反 CHECK 约束

```
UPDATE sales.Lineitem SET shipdate = '2015-01-05', receiptdate = '2015-01-01'
WHERE orderkey = 5005 AND linenumber = 1;
```

实验总结：

5. 思考题

(1) 请分析哪些完整性约束只针对单个属性，哪些完整性约束可以针对多个属性？哪些只针对一个表，哪些针对多个表？

(2) 对表中某一列数据类型进行修改时，要修改的列是否必须为空列？

实验 4　触发器实验

触发器实验只有一个必修实验项目(参见表 7)，该实验项目为设计型实验项目。

表 7　"触发器"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
触发器实验	必修	较难	2	第 5 章 5.7

1. 实验目的

掌握数据库触发器的设计和使用方法。

2. 实验内容和要求

定义 BEFORE 触发器和 AFTER 触发器。能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。

3. 实验重点和难点

实验重点：触发器的定义。

实验难点：利用触发器实现较为复杂的用户自定义完整性。

4. 实验报告示例

实 验 报 告				
题目：触发器实验	姓名		日期	

(1) AFTER 触发器

① 在 Lineitem 表上定义一个UPDATE 触发器，当修改订单明细(即修改订单明细价格 extendedprice、折扣 discount、税率 tax)时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。

```
total price = totalprice+extendedprice * (1-discount) * (1+tax) )
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_UPDATE
AFTER UPDATE OF extendedprice,discount,tax ON Lineitem
  FOR EACH ROW
  AS
  DECLARE
    L_valuediff REAL;
BEGIN
  /* 订单明细修改后,计算订单含税折扣价总价的修正值 */
  L_valuediff = NEW.extendedprice * (1-NEW.discount) * (1+NEW.tax) -
              OLD.extendedprice * (1-OLD.discount) * (1+OLD.tax);
              /* 更新订单的含税折扣价总价 */
  UPDATE Orders SET totalprice = totalprice + L_valuediff
  WHERE orderkey = NEW.orderkey;
END;
```

② 在 Lineitem 表上定义一个INSERT 触发器，当增加一项订单明细时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_INSERT
```

```
AFTER INSERT ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
      L_valuediff REAL;
BEGIN
  L_valuediff = NEW.extendedprice * (1 - NEW.discount) * (1 + NEW.tax);
                /* 增加订单明细项后,计算订单含税折扣价总价的修正值 */
  UPDATE Orders SET totalprice = totalprice + L_valuediff
                /* 更新订单的含税折扣价总价 */
  WHERE orderkey = NEW.orderkey;
END;
```

③ 在 Lineitem 表上定义一个DELETE 触发器,当删除一项订单明细时,自动修改订单 Orders 的 TotalPrice,以保持数据一致性。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_DELETE
AFTER DELETE ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
      L_valuediff REAL;
BEGIN
      L_valuediff = - OLD.extendedprice * (1 - OLD.discount) * (1 + OLD.tax);
                /* 删除订单明细项后,计算订单含税折扣价总价的修正值 */
      UPDATE Orders SET totalprice = totalprice + L_valuediff
                /* 更新订单的含税折扣价总价 */
WHERE orderkey = NEW.orderkey;
END;
```

④ 验证触发器 TRI_Lineitem_Price_UPDATE。

```
/* 查看 1854 号订单的含税折扣总价 totalprice */
SELECT totalprice
FROM Orders
WHERE orderkey = 1854;
/* 激活触发器:修改 1854 号订单第一个明细项的税率,该税率增加 0.5% */
UPDATE Lineitem SET tax = tax + 0.005
WHERE orderkey = 1854 AND linenumber = 1;
```

```
/* 再次查看 1854 号订单的含税折扣总价 totalprice 是否有变化,如有变化,则是触发器起作用了,否则触发器没有起作用 */
SELECT totalprice
FROM Orders
WHERE orderkey = 1854;
```

(2) BEFOR 触发器

① 在 Lineitem 表上定义一个BEFORE UPDATE 触发器,当修改订单明细中的数量(quantity)时,先检查供应表 PartSupp 中的可用数量 availqty 是否足够。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE UPDATE OF quantity ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
        L_valuediff  INTEGER;
        L_availqty  INTEGER;
BEGIN
        /* 计算订单明细项修改时,订购数量的变化值 */
        L_valuediff = NEW.quantity - OLD.quantity;
        /* 查询当前订单明细项对应零件供应记录中的可用数量 */
        SELECT availqty INTO L_availqty
        FROM PartSupp
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;

    IF (L_availqty - L_valuediff  >= 0) THEN
          BEGIN
            /* 如果可用数量可以满足订单订购数量,则提示 ENOUGH */
            RAISE NOTICE 'Available quantity is ENOUGH';
            /* 修改当前订单明细项对应零件供应记录中的可用数量 */
            UPDATE PartSupp
            SET availqty = availqty - L_valuediff
            WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
          END;
    ELSE
        /* 如果可用数量不能满足订单订购数量,则更新过程异常中断 */
        RAISE EXCEPTION 'Available quantity is NOT ENOUGH';
    END IF;
```

```
END;
```

② 在 Lineitem 表上定义一个BEFORE INSERT 触发器，当插入订单明细，先检查供应表 PartSupp 中的可用数量 availqty 是否足够。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE INSERT ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
        L_valuediff   INTEGER;
        L_availqty   INTEGER;
BEGIN
        L_valuediff = NEW.quantity;      /*获得插入订单明细项的订购数量*/
        /*查询当前订单明细项对应零件供应记录中的可用数量*/
        SELECT availqty INTO L_availqty
        FROM PartSupp
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;

        IF (L_availqty - L_valuediff  >= 0) THEN
              BEGIN
                    /*如果可用数量可以满足订单订购数量,则提示 ENOUGH*/
                    RAISE NOTICE 'Available quantity is ENOUGH';
                    /*修改当前订单明细项对应零件供应记录中的可用数量*/
                    UPDATE PartSupp
                    SET availqty = availqty - L_valuediff
                    WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
              END;
        ELSE
              /*如果可用数量不能满足订单订购数量,则插入过程异常中断。*/
              RAISE EXCEPTION 'Available quantity is NOT ENOUGH';
        END IF;
END;
```

③ 在 Lineitem 表上定义一个 BEFORE DELETE 触发器，当删除订单明细时，该订单明细项订购的数量要归还对应的零件供应记录。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE DELETE ON Lineitem
    FOR EACH ROW
```

```
    AS
    DECLARE
        L_valuediff  INTEGER;
        L_availqty   INTEGER;
BEGIN
        /* 获得删除订单明细项的订购数量 */
        L_valuediff = -OLD.quantity;
        /* 修改当前订单明细项对应零件供应记录中的可用数量 */
        UPDATE PartSupp
        SET availqty = availqty - L_valuediff
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
END;
```

④ 验证触发器 TRI_Lineitem_Quantity_UPDATE。

```
/* 查看 1854 号订单第 1 个明细项的零件和供应商编号、订购数量、可用数量 */
SELECT L.partkey, L.suppkey, L.quantity, PS.availqty
FROM Lineitem L, PartSupp PS
WHERE L.partkey = PS.partkey AND L.suppkey = PS.suppkey AND
      L.orderkey = 1854 AND L.linenumber = 1;
/* 激活触发器:修改 1854 号订单第 1 个明细项的订购数量 */
UPDATE Lineitem SET  quantity = quantity + 5
WHERE orderkey = 1854  AND linenumber = 1;

/* 再次查看 1854 号订单第 1 个明细项的相关信息,以验证触发器是否起作用 */
SELECT L.partkey, L.suppkey, L.quantity, PS.availqty
FROM Lineitem L, PartSupp PS
WHERE L.partkey = PS.partkey AND L.suppkey = PS.suppkey AND
      L.orderkey = 1854 AND L.linenumber = 1;
```

(3) 删除触发器

删除触发器 TRI_Lineitem_Price_UPDATE。

```
DROP TRIGGER TRI_Lineitem_Price_UPDATE ON Lineitem;
```

实验总结:

5. 思考题

试设计一个 AFTER 触发器，当 Lineitem 表中的 quantity 变化时，自动计算 Lineitem 表中的 extendedprice 值，同时也要修改 PartSupp 中的 availqty 值（提示：extendedprice = quantity * Part.retailprice）。

实验 5　数据库设计实验

数据库设计和应用开发实验是一个大的实验项目，共包括 4 个实验，即实验 5、实验 6、实验 7 和实验 8。实验 5 只有一个必修实验项目（参见表 8），针对第 7 章数据库设计中数据库概念设计、逻辑设计和物理设计等内容进行实验。实验 6 和实验 7 针对第 8 章内容，实验 8 则是一个综合性的数据库设计与应用开发大作业。

表 8　“数据库设计”实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
数据库设计	必修	适中	4	第 7 章

1. 实验目的

掌握数据库设计基本方法及数据库设计工具。

2. 实验内容和要求

掌握数据库设计基本步骤，包括数据库概念结构设计、逻辑结构设计，物理结构设计，数据库模式 SQL 语句生成。能够使用数据库设计工具进行数据库设计。

3. 实验重点和难点

实验重点：概念结构设计、逻辑结构设计。

实验难点：逻辑结构设计。逻辑结构设计虽然可以按照一定的规则从概念结构转换而来，但是由于概念结构通常比较抽象，较少考虑更多细节，因此转换而成的逻辑结构还需要进一步调整和优化。逻辑结构承接概念结构和物理结构，处于核心地位，因而是数据库设计的重点，也是难点。

4. 实验报告示例

实　验　报　告				
题目：数据库设计实验	姓名		日期	
设计一个采购、销售和客户管理应用数据库。其中，一个供应商可以供应多种零件，一种零件也可以有多个供应商。一个客户订单可以订购多种供应商供应的零件。客户和供				

应商都分属不同的国家,而国家按世界五大洲八大洋划分地区。请利用 PowerDesigner 或者 ERwin 等数据库设计工具设计该数据库。

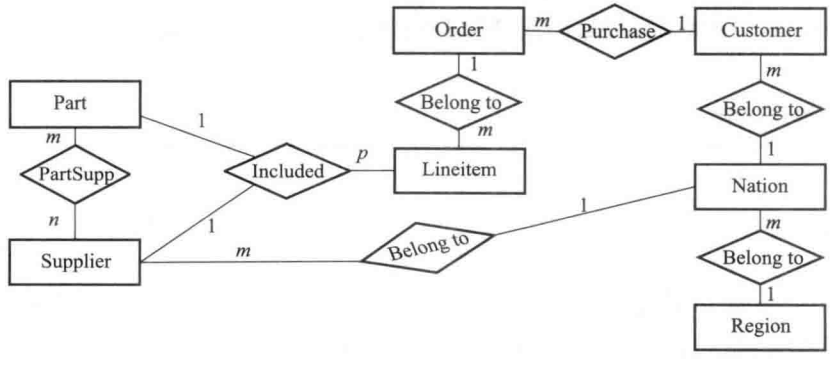
(1) 数据库概念结构设计

识别出零件 Part、供应商 Supplier、客户 Customer、订单 Order、订单项 Lineitem、国家 Nation、地区 Region 等 7 个实体。每个实体的属性、码如下。

- 零件 Part:零件编号 partkey、零件名称 name、零件制造商 mfgr、品牌 brand、类型 type、大小 Size、零售价格 retailprice、包装 container、备注 comment。主码:零件编号 partkey。
- 供应商 Supplier:供应商编号 suppkey、供应商名称 name、地址 address、国籍 nation、电话 phone、备注 comment 等。主码:供应商编号 suppkey。
- 客户 Customer:客户编号 custkey、客户名称 name、地址 address、电话 phone、国籍 nation、备注 comment。主码:客户编号 custkey。
- 订单 Order:订单编号 orderkey、订单状态 status、订单总价 totalprice、订单日期 orderdate、订单优先级 orderpriority、记账员 clerk、运送优先级 shippriority、备注 comment。主码:订单编号 orderkey。
- 订单项 Lineitem:订单项编号 linenumber、所订零件号 partkey、所订零件供应商号 suppkey、零件数量 quantity、零件总价 extendedprice、折扣 discount、税率 tax、退货标记 returnflag 等。主码:订单项编号 linenumber。
- 国家 Nation:国家编号 nationkey、国家名称 name、所属地区 region、备注 comment。主码:国家编号 nationkey。
- 地区 Region:地区编号 regionkey、地区名称 name、备注 comment。主码:地区编号 regionkey。

根据实际语义,分析实体之间的联系,确定实体之间一对一,一对多和多对多联系。

实体-联系图(E-R 图)如下:



（2）数据库逻辑结构设计

按照数据库设计原理中概念结构转化成逻辑结构的规则，每个实体转换成一个关系，多对多的联系也转换成一个关系。因此，根据上述 E-R 图设计数据库逻辑结构（参见实验 1.1 数据库定义中的图 2 TPC-H 数据库模式图）。

（3）数据库物理结构设计

数据库物理结构首先根据逻辑结构自动转换生成，然后根据应用需求设计数据库的索引结构、存储结构。

（4）数据库模式 SQL 语句生成

生成 KingbaseES 数据库管理系统的 SQL 语句参见实验 1.1 数据库定义。

实验总结：

5. 思考题

试选择一个应用，练习数据库设计。

实验 6　存储过程实验

存储过程实验包含三个实验项目（参见表 9），其中 2 个必修实验项目，1 个选修实验项目，均为设计性实验项目。

表 9　“存储过程”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 6.1	存储过程实验	必修	较难	2	第 8 章 8.2,8.3
实验 6.2	自定义函数实验	选修	较难	2	第 8 章 8.2,8.3
实验 6.3	游标实验	选修	较难	2	第 8 章 8.3

实验 6.1　存储过程实验

1. 实验目的

掌握数据库 PL/SQL 编程语言，以及数据库存储过程的设计和使用方法。

2. 实验内容和要求

存储过程定义,存储过程运行,存储过程更名,存储过程删除,存储过程的参数传递。掌握 PL/SQL 编程语言和编程规范,规范设计存储过程。

3. 实验重点和难点

实验重点:存储过程定义和运行。

实验难点:存储过程的参数传递方法。

4. 实验报告示例

实 验 报 告				
题目:存储过程实验	姓名		日期	

(1) 无参数的存储过程

① 定义一个存储过程,更新所有订单的(含税折扣价)总价。

