# 实验2: SQL的数据操作实验

1. **实验目的：**
2. 熟练掌握交互式的SQL语言的各种查询语句。
3. 掌握索引的创建和使用。
4. 掌握查看SQL语句的查询执行计划，并能够基于查询计划对查询进行调优。

**二、实验要求**

1. 熟悉SQL查询语句，包括单表查询、多表查询、嵌套查询、分组聚合查询、排序等。写出各查询语句，分析可能的查询结果。
2. 针对多值属性，创建索引，分析可能的结果。
3. 针对涉及多表的查询，至少创建三个以上的视图，之后，基于视图进行查询，分析查询结果。
4. 基于视图插入和删除数据，分析结果。

**三、实验内容**

（一）SQL基本查询

1. 实验目的

学习使用SQL DML语言进行数据库的查询操作，查看查询结果。

能够根据SQL语句的错误提示，对SQL语句进行修改和调试。

2. 实验内容

基于“学生-课程”数据库模式，按照要求编写完成相应查询功能的SQL语句。如果编写的查询语句数据库无法执行，请根据数据库给出的提示进行调试。

对于执行成功的SQL查询语句，检查查询结果是否和查询语义相符。

3. 实验准备知识

PostgreSQL的SQL语法结构与标准SQL语句的语法结构基本一致。

在实验中完成“学生-课程”数据库Enrolled的创建，其中包括关系表：

Student (Sno, Sname, Ssex, Sage, Sdept)

Course (Cno, Cname, Chour, Ccredit, Cpno)

