《数据库技术实验》报告

|  |  |  |
| --- | --- | --- |
| **第 九 次实验** | **日期：2022-12-20** | **得分：** |
| **学号：** | **姓名：*Steven*** | **专业：智能科学与技术** |

一、实验目的

1. 了解查询优化的基本原理和方法

2. 了解建立索引优化查询的基本方法

二、实验内容

（一）实验题

1. 在Student\_Course数据库上，增加Student表中的学生数为1000以上、Course表的课程数为100以上，SC表中的选课数为10000以上，自行设计5个以上查询（包括选择查询、连接查询，以及有无索引），查看、分析其执行计划。

2. 在TCP\_H数据库上自行设计5个以上查询（包括选择查询、连接查询，以及有无索引），查看、分析其执行计划。

（二）思考题

1. 查询优化方法主要包括哪两类？它们如何实现查询优化？

三、实验过程及实验结果

1. 在Student\_Course数据库上，增加Student表中的学生数为1000以上、Course表的课程数为100以上，SC表中的选课数为10000以上，自行设计5个以上查询（包括选择查询、连接查询，以及有无索引），查看、分析其执行计划。

(1) 创建数据

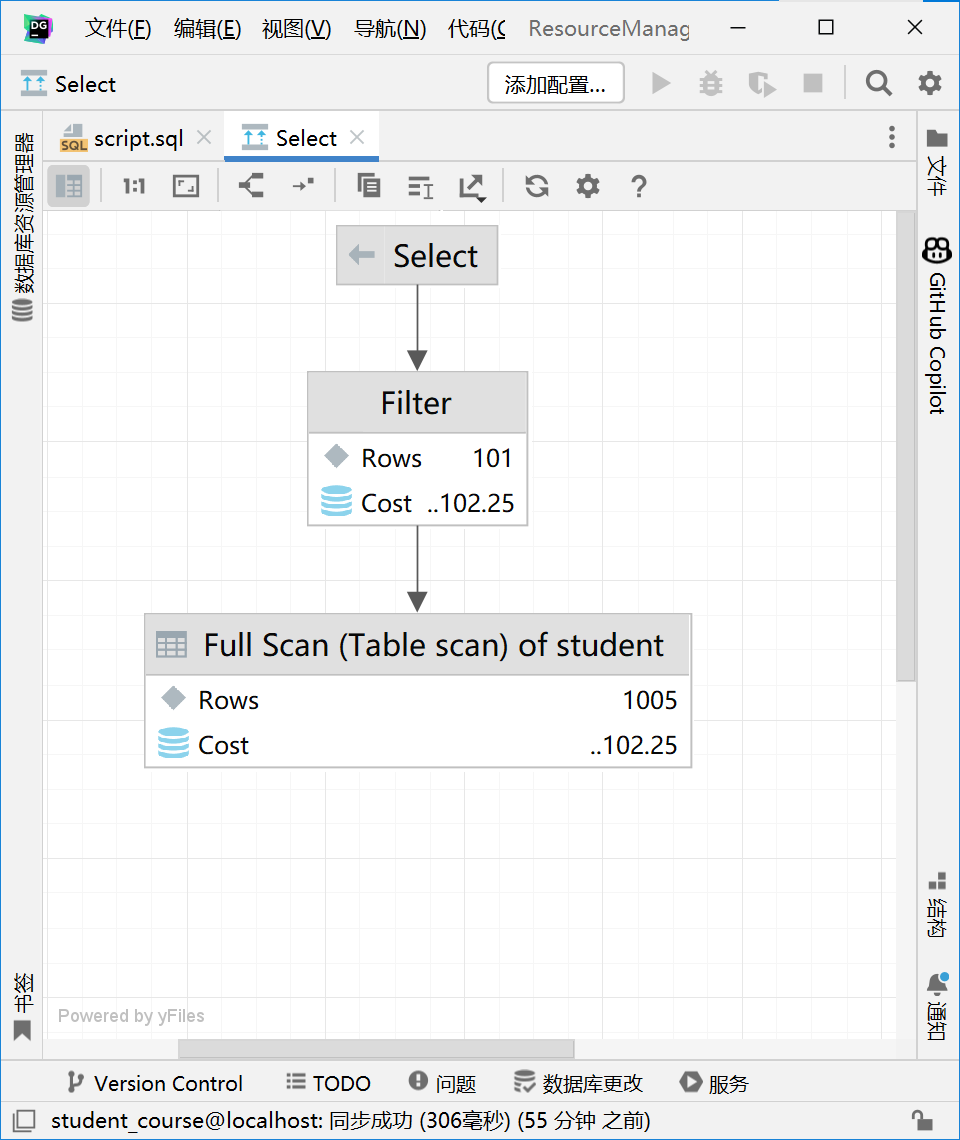
首先要将数据库中的数据添加到1000或10000条及以上，这一步使用如下的存储过程实现。

