《数据库原理(1)》实验报告

实验二

姓名：严昕宇 学号：20121802 实验日期：2022.12.08

实验环境：

实验设备：Lenovo Thinkbook16+ 2022

操作系统：Windows 11 22H2

数据库管理系统：MySQL 8.0.31

数据库管理软件：JetBrains DataGrip 2022.3

实验内容：

对 school 数据库做如下查询：

1. 验证在1000万个以上记录时在索引和不索引时的查询时间区别。
2. 查询2011年进校年龄大于20岁的男学生的学号与姓名。
3. 检索刘晓明不学的课程的课程号。
4. 查询计算机学院男生成绩及格、教授开设的课程的课程号、课名、开课教师姓名，按开课教师升序，课程号降序排序。
5. 检索学号比张颖同学大，年龄比张颖同学小的同学学号、姓名。
6. 检索同时选修了“08305001”和“08305002”的学生学号和姓名。
7. 查询每个学生选课情况（包括没有选修课程的学生）。

**实验过程1：**

验证在1000万个以上记录时在索引和不索引时的查询时间区别

① 连接数据库

先连接MySQL数据库，进入到school数据库中。

在school中有一张临时的表test。

② 查看表中的索引

使用show index from 表名;查看表中的索引。

|  |
| --- |
| show index from school.test; |

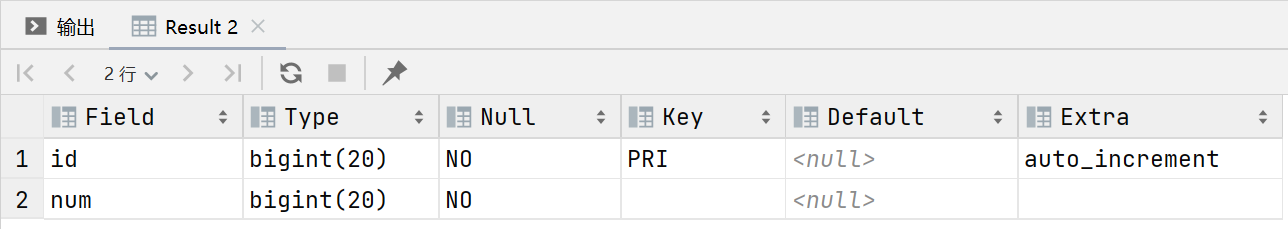
由于test表中，现在还没有任何数据，因此也没有索引，查询结果为空Empty

③ 随机填充数据

为了对比有无索引的效率差距，需要插入大量的数据，此处使用代码，来批量插入数据到test表中。

先使用desc 表名; 查看表的字段。

|  |
| --- |
| desc school.test; |



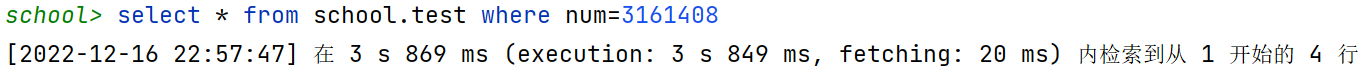
接下来使用Python的pymysql模块，配合随机函数，在数据库中插入随机测试数据。

|  |
| --- |
| *# coding=utf-8* import random from pymysql import connect  def RandomInsert():  conn = connect(  host="localhost",  port=3306,  user="root",  password="\*\*\*\*", *# 密码略* database="school",  charset="utf8"  )  cursor = conn.cursor()  for i in range(10000000):  sql = "insert into test values({id},{i}');".format(id=i, i=random.randint(1, 10000000))  try:  cursor.execute(sql)  conn.commit()  except Exception as error:  print(error)  cursor.close()  conn.close()  if \_\_name\_\_ == '\_\_main\_\_':  RandomInsert() |

④ 无索引时查询

|  |
| --- |
| select *\** from school.test where num=3161408; |

执行时间：



⑤ 创建索引

使用create index 索引名 on 表名(字段名称(长度)); 来创建索引。

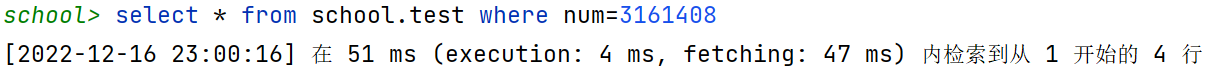
值得注意的是，如果指定的字段类型是字符串，需要指定长度，建议长度与数据表中定义字段时的长度一致。如果字段类型不是字符串，不用声明长度。

|  |
| --- |
| create index stu\_index on school.test(num); |

⑥ 创建索引后查询

创建索引后，执行相同的查询语句。

执行时间：



实验结果1：

通过以上步骤可知，创建索引后的查询时间相较于创建前有一定缩短。而在实际的项目中，数据远比一千万条要多，此时索引带来的效率提升效果会更明显。

当然，不是所有的表都需要建立索引。如果表中数据很少，没有必要建立索引。如果一个表中的数据增删很频繁，不适合索引，因为只要数据发生增减，索引就要重新建立，建立索引是很耗时的。频繁建立索引，反而效率更低了。索引只适合查询操作频繁的表。

**实验过程2：**

查询2011年进校年龄大于20岁的男学生的学号与姓名

|  |
| --- |
| select xh as '姓名', xm as '学号' from school.s where xb = '男'  and csrq < '1991-09-01'; |

