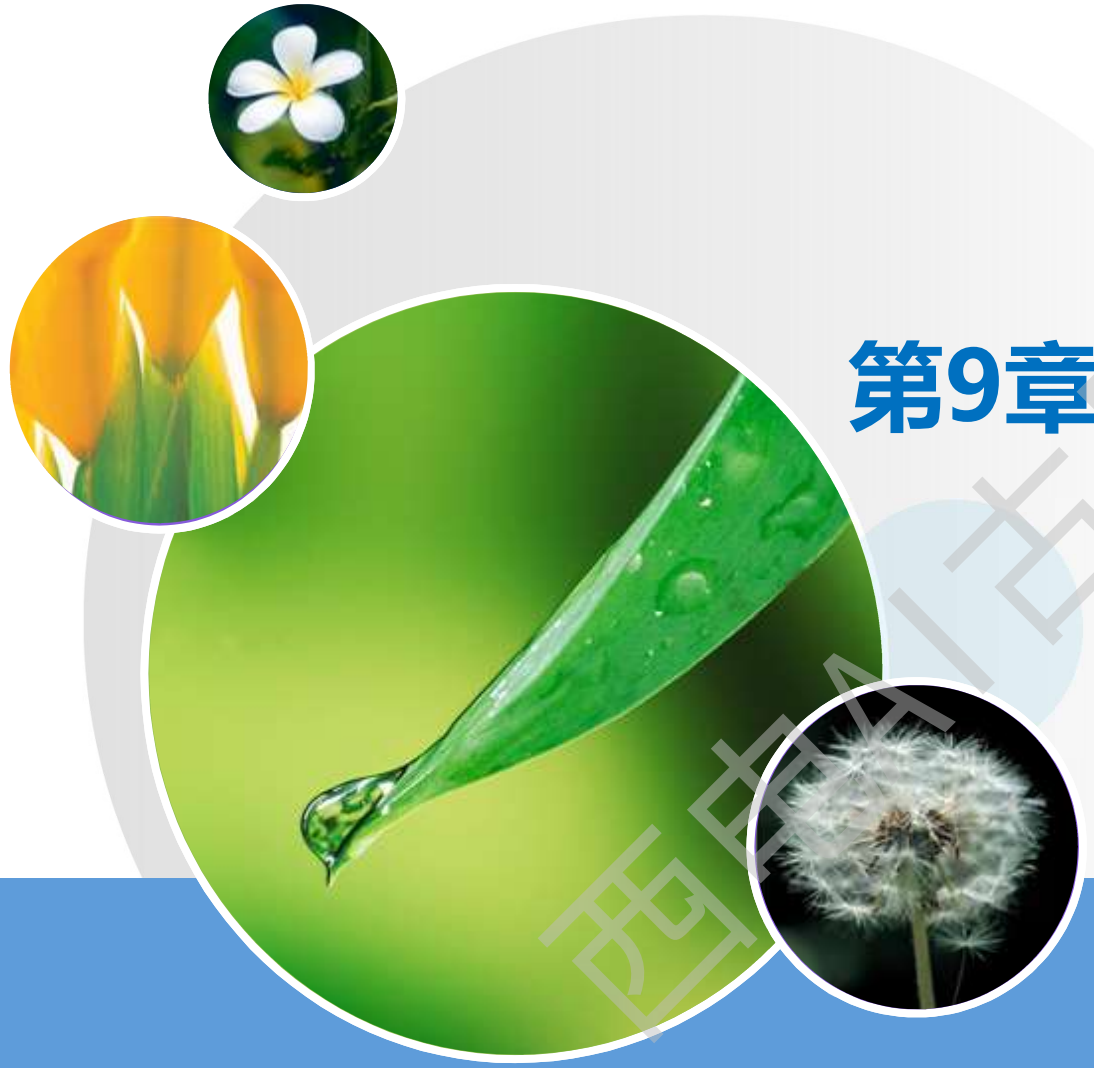

第9章 MySQL索引

人工智能学院
古 晶

西安电子科技大学

第9章MySQL索引

- 9.1 索引
- 9.2 索引的定义与管理
- 9.3 索引的设计原则和注意事项
- 9.4 知识点小结
- 本章实验

第9章MySQL索引

- 索引是一种特殊的数据库结构，其作用相当于一本书的目录，可以用来**快速查询数据库表中的特定记录**。
- 索引是**提高数据库性能**的重要方式。
- 本章将介绍索引的含义和作用、索引定义的原则和创建索引的方法以及查看索引和删除索引的方法。