```
/*即根据订单明细表,计算每个订单的总价,更新 ORDER 表*/
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice()
AS
BEGIN
  UPDATE Orders SET totalprice =
        (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
         FROM Lineitem
         WHERE Orders.orderkey = Lineitem.orderkey);
END;
```

② 执行存储过程 Proc_CalTotalPrice()

```
CALL Proc_CalTotalPrice();
```

(2) 有参数的存储过程

① 定义一个存储过程,更新给定订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Order(okey INTEGER)
AS
BEGIN
  UPDATE Orders SET totalprice =
        (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
         FROM Lineitem
         WHERE Orders.orderkey = Lineitem.orderkey AND
               Lineitem.orderkey = okey);
END;
```

② 执行存储过程。

```
CALL Proc_CalTotalPrice4Order(5365);  /*带参数的调用*/
```

(3) 有局部变量的存储过程

① 定义一个存储过程,更新某个顾客的所有订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Customer(p_custname CHAR(25))
AS
DECLARE
   L_custkey INTEGER;
BEGIN
  SELECT custkey INTO L_custkey  /*查找给定客户名对应的客户编号*/
  FROM Customer
  WHERE name =TRIM(p_custname);
                /*TRIM 是系统函数,截去字符串前后空格*/
  UPDATE Orders SET totalprice =/*修改指定客户编号的所有订单的总价*/
         (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
          FROM Lineitem
          WHERE Orders.orderkey = Lineitem.orderkey AND
                Orders.custkey = L_custkey);
END;
```

② 执行存储过程 Proc_CalTotalPrice4Customer()。

```
CALL Proc_CalTotalPrice4Customer('陈楷丰');
```

③ 查看存储过程执行结果。

```
SELECT * FROM Orders
WHERE custkey = (SELECT custkey FROM Customer WHERE name = '陈楷丰');
```

(4) 有输出参数的存储过程

① 定义一个存储过程,更新某个顾客的所有订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Customer2(p_custname CHAR(25),
OUT p_totalprice REAL)
AS
DECLARE
   L_custkey INTEGER;
BEGIN
   SELECT custkey INTO L_custkey  /*查找给定客户名对应的客户编号*/
   FROM Customer
   WHERE name = trim(p_custname);
```

```
        RAISE NOTICE 'custkey is %', L_custkey;      /* 提示客户编号信息 */

        UPDATE Orders SET totalprice =       /* 修改指定客户编号的所有订单的总价 */
                (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
                 FROM Lineitem
                 WHERE Orders.orderkey =
                       Lineitem.orderkey AND Orders.custkey = L_custkey);
               /* 查找指定客户编号的所有订单的总价的和,并通过输出参数输出该值 */
        SELECT SUM(totalprice) INTO p_totalprice
        FROM Orders WHERE custkey = L_custkey;
    END;
```

② 执行存储过程 Proc_CalTotalPrice4Customer2()

```
/* 有输出参数的存储过程,调用的时候,生成临时表,以结果集的形式显示出来 */
CALL Proc_CalTotalPrice4Customer2('陈楷丰',null);
```

③ 查看存储过程执行结果

```
/* 检查下列 SQL 语句执行结果与上述存储过程执行结果是否一致 */
SELECT SUM(totalprice)
FROM Orders
WHERE custkey = (SELECT custkey
                 FROM Customer
                 WHERE name = '陈楷丰');
```

(5) 修改存储过程

① 修改存储过程名 Proc_CalTotalPrice4Order 为 CalTotalPrice4Order。

```
ALTER PROCEDURE Proc_CalTotalPrice4Order RENAME TO
CalTotalPrice4Order;
```

② 编译存储过程 CalTotalPrice4Order。

```
ALTER PROCEDURE CalTotalPrice4Order(okey INTEGER) COMPILE;
```

(6) 删除存储过程

删除存储过程 CalTotalPrice4Order。

```
DROP PROCEDURE CalTotalPrice4Order;
```

实验总结:

存储过程、用户自定义函数可以通过 CALL 和 SELECT 语句调用。需要说明的是:

① 存储过程、用户自定义函数如果带有 OUT 或 INOUT 参数,则参数对应位置在调用时必须使用 NULL 或其他常量占位。运行所得是一个结果集,结果集由一条或多条

RECORD 组成，每条 RECORD 中字段的顺序是 OUT 或 INOUT 参数对应的字段在前，最后返回 RETURN 语句对应的字段。

② SELECT 调用，就是执行普通的 SELECT 语句。对于存储过程，不能和其他任何常量、函数、存储过程等一并构成表达式使用，只能单独作为一个表达式出现在 SELECT 语句中。对于用户自定义函数，如果没有 OUT 或 INOUT 参数，可以和其他常量、变量、对象名如字段名等组合成表达式使用。带有 OUT 或 INOUT 参数的函数不可以参与表达式的计算。

5. 思考题

（1）试总结几种调试存储过程的方法。

（2）存储过程中的 SELECT 语句与普通的 SELECT 语句格式有何不同？执行方法有何不同？

实验 6.2　自定义函数实验

1. 实验目的

掌握数据库 PL/SQL 编程语言以及数据库自定义函数的设计和使用方法。

2. 实验内容和要求

自定义函数定义，自定义函数运行，自定义函数更名，自定义函数删除，自定义函数的参数传递。掌握 PL/SQL 和编程规范，规范设计自定义函数。

3. 实验重点和难点

实验重点：自定义函数的定义和运行。

实验难点：自定义函数的参数传递方法。

4. 实验报告示例

实 验 报 告				
题目：自定义函数实验	姓名		日期	

（1）无参数的自定义函数

① 定义一个自定义函数，更新所有订单的（含税折扣价）总价，并返回所有订单的总价之和。

```
/*该自定义函数与实验 6.1 中 Proc_CalTotalPrice()存储过程类似，区别在于该自定义函数具有一个 REAL 类型的返回值。*/
```

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice( )
RETURN REAL
AS
DECLARE
    res  REAL;
BEGIN
    UPDATE Orders SET totalprice =     /* 更新所有订单的含税折扣价总价 */
           (SELECT SUM(extendedprice * (1-discount) * (1+tax))
            FROM Lineitem
            WHERE Orders.orderkey=Lineitem.orderkey);
     SELECT SUM(totalprice) INTO res   /* 计算所有订单的含税折扣价总价之和 */
     FROM Orders;
     RETURN res;      /* 返回总价之和 */
END;
```

② 执行自定义函数 FUN_CalTotalPrice( )。

```
SELECT FUN_CalTotalPrice( );
/* 执行自定义函数,其返回值以结果集的方式返回和显示。 */
```

(2) 有参数的自定义函数

① 定义一个自定义函数,更新并返回给定订单的总价。

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Order(p_okey INTEGER)
RETURN REAL
AS
DECLARE
    res REAL;
BEGIN UPDATE Orders SET totalprice =   /* 更新给定编号的订单的含税折扣价总价 */
    (SELECT SUM(extendedprice * (1-discount) * (1+tax))
     FROM Lineitem
     WHERE Orders.orderkey=Lineitem.orderkey AND Lineitem.orderkey=p_okey);
     SELECT totalprice INTO res          /* 查找给定订单的总价 */
     FROM Orders;              WHERE orderkey=p_okey;
     RETURN res;       /* 返回给定订单的总价 */
END;
```

② 执行自定义函数 FUN_CalTotalPrice4Order( )

```
/* 更新并返回 5365 号订单的总价 */
        CALL FUN_CalTotalPrice4Order(5365);
```

(3) 有局部变量的自定义函数

① 定义一个自定义函数,计算并返回某个顾客的所有订单的总价。

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Customer(p_custname CHAR(25))
RETURN REAL
AS
DECLARE
   L_custkey   INTEGER;        /* 局部变量 L_custkey */
   res   REAL;
BEGIN
   SELECT custkey INTO L_custkey /* 查找给定客户名的客户编号 */
   FROM Customer
   WHERE name=trim(p_custname);

   RAISE NOTICE 'custkey is %',L_custkey; /* 提示客户编号信息 */

   /*更新指定客户编号的所有订单的含税折扣价总价 */
   UPDATE Orders SET totalprice=
          (SELECT SUM(extendedprice * (1-discount) * (1+tax))
           FROM Lineitem
           WHERE Orders.orderkey=Lineitem.orderkey AND
                 Orders.custkey=L_custkey);
   /* 计算指定客户编号的所有订单的含税折扣价总价之和 */
   SELECT SUM(totalprice) INTO res
   FROM Orders
   WHERE custkey=L_custkey;
   RETURN res;  /*返回总价之和*/
END;
```

② 执行自定义函数 FUN_CalTotalPrice4Customer()。

```
SELECT  FUN_CalTotalPrice4Customer('陈楷丰');
```

(4) 有输出参数的自定义函数

① 定义一个自定义函数,计算并返回某个顾客的所有订单的总价。

/*该函数定义一个输入参数 p_custname,一个输出参数 p_totalprice,还有一个返回值类型 REAL,通过输出参数的定义,该函数可以返回两个或者两个以上的值。该函数与 FUN_CalTotalPrice4Customer ()基本类似,区别只在于该函数多了一个输出参数。*/

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Customer2(p_custname
CHAR(25),OUT p_totalprice REAL)
```

```
    RETURN REAL
    AS
    DECLARE
      L_custkey  INTEGER;
      res  REAL;
    BEGIN
      SELECT custkey INTO L_custkey
      FROM Customer
      WHERE name=trim(p_custname);

      RAISE NOTICE 'custkey is %',L_custkey;

      UPDATE Orders SET totalprice=
            (SELECT SUM(extendedprice * (1-discount) * (1+tax))
             FROM Lineitem
             WHERE Orders.orderkey=Lineitem.orderkey AND
                   Orders.custkey=L_custkey);

      SELECT SUM(totalprice) INTO p_totalprice
      FROM Orders WHERE custkey=L_custkey;

      Res:= p_totalprice;
      RETURN res;
    END;
```

② 执行自定义函数 FUN_CalTotalPrice4Customer2()。

```
SELECT FUN_CalTotalPrice4Customer2('陈楷丰',null);
```

(5) 修改自定义函数

① 修改自定义函数名 FUN_CalTotalPrice4Order 为 CalTotalPrice4Order。

```
ALTER FUNCTIONFUN_CalTotalPrice4Order RENAME TO CalTotalPrice4Order;
```