|  |
| --- |
| CREATE PROCEDURE *`batch\_insert\_student`*() BEGIN  DECLARE i INT;  SET i = 1;  WHILE i <= 1000  DO  INSERT INTO student  SET sno =*concat*(20191060005 + i),  sname=*concat*('学生', i),  ssex=*if*(i % 2 = 0, '男', '女'),  sage=*if*(i % 2 = 0, 20, 21),  sdept='计算机科学与技术';  SET i = i + 1;  end while; END;  CREATE PROCEDURE *`batch\_insert\_course`*() BEGIN  DECLARE i INT;  SET i = 7;  WHILE i <= 100  DO  INSERT INTO course  SET cno=*concat*(i),  cname= *concat*('课程', i - 6),  cpno=*concat*(i - 6),  ccredit=*if*(i % 2 = 0, 2, 3);  SET i = i + 1;  END WHILE; END;  CREATE PROCEDURE *`batch\_insert\_sc`*() BEGIN  DECLARE i INT;  DECLARE j INT;  SET i = 1;  WHILE i <= 1000  DO  SET j = 1;  WHILE j <= 10  DO  INSERT INTO sc  SET sno=*concat*(20191060005 + i),  cno=*concat*(j),  grade=*if*(i % 2 = 0, 60, 70);  SET j = j + 1;  END WHILE;  SET i = i + 1;  END WHILE; END; |

(2) 无索引单表查询的执行计划

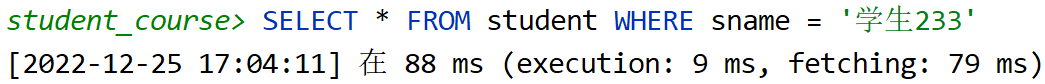
以下列查询语句为例，Student表中并未为sname列建立索引。

|  |
| --- |
| SELECT *\** FROM student WHERE sname = '学生233'; |



（执行计划图示）

执行过程如上图，由于没有索引，该过程需要执行全表扫描，花费时间应当较长。



（fetching用时79ms）

(3) 主码索引单表查询的执行计划

以下列查询语句为例，sno为Student表的主码，包含主码索引。

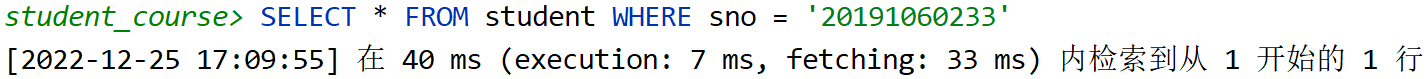
|  |
| --- |
| SELECT *\** FROM student WHERE sno = '20191060233'; |

日程表

描述已自动生成

（执行计划图示）

执行过程如上图所示，由于主码索引都为Unique类型，所以该执行语句仅需找到特定的行即可，Rows为1，查询时间应当很短。



（fetching用时40ms）

(4) 无索引连接查询的执行计划

以下列查询语句为例，限定条件为无索引的sname。

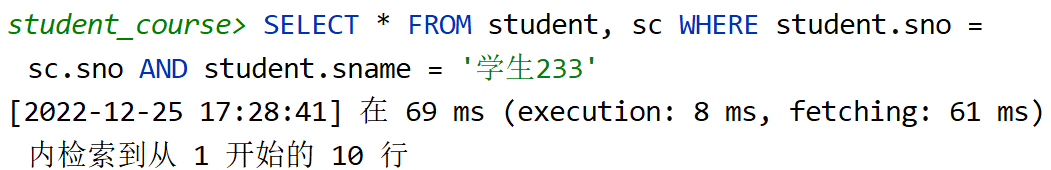
|  |
| --- |
| SELECT *\** FROM student, sc WHERE student.sno = sc.sno AND student.sname = '学生233'; |

图示

描述已自动生成

（执行计划图示）

如图所示，该过程使用嵌套循环完成。对于sc表，由于sno恰好是PRIMARY KEY，因此只需要根据Unique索引检索即可，Rows=10是因为sc中存在10条相关记录。而对于Student表，由于sname没有索引，只能全表查询，也可以看到Cost即时间开销主要都花费在Student表的检索上。