第9章MySQL索引

- 9.1 索引
- 9.2 索引的定义与管理
- 9.3 索引的设计原则和注意事项
- 9.4 知识点小结
- 本章实验

9.1 索引概述

- **目的**：**优化数据库的查询速度**。
- 所有MySQL列类型都可以被索引，对相关**列**使用索引是提高select操作性能的最佳途径。
- 不同的存储引擎定义了每一个表的最大索引数量和最大索引长度，所有存储引擎对每个表至少支持16个索引，总索引长度至少为256字节。
- **索引分为**哈希索引、B树索引。
- InnoDB和MyISAM支持B树索引，MyISAM支持哈希索引、B树索引但默认的是哈希索引。

9.1 索引的优点与缺点

- 索引的**优点**：

（1）通过创建唯一性索引，可以保证数据库表中每一行数据的**唯一性**。

（2）可以大大加快数据的**检索速度**，这也是创建索引的最主要的原因。

（3）可以加速表和表之间的**连接**，特别是在实现数据的参考完整性方面特别有意义。

（4）在使用分组和排序子句进行数据检索时，同样可以显著**减少查询中分组和排序的时间**。

（5）通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

9.1 索引的优点与缺点

- 索引的**缺点**：

（1）创建索引和维护索引要**耗费时间**，这种时间随着数据量的增加而增加。

（2）索引需要占**物理空间**，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

（3）当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就**降低了数据的维护速度**。

9.1 索引的特征

- 索引的**特征**：
  - **唯一性索引**：
    - 保证在索引列中的全部数据是唯一的，不会包含冗余数据。
    - 当在表中创建主键约束或者唯一性键约束时，MySQL自动创建一个唯一性索引；
  - **复合索引**：
    - 一个索引创建在两个列或者多个列上。
    - 原则：**长度不能太长，不能跨表建立，认真排列列的顺序**

9.1 索引的分类

- **普通索引**
  - 不附加任何限制条件
  - 可以创建在任何数据类型中，其值是否唯一和非空由字段本身的完整性约束条件决定。
  - 建立索引以后，查询时可以通过**索引**进行查询。
- **唯一性索引**
  - 使用**UNIQUE**参数可以设置索引为唯一性索引。
  - 在创建唯一性索引时，限制该索引的值必须是唯一的。
  - 通过唯一性索引，可以更快速地确定某条记录。**主键**就是一种特殊唯一性索引。

9.1 索引的分类

口 全文索引

- 使用**FULLTEXT**参数可以设置索引为全文索引。
- 全文索引**只能创建在CHAR、VARCHAR或TEXT类型的字段上**。
- 查询数据量**较大**的字符串类型的字段时，使用全文索引可以提高查询速度。
- 在默认情况下，全文索引的搜索执行方式**不区分大小写**。但索引的列使用二进制排序后，可以执行区分大小写的全文索引。

9.1 索引的分类

口 单列索引

- 在表中的**单个字段**上创建索引。
- 单列索引只根据该字段进行索引。
- 单列索引可以是普通索引，也可以是唯一性索引，还可以是全文索引。只要保证该索引只对应一个字段即可。

口 多列索引

- 多列索引是在表的多个字段上创建一个索引。
- 该索引指向创建时对应的多个字段，可以通过这几个字段进行查询。但是，只有查询条件中使用了这些字段中第一个字段时，索引才会被使用。

9.1 索引的分类

口 空间索引

- 使用**SPATIAL**参数可以设置索引为空间索引。
- 空间索引只能建立在空间数据类型上，这样可以提高系统获取空间数据的效率。
- MySQL中的空间数据类型包括GEOMETRY和POINT、LINESTRING和POLYGON等。
- 目前只有MyISAM存储引擎支持空间检索，而且索引的字段不能为空值。

第9章MySQL索引

- 9.1 索引
- **9.2 索引的定义与管理**
- 9.3 索引的设计原则和注意事项
- 9.4 知识点小结
- 本章实验

9.2 创建索引

- 创建索引是指在某个表的一列或多列上建立一个索引。
- **创建**索引方法：
  - **直接创建索引**
    1)在**创建表**的时候创建索引。
    2)在**已存在的表**上创建索引
    3)使用**alter table**语句来创建索引。
  - **间接创建索引**
    - **例如**：在表中定义**主键**约束或者**唯一**性键约束时，同时也创建了索引。

9.2 在创建表的时候创建索引

- **语法格式**:

```
CREATE TABLE tbl_name(
字段名称 字段类型 [完整性约束条件],
…,
[UNIQUE|FULLTEXT|SPATIAL] INDEX|KEY [索引名称](字段名称[(长度)] [ASC|DESC])
);
```

- **INDEX**或**KEY**参数用来指定字段为索引，索引名参数是用来指定要创建索引的名称。字段名称参数用来指定索引索要关联的字段的名称，”长度”参数用来指定索引的长度，ASC用来指定为升序，DESC用来指定为降序。

9.2 创建索引示例

- **示例**：创建**普通索引**

```
CREATE TABLE t_test4(
    id TINYINT UNSIGNED,
    username VARCHAR(20),
    INDEX in_id(id),
    KEY in_username(username)
    );
```

查看多列索引是否成功:
SHOW CREATE TABLE t_test4;

- **示例**：创建一个**唯一索引**

```
CREATE TABLE t_test5(
    id TINYINT UNSIGNED AUTO_INCREMENT KEY,
    username VARCHAR(20) NOT NULL UNIQUE,
    card CHAR(18) NOT NULL,
    UNIQUE KEY uni_card(card)
    );
```

注：一个表可有多个唯一性索引，但只能有一个主键索引

9.2 创建索引示例

- **示例**：创建一个**全文索引**

```
CREATE TABLE t_test6(
    id TINYINT UNSIGNED AUTO_INCREMENT KEY,
    username VARCHAR(20) NOT NULL UNIQUE,
    userDesc VARCHAR(20) NOT NULL,
    FULLTEXT INDEX full_userDesc(userDesc)
    );
```

- **示例**：创建**单列索引**

```
CREATE TABLE t_test7(
    id TINYINT UNSIGNED AUTO_INCREMENT KEY,
    t_test1 VARCHAR(20) NOT NULL,
    t_test2 VARCHAR(20) NOT NULL,
    t_test3 VARCHAR(20) NOT NULL,
    t_test4 VARCHAR(20) NOT NULL,
    INDEX in_test1(t_test1)
 );
```

9.2 创建索引示例

- **示例**：创建**多列索引**

```
CREATE TABLE t_test8(
    id TINYINT UNSIGNED AUTO_INCREMENT KEY,
    t_test1 VARCHAR(20) NOT NULL,
    t_test2 VARCHAR(20) NOT NULL,
    t_test3 VARCHAR(20) NOT NULL,
    t_test4 VARCHAR(20) NOT NULL,
    INDEX mul_t1_t2_t3(t_test1, t_test2, t_test3)
    );
```

- **示例**：创建**空间索引**

```
CREATE TABLE t_test10(
    id TINYINT UNSIGNED AUTO_INCREMENT KEY,
    t_test GEOMETRY NOT NULL,
    SPATIAL INDEX spa_test(t_test)（SPATIAL指定空间类型）
    )ENGINE=MyISAM;
```

9.2 另外两种创建方式

- 在**已存在**的表上创建索引
  - **语法格式**:

    CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名称 ON 表名 {字段名称[(长度)] [ASC|DESC]}
- 使用ALTER TABLE语句来创建索引
  - **语法格式**:

    ALTER TABLE tbl_name ADD [UNIQUE| FULLTEXT | SPATIAL] INDEX 索引名称(字段名称[(长度)] [ASC|DESC]);

    注：添加的同时加指定索引名称。名称的作用是在**删除**时需要使用索引名称。

9.2 在已经存在的表创建索引示例

- **示例**：创建索引索引名称

```
CREATE INDEX in_id ON t_test4(id);
```

- 加上原来的索引（第一个普通索引）

```
ALTER TABLE t_test4 ADD INDEX in_username(username);
```

9.2 通过约束，创建索引示例

- 创建表vipuser12：

```
CREATE TABLE `vipuser12` (
  `id` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `card` char(18) NOT NULL,
  `test` varchar(20) NOT NULL,
  `test1` char(32) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

- 通过约束为vipuser12添加索引

```
ALTER TABLE vipuser12 ADD CONSTRAINT symbol  UNIQUE KEY uni_card(card);
```

- 添加复合（联合）索引

```
ALTER TABLE vipuser12 ADD CONSTRAINT symbol UNIQUE INDEX mulUni_test_test1(test,test1);
```

注：索引名称要放在最后面

9.2 查看索引

- **语法格式**:

  SHOW INDEX FROM table_name [FROM db_name]

- 语法的另一种形式。这两个语句是等价的:

  SHOW INDEX FROM mytable FROM mydb;

  SHOW INDEX FROM mydb.mytable;

- SHOW KEYS是SHOW INDEX的同义词。
- 也可以使用以下命令列举一个表的索引。

  MySQLshow -k db_name table_name

  SHOW INNODB STATUS语法

  SHOW INNODB STATUS

9.2 删除索引

- 已经被建立且不经常使用的索引，一方面可能会占有系统资源，另一方面也可能导致更新速度下降，会极大地影响数据表的性能。
- 删除索引可以使用**ALTER TABLE**或**DROP INDEX**语句来实现。DROP INDEX可以在ALTER TABLE内部作为一条语句处理。
- **语法格式**：

```
DROP INDEX index_name ON table_name ;
ALTER TABLE table_name DROP INDEX index_name ;
ALTER TABLE table_name DROP PRIMARY KEY ;
```

删除了table_name中的索引index_name

在最后一条语句中，只删除PRIMARY KEY索引，因为一个表只可能有一个PRIMARY KEY索引，因此**不需要指定索引名。**

9.2 删除索引示例

- 删除两个唯一性索引**示例**:

```
ALTER TABLE vipuser12 DROP KEY uni_card;
ALTER TABLE vipuser12 DROP KEY mulUni_test_test1;
```

第9章MySQL索引

- 9.1 索引
- 9.2 索引的定义与管理
- **9.3 索引的设计原则和注意事项**
- 9.4 知识点小结
- 本章实验

9.3 索引的设计原则和注意事项

1.索引的**设计原则**：

（1）选择**唯一性**索引

唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录。

（2）为经常需要**排序**、**分组**和**联合**操作的字段建立索引

经常需要ORDER BY、GROUP BY、DISTINCT和UNION等操作的字段，排序操作会浪费很多时间。如果为其建立索引，可以有效地避免排序操作。

（3）为常作为**查询条件**的字段建立索引

9.3 索引的设计原则和注意事项

（4）限制索引的**数目**

索引的数目不是越多越好。每个索引都需要占用磁盘空间，索引越多，需要的磁盘空间就越大。修改表时，对索引的重构和更新很麻烦。越多的索引，会使更新表变得很浪费时间。

（5）尽量使用**数据量少**的索引

如果索引的值很长，那么查询的速度会受到影响。

（6）尽量使用**前缀**来索引

如果索引字段的值很长，最好使用值的前缀来索引。

（7）删除**不再使用**或者**很少使用**的索引

9.3 索引的设计原则和注意事项

2.合理使用索引注意事项

（1）在**经常需要搜索的列**上，可以加快搜索的速度。

（2）在**作为主键的列**上，强制该列的唯一性和组织表中数据的排列结构。

（3）在经常用在**连接的列**上，这些列主要是一些外键，可以加快连接的速度。

（4）在经常需要**根据范围进行搜索**的列上创建索引，因为索引已经排序，其指定的范围是连续的。

（5）在经常需要**排序**的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间。

（6）经常使用在**WHERE子句**中的列上创建索引，加快条件的判断速度。

9.3 索引的设计原则和注意事项

3.不合理使用索引的注意事项

（1）对于那些在查询中**很少使用**或者参考的列不应该创建索引。

（2）对于那些只有**很少数据值**的列也不应该增加索引。

（3）对于那些**定义为text、image和bit数据类型的列不应该增加索引**。

主要是由于列的数据量要么相当大，要么取值很少。

（4）当**修改性能**远远大于**检索性能**时，不应该创建索引。

由于修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改性能远远大于检索性能时，不应该创建索引。

第9章MySQL索引

- 9.1 索引
- 9.2 索引的定义与管理
- 9.3 索引的设计原则和注意事项
- **9.4 知识点小结**
- 本章实验

9.4 知识点小结

本章知识小结：

- MySQL数据库的索引的基础知识
- 创建索引的方法和删除索引的方法
- 设计索引的基本原则

第9章MySQL索引

- 9.1 索引
- 9.2 索引的定义与管理
- 9.3 索引的设计原则和注意事项
- 9.4 知识点小结
- **本章实验**

本章实验

口 实验内容：

（7）在创建student表时，设置学号id为唯一索引，创建完后查看索引。

Thanks!

See you