② 编译自定义函数 CalTotalPrice4Order。

```
ALTER FUNCTION CalTotalPrice4Order(okey INTEGER) COMPILE;
```

(6) 删除自定义函数

删除自定义函数 CalTotalPrice4Order。

```
DROP FUNCTION CalTotalPrice4Order;
```

实验总结：

5. 思考题

（1）试分析自定义函数与存储过程的区别与联系。

（2）如何使得自定义函数可以返回多个值？如何利用？

实验 6.3　游标实验

1. 实验目的

掌握 PL/SQL 游标的设计、定义和使用方法，理解 PL/SQL 游标按行操作和 SQL 按结果集操作的区别和联系。

2. 实验内容和要求

游标定义、游标使用。掌握各种类型游标的特点、区别与联系。

3. 实验重点和难点

实验重点：游标定义和使用。

实验难点：游标类型。

4. 实验报告示例

实 验 报 告				
题目：游标实验	姓名		日期	

（1）普通游标

① 定义一个存储过程，用游标实现计算所有订单的总价。

```
CREATE OR REPLACE PROCEDURE ProcCursor_CalTotalPrice( )
AS  /* 定义存储过程,循环计算和更新每个订单的总价 totalprice */DECLARE
   L_orderkey    INTEGER;
   L_totalprice   REAL;
   CURSOR mycursor FOR    /* 定义下面 SELECT 语句的游标 */
      SELECT orderkey,totalprice
      FROM Orders;
BEGIN
   OPEN mycursor;    /*  打开游标  */
   LOOP   /* 对每个订单记录循环,修改其 totalprice 属性值 */
      FETCH mycursor INTO L_orderkey,L_totalprice; /* 获取游标当前记录 */
      IF mycursor%NOTFOUND THEN
         EXIT;      /* 游标找不到记录,则退出 */
      END IF;
```

```
        /* 计算当前订单所有明细项目的含税折扣价格总和 */
        SELECT SUM(extendedprice * (1-discount) * (1+tax)) INTO L_totalprice
        FROM Lineitem
        WHERE orderkey=L_orderkey;
        /* 修改当前订单的 totalprice 属性值 */
        UPDATE Orders SET totalprice=L_totalprice
        WHERE orderkey=L_orderkey;
    END LOOP;
    CLOSE mycursor;  /* 关闭游标 */
END;
```

② 执行存储过程 ProcCursor_CalTotalPrice()。

```
CALL  ProcCursor_CalTotalPrice();
```

(2) REFCURSOR 类型游标

① 定义一个存储过程,用游标实现计算所有订单的总价。

```
/* 该存储过程与 ProcCursor_CalTotalPrice()存储过程类似,唯一区别就是所定义的游标类型不同 */
CREATE OR REPLACE PROCEDURE ProcRefCursor_CalTotalPrice()
AS
DECLARE
        L_orderkey   INTEGER;
        L_totalprice  REAL;
        mycursor REFCURSOR;          /* 定义 REFCURSOR 类型游标 */
BEGIN
        /* 打开 REFCURSOR 类型游标,游标记录集为 Orders 订单表所有记录 */
        OPEN mycursor FOR SELECT orderkey,totalprice FROM Orders;
        LOOP
            FETCH mycursor INTO L_orderkey,L_totalprice;
            IF mycursor%NOTFOUND THEN
              EXIT;
            END IF;
            SELECT SUM(extendedprice * (1-discount) * (1+tax))  INTO L_totalprice
            FROM Lineitem
            WHERE orderkey=L_orderkey;

            UPDATE Orders SET totalprice=L_totalprice
            WHERE orderkey=L_orderkey;
```

```
        END LOOP;
      CLOSE mycursor;
    END;
```

② 执行存储过程 ProcRefCursor_CalTotalPrice( )。

```
CALL ProcRefCursor_CalTotalPrice( );
```

(3) 记录变量与游标

① 定义一个存储过程,用游标实现计算所有订单的总价。

```
/* 该存储过程与 ProcCursor_CalTotalPrice( )存储过程类似,唯一区别就是所定义的游标结果类型为记录类型的变量 */
    CREATE OR REPLACE PROCEDURE ProcRecCursor_CalTotalPrice( )
    AS
    DECLARE
      L_totalprice   REAL;
      res RECORD;                    /* 定义游标结果为记录类型变量 */
      CURSOR mycursor FOR
          SELECT orderkey,totalprice
          FROM Orders;
      BEGIN
        OPEN mycursor;
        LOOP
          /* 从游标中取出的结果保存在 res 记录类型的变量中 */
          FETCH mycursor INTO res;
          IF mycursor%NOTFOUND THEN EXIT;
          END IF;
          /* 从记录变量 res 中获得当前订单的订单编号,计算当前订单的所有明细项目的含税折扣价总和 */
          SELECT SUM( extendedprice * (1-discount) * (1+tax)) INTO L_totalprice
          FROM Lineitem
          WHERE orderkey=res.orderkey;

          UPDATE Orders SET totalprice=L_totalprice
          WHERE orderkey =res.orderkey;
        END LOOP;
        CLOSE mycursor;
    END;
```

② 执行存储过程 ProcRecCursor_CalTotalPrice( )

```
CALL ProcRecCursor_CalTotalPrice( );
```

(4) 带参数的游标

① 定义一个存储过程,用游标实现计算指定国家的用户订单的总价。

```
/* 该存储过程与 ProcCursor_CalTotalPrice( )存储过程类似,区别在于:该存储过程带有一个参数、所定义的游标为带参数的游标、游标结果定义为记录类型。 */
CREATE OR REPLACE PROCEDURE ProcParaCursor_CalTotalPrice( p_nationname CHAR(20))
AS
DECLARE
  L_totalprice   REAL;
  res    RECORD;          /* 定义游标结果为记录类型变量 */
  /* 定义一个带参数的游标,查询指定国家名称的顾客的订单及其总价 */
  CURSOR mycursor( c_nationname CHAR(20)) FOR
          SELECT O.orderkey, O.totalprice
          FROM Orders O, Customer C, Nation N
          WHERE O.custkey = C.custkey AND c.nationkey = N.nationkey AND
                  TRIM( N.name) = TRIM( c_nationname);
BEGIN
  /* 打开游标:把存储过程的参数 p_nationname 传递给游标的参数 c_nationname */
  OPEN mycursor( p_nationname);
  LOOP
      FETCH mycursor INTO res;
      IF mycursor % NOTFOUND THEN
          EXIT;
      END IF;
      SELECT SUM( extendedprice * (1-discount) * (1+tax)) INTO L_totalprice
      FROM Lineitem
      WHERE orderkey = res.orderkey;

      UPDATE Orders SET totalprice = L_totalprice
      WHERE orderkey  = res.orderkey;
    END LOOP;
  CLOSE mycursor;
END;
```

② 执行存储过程 ProcParaCursor_CalTotalPrice( )。

```
CALL ProcParaCursor_CalTotalPrice('中国');      /* 计算中国用户订单的总价 */
```

实验总结：

① REFCURSOR 类型的游标定义一个游标引用变量，只是在打开该类型游标时才指定具体的 SELECT 语句以便产生游标的结果集。因此 REFCURSOR 实质上是定义了一个动态游标，可以灵活方便地根据程序运行时情况动态设置游标的 SELECT 查询结果集。

② 从任务(1)可以看出，游标可以实现对数据库记录逐条处理，而不是整个结果集一起处理，因此游标是在 PL/SQL 语言中实现过程化处理的核心功能。

③ 从任务(3)看出，记录变量对于游标结果记录的处理很方便，通过记录变量可以直接访问记录的每个属性，而无须为记录的每个属性定义相应的变量。

④ PL/SQL 语言中%TYPE、%ROWTYPE 和 RECORD 等数据类型的区别与联系如下：

- %TYPE 提供一个表字段数据类型的变量，例如在 users 表里面有一个字段叫 user_id，要声明一个和 users.user_id 类型相同的变量，可以写为 user_id users.user_id%TYPE。缺点是：通过使用 %TYPE，必须知道所引用的结构的数据类型。优点是：如果被引用项的表字段数据类型变化了也不需要修改 PL/SQL 的过程体定义。
- %ROWTYPE，一个复合类型变量，叫做行变量。这样的一个变量可以保存一次 SELECT 命令结果的完整一行，只要命令的字段集匹配该变量声明的类型。一个行变量可以声明为和一个现有的表或者视图的行类型相同，方法是使用 table_name%ROWTYPE 表示。在一个行类型的变量中，只可以访问用户定义的表中行的属性。
- RECORD 记录变量类似行类型变量，但是它们没有预定义的结构，而是在 SELECT 命令中获取实际的行结构，这点和 Oracle 不同，在 KingbaseES 的 PL/SQL 的过程体中，正是这种预先没有确定结构的特性为用户提供了极大的灵活性。

5. 思考题

试分析说明 REFCURSOR 类型游标的优点。

实验 7　数据库应用开发实验

数据库应用开发实验分为 2 个选修实验项目(参见表 10)。该实验项目可以任选其一，也可以单独作为一个大作业，或者与数据库设计实验一起作为一个大作业布置给学生，让学生充分利用课外时间完成该大作业。该实验项目为综合型实验项目。

表 10 “数据库应用开发”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 7.1	基于 ODBC 的数据库应用开发实验	选修	较难	4	第 8 章 8.4
实验 7.2	基于 JDBC 的数据库应用开发实验	选修	较难	4	第 8 章 8.6

实验 7.1 基于 ODBC 的数据库应用开发实验

1. 实验目的

掌握基于 ODBC 驱动的数据库应用开发方法。

2. 实验内容和要求

设置 ODBC 驱动数据源,基于 ODBC 驱动的数据库连接方法,实现数据库数据操纵等应用开发常见功能。

3. 实验重点和难点

实验重点:基于 ODBC 驱动的数据库连接方法、数据库数据操纵功能等。

实验难点:不同的数据库应用开发工具具有不同的开发框架和模式。能够较为熟练地使用所选择的应用开发工具,是实现本实验的难点。

4. 实验报告示例

实 验 报 告				
题目:基于 ODBC 的数据库应用开发实验	姓名		日期	
在本实验中,以 VC++6.0 开发环境、KingbaseES 和 SQL Server 数据库为例,实现一个完整的示例程序,其源代码 ODBCTest.c 附在思考题之后。该程序实现把 KingbaseES 的 Samples 数据库中 S-C 模式下的 Student 表数据复制到 SQL Server 的 SamplesBackup 数据库中相同模式下的 Student 表中。为简单起见,也可以在 KingbaseES 中创建 SamplesBackup 数据库,建立 S-C 模式和 Student 表,然后实现上述复制功能。该程序的框架如下。				

（1）配置 ODBC 驱动

配置 KingbaseES ODBC 驱动数据源，共有两种方法：

方法一：运用数据源管理工具来进行配置。打开 Winodws 的“开始/设置/控制面板/管理工具/数据源(ODBC)”，出现“ODBC 数据源管理器”界面，点击“添加”按钮，出现“创建数据源”界面，选择希望安装数据源的驱动程序类别为“KingbaseES ODBC Driver”，出现“建立新的数据源到 KingbaseES”界面，然后设置好数据源的名称、描述信息及服务器的名称等参数即可。同样的方法，可以设置 SQL Server 数据库中的数据源。

方法二：使用 Driver Manager 提供的 ConfigDsn 函数来增加、修改或删除数据源。这种方法特别适用于在应用程序中创建的临时使用的数据源。

（2）基于 ODBC 驱动的数据库连接方法

```
/* Step 1 :定义句柄和变量 */
/* Step 2 :初始化环境 */
/* Step 3 :建立连接 */
```

（3）基于 ODBC 驱动的数据库数据操纵方法

```
/* Step 4 :初始化语句句柄 */
/* Step 5 :两种方式执行语句 */
    /*预编译带有参数的语句 */
    /*直接执行 SQL 语句*/
/* Step 6:处理结果集并执行预编译后的语句*/
```

