10. 索引特性（重点）

索引：提高数据库的性能，索引是物美价廉的东西了。不用加内存，不用改程序，不用调sql，只要执行正确的`create index`，查询速度就可能提高成百上千倍。但是天下没有免费的午餐，查询速度的提高是以插入、更新、删除的速度为代价的，这些写操作，增加了大量的IO。所以它的价值，在于提高一个海量数据的检索速度。

常见索引分为：

- 主键索引(primary key)
- 唯一索引(unique)
- 普通索引(index)
- 全文索引(fulltext)--解决中子文索引问题。

案例：

先整一个海量表，在查询的时候，看看没有索引时有什么问题？

```
--构建一个8000000条记录的数据
--构建的海量表数据需要有差异性，所以使用存储过程来创建， 拷贝下面代码就可以了，暂时不用理解

-- 产生随机字符串
delimiter $$
create function rand_string(n INT)
returns varchar(255)
begin
 declare chars_str varchar(100) default
   'abcdefghijklmnopqrstuvwxyzABCDEFJHIJKLMNOPQRSTUVWXYZ';
 declare return_str varchar(255) default '';
 declare i int default 0;
 while i < n do
   set return_str =concat(return_str,substring(chars_str,floor(1+rand()*52),1));
   set i = i + 1;
   end while;
  return return_str;
  end $$
delimiter ;

--产生随机数字
delimiter $$
create function rand_num()
returns int(5)
begin
 declare i int default 0;
 set i = floor(10+rand()*500);
return i;
end $$
delimiter ;

--创建存储过程，向雇员表添加海量数据
delimiter $$
create procedure insert_emp(in start int(10),in max_num int(10))
```

```
begin
declare i int default 0;
 set autocommit = 0;
 repeat
  set i = i + 1;
  insert into EMP values ((start+i)
,rand_string(6),'SALESMAN',0001,curdate(),2000,400,rand_num());
  until i = max_num
 end repeat;
 commit;
end $$
delimiter ;

-- 执行存储过程，添加8000000条记录
call insert_emp(100001, 8000000);
```

到此，已经创建出了海量数据的表了。

- 查询员工编号为998877的员工

  ```
  select * from EMP where empno=998877;
  ```

  可以看到耗时4.93秒，这还是在本机一个人来操作，在实际项目中，如果放在公网中，假如同时有1000个人并发查询，那很可能就死机。

- 解决方法，创建索引

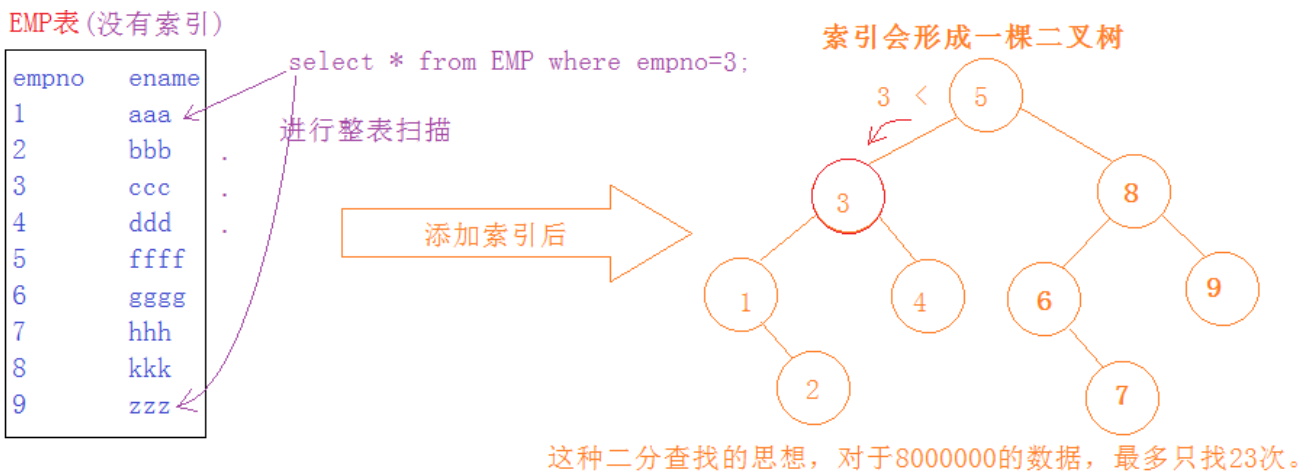
  ```
  alter table EMP add index(empno);
  ```

- 换一个员工编号，测试看看查询时间

  ```
  select * from EMP where empno=123456;
  ```

10.1 基本原理

索引理论的深度理解参见推荐书籍：《数据库系统概念》的第11章



索引的说明：

- 占用磁盘空间
- 当添加一条记录，除了添加到表中，还要维护二叉树，速度有影响，但不大。
- 当我们添加一个索引，不能够解决所有查询问题，需要分别给字段建立索引；例如 select * from EMP where ename='abcdef';
- 索引是以空间换时间

10.2 创建索引

10.2.1 创建主键索引

- 第一种方式

```
-- 在创建表的时候，直接在字段名后指定 primary key
create table  user1(id int primary key, name varchar(30));
```

- 第二种方式：

```
-- 在创建表的最后，指定某列或某几列为主键索引
create table  user2(id int, name varchar(30), primary key(id));
```

- 第三种方式：

```
create table  user3(id int, name varchar(30));
-- 创建表以后再添加主键
alter table user3 add primary key(id);
```

主键索引的特点：

- 一个表中，最多有一个主键索引，当然可以使符合主键
- 主键索引的效率高（主键不可重复）
- 创建主键索引的列，它的值不能为null，且不能重复
- 主键索引的列基本上是int

10.2.2 唯一索引的创建

- 第一种方式

```
-- 在表定义时，在某列后直接指定unique唯一属性。
create table user4(id int primary key, name varchar(30) unique);
```

- 第二种方式

```
-- 创建表时，在表的后面指定某列或某几列为unique
create table user5(id int primary key, name varchar(30), unique(name));
```

- 第三种方式

```
create table user6(id int primary key, name varchar(30)) ;
alter table user6 add unique(name);
```

唯一索引的特点：

- 一个表中，可以有多个唯一索引
- 查询效率高
- 如果在某一列建立唯一索引，必须保证这列不能有重复数据
- 如果一个唯一索引上指定not null，等价于主键索引

10.2.3 普通索引的创建

- 第一种方式

```
create table user8(id int primary key,
     name varchar(20),
     email varchar(30),
     index(name) --在表的定义最后，指定某列为索引
);
```

第二种方式

```
create table user9(id int primary key,  name varchar(20),  email varchar(30));
alter table user9 add index(name); --创建完表以后指定某列为普通索引
```

- 第三种方式

```
create table user10(id int primary key,  name varchar(20),  email varchar(30));
-- 创建一个索引名为 idx_name 的索引
create index idx_name on user10(name);
```

普通索引的特点：

- 一个表中可以有多个普通索引，普通索引在实际开发中用的比较多
- 如果某列需要创建索引，但是该列有重复的值，那么我们就应该使用普通索引

10.2.3 全文索引的创建

当对文章字段或有大量文字的字段进行检索时，会使用到全文索引。MySQL提供全