SC (Sno, Cno, Grade)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验报告2.1 | | | | |
| 题目：SQL基本查询 | 姓名 | 杨世蛟 | 班级 | 1905 |
| 学号 | 20194652 | 时间 | 10.23 |
| 完成以下实验内容，其中标注“\*”的需要自己编写SQL查询语句并写在实验报告中。  1. 导入实验数据  使用SQL脚本导入Enrolled数据库各关系表的数据。  脚本文件包括create table.txt, insert data.txt    导入成功  2. 单表查询，在Enrolled数据库上完成以下查询SQL语句的编写  （1）（单表查询-投影）求选修了课程的学生的学号  1）要求：不消除重复元组，写出其SQL语句      2）要求：消除重复元组，写出其SQL语句    3）以下查询结果与上面哪个结果相同？  select Student.Sno from Student, SC where Student.Sno=SC.Sno  重复的那个  （2）（单表查询-投影）求开设的课程号、课程名、学时和学分  要求：对查询结果的列重新命名为“课号”，“课名”，“学时”和“学分”    （3）（单表查询）求计算机系（‘CS’）和数学系（‘MA’）的学生学号、姓名和年龄      要求：两种以上SQL语句写法  （4）（单表查询）求不是数学系、计算机系的学生的情况  要求：两种以上SQL语句写法      （5）求全体学生的信息  要求：按年龄升序排列    （6）求计算机系年龄在18~20岁之间的学生姓名和年龄  要求：两种以上SQL语句写法      （7）求姓名是以”李”开头的计算机系学生    3 完成多表查询语句   1. 求选修了数据库课程的学生的学号、姓名、成绩      1. 求学号为末尾是5的学生的学号和所选修的课程的学分      1. 求选修“数据库原理”课程且成绩为80分以上的学生的学号、姓名和成绩      1. 求计算机系每个学生的姓名，选修的课程名和成绩        1. 求每一门课程的间接先行课号     （13）查询没有选修任何课程的学生姓名\*    4 完成分组聚合查询语句  （14）求每个学生的学号及平均成绩    （15）求男学生（ssex=‘M’）每一年龄组中超过1人的年龄组及人数     1. 查询每个系的学生中年龄最大的学生姓名、年龄     以下语句是否能够执行？如不能执行请说明原因，并修改为正确的语句。  Select sname, max(sage)  From student  Group by sdept  非聚合属性需要在group by中出现  （17）查询计算机系（CS）选课人数超过5人的课程的课程名\*    5 使用嵌套查询语句完成以下查询  （18）查询没有选修“数据库原理”课程的学生\*    （19）求选修了全部课程的学生的学号    （20）求选修课程记录中，男生里成绩最好和最差的学生的姓名，性别和成绩，以及女生里成绩最好和最差的学生的姓名，性别和成绩 \* | | | | |
| 实验总结：  1 实验中遇到的难点有哪些？  导入数据有问题  2 实验自我评价。  可以 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验报告2.2 | | | | |
| 题目：SQL查询与查询执行计划分析 | 姓名 | 杨世蛟 | 班级 | 1905 |
| 学号 | 20194652 | 时间 | 10.23 |
| 基于TPC-H数据库模式执行或编写查询语句：  （1）连接查询语句  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  1)SQL语句1  Select P.P\_NAME, P.P\_mfgr, P.P\_RETAILPRICE, PS.PS\_SUPPLYCOST  From tpch.part P, tpch.partsupp PS  Where P.P\_RETAILPRICE>PS.PS\_SUPPLYCOST  Limit 200  2) SQL语句2  Select P.P\_NAME, P.P\_mfgr, P.P\_RETAILPRICE, PS.PS\_SUPPLYCOST  From tpch.part P, tpch.partsupp PS  Where P.P\_RETAILPRICE>PS.PS\_SUPPLYCOST  AND P.P\_PARTKEY =PS.PS\_SUPPKEY  Limit 200  比较以上两个查询的执行计划，并说明哪个查询符合要求    第二个，第一个不对应  （2）三表连接的查询计划  查询顾客“Customer#0000000??”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，编写SQL查询语句，并查看执行计划（截图）。其中“??”为自己的学号后两位。      （3）不相关子查询  查询供应商名“Supplier#0000000??”所供应的零件信息。其中“??”为自己的学号后两位。  1）使用IN的不相关子查询实现  2）使用连接操作实现    3）对比以上两个SQL语句，查看系统生成的查询执行计划是否相同。      不同    换了写法还是不同  又换了，还是不同    发现使用in的时候，postgres总是倾向于使用hash join，而非index join  （4）简单相关子查询  查询没有购买过“Brand#13”品牌产品的顾客ID和姓名，编写带有EXISTS的SQL语句，并查看执行计划。    （5）双层嵌套相关子查询  查询至少购买过顾客“Customer#0000000??”所购买过的全部零件的顾客姓名，其中“??”为自己的学号后两位。编写SQL语句，运行并查看执行计划。      （6）子查询派生表  查询订单平均额度超过10000的法国（FRANCE）顾客信息。要求使用FROM子句中的派生表实现。编写SQL语句，并查看执行计划 | | | | |
| 实验总结：  1 实验中遇到的难点有哪些？  无  2 实验自我评价。  还行 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验报告2.3 | | | | |
| 题目： SQL数据更新实验 | 姓名 | 杨世蛟 | 班级 | 1905 |
| 学号 | 20194652 | 时间 | 10.23 |