（4）中断基于 ODBC 驱动的数据库连接

```
/* Step 7 中止处理*/
```

实验总结：

5. 思考题

（1）试修改源程序 ODBCTest.c，实现 TPC-H 数据库 SALES 模式下所有表数据的复制。

（2）请调查目前比较流行的软件开发环境在基于 ODBC 驱动开发数据库应用方面各有什么优缺点？

VC++6.0 环境中编程实现访问数据库的源程序 ODBCTest.c：

```
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <sqltypes.h>
#define SNO_LEN 30
#define NAME_LEN 50
#define DEPART_LEN 100
#define SSEX_LEN 5
int main( )
{
    /* Step 1 :定义句柄和变量 */
    //以 king 开头的表示的是连接 KINGBASEES 的变量
    //以 server 开头的表示的是连接 SQLSERVER 的变量
     SQLHENV kinghenv,serverhenv;        //环境句柄
     SQLHDBC kinghdbc,serverhdbc;        //连接句柄
     SQLHSTMT kinghstmt,serverhstmt;     //语句句柄
     SQLRETURN ret;                      //结果返回集
     SQLCHAR sName[ NAME_LEN ],sDepart[ DEPART_LEN ],sSex[ SSEX_LEN ],
             sSno[ SNO_LEN ];
     SQLINTEGER sAge;
     SQLINTEGER cbAge=0,cbSno=SQL_NTS,cbSex=SQL_NTS,
                cbName=SQL_NTS,cbDepart=SQL_NTS;
      /* Step 2 :初始化环境 */
      //分配环境句柄
      ret=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&kinghenv);
      ret=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&serverhenv);
      //设置管理环境的属性
      ret=SQLSetEnvAttr(kinghenv,SQL_ATTR_ODBC_VERSION,(void*)SQL_OV_ODBC3,0);
      Vret=SQLSetEnvAttr(serverhenv,SQL_ATTR_ODBC_VERSION,(void*)SQL_OV_ODBC3,0);
      /* Step 3 :建立连接 */
      //分配连接句柄
      ret=SQLAllocHandle(SQL_HANDLE_DBC,kinghenv,&kinghdbc);
      ret=SQLAllocHandle(SQL_HANDLE_DBC,serverhenv,&serverhdbc);
      ret=SQLConnect(kinghdbc,                    //连接 KingbaseES
                     "KingbaseES ODBC",SQL_NTS,
```

```
                    "SYSTEM",SQL_NTS,
                    "MANAGER",SQL_NTS);
if (!SQL_SUCCEEDED(ret))                    //连接失败时返回错误值
    return-1;
ret=SQLConnect(serverhdbc,                  //连接 SQLServer
                    "SQLServer",SQL_NTS,
                    "sa",SQL_NTS,
                    "sa",SQL_NTS);
if (!SQL_SUCCEEDED(ret))
    return-1;                               //连接失败时返回错误值
/* Step 4 :初始化语句句柄 */
ret=SQLAllocHandle(SQL_HANDLE_STMT,kinghdbc,&kinghstmt);
ret=SQLSetStmtAttr(kinghstmt,SQL_ATTR_ROW_BIND_TYPE,(SQLPOINTER)SQL_BI
    ND _BY_COLUMN,SQL_IS_INTEGER);//设置语句选项
ret=SQLAllocHandle(SQL_HANDLE_STMT,serverhdbc,&serverhstmt);
/* Step 5 :两种方式执行语句 */
/*预编译带有参数的语句 */
//需要多次执行插入,因此预先声明插入语句
ret=SQLPrepare (serverhstmt,"INSERT INTO STUDENT
                (SNO,SNAME,SSEX,SAGE,SDEPT) VALUES (?,?,?,?,?)"
                SQL_NTS);
if (ret == SQL_SUCCESS || ret == SQL_SUCCESS_WITH_INFO)
{   ret=SQLBindParameter(serverhstmt,1,SQL_PARAM_INPUT,     //绑定参数
    SQL_C_CHAR,SQL_CHAR,SNO_LEN,0,sSno,0,&cbSno);
    ret=SQLBindParameter(serverhstmt,2,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,NAME_LEN,0,sName,0,&cbName);
    ret=SQLBindParameter(serverhstmt,3,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,2,0,sSex,0,&cbSex);
    ret=SQLBindParameter(serverhstmt,4,SQL_PARAM_INPUT,
    SQL_C_LONG,SQL_INTEGER,0,0,&sAge,0,&cbAge);
    ret=SQLBindParameter(serverhstmt,5,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,DEPART_LEN,0,sDepart,0,&cbDepart);
}
/*执行 SQL 语句*/
ret=SQLExecDirect(kinghstmt,"SELECT * FROM STUDENT",SQL_NTS);
if (ret == SQL_SUCCESS || ret == SQL_SUCCESS_WITH_INFO)
    { //将结果集中的属性列一一绑定至变量
```

```
    ret=SQLBindCol(kinghstmt,1,SQL_C_CHAR,sSno,SNO_LEN,&cbSno);
    ret=SQLBindCol(kinghstmt,2,SQL_C_CHAR,sName,NAME_LEN,&cbName);
    ret=SQLBindCol(kinghstmt,3,SQL_C_CHAR,sSex,SSEX_LEN,&cbSex);
    ret=SQLBindCol(kinghstmt,4,SQL_C_LONG,&sAge,0,&cbAge);
    ret=SQLBindCol(kinghstmt,5,SQL_C_CHAR,sDepart,DEPART_LEN,&cbDepart);
    }
/* Step 6 :处理结果集并执行预编译后的语句 */
while ((ret=SQLFetch(kinghstmt)) !=SQL_NO_DATA_FOUND)
    {
      if(ret== SQL_ERROR)                    //错误处理
          printf("Fetch error\n");
      else ret=SQLExecute(serverhstmt);       //执行语句
    }
/* Step 7 :中止处理 */
SQLFreeHandle(SQL_HANDLE_STMT,kinghstmt);
SQLDisconnect(kinghdbc);
SQLFreeHandle(SQL_HANDLE_DBC,kinghdbc);
SQLFreeHandle(SQL_HANDLE_ENV,kinghenv);
SQLFreeHandle(SQL_HANDLE_STMT,serverhstmt);
SQLDisconnect(serverhdbc);
SQLFreeHandle(SQL_HANDLE_DBC,serverhdbc);
SQLFreeHandle(SQL_HANDLE_ENV,serverhenv);
return 0;
}
```

实验 7.2 基于 JDBC 的数据库应用开发实验

1. 实验目的

掌握基于 JDBC 驱动的数据库应用开发方法。

2. 实验内容和要求

基于 JDBC 驱动的数据库连接方法,实现数据库数据操纵等应用开发常见功能。

3. 实验重点和难点

实验重点:基于 JDBC 驱动的数据库连接方法、数据库数据操纵功能等。

实验难点:不同的数据库应用开发工具具有不同的开发框架和模式。能够较为熟练地使用所选择的应用开发工具,是实现本实验的难点。

4. 实验报告示例

<table>
<tr><td colspan="5">实 验 报 告</td></tr>
<tr><td>题目：基于 JDBC 的数据库应用开发实验</td><td>姓名</td><td></td><td>日期</td><td></td></tr>
<tr><td colspan="5">在本实验中，以 Eclipse 开发环境和 KingbaseES 数据库为例，实现一个完整的示例程序，其源代码 JDBCTest.java 附在思考题之后。该程序实现把 KingbaseES 的 Samples 数据库中 S-C 模式下的 Student 表数据复制到 SQL Server 的 SamplesBackup 数据库中相同模式的相同表中。为简单起见，也可以在 KingbaseES 中创建 SameplesBackup 数据库，建立 S-C 模式和 Student 表，然后实现上述复制功能。该程序的框架如下：
（1）基于 JDBC 驱动的数据库连接方法
/* Step 1 定义句柄和变量 */
/* Step 2 初始化环境 */
/* Step 3 :建立连接 */
（2）基于 JDBC 驱动的数据库数据操纵方法
/* Step 4 :初始化语句句柄 */
/* Step 5 :两种方式执行语句 */
/* 预编译带有参数的语句 */
/*直接执行 SQL 语句*/
/* Step 6:处理结果集并执行预编译后的语句*/
（3）中断基于 JDBC 驱动的数据库连接
/* Step 7 中止处理*/</td></tr>
<tr><td colspan="5">实验总结：</td></tr>
</table>

5. 思考题

（1）试修改源程序 JDBCTest.java，实现 TPC-H 数据库 SALES 模式下所有表的复制。

（2）请调查目前比较流行的软件开发环境在基于 JDBC 驱动开发数据库应用方面各有哪些优缺点？

Eclipse 环境中编程实现访问数据库的源程序 JDBCTest.java：

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Properties;

public class JDBCTest {
    public static void main(String[ ] args) {

        /* Step 1 :定义句柄和变量 */
        String sSno,sName,sSex,sDepart;
        Integer sAge;

        /* Step 2 :初始化环境 */
        /* Step 3 :建立连接 */

        String driverName = "com.kingbase.Driver";                    //连接源数据库
        String src_url = "jdbc:kingbase: //localhost:54321/SAMPLES";
        Properties src_info = new Properties( );
        src_info.put ("user","SYSTEM");
        src_info.put ("password","MANAGER");
        src_info.put ("EscapeProcessing","false");

        String dst_url = "jdbc:kingbase://localhost:54321/SameplesBackup";  //连接目标数据库
        Properties dst_info = new Properties( );
        dst_info.put ("user","SYSTEM");
        dst_info.put ("password","MANAGER");
        dst_info.put ("EscapeProcessing","false");

        try{
            Class.forName( driverName);
            Connection src_conn = DriverManager.getConnection( src_url,src_info);
            Connection dst_conn = DriverManager.getConnection( dst_url,dst_info);

            /* Step 4 :初始化语句句柄 */
            Statement src_stmt = src_conn.createStatement( );
            PreparedStatement dst_prestmt = dst_conn.prepareStatement ("INSERT INTO \"S-C\".
                                                                    STUDENT (SNO, SNAME,
                                                                    SSEX,SAGE,SDEPT) VAL-
                                                                    UES (?,?,?,?,?)");
```

```
        /* Step 5 :两种方式执行语句 */
        /* 预编译带有参数的语句 */
        /* 直接执行 SQL 语句 */
        ResultSet rs = src_stmt.executeQuery( "SELECT * FROM \"S-C\".Student" );

        /* Step 6 :处理结果集并执行预编译后的语句 */
        while ( rs.next( ) )
          {
            sSno = rs.getString( "SNO" );
            sName = rs.getString( "SNAME" );
            sSex = rs.getString( "SSEX" );
            sAge = rs.getInt( "SAGE" );
            sDepart = rs.getString( "SDEPT" );

            dst_prestmt.setString( 1, sSno );
            dst_prestmt.setString( 2, sName );
            dst_prestmt.setString( 3, sSex );
            dst_prestmt.setInt( 4, sAge );
            dst_prestmt.setString( 5, sName );

            dst_prestmt.executeUpdate( );