（fetching用时61ms）

(5) 主码索引连接查询的执行计划

以下列查询语句为例，限定条件为主键索引的sno。

|  |
| --- |
| SELECT *\** FROM student, sc WHERE student.sno = sc.sno AND student.sno = '20191060233'; |

图示

描述已自动生成

（执行计划图示）

如图所示，此时执行计划就大不相同了，该过程只需要对sc中的10条数据进行查询即可，因为主键为Unique Index，对其扫描花费的时间将大大降低。

文本

描述已自动生成

（fetching用时27ms）

(5) 无索引+子查询的执行计划

以下列查询语句为例，限定条件为子查询的结果。

|  |
| --- |
| SELECT *\** FROM student WHERE sname = (SELECT sname FROM student WHERE sno = '20191060233'); |

图示

描述已自动生成

（执行计划图示）

如图所示，子查询是sno=’’的形式，这部分也是对PRIMARY KEY的检索，因此Rows为1，Cost为0，几乎没有开销。而对于Student表则因为根据sname检索，没有索引，执行的是全表查询，所以该部分开销很高。

文本

描述已自动生成

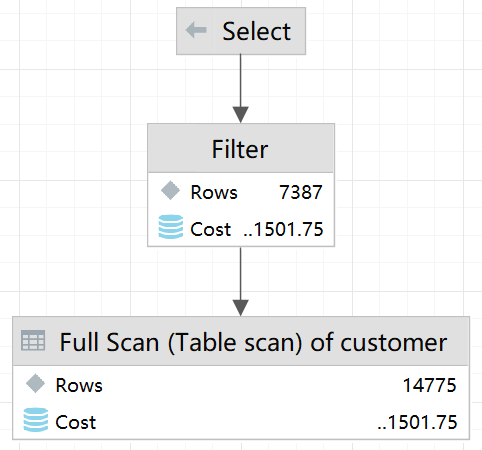
（fetching用时91ms）

2. 在TCP\_H数据库上自行设计5个以上查询（包括选择查询、连接查询，以及有无索引），查看、分析其执行计划。

1. 无索引单表查询的执行计划

以下列查询语句为例，customer表中没有为nationkey建立索引。

|  |
| --- |
| SELECT *\** FROM customer WHERE nationkey <= 200; |



（执行计划图示）

执行过程如上图，由于没有建立索引，该查询需要全表扫描，消耗时间较长。

图形用户界面, 文本

描述已自动生成

（fetching用时58ms）

(2) 主码索引单表查询的执行计划

以下列查询语句为例，nation表中nationkey为主键。

|  |
| --- |
| SELECT *\** FROM nation WHERE nationkey = 100; |

日程表

描述已自动生成

（执行计划图示）

如上图所示，该过程只需要查找到主码为100的一行即可，由于建立了索引，该过程的开销非常小。

图形用户界面, 文本, 应用程序

描述已自动生成

（fetching用时16ms）

(3) 无索引连接查询的执行计划

以下列查询语句为例。

|  |
| --- |
| SELECT *\** FROM supplier, nation; |

图示

描述已自动生成

（执行计划图示）

和1(4)的处理方式类似，只不过两个数据表都需要进行全表扫描，时间开销会很大。

图片包含 文本

描述已自动生成

（fetching用时66ms）

(4) 主码索引连接查询的执行计划

以下列查询语句为例，where子句中条件与主键相关。

|  |
| --- |
| SELECT *\** FROM customer, orders WHERE customer.custkey = orders.custkey AND orders.orderkdy <= 100; |

图形用户界面, 表格

描述已自动生成

（执行计划图示）

对于customer表，只需要使用custkey进行检索，而custkey又是PRIMARY主键，所以该部分时间开销小，而对于orders表来说，WHERE子句为主键的一个数据范围，所以尽管查询策略为Index range scan，但同样时间开销较小，整个执行过程应当较快。