| （1）INSERT语句  1）完整数据列  插入一条学生记录  insert into Student values('20150020', '苏三', 'M', 20,'CS');  2）部分数据列  插入一条学生记录  insert into Student(Sno, Sname, Ssex) values('20150021', '陈东', 'M');    3）重复执行插入语句  编写SQL语句将本寝室同学信息插入数据库    （2）插入批量数据  创建一个计算机系学生表，把所有计算机系学生插入到新表中  CREATE TABLE CS\_S (  Sno VARCHAR (20),  Sname VARCHAR (30),  Ssex VARCHAR (2),  Sage INTEGER,  Sdept VARCHAR (30),  PRIMARY KEY (Sno))  或者  CREATE TABLE CS\_S AS SELECT \* FROM Student WITH NO DATA;  插入计算机系学生数据  insert into CS\_S  SELECT \*  FROM Student  WHERE Sdept='CS';        （3）修改记录  1）修改部分记录  编写SQL语句，修改C4课程的成绩，将成绩不足60分的修改为60分，记录返回结果    2）修改部分记录-子查询  编写SQL语句，修改信息系学生的“C3”课程成绩，将成绩加10分。    （4）删除记录  编写SQL语句，删除姓名为“李东”的学生信息，及其选修的课程信息。  1）先删除选课记录    2）再删除学生记录 | | | | |
| 实验总结：  1 实验中遇到的难点有哪些？  无  2 实验自我评价。  还行 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验报告2.4 | | | | |
| 题目：基于TPC-H查询语句的OLAP测试 | 姓名 | 杨世蛟 | 班级 | 1905 |
| 学号 | 2019652 | 时间 | 10.23 |
| （1）Q1价格统计报告查询  Q1语句是查询lineItems的一个定价总结报告。在单个表lineitem上查询某个时间段内，对已经付款的、已经运送的等各类商品进行统计，包括业务量的计费、发货、折扣、税、平均价格等信息。  Q1语句的特点是：带有分组、排序、聚集操作并存的单表查询操作。这个查询会导致表上的数据有95%到97%行被读取到。  select  l\_returnflag, --H返回标志  l\_linestatus,  sum(l\_quantity) as sum\_qty, --总的数量  sum(l\_extendedprice) as sum\_base\_price, --聚集函数操作  sum (l\_extendedprice \* (1 - l\_discount)) as sum\_disc\_price,  sum (l\_extendedprice \* (1 - l\_discount) \* (1 + l\_tax)) as sum\_charge,  avg(l\_quantity) as avg\_qty,  avg(l\_extendedprice) as avg\_price,  avg(l\_discount) as avg\_disc,  count(\*) as count\_order --每个分组所包含的行数  from  tpch.lineitem  where  l\_shipdate <= date'1998-06-01' - interval '90' day --时间段是随机生成的  group by --分组操作  l\_returnflag,  l\_linestatus  order by --排序操作  l\_returnflag,  l\_linestatus;  执行以上查询，并记录执行时间和执行计划。    （2）Q2: 最小代价供货商查询  Q2语句查询获得最小代价的供货商。得到给定的区域内，对于指定的零件（某一类型和大小的零件），哪个供应者能以最低的价格供应它，就可以选择哪个供应者来订货。  Q2语句的特点是：带有排序、聚集操作、子查询并存的多表查询操作。查询语句没有从语法上限制返回多少条元组，但是TPC-H标准规定，查询结果只返回前100行（通常依赖于应用程序实现）。  Q2的查询语句如下：  select  s\_acctbal, s\_name, n\_name, p\_partkey, p\_mfgr, s\_address, s\_phone, s\_comment /\*查询供应者的帐户余额、名字、国家、零件的号码、生产者、供应者的地址、电话号码、备注信息 \*/  from  part, supplier, partsupp, nation, region --五表连接  where  p\_partkey = ps\_partkey  and s\_suppkey = ps\_suppkey  and p\_size =[size]--指定大小，在区间[1, 50]内随机选择  and p\_type like '%[TYPE]' --指定类型，在TPC-H标准指定的范围内随机选择  and s\_nationkey = n\_nationkey  and n\_regionkey = r\_regionkey  and r\_name = '[REGION]' --指定地区，在TPC-H标准指定的范围内随机选择  and ps\_supplycost = ( --子查询  select  min(ps\_supplycost) --聚集函数  from  tpch.partsupp, tpch.supplier, tpch.nation, tpch.region --与父查询的表有重叠  where  p\_partkey = ps\_partkey  and s\_suppkey = ps\_suppkey  and s\_nationkey = n\_nationkey  and n\_regionkey = r\_regionkey  and r\_name = '[REGION]'  )  order by --排序  s\_acctbal desc,  n\_name,  s\_name,  p\_partkey;  修改并执行以上查询，并记录执行时间和执行计划。    （3）Q5: 某地区供货商为公司带来的收入查询  Q5语句查询得到通过某个地区零件供货商而获得的收入（收入按sum(l\_extendedprice \* (1 -l\_discount))计算）统计信息。可用于决定在给定的区域是否需要建立一个当地分配中心。  Q5语句的特点是：带有分组、排序、聚集操作、子查询并存的多表连接查询操作。  Q5的查询语句如下：  select  n\_name,  sum(l\_extendedprice \* (1 - l\_discount)) as revenue --聚集操作  from  tpch.customer, tpch.orders, tpch.lineitem, tpch.supplier, tpch.nation, tpch.region --六表连接  where  c\_custkey = o\_custkey  and l\_orderkey = o\_orderkey  and l\_suppkey = s\_suppkey  and c\_nationkey = s\_nationkey  and s\_nationkey = n\_nationkey  and n\_regionkey = r\_regionkey  and r\_name = '[REGION]' --指定地区，在TPC-H标准指定的范围内随机选择  and o\_orderdate >= date '[DATE]' --DATE是从1993年到1997年中随机选择的一年的1月1日  and o\_orderdate < date '[DATE]' + interval '1' year  group by n\_name --按名字分组  order by revenue desc; --按收入降序排序，注意分组和排序子句不同      （4）Q14: 促销效果查询  Q14语句查询获得某一个月的收入中有多大的百分比是来自促销零件。