            System.out.println( sSno+sName+sSex+sDepart+sAge );
          }
        /* Step 7 :中止处理 */
          rs.close( );
          src_stmt.close( );
          dst_prestmt.close( );
          src_conn.close( );
          dst_conn.close( );
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage( ) );
        }
    }
}
```

实验 8 数据库设计与应用开发大作业

实验 8 是数据库设计与应用开发大作业(参见表 11),是一个综合型的实验项目。

表 11 "数据库事务管理"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
数据库设计与应用开发大作业	必修	较难	8	第 3-8 章

1. 实验目的

掌握综合运用数据库原理、方法和技术进行数据库应用系统分析、设计和开发的能力。

2. 实验内容和要求

为某个部门或单位开发一个数据库应用系统,具体内容包括:对某个部门或单位业务和数据进行调查,系统分析,系统设计,数据库设计,数据库创建和数据加载,数据库应用软件开发,系统测试,系统分析设计和开发文档撰写,软件、文档和数据库提交,数据库应用系统运行演示和大作业汇报。

能够针对某个部门或单位的应用需求,通过系统分析,从数据库数据和应用系统功能两方面进行综合设计,实现一个完整的数据库应用系统。同时培养团队合作精神。要求 5~6 位同学组成一个开发小组,每位同学承担不同角色(如项目管理员、DBA、系统分析员、系统设计员、系统开发员、系统测试员)。撰写系统设计和开发文档;提交系统文档、数据库应用软件和数据库。每个小组进行 60 分钟的报告和答辩,讲解设计方案,演示系统运行,汇报分工与合作情况。

3. 实验重点和难点

实验重点:数据库设计,数据库应用软件开发。

实验难点:综合运用系统分析与设计方法,从数据和功能两方面协调设计一个完整的数据库应用系统。熟练掌握和运用一个主流数据库应用开发工具进行数据库应用软件开发。

4. 实验报告示例

实 验 报 告				
题目:数据库设计与应用开发大作业	姓名		日期	
在"数据库系统概论"精品课程网站上有一些大作业实验报告示例,可供读者参考学习。				

数据库设计和应用开发步骤主要包括：

① 系统调查

② 系统分析

③ 系统设计

④ 数据库设计

⑤ 数据库创建和数据加载

⑥ 数据库应用软件的功能设计和开发

⑦ 数据库应用系统测试

⑧ 分析设计和开发文档撰写

⑨ 应用软件、数据库和文档提交

⑩ 应用系统演示和汇报

实验总结：

5. 思考题

① 数据库应用系统的功能设计对数据库设计有何影响？

② 如何协调数据库应用系统的功能设计与数据库设计？

③ 数据库应用开发中使用的程序设计语言的数据类型与数据库的数据类型如何互操作？

实验 9　数据库监视与性能优化

数据库监视与性能优化实验包括三个选修实验项目(参见表 12),为验证性与设计性相结合的实验项目。

表 12　“数据库监视与性能优化”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 9.1	数据库查询性能调优实验	选修	难	4	第 9 章
实验 9.2	数据库性能监视实验	选修	难	4	第 9 章
实验 9.3	数据库系统配置参数调优实验	选修	难	4	第 9 章

实验 9.1 数据库查询性能调优实验

1. 实验目的

理解和掌握数据库查询性能调优的基本原理和方法。

2. 实验内容和要求

学会使用 EXPLAIN 命令分析查询执行计划、利用索引优化查询性能、优化 SQL 语句，以及理解和掌握数据库模式规范化设计对查询性能的影响。能针对给定的数据库模式，设计不同的实例验证查询性能优化效果。

3. 实验重点和难点

实验重点：利用索引优化查询性能、优化 SQL 语句。

实验难点：数据库模式规范化设计对查询性能的影响。

4. 实验报告示例

实 验 报 告				
题目：数据库查询性能调优	姓名		日期	

(1) 使用 EXPLAIN 命令查看查询执行计划

查看 Part、PartSupp、Supplier 三个表连接查询的查询执行计划。

```
EXPLAIN  SELECT *
FROM Part P,Partsupp PS,Supplier S
WHERE P.partkey=PS.partkey AND PS.suppkey=S.suppkey
        AND P.name='发动机'
ORDER BY S.acctbal desc,S.name,P.partkey;
/*该 SQL 语句是要查询零件名为"发动机"的零件、供应该零件的供应商以及供应等详细信息，
并按照供应商的账户余额(降序)、供应商名称(升序)、零件编号(升序)，排序输出结果*/
```

(2) 利用索引优化查询性能

建立索引，优化 SQL 查询性能。

```
CREATE INDEX IDX_part_name ON Part(name); /*在 Part 表的 name 属性上建立索引*/

EXPLAIN  SELECT *
FROM Part P,Partsupp PS,Supplier S
WHERE P.partkey=PS.partkey AND PS.suppkey=S.suppkey
        AND P.name='发动机'
ORDER BY S.acctbal desc,S.name,P.partkey;
```

比较在 Part 表的 name 上有索引和无索引时，两种执行计划有何异同，并实际执行该查询，验证有索引和无索引时此查询语句的执行性能。

(3) 优化 SQL 语句

① IN 与 EXISTS 查询

```
SELECT *
FROM Orders
WHERE orderkey IN(SELECT orderkey
                  FROM  Lineitem
                  WHERE partkey IN(SELECT partkey
                                   FROM  Part
                                   WHERE name='发动机'));
```

一般地，使用 EXISTS 查询效率要高于 IN 查询，改写 SQL 语句如下：

```
SELECT *
FROM Orders O
WHERE EXISTS(SELECT *
             FROM Lineitem L
             WHERE O.orderkey=L.orderkey AND
                   EXISTS(SELECT *
                          FROM Part P
                          WHERE P.partkey=L.partkey AND name='发动机'));
```

比较两种执行计划，并实际测试执行性能哪种情况好。

② 尽可能使用不相关子查询，避免使用相关子查询。不相关子查询一般比相关子查询执行效率要高，在可能的情况下，改写相关子查询为不相关子查询。

查找这样的订单，其总价大于该顾客所购商品的平均总价。

相关子查询：

```
SELECT *
FROM Orders O1
WHERE O1.totalprice > (SELECT AVG(O2.totalprice)
                       FROM Orders O2
                       WHERE O2.custkey=O1.custkey)
```

不相关子查询：

```
SELECT *
FROM Orders O1,(SELECT O2.custkey,AVG(O2.totalprice) AS avgprice
FROM Orders O2
GROUP BY O2.custkey) AVG1
```

```
/*子查询生成临时派生表,取名为 AVG1*/
    WHERE O1.custkey=AVG1.custkey AND O1.totalprice > AVG1.avgprice;
```

比较两种执行计划,并实际测试执行性能哪种情况好。

(4) 数据库模式规范化设计对查询性能的影响

分析 TPC-H 数据库模式中是否存在不规范化的设计。该设计在海量数据的情况下查询效率怎样? 如何在设计上进一步提高海量数据的查询效率?

第三范式在一定程度上减少了不必要的冗余,提高了数据库的查询效率,但是如果数据量大且需要大量联合查询的时候,第三范式设计又可能会影响查询效率。TPC-H 中存在的不规范的设计如下:

① Orders 表中订单的 totalprice 由 Lineitem 表中该订单的所有订单项的 extendedprice、discount 和 tax 等属性计算得出,即 totalprice = SUM(Lineitem.extendedprice×(1-Lineitem.discount)×(1+Lineitem.tax)

该项设计虽然不规范,但大大提高了客户所有订单总金额的查询效率。例如,查询所有购物总金额大于 20 万的客户编号及其购物总金额:

```
SELECT custkey,SUM(totalprice)
FROM Orders
GROUP BY custkey
HAVING SUM(totalprice) > 200000.00;
```

如果不用 Totalprice 字段,SQL 查询语句则为:

```
SELECT O.custkey,SUM(L.extendedprice * (1-L.discount) * (1+L.tax)) AS sumprice
FROM Orders O,Lineitem L
WHERE O.orderkey=L.orderkey
GROUP BY O.custkey
HAVING sumprice > 200000.00;
```

② Lineitem 表中订单明细价格 extendedprice 由 quantity 和 Part 表中的 retailprice 得出,即 extendedprice=quantity× Part.retailprice。

该项设计虽然不规范,但大大提高了客户订单总金额的查询效率。例如,查询所有购物零售总金额(折扣和加税之前的价格,即 extendedprice)大于 1 万的订单对应的客户编号及金额:

```
SELECT O.custkey,SUM(L.extendedprice) AS sumextprice
FROM Orders O,Lineitem L
WHERE O.orderkey=L.orderkey
GROUP BY O.custkey
HAVING sumextprice > 10000.00;
```

如果不使用 extendedprice，SQL 查询则为：

```
SELECT O.custkey,SUM(L.quantity * P.retailprice) AS sumextprice
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
GROUP BY O.custkey
HAVING sumextprice > 10000.00;
```

实验总结：

5. 思考题

对实验 1.3 中的查询，使用不同的 SQL 语句来表达，比较它们的查询效率，体会并总结查询优化的技巧和方法。

实验 9.2　数据库性能监视实验

1. 实验目的

了解所使用的 DBMS 提供的数据库性能监视功能，学习数据库查询性能监视的基本原理和方法。

2. 实验内容和要求

使用 KingbaseES 的数据库性能监视工具，通过标准统计视图和统计访问函数查看数据库系统收集到的性能统计信息、ANALYZE 更新数据库统计信息，通过专门工具监视系统系性能。希望能够熟悉数据库系统有关性能统计信息的标准视图和统计访问函数，了解如何通过系统收集到的性能数据监视系统性能。

3. 实验重点和难点

实验重点：通过标准统计视图和统计访问函数查看数据库系统收集到的性能统计信息。

实验难点：分析性能统计信息，找出存在的性能问题。

4. 实验报告示例

实　验　报　告				
题目：数据库性能监视实验	姓名		日期	
（1）查看系统标准统计视图和统计访问函数 ① KingbaseES 部分标准统计视图				

sys_stat_activity：每个服务器进程一行，显示数据库、进程、用户、当前查询等信息。只有在打开stats_command_string参数时，才能得到报告当前查询的相关信息的各个字段。

sys_stat_database：每个数据库一行，显示数据库与该数据库连接的活跃服务器进程数、提交的事务总数以及在该数据库中回滚数目的总数、读取的磁盘块的总数，以及缓冲区命中的总数。

sys_stat_all_tables：当前数据库中每个表一行，显示模式和表信息、发起的顺序扫描的总数、顺序扫描抓取的有效数据行的数目、发起的索引扫描的总数、索引扫描抓取的有效数据行的数目，以及插入、更新和删除的行的总数。

sys_stat_all_indexes：对当前数据库的每个索引，显示输入索引所在表和索引，模式、表和索引名，包括使用了该索引的索引扫描总数、索引扫描返回的索引记录的数目、使用该索引的简单索引扫描抓取的有效的表中数据行数。

sys_statio_all_tables：当前数据库中每个表一行，显示表、模式和表名，包含从该表中读取的磁盘块总数、缓冲区命中的次数、在该表上所有索引的磁盘块读取和缓冲区命中总数等信息。

sys_statio_all_indexes：当前数据库中每个索引一行，显示表和索引OID、模式、表和索引名、该索引的磁盘块读取和缓冲区命中的数目。

sys_statio_all_sequences：当前数据库中每个序列对象一行，显示序列的OID、模式和序列名、序列磁盘读取和缓冲区命中的数目。

② KingbaseES部分统计访问函数

sys_stat_get_db_numbackends(oid)，integer，数据库活跃的服务器进程数目

sys_stat_get_db_xact_commit(oid)，bigint，数据库中已提交的事务数量

sys_stat_get_db_xact_rollback(oid)，bigint，数据库中回滚的事务数量

sys_stat_get_tuples_inserted(oid)，bigint，插入表中的元组数量

sys_stat_get_tuples_updated(oid)，bigint，在表中已更新的元组数量

sys_stat_get_tuples_deleted(oid)，bigint，从表中删除的元组数量

sys_stat_get_blocks_fetched(oid)，bigint，表或者索引的磁盘块被存取的数量

sys_backend_pid()，integer，附着在当前会话上的服务器进程ID

sys_stat_get_backend_pid(integer)，integer，给出的服务器进程的进程号

sys_stat_get_backend_dbid(integer)，oid，指定服务器进程的数据库ID

sys_stat_get_backend_userid(integer)，oid，指定服务器进程的用户ID

sys_stat_get_backend_activity(integer)，text，服务器进程的当前活跃查询

sys_stat_get_backend_client_port(integer)，integer，连接到给定服务器的客户端口

sys_stat_reset()，boolean，重置所有当前收集的统计

(2) 查看数据库管理系统当前活动情况

① 查看当前进程数

```
SELECT COUNT( * )
```

```
    FROM (SELECT sys_stat_get_backend_idset() AS backendid) AS s;
    /* sys_stat_get_backend_idset() 函数为:获得服务器后台进程集合 */
```

② 查看当前进程的详细活动

```
SELECT * FROM sys_stat_activity;
```

③ 查看正在进行的 SQL 操作

```
SELECT sys_stat_get_backend_pid(s.backendid) AS procpid,
       sys_stat_get_backend_activity(s.backendid) AS current_query
FROM (SELECT sys_stat_get_backend_idset() AS backendid) AS s ;
```

(3) 更新数据库统计信息

① 用 ANALYZE 更新数据库统计信息