图形用户界面, 文本, 应用程序

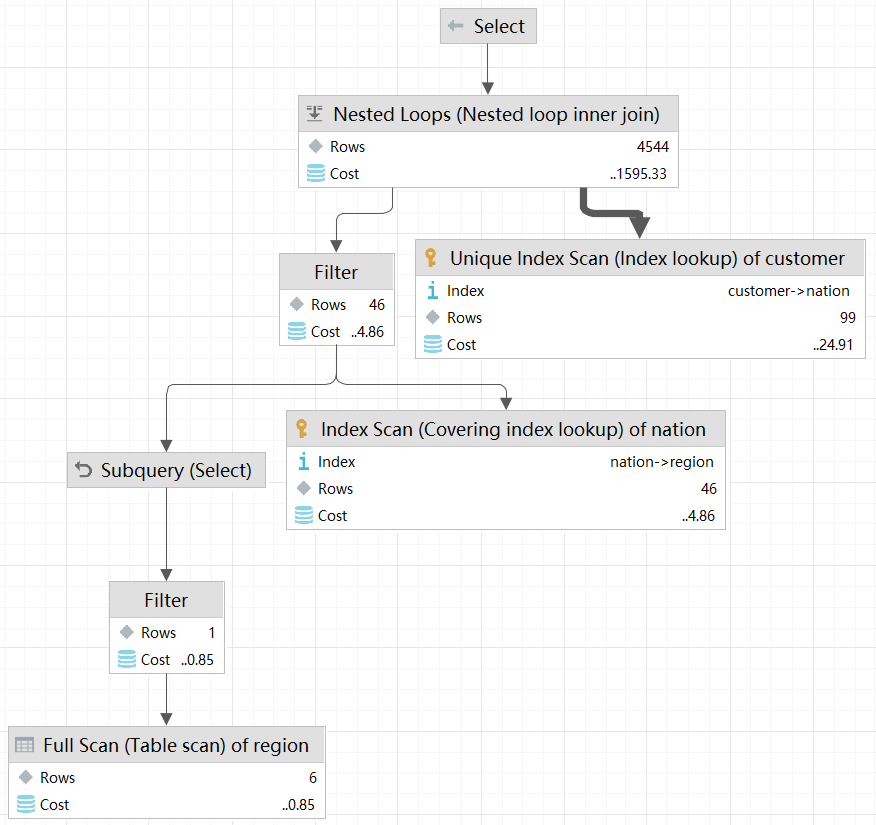
描述已自动生成

（fetching用时20ms）

(5) 无索引+子查询的执行计划

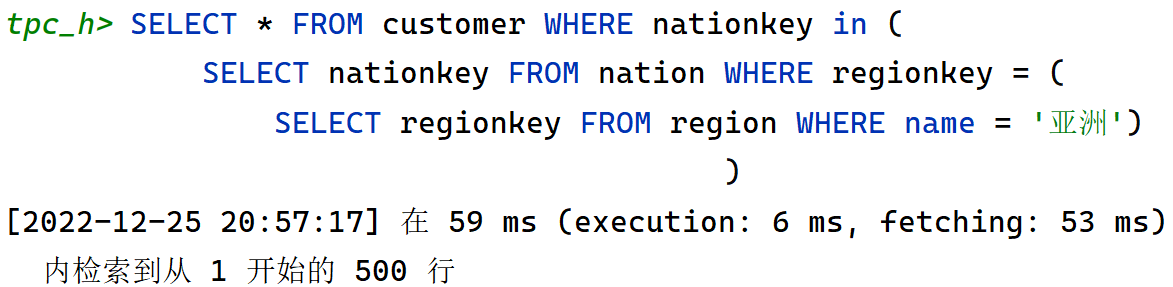
以下列查询语句为例，这是一个多层的子查询结构。

|  |
| --- |
| SELECT *\** FROM customer WHERE nationkey in (SELECT nationkey FROM nation WHERE regionkey = (SELECT regionkey FROM region WHERE name = '亚洲')); |



（执行计划图示）

这棵树的查询结构仍然是嵌套循环，从内到外依次是region表、nation表、customer表。对于region表，由于按照name检索，而name没有索引，所以为全表扫描，返回多个结果。对于nation表，由于按照nationkey检索，有主键索引，但是内层返回多个结果，所以Rows=46而非1，也有一定的时间开销；最后是最外层的customer表，同样是按照主键检索，但由于检索类型为包含，所以检索范围仍然较大，时间开销较长。



（fetching用时53ms）

四、思考题

1. 查询优化方法主要包括哪两类？它们如何实现查询优化？

主要分为代数优化和物理优化。

**代数优化：**通过对关系代数表达式的等价变换来提高查询效率，即换用等价但运算更少的关系代数表达式替代原本的表达式，并不涉及底层的存取路径。

**物理优化：**选择高效合理的操作算法或存取路径，求得优化的查询计划。选择的方法包括(1)基于规则的启发式优化、(2)基于代价估算的优化、(3)两者结合的优化方法。