**数据库实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实 验 人：** | **凌国明** | | **学 号：** | **21307077** | **日 期：** | **2023年10月5日** |
| **院（系）：** | **计算机学院** | | | **专业（班级）：** | **计算机科学与技术** | |
| **联系方式：** |  | | |  |  | |
| **实验题目：** | | **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。

* + - 1. **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。

* + - 1. **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。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat

* + - 1. **实验过程**

1) 创建唯一索引

在零件表的零件编号字段上创建唯一索引。

create unique index Idx\_part\_key on PART(PARTKEY);

注：对于零件名称这种可能会有重复键值的属性是无法创建唯一索引的。

截图如下：



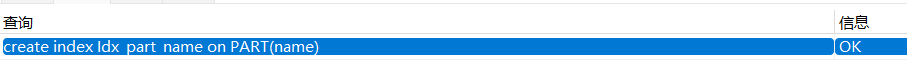
2) 创建函数索引（对某个属性的函数创建索引，称为函数索引）

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。

由于SQL SERVER、MySQL均不支持函数索引，从而将该实验改为“在零件表的零件名称字段上创建索引”

create index Idx\_part\_name on PART(name)

截图如下：

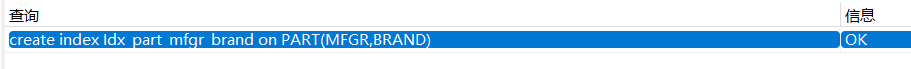


3) 创建复合索引（对两个及两个以上的属性创建索引，称为复合索引）

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 index Idx\_part\_mfgr\_brand on PART(MFGR,BRAND);

截图如下：

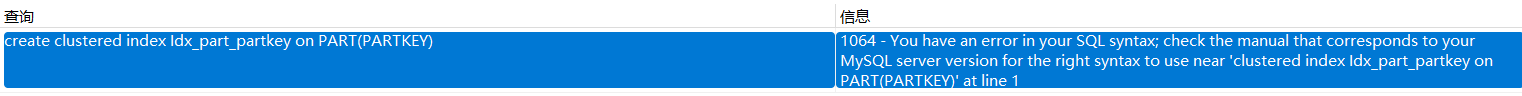


4) 创建聚簇索引

在零件表的partkey字段上创建一个聚簇索引。

create clustered index Idx\_part\_partkey on PART(PARTKEY);

注意：对于 MySql、Sql server，由于主键的列自动存在聚簇索引，因此重复添加聚簇索引会失败！



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(NAME) USING HASH;

截图如下：



6) 修改索引名称

将零件表的索引Idx\_part\_name\_hash修改为Idx\_part\_name\_hash\_new。

ALTER TABLE Part RENAME INDEX Idx\_part\_name\_hash TO Idx\_part\_name\_hash\_new

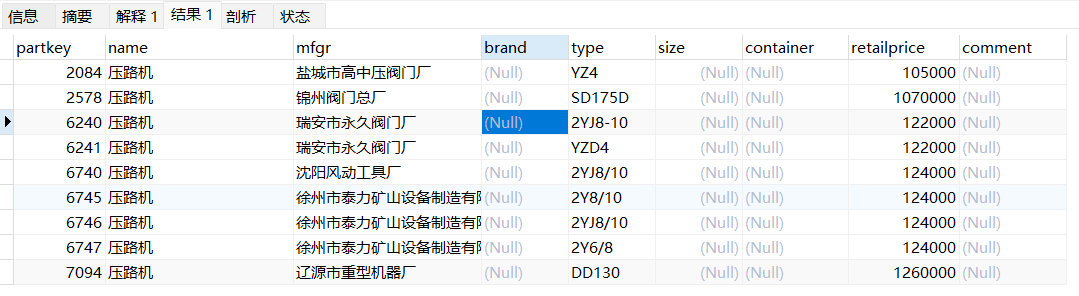
截图如下：

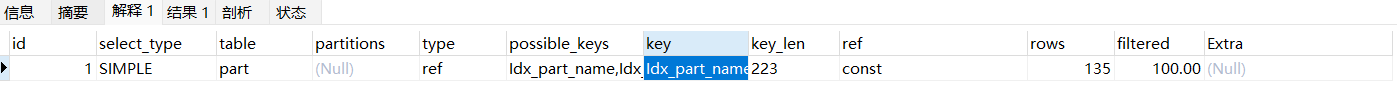


7) 分析某个SQL查询语句执行时是否使用了索引

select \* from part where name='压路机'

截图如下：





可以看到查询时使用了名为Idx\_part\_name的索引

8) 验证索引效率

* 先创建如下函数：

set global log\_bin\_trust\_function\_creators=TRUE;

CREATE FUNCTION TestIndex(p\_partname CHAR(55) ) RETURNS DOUBLE

BEGIN

DECLARE begintime DATETIME(6);

DECLARE endtime DATETIME(6);

DECLARE durationtime DOUBLE;

DECLARE name\_count INTEGER;

SELECT sysdate(6) INTO begintime; /\*记录查询执行的开始时间\*/

SELECT count(\*) FROM Part WHERE name=p\_partname into name\_count;

SELECT sysdate(6) INTO endtime; /\*记录查询执行的结束时间\*/

SELECT TimeDiff(endtime,begintime) into durationtime; /\*计算并返回查询执行时间,时间单位为秒s\*/

return durationtime;

END;

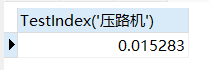
* 然后，无索引查询情况：

/\*查看当零件表Part数据规模比较大, 但无索引时的执行时间\*/

/\*删除零件表Part中所有基于属性 name 建立的索引\*/

SELECT TestIndex('压路机') ;

截图如下：



* 再，有索引查询情况：

先创建索引：

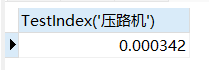
CREATE INDEX part\_name ON Part(name);/\*在零件表的零件名称字段上创建索引\*/

/\*查看零件表Part数据规模比较大, 有索引时的执行时间\*/

再查询：

SELECT TestIndex('压路机') ;

截图如下：



* 结论：

可以看出使用索引在查询时的效率大大提升。

* + - 1. **与实验结果相关的文件**

无

* + - 1. **实验总结**

通过这次实验，我深刻理解了索引在数据库中的重要性以及如何有效地创建、修改和删除索引以提高查询性能。

首先，我学会了索引的设计原则和技巧。在实验中，我创建了唯一索引、函数索引和复合索引，根据数据库模式和特定应用需求选择了适当的索引类型。这使我更好地了解了何时以及如何使用不同类型的索引来优化查询。

其次，我学会了如何使用EXPLAIN命令来分析SQL查询的执行计划，以确定是否使用了所创建的索引。这个步骤对于验证索引的有效性至关重要。通过观察查询计划，我能够分析索引的使用情况，并了解哪些查询受益于索引，以及为什么。

最具挑战性的部分是设计SQL查询来验证索引的有效性。需要确保查询测试涵盖了不同的查询模式，以便全面评估索引的性能提升。

总的来说，这次实验帮助我掌握了索引的创建、分析和优化技巧。我现在更加了解如何为数据库表选择正确的索引，并且知道如何使用工具来评估索引的性能。这将在未来的数据库设计和优化工作中大有裨益。