```
ANALYZE;
```

② 用 ANALYZE 更新数据表的统计信息

```
ANALYZE Sales.Orders;
```

实验总结:

5. 思考题

试总结所用的 DBMS 的性能监视工具的功能和使用方法。

实验 9.3　数据库系统配置参数调优实验

1. 实验目的

了解数据库系统级参数和连接级参数的配置和调优的基本原理和方法。了解用户可以通过修改这些参数设置来调整系统运行时配置,以优化系统性能。

2. 实验内容和要求

熟悉和了解数据库各级参数的作用以及配置,包括系统级参数配置和调优、数据库级参数配置和调优、会话(连接)级参数配置和调优。

3. 实验重点和难点

实验重点:数据库级参数配置和调优、连接级参数配置和调优。

实验难点:系统级参数配置和调优。

4. 实验报告示例

实 验 报 告				
题目:数据库系统配置参数调优实验	姓名		日期	

(1) 显示参数的命令

① 显示 XXX 参数。

```
SHOW  shared_buffers;  /* 专门的显示系统会话值的命令,可以显示所有参数的值 */
```

② 显示 xxx 参数。

```
SELECT  shared_buffers;  /* 只能显示一个参数的值 */
```

(2) 设置会话级参数

① 设置提交延迟时间参数为 10 s,对本次会话有效。

```
SET SESSION COMMIT_DELAY to 10;
```

② 设置提交延迟时间参数为 10 s,对本次提交事务有效。

```
SET LOCAL COMMIT_DELAY to 10;
```

(3) 修改数据库默认参数

设置 test 数据库的密码长度为 10 个字符,该设置覆盖数据库的默认设置,对该数据库上启动的每个会话都有效。

```
ALTER DATABASE test SET password_length TO 10;
```

(4) 设置系统级或全局级参数配置

① 关闭自动清理存储空间开关,自动修改数据库系统配置文件,该修改将在重启数据库服务器后才生效。

```
ALTER SYSTEM SET autovacuum = off
```

② 设置数据库监听的地址,自动修改数据库系统配置文件,该修改将在重启数据库服务器后才生效。

```
ALTER SYSTEM SET listen_addresses = '127. 0. 0. 1'
```

(5) 优化系统级参数

优化 KingbaseES 数据库管理系统的 shared_buffers 参数

```
ALTER SYSTEM SET shared_buffers = 80000;
```

该参数是很重要的参数,数据库管理系统通过 shared_buffers 与内核和磁盘打交道,一方面,该参数应该设置得足够大来应付通常的表访问操作,让更多的数据缓存在 shared_buffers 中。通常设置为实际 RAM 的 10%,比如设置 80 000 个缓冲区(每个缓冲区一般为

8 KB,8 万个缓冲区大小为 625 MB)。将所有的内存都给 shared_buffers 将导致没有内存来运行程序。另一方面,它应该足够小,以避免操作系统因缺少内存而将页面换进换出。

(6) 优化会话级参数

① 优化 KingbaseES 数据库管理系统的 work_mem 参数。

```
SET work_mem = 61440;
```

在执行排序操作时,系统会根据 work_mem 的大小决定是否将一个大的结果集拆分为几个小的和 work_mem 差不多大小的临时文件。显然,拆分的结果会降低排序的速度。因此增加 work_mem 有助于提高排序的速度。通常设置为实际 RAM 的 2%~4%,还要根据需要排序结果集的大小而定,比如 61 440(60 MB)。

② 优化 KingbaseES 数据库管理系统的 temp_buffers 参数。

```
SET temp_buffers = 2048;  /* 设置临时缓冲区为 2 GB */
```

该参数称之为临时缓冲区,用于数据库会话访问临时表数据,系统默认值为 8 MB。可以在单独的 session 中对该参数进行设置,尤其是需要访问比较大的临时表时将会有显著的性能提升。

实验总结:

5. 思考题

试通过一个实际例子,测试不同的 temp_buffers 大小对临时表查询性能的影响。

实验 10　数据库恢复技术实验

数据库恢复技术实验包括三个选修实验项目(参见表 13)。该实验项目为验证性实验项目。

表 13　"数据库恢复技术"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 10.1	事务实验	选修	适中	2	第 10 章
实验 10.2	数据库备份实验	选修	适中	2	第 10 章
实验 10.3	数据库恢复实验	选修	适中	2	第 10 章

实验 10.1 事务实验

1. 实验目的

掌握数据库事务管理的基本原理以及事务的编程方法。

2. 实验内容和要求

设计几个典型的事务应用,包括显式事务、事务提交、事务回滚、隐式事务等。

3. 实验重点和难点

实验重点:显式事务的编写。

实验难点:把事务的编写和存储过程的设计与使用结合起来。

4. 实验报告示例

实 验 报 告				
题目:事务实验	姓名		日期	

(1) 显式事务的编写

① 创建一个事务,当用户购买零件时,插入订单明细和订单记录,修改供应基本表以保持数据一致性。

```
BEGIN TRANSACTION;          /* 明显写出事务开始和后面的事务提交 */
INSERT INTO Orders(orderkey,custkey,orderstatus,totalprice,   /* 插入订单记录 */
                   orderdate,orderpriority,clerk,shippriority,comment)
VALUES(1021,3,'P',0.0,CURRENT_DATE,'Common','Mike',1,null);
INSERT INTO Lineitem(orderkey,partkey,suppkey,          /* 插入订单明细记录 */
       linenumber,quantity,extendedprice,discount,tax,returnflag,linestatus,
       shipdate,commitdate,receiptdate,shipinstruct,shipmode,comment)
VALUES(1021,479,1,1,2,0.0,0.3,0.08,'F','C',DATEADD('day',3,
       CURRENT_DATE),DATEADD('day',3,CURRENT_DATE),
       CURRENT_DATE,'BOX','GENERAL',NULL);
                   /* 修改订单明细项目对应的零件供应记录中的可用数量 */
UPDATE PartSupp SET availqty=availqty-2
WHERE partkey =479 AND suppkey =1;
COMMIT TRANSACTION;                    /* 明显写出事务开始和事务提交 */
```

② 创建一个事务,当用户撤销某个用户购买记录时,删除订单明细(假设只有一项订单明细)和订单记录,然后修改供应基本表以保持数据一致性。

```
BEGIN TRANSACTION;
   DELETE FROM Lineitem          /* 先删除订单明细记录 */
   WHERE orderkey=1021 AND linenumber=1;
```

```
    DELETE  FROM  Orders              /*再删除订单记录*/
    WHERE  orderkey=1021;
    /*修改订单明细记录对应的零件供应记录中的可用数量*/
    UPDATE PartSupp  SET  availqty=availqty+2
    WHERE partkey=479 AND suppkey=1;
    COMMIT TRANSACTION;
```

③ 创建一个存储过程，当用户撤销某个用户购买记录时，修改供应基本表，删除订单明细（可以是多项）和订单记录，保持数据一致性。

```
    /*该存储过程就是一个事务*/
    /*创建存储过程 PRC_DeleteOrder：删除输入的订单。首先逐个删除订单明细记录，最后删除
  订单记录 */
    CREATE OR REPLACE PROCEDURE PRC_DeleteOrder(P_orderkey INTEGER)
    AS
    DECLARE          /*定义一个带参数的游标，检索给定订单号的订单明细记录*/
      CURSOR Cursor_Lineitem(L_orderkey INTEGER)
        FOR SELECT * FROM Sales.Lineitem WHERE orderkey=L_orderkey;
      res Lineitem%ROWTYPE;              /*定义订单明细记录类型变量 res*/
    BEGIN
      OPEN Cursor_Lineitem(P_orderkey);
    /*打开游标：把存储过程的参数传递给游标的参数*/
    LOOP                          /*针对给定订单的明细记录循环，逐个删除订单明细记录 */
      FETCH Cursor_Lineitem INTO res;          /*从游标获取当前订单明细记录 */
      IF Cursor_Lineitem%NOTFOUND THEN
        EXIT;
      END IF;
      /*修改零件供应表中对应的可用数量 */
      UPDATE Sales.PartSupp SET availqty=availqty+res.quantity
      WHERE partkey=res.partkey AND suppkey=res.suppkey;
      /*删除当前订单明细记录 */
      DELETE FROM  Sales.Lineitem
      WHERE orderkey=P_orderkey AND linenumber=res.linenumber;
    END LOOP;
    CLOSE Cursor_Lineitem;          /* 关闭游标 */
    DELETE FROM  Sales.Orders      /* 删除订单记录 */
    WHERE orderkey=P_orderkey;
```

```
    END;
```

(2) 显式事务的编写(带有回滚)

创建一个事务,当用户购买零件时,插入订单明细(假设只购买一项明细)和订单记录,修改供应基本表,以保持数据一致性。

```
    /* 创建存储过程,完成用户购买零件事务 */
    CREATE OR REPLACE PROCEDURE PRC_Purchase4OneItem(p_orderkey INTEGER,p_custkey
INTEGER,p_partkey INTEGER,p_suppkey INTEGER,p_purchaseqty INTEGER)
    AS
    DECLARE
      L_availqty  INTEGER;
      BEGIN
      BEGIN TRANSACTION
      /* 插入订单记录 */
      INSERT INTO Sales.Orders(orderkey,custkey,orderstatus,totalprice,
            orderdate,orderpriority,clerk,shippriority,comment)
      VALUES(P_orderkey,P_custkey,'P',0.0,CURRENT_DATE,
            'Common','Mike',1,null);
      /*插入订单明细记录 */
      INSERT INTO Sales.Lineitem(orderkey,partkey,suppkey,linenumber,
          quantity,extendedprice,discount,tax,returnflag,linestatus,
          shipdate,commitdate,receiptdate,shipinstruct,shipmode,comment)
      VALUES(P_orderkey,P_partkey,P_suppkey,1,P_purchaseqty,0.0,0.3,0.08,
            'F','C',  DATEADD('day',3,CURRENT_DATE),
            DATEADD('day',3,CURRENT_DATE),
            CURRENT_DATE,'BOX','GENERAL',NULL);

      /*查询订单明细记录对应的零件供应记录中的可用数量*/
      SELECT availqty INTO L_availqty
      FROM Sales.PartSupp
      WHERE partkey=P_partkey AND suppkey=P_suppkey;
      IF (L_availqty > P_purchaseqty)  THEN
        /*如果可用数量大于订购数量,则修改零件供应记录,提交事务*/
        UPDATE Sales.PartSupp  SET  availqty=availqty-P_purchaseqty
        WHERE partkey=P_partkey AND suppkey=P_suppkey;
        COMMIT TRANSACTION;
      ELSE
```

```
    ROLLBACK TRANSACTION;                    /* 可用数量不够,回滚事务 */
    END IF;
END;

CALL PRC_Purchase4OneItem (1021,3,479,1,20);      /* 执行存储过程,购买零件 */

/* 查看 Orders、Lineitem 和 PartSupp 三个表中相应记录,验证存储过程的事务执行正确性 */
SELECT  * FROM Orders WHERE orderkey = 1021;
SELECT * FROM Lineitem WHERE orderkey = 1021;
SELECT * FROM PartSupp WHERE partkey = 479 AND suppkey = 1;

CALL PRC_DeleteOrder(1021);                  /* 执行存储过程,删除指定订单 */
CALL PRC_Purchase4OneItem (1021,3,479,1,40000);/* 重新执行存储过程,购买零件 */

/* 再次查看 Orders、Lineitem 和 PartSupp 三个表中相应记录,验证存储过程的事务执行正确性 */
SELECT  * FROM Orders WHERE orderkey = 1021;
SELECT * FROM Lineitem WHERE orderkey = 1021;
SELECT * FROM PartSupp WHERE partkey = 479 AND suppkey = 1;
```

5. 思考题

试分析存储过程与事务的联系和区别。

实验 10.2　数据库备份实验

1. 实验目的

掌握数据库数据转储备份的方法。

2. 实验内容和要求

了解数据转储备份的概念。学习实际使用的数据库系统(如 Kingbase)中数据库逻辑备份、物理备份 、增量备份和完全备份的概念和使用方法。利用数据库管理系统提供的备份工具实现各种数据库备份策略。

说明:数据库备份也称数据库转储。物理备份是复制数据库物理文件;逻辑备份将数据库对象的定义和数据导出到指定文件中;完全备份是备份整个数据库;增量备份是只备份上次备份以来有变化的数据。

3. 实验重点和难点

实验重点:逻辑备份,物理备份。

实验难点:增量备份。

4. 实验报告示例

实 验 报 告				
题目:数据备份实验	姓名		日期	

实际使用的数据库管理系统所提供的备份功能、备份命令都不尽相同。我们以 Kingbase 为例进行实验。

KingbaseES 提供逻辑备份有三种备份模式:全库备份、模式备份、表备份。

全库备份是指备份单个数据库中所有的用户可备份的对象;模式备份是指备份用户指定的模式和模式所包含的对象;表备份是指备份指定的表和表的数据。

KingbaseES 提供了图形界面的逻辑备份还原工具 javatools.bat JDump,并提供了命令行的逻辑备份工具 sys_dump 和 exp。sys_dump 是 KingbaseES 专有的逻辑备份工具,而 exp 是兼容 Oracle 的逻辑备份工具,exp 的使用依赖于配置的 Kingbase 服务,所以需要在使用前配置 sys_service.conf 文件。

KingbaseES 提供物理全系统备份,指对“全系统”进行备份,备份结果形成一个备份集。备份机制是通过复制 KingbaseES 系统,以统一的格式形成一个全系统的拷贝,以备恢复到一个一致的数据状态。物理全系统备份支持两种类型:物理全系统联机备份和物理全系统脱机备份。KingbaseES 提供了图形界面的物理备份还原工具 javatools.bat JBackup,并提供了命令行的物理备份还原工具 sys_backup。

本实验使用 KingbaseES 命令行工具 sys_dump 和 sys_backup 备份数据库。打开命令窗口,设置 KingbaseES 的可执行文件所在的目录为当前路径,完成如下所有实验。本书假设 KingbaseES 安装后的可执行文件所在目录为 c:\Kingbase\ES\V7\bin。

(1) 逻辑备份整个数据库

① 备份为二进制文件。

```
/*-F c | p:  c 表示备份文件为二进制格式,p 表示备份文件为 SQL 文件,不指明该参数默认为二进制格式;-f 参数指定备份文件名*/
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-F c-f d:\dumpfile.dmp TPCH
```

② 备份为 SQL 文件。

```
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-F p-f d:\dumpfile.sql  TPCH
```

(2) 逻辑备份多个表

备份 TPC-H 的 Orders,Lineitem,PartSupp 等多个表。

```
    /*-t 参数指定备份的表名*/
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.Orders  -f d:\dumpfile.dmp
TPCH
```

```
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.Lineitem  -f d:\dumpfile.dmp
TPCH
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.PartSupp  -f d:\dumpfile.dmp
TPCH
```

(3) 逻辑备份数据库中指定的模式

备份数据库 TPCH 的 SALES 模式。

```
/*-n 参数指定备份模式名*/
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-n Sales  -f d:\dumpfile.dmp TPCH
```

(4) 利用 COPY TO 命令备份数据

备份数据库 TPCH 中 SALES 模式的 Lineitem 表到物理文件 lineitem.csv 中。

```
/*该命令不是 KingbaseES 的命令行工具,需要在 KingbaseES 的查询分析器中执行。FORCE QUOTE
comment:强制复制表中 comment 属性的所有非空数值,两边使用 QUOTE 指定的标记符*/
COPY Sales.Lineitem TO 'd:\lineitem.csv' WITH CSV QUOTE '"' FORCE QUOTE comment;
```

(5) 联机全系统物理备份

设置数据库管理系统为归档模式,然后联机进行全系统的物理备份数据库 TPC-H。

① 首先找到数据库管理系统的配置文件,如 KingbaseES 数据库的配置文件为 Data 目录下的 Kingbase.conf,找到其中 log_archive_start 参数设置为 on,同时设置 log_archive_dest 为一个已经存在的存放归档日志的文件夹名称,然后启动数据库服务器。

② 利用BACKUP 命令联机备份数据库 TPC-H。

```
/*BACKUP 命令不是 KingbaseES 的命令行工具,需要在查询分析器中执行。NAME 参数指定备份
集的名称,TYPE FULL 参数指定为全系统备份,FILEPATH 指定备份数据存放的文件夹,该文件夹可以存
放多个备份集成员。*/
BACKUP NAME week1 TYPE FULL FILEPATH 'd:\Kingbase\archive';
```

(6) 脱机全系统物理备份

停止数据库服务器运行,然后脱机进行全系统的物理备份数据库 TPC-H。

① 利用 KingbaseES 控制管理器停止数据库服务器运行。

② 脱机物理备份数据库 TPC-H。

```
/*-b 参数指定脱机备份,-D 参数指定要备份的数据目录,-n 参数指明备份名,-p 参数指定备份集
路径*/
sys_backup-b-Dc:\Kingbase\ES\V7\data-n offline_backup-P d:\Kingbase\backup
```

(7) 物理联机增量备份

先做一次全系统物理备份作为基准备份,然后修改数据,做增量备份。

① 完全备份,得到完全备份集 week1。

```
/* BACKUP 命令不是 KingbaseES 的命令行工具,需要在查询分析器中执行。*/
BACKUP NAME week1 TYPE FULL FILEPATH 'd:\Kingbase\archive';
```

② 插入数据,修改数据库目录。

```
INSERT INTO Sales.Customer(custkey,name,address,nationkey,phone,acctbal,mktsegment,comment) VALUES(999888,'李四',null,null,null,null,null,null);
```

③ 联机增量备份(其基础备份为 TPC-H_base1),得到增量备份集 diff1。

```
/* FILEPATH 指定增量备份数据存放的文件夹*/
BACKUP NAMEtpch_diff1 FILEPATH 'd:\kingbase\archive\' TYPE DIFFERENTIAL INCREMENT;
```

(8) 物理脱机增量备份

停止数据库服务器运行,然后脱机进行全系统的物理增量备份数据库 TPC-H。

① 利用 KingbaseES 的控制管理器停止数据库服务器。

② 进行脱机完全备份,如果没有停止服务器,就进行脱机完全备份,会报错。

```
/*-b 参数指定脱机备份,-D 参数指定要备份的数据目录,-n 参数指明备份名,-p 参数指定备份集路径*/
sys_backup-b-D c:\Kingbase\ES\V7\data-n offline_backup_full-P d:\Kingbase\backup
```

③ 利用 KingbaseES 控制管理器启动服务器,连接数据库后增加数据

```
CREATE TABLE Sales.Tab(col1 INT,col2 TEXT);
INSERT INTO Sales.Tab VALUES(1,'one');
```

④ 利用 KingbaseES 控制管理器停止数据库服务器。

⑤ 进行脱机差异增量备份,该增量备份的基础备份是第二步的完全备份或差异增量备份。

```
sys_backup-b-n offline_backup_increment2-D c:\Kingbase\ES\V7\data-P d:\Kingbase\backup-t differential
```

实验总结:

5. 思考题

请阅读所用的数据库系统联机帮助,设计一个物理脱机增量备份方案。

实验 10.3　数据库恢复实验

1. 实验目的

掌握数据库逻辑恢复和物理恢复的方法。

2. 实验内容和要求

设计数据库恢复策略，实现数据库恢复，包括数据库逻辑恢复、物理恢复、增量恢复和完全恢复等。数据库恢复也称数据库还原。

3. 实验重点和难点

实验重点：逻辑恢复，物理恢复。

实验难点：增量恢复。

4. 实验报告示例

实 验 报 告				
题目：数据恢复实验	姓名		日期	

"逻辑还原"是指利用备份把数据库从一个快照转化到另一个快照的过程（数据库在某个时刻的一个状态称为一个快照）。

KingbaseES 逻辑还原是将备份文件中的数据库对象和数据还原到指定数据库。逻辑还原时，若不指定还原对象，则对备份文件中的所有备份对象进行还原。逻辑还原方式可以有三种选择：

- 还原整个备份文件。
- 还原指定对象（表、索引、存储过程、触发器）。
- 根据对象列表文件还原。

KingbaseES 提供了图形界面的逻辑备份还原工具 javatools.bat JDump，并提供了命令行工具 sys_restore 和 imp。sys_restore 是 KingbaseES 专有的逻辑备份工具，而 imp 是兼容 Oracle 的逻辑还原工具。imp 的使用依赖于配置的 Kingbase 服务，所以需要在使用前配置 sys_service.conf 文件。

"物理还原"指的是通过备份集和归档日志将数据库转化为一致性状态的过程。还原时可以采用两种策略：

- 只使用增量备份的备份集进行恢复。
- 使用增量备份的备份集+归档日志+尾日志进行恢复。增量备份的脱机恢复对用户来说是透明的。

系统会自动判断是完全备份还是增量备份。如果是增量备份，系统会依次恢复本备份

集到基础备份间的所有备份集。增量恢复成功后,可以是一个不完整恢复,此时将会为恢复后数据目录中的日志文件生成新的时间线,这点和完全备份是一致的。

KingbaseES 提供了图形界面的物理备份还原工具 javatools.bat JBackup,并提供了命令行工具 sys_backup-r,该命令可以使用联机和脱机中的完全备份和增量备份进行脱机恢复。

本实验使用 KingbaseES 命令行工具 sys_restore 和 sys_backup-r 还原数据库。打开命令窗口,设置 KingbaseES 的可执行文件的所在目录为当前路径,完成如下所有实验。本书假设 KingbaseES 安装后的可执行文件所在目录为 c:\Kingbase\ES\V7\bin。

(1) 逻辑还原整个数据库

从二进制备份文件还原整个数据库。

```
/*-d 参数指明数据库名,-clean 参数指明还原之前删除已经存在的数据库对象*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH d:\dumpfile.dmp
```

(2) 逻辑恢复多个表

恢复数据库 TPC-H 的 Orders,Lineitem,PartSupp 等多个表。

```
/*-t 参数指定要恢复的表名*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.Orders d:\
dumpfile.dmp
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.Lineitem d:\
dumpfile.dmp
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.PartSupp d:\
dumpfile.dmp
```

(3) 逻辑恢复指定模式

恢复 TPCH 数据库的 SALES 模式。

```
/*-d 参数指明数据库名,实质与恢复整个数据库的命令一致*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH d:\dumpfile.dmp
```

(4) 利用 COPY FROM 命令恢复数据

从物理文件 lineitem.csv 恢复 TPCH 数据库 SALES 模式中的 Lineitem 表。

```
/*该命令需要在 KingbaseES 的查询分析器中执行。WITH CSV 指明输入数据文件格式;
QUOTE:指明 CSV 文件中所有非空数值两边使用的标记符,默认为双引号。*/
COPY Sales.Lineitem FROM 'd:\lineitem.csv' WITH CSV QUOTE '"';
```

说明:执行 COPY FROM 命令时,DBMS 进行数据完整性检查。如果要恢复的表已存有数据,可能由于数据重复引起完整性检查失败,从而导致 COPY FROM 命令执行失败。解决的办法之一是,如确认表中已有数据已失效,可以先执行 TRUNCATE Sales.Lineitem 命令清空该表中的数据,然后执行上述 COPY FROM 命令。

(5) 全系统恢复物理数据库

① 利用联机和脱机中的完全备份进行全系统恢复物理数据库 TPCH。

使用命令行工具 sys_backup,全系统恢复物理数据库 TPCH 到新路径。

```
/* -r 指定使用物理全系统恢复功能;-P 指明完全备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径。 */
sys_backup-r-P d:\Kingbase\archive\week1-N c:\Kingbase\data
```

说明:如果目的路径是备份前数据所在路径,则要删除该路径之后再恢复,否则由于该路径下有数据而非空导致恢复失败;如果目的路径是一个新路径,则要修改 KingbaseES 安装路径中 config 目录下 instance.conf 文件中的 data_dir 和 log_dir 为该新路径,以使 KingbaseES 启动时从该新路径中装载数据库数据。

② 利用联机和脱机中的增量备份进行全系统恢复物理数据库 TPCH。

使用命令行工具 sys_backup,全系统恢复物理数据库 TPCH 到新路径。

```
/* -r 指定使用物理全系统恢复功能;-P 指明增量备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径。 */
sys_backup-r-P d:\Kingbase\archive\tpch_diff2-N c:\Kingbase\data1
```

说明:该命令实际上与利用完全备份集进行恢复是一样的,只是-P 参数指明的是最后一次增量备份集所在的路径。系统先自动找到增量备份对应的最后一次完全备份做全系统物理数据库恢复,然后按照增量备份集的时间顺序依次恢复各个增量备份集。

③ 使用归档的日志和尾日志文件,恢复备份名为 ONLINE_B2 的备份到新位置

```
/* -P 指明备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径;-A: 指明归档日志保存的位置;-D: 恢复时,是尾日志所在的数据目录的路径值。尾日志通常在 KingbaseES 系统运行的数据目录中,所以在基于归档日志的恢复中,需要指定这个目录 */
sys_backup-r-P d:\Kingbase\archive\week2-N c:\Kingbase\recover\data-A d:\Kingbase\archive
-D c:\Kingbase\ES\V7\data
```

实验总结:

5. 思考题

请阅读所用的数据库系统联机帮助,设计一个物理脱机增量恢复方案。

实验 11 并发控制实验

并发控制为 1 个实验项目(参见表 14),为选修的设计性实验项目。

表 14 "并发控制"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
并发控制实验	选修	适中	2	第 11 章

1. 实验目的

掌握数据库并发控制的基本原理及其应用方法。

2. 实验内容和要求

验证并发操作带来的数据不一致性问题,包括丢失修改、不可重复读和读"脏"数据等情况。要求通过取消查询分析器的自动提交功能,创建两个不同的用户,分别登录查询分析器,同时打开两个客户端;通过 SQL 语言设计具体例子展示不同的封锁级别的应用场景,验证各种封锁级别的并发控制效果,以进一步理解封锁技术是如何解决事务并发导致的问题。

3. 实验重点和难点

实验重点:并发操作带来的数据不一致性问题。

实验难点:设计具体的例子演示各种封锁级别。

4. 实验报告示例

实 验 报 告				
题目:并发控制实验	姓名		日期	

DBMS 通常提供四级隔离级别:读未提交(read uncommitted)、读已提交(read committed)、可重复读(repeatable read)、可串行化(serializable)。

四级隔离级别与三级封锁协议大致对应关系如下:

- 读未提交隔离级别提供最大的事务并发度,但仅能避免丢失修改,对应一级封锁协议。
- 读已提交隔离级别提供的事务并发度减弱,能够避免丢失修改和脏读,对应二级封锁协议。
- 可重复读隔离级别提供的事务并发度进一步减弱,能够避免丢失修改、脏读和不可重复读,对应增强的二级封锁协议。

• 可串行化隔离级别提供最小的事务并发度，能够避免所有的事务并发控制问题，对应三级封锁协议。

read commited 是 KingbaseES 默认的事务隔离级别。运行在该隔离级别的事务中，查询语句只能看到该查询开始执行之前提交的数据，而不会看到任何未提交的数据或查询执行期间并发的其他事务提交的数据，但是该查询可以看到本事务中查询之前执行的数据更新。

serializable 提供了更加严格的事务隔离级别。在该事务隔离级别下，事务好像没有并发，是串行执行的。运行在 serializable 隔离级别的事务中，查询语句只能看到该事务开始执行之前提交的数据，而不会看到任何未提交的数据或事务执行期间并发的其他事务提交的数据。

repeatable read 向上兼容 serializable。read uncommitted 向上兼容 read commited。

(1) 实验准备

① 数据准备：给 Sales 模式下的 PartSupp 表增加一条记录。

```
/*插入零件供应记录 */
INSERT INTO Sales.PartSupp(partkey,suppkey,availqty,supplycost,comment)
VALUES(1,1,0,0.0,null);
/*设置指定零件供应记录的可用数量为 0 */
UPDATE Sales.PartSupp
SET availqty=0
WHERE partkey=1 AND suppkey =1;
```

② 创建两个超级用户：SYSTEM1 和 SYSTEM2。

```
CREATE USER SYSTEM1 WITH SUPERUSER PASSWORD 'MANAGER';
CREATE USER SYSTEM2 WITH SUPERUSER PASSWORD 'MANAGER';
```

③ 分别以 SYSTEM1 和 SYSTEM2 两个用户登录查询分析器。

假设 SYSTEM1 用户登录打开的查询分析称为客户端 A，SYSTEM2 用户登录打开的查询分析器称为客户端 B。设置客户端 A 和客户端 B 的当前数据库为 TPCH。取消两个查询分析器的自动提交事务的功能，改为显式事务提交，即需要执行 COMMIT 命令提交事务。

```
SHOW TRANSACTION ISOLATION LEVEL;
```

(2) 在“读已提交”隔离级别下，验证丢失修改的情况。

① 查看 Sales 模式下 PartSupp 表的 availqty 值为 0。

分别在客户端 A 和客户端 B 执行如下 SQL 语句。

```
/*查看零件供应记录 */
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

② 在客户端 A 执行如下 SQL 语句修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty =availqty-5
WHERE partkey=1 AND suppkey=1;
/*查看 PartSupp 表的 availqty 值为修改后的值*/
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

③ 在客户端 B 执行如下 SQL 语句也修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty =availqty-5
WHERE partkey=1 AND suppkey=1;
/*查看 PartSupp 表的 availqty 值为修改后的值*/
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

④ 在客户端 A 执行如下 SQL 语句提交事务：

```
COMMIT;
```

⑤ 在客户端 B 执行如下 SQL 语句提交事务：

```
COMMIT;
```

⑥ 在客户端 A 和 B 分别执行如下 SQL 语句，查看 PartSupp 表的 availqty 值。

```
/*看到修改后且已提交的 availqty 值*/
SELECT * FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

上述操作过程可以用下面的表格示意表达：

T_A 终端 A 提交的事务	T_B 终端 B 提交的事务
Read(availqty)=22	
	Read(availqty)=22
Update(availqty)=availqty-5	
Read(availqty)=17	
	Update(availqty) = availqty-5
	等待...
Commit	
	Update(availqty) = availqty-5
	执行完成
	Read(availqty)=12
	Commit;

(3) 在“读已提交”隔离级别下，验证避免脏读的情况

① 查看 Sales 模式下 PartSupp 表的 availqty 值为 0。

分别在客户端 A 和客户端 B 执行如下 SQL 语句。

```
/* 查看零件供应记录 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

② 在客户端 A 执行如下 SQL 语句,修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty = 22 WHERE partkey = 1 AND suppkey = 1;
/* 查看 PartSupp 表的 availqty 值为修改后的值 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

③ 在客户端 B 执行如下 SQL 语句,查看 PartSupp 表的 availqty 值。

```
/* 看不到 A 客户端修改后但还未提交的 availqty 值 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

④ 在客户端 A 执行如下 SQL 语句提交事务:

```
COMMIT;
```

⑤ 在客户端 B 执行如下 SQL 语句,查看 PartSupp 表的 availqty 值:

```
/* 看到 A 客户端修改后且已提交的 availqty 值 */
SELECT * FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

上述操作过程可以用下面的表格示意表达:

T_A 终端 A 提交的事务	T_B 终端 B 提交的事务
Read(availqty) = 0	
	Read(availqty) = 0
update(availqty) = 22	
Read(availqty) = 22	
	Read(availqty) = 0
Commit	
	Read(availqty) = 22

(4) 在“读已提交”隔离级别下,验证出现不可重复读的现象

① 客户端 A 执行如下 SQL 语句,读取指定零件的 availqty 值。

```
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

② 客户端 B 执行 SQL 语句,修改指定零件的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty = 33 WHERE partkey = 1 AND suppkey = 1;
```

③ 客户端 A 再次读取指定零件的 availqty 值。

```
/* 发现这次读取的值与上次读取的值一样,因为客户端 B 还没有提交 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

④ 客户端 B 提交事务。

```
COMMIT;
```

⑤ 客户端 A 再次读取指定零件的 availqty 值。

```
/* 发现这次读取的值与上次读取的值不一样了,发生了不可重复现象 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

⑥ 客户端 A 提交事务。

```
COMMIT;
```

(5) 在“读已提交”隔离级别下,验证出现“幻影”的现象

① 客户端 A 执行如下 SQL 语句,读取指定供应商的零件供应记录。

```
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

② 客户端 B 执行 SQL 语句,插入指定供应商新的零件供应记录。

```
INSERT INTO Sales.PartSupp(partkey, suppkey, availqty, supplycost, comment) VALUES(2, 1, 0,
0.0, null);
```

③ 客户端 A 再次读取指定供应商的零件供应记录。

```
/* 发现这次读取的值与上次读取的值一样,因为客户端 B 还没有提交 */
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

④ 客户端 B 提交事务。

```
COMMIT;
```

⑤ 客户端 A 再次读取指定供应商的零件供应记录。

```
/* 发现这次读取的值与上次读取的值不一样了,发生了“幻影”现象 */
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

⑥ 客户端 A 提交事务。

```
COMMIT;
```

(6) 在“可串行化”隔离级别下,验证出现“幻影”的现象

修改 KingbaseES 的 data 文件下的 kingbase.conf 文件,设置 default_transaction_isolation 为 'serializable',重新启动数据库,然后重新执行本实验中(2)、(3)和(4)等三项任务。在“可串行化”隔离级别下,避免脏读、避免不可重复读和避免“幻影”。

实验总结:

不可重复读分为三种情况,一是由于修改导致不可重复读;二是由于插入新记录导致不可重复读,第二次读取时会多出一些新记录;三是由于删除记录导致不可重复读,即第二次读取时有些旧记录“消失”了。后面两种情况称为“幻影”现象。

5. 思考题

试分析在“读已提交”隔离级别下,设计一个应用程序以避免“不可重复读”和“幻影”。

五 实验考核标准和评价方式

1. 实验考核标准

数据库课程实验考核内容主要分为:实验过程、实验代码和实验报告三部分(实验考核评分标准表参见表 15)。实验过程考核主要考查学生实验课程出勤、实验态度、实验任务完成情况、实验过程中分析问题和解决问题的能力等方面;实验代码考核主要是考查学生编写的 SQL 代码规范性、正确性和代码质量等方面;实验报告考核主要是考查实验报告内容完整性、正确性和规范性等方面。

数据库课程实验通常由教师选择实验手册中的若干个小实验组成,实验成绩通常占课程总成绩的 30%~50%,实验过程、实验代码和实验报告可分别占三分之一的分数。考核标准以实验成绩 100 分计算,最终实验成绩可以根据其占课程总成绩的百分比折算。

表 15 实验考核评分标准表

考核内容	评分项	评分标准	分数	备注
1. 实验过程	1.1 实验课程出勤	全勤 缺勤 2 次以下 缺勤 3 次以上	10 8 6	
	1.2 实验态度	积极认真 一般 态度不端正	10 7 5	
	1.3 实验任务完成情况	全部完成 缺项 1 次 缺项 2 次以上	10 8 6	
	1.4 分析问题和解决问题的能力	良好 一般 较差	10 7 6	

续表

考核内容	评分项	评分标准	分数	备注
2. 实验代码	2.1 代码规范性	符合 SQL 编程规范 基本规范 不够规范	10 8 6	
	2.2 代码正确性	运行结果全部正确 错误 2 项以下 错误 3 项以上	10 7 5	
	2.3 代码质量	质量较高 一般 质量较差	10 7 5	
3. 实验报告	3.1 内容完整性	内容完整不缺项 缺 1 项 缺 2 项以上	10 8 5	
	3.2 内容正确性	内容全部正确 错误 1 项 错误 2 项以上	10 8 6	
	3.3 规范性	内容符合实验报告撰写规范 基本规范,字体格式等不统一 不规范	10 8 5	

表 15 中实验考核标准可以针对每个小实验评定分数,然后加权求和得到总体实验成绩;也可以针对所有实验总体完成情况一次性评定分数。

2. 实验评价方式

实验评价从不同角度来看有多种方式。例如,

① **教师评价**:教师是传统的最主要的评价主体。实验课教师全程跟踪和监督每个学生的实验情况,依据实验考核标准,作出详细的评价。

② **学生互相评价**:学生互相之间是很了解各自试验情况的。通过学生之间互相匿名评价,引入监督机制,可以更好地促进实验成绩公平公正的评价。

③ **学生演示汇报和答辩**:在大作业实验结束阶段,每个小组进行 60 分钟的报告和答辩,

制作幻灯片，讲解设计方案，演示系统运行，汇报分工与合作情况。回答老师和同学们的提问。根据整个小组实验情况进行评价，给出总体实验成绩，然后根据小组中每个成员的完成情况，评价每个小组成员的实验成绩。

上述各种评价方式可以按需组合，形成各具特色的实验评价方式。