实验结果2：

图形用户界面, 应用程序, 表格

描述已自动生成

**实验过程3：**

检索刘晓明不学的课程的课程号

|  |
| --- |
| select distinct kh as '课程号' from school.c where kh not in (select e.kh  from e,  s  where e.xh = s.xh  and s.xm = '刘晓明') |

实验结果3：

表格

描述已自动生成

**实验过程4：**

查询计算机学院男生成绩及格、教授开设的课程的课程号、课名、开课教师姓名，按开课教师升序，课程号降序排序

|  |
| --- |
| select c.kh as '课程号', c.km as '课名', t.xm as '开课教师姓名' from school.c,  school.t,  school.s,  school.e,  school.d where e.xh = s.xh = c.kh  and s.yxh = d.yxh  and d.Mc = '计算机学院'  and s.xb = '男'  and e.zpcj is not null  and e.zpcj >= 60  and t.gh = e.gh  and t.xl = '教授' order by t.xm asc, c.kh desc; |

实验结果4：

图形用户界面, 应用程序

描述已自动生成

**实验过程5：**

检索学号比张颖同学大，年龄比张颖同学小的同学学号、姓名

|  |
| --- |
| select x.xh as '学号', x.xm as '姓名' from school.s as x,  school.s as y where y.xm = '张颖'  and x.xh > y.xh  and x.csrq > y.csrq; |

实验结果5：

图形用户界面, 文本, 应用程序, 表格

描述已自动生成

**实验过程6：**

检索同时选修了“08305001”和“08305002”的学生学号和姓名。

|  |
| --- |
| select distinct s.xh as '学号', s.xm as '姓名' from school.s,  school.e where e.xh in  (select xh  from e  where kh = '08305001'  and xh in  (select xh  from e  where kh = '08305002'))  and s.xh = e.xh; |

实验结果6：

表格

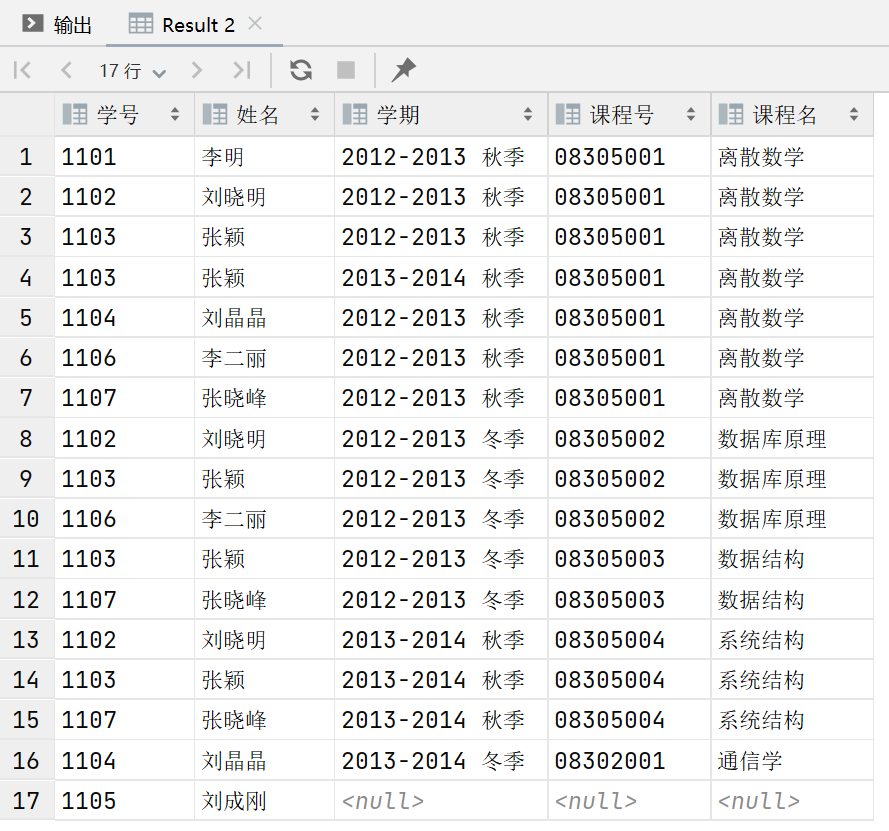
描述已自动生成

**实验过程7：**

查询每个学生选课情况（包括没有选修课程的学生）

|  |
| --- |
| select s.xh as '学号', s.xm as '姓名', e.Xq as '学期',e.kh as '课程号', c.km as '课程名' from school.s  left join e on s.xh = e.xh  left join c on c.kh = e.kh |

实验结果7：



实验体会：

本次实验是我第一次使用结构化查询语言SQL。通过动手实验，我初步掌握了SQL的基础语法规则。与此同时，我也发现相比阅读书本上的SQL语句，在实验中亲自动手使用SQL语句，才是更好的学习方法，能加深我的记忆与理解。

当然，在实验过程中，我也遇到了一些困难。通过上网查询文献和资料，并理解其背后的逻辑，我完成了实验内容。与此同时，通过与其他同学的交流，我发现SQL语句相关问题的解决方案是有多种多样的，正所谓“条条大路通罗马”，每个人由于其不同思路，都有属于自己的独特解决方法。