用以监视促销带来的市场反应。  Q14语句的特点是：带有分组、排序、聚集、子查询、左外连接操作并存的查询操作。  Q14的查询语句如下：  select  100.00 \* sum (case  when p\_type like 'PROMO%' --促销零件  then l\_extendedprice\*(1-l\_discount) --某一特定时间的收入  else 0  end) / sum (l\_extendedprice \* (1 - l\_discount)) as promo\_revenue  from  tpch.lineitem, tpch.part  where  l\_partkey = p\_partkey  and l\_shipdate >= date '[DATE]' -- DATE是从1993年到1997年中任一年的任一月的一号 例1995-01-01  and l\_shipdate < date '[DATE]' + interval '1' month;    （5）Q18: 大订单顾客查询  Q18语句查询获得比指定供货量大的供货商信息。可用于决定在订单量大，任务紧急时，验证否有充足的供货商。  Q18语句的特点是：带有分组、排序、聚集、IN子查询操作并存的三表连接操作。查询语句没有从语法上限制返回多少条元组，但是TPC-H标准规定，查询结果只返回前100行（通常依赖于应用程序实现）。  Q18的查询语句如下：  select  c\_name, c\_custkey, o\_orderkey, o\_orderdate, o\_totalprice, --基本信息  sum(l\_quantity) --订货总数  from  tpch.customer, tpch.orders, tpch.lineitem  where  o\_orderkey in ( --带有分组操作的IN子查询  select  l\_orderkey  from  tpch.lineitem  group by  l\_orderkey having  sum(l\_quantity) > [QUANTITY] -- QUANTITY是位于312到315之间的任意值  )  and c\_custkey = o\_custkey  and o\_orderkey = l\_orderkey  group by  c\_name,  c\_custkey,  o\_orderkey,  o\_orderdate,  o\_totalprice  order by  o\_totalprice desc,  o\_orderdate  limit 100;      （6）Q21: 不能按时交货供货商查询  Q21语句查询获得不能及时交货的供货商。  Q21语句的特点是：带有分组、排序、聚集、EXISTS子查询、NOT EXISTS子查询操作并存的四表连接操作。查询语句没有从语法上限制返回多少条元组，但是TPC-H标准规定，查询结果只返回前100行（通常依赖于应用程序实现）。  Q21的查询语句如下：  select  s\_name, count (\*) as numwait  from  tpch.supplier, tpch.lineitem l1, tpch.orders, tpch.nation  where  s\_suppkey = l1.l\_suppkey  and o\_orderkey = l1.l\_orderkey  and o\_orderstatus = 'F'  and l1.l\_receiptdate > l1.l\_commitdate  and exists ( --EXISTS子查询  select  \*  from  tpch.lineitem l2  where  l2.l\_orderkey = l1.l\_orderkey  and l2.l\_suppkey <> l1.l\_suppkey  )  and not exists ( --NOT EXISTS子查询  select  \*  from  tpch.lineitem l3  where  l3.l\_orderkey = l1.l\_orderkey  and l3.l\_suppkey <> l1.l\_suppkey  and l3.l\_receiptdate > l3.l\_commitdate  )  and s\_nationkey = n\_nationkey  and n\_name = '[NATION]' --TPC-H标准定义的任意值  group by  s\_name  order by  numwait desc,  s\_name  limit 100;      对于以上6个查询，完成下表，其中优化部分为选做   |  |  |  |  | | --- | --- | --- | --- | | 查询 | 执行时间ms | 优化方法\* | 优化后执行时间\*ms | | Q1 | 1769.709 | 下推谓词  并行sort  因为group by和order by出现重复，所以排序后再聚合  内部是parallel seq scan并提前进行部分聚合 |  | | Q2 | 223.479ms | parallel index scan  like中用parallel index scan再merge  剩下的表用hash join，并下推谓词。然后再sort  两表进行merge join  子计划是进行index scan，然后取min |  | | Q5 | 205.606ms | 嵌套的hash join  index scan  最外层的hash join是并行的  提前选择 |  | | Q14 | 300.669ms | partial aggregate  parallel hash join  parallel seq scan  parallel hash  最后gather  这个查询就是找满足一定谓词的tuple做计算，对于这种简单的查询，postgres可以放心的用并行加速 |  | | Q18 | 2949.104ms | index scan  parallel seq scan  parallel hash  partial group aggregate  这里分成两个worker，分别做aggregate，同时在内部提前排序。最后再做gather merge 和 finalize aggregate |  | | Q21 | 580.904ms | parallel seq scan  hash join  index scan |  | | 执行查询的机器配置：  CPU:i7 9th gen  内存:16G  PostgreSQL版本：12.8 | | | | | | | | |
| 实验总结：  1 能想到何种方法提高查询的效率？  上面写了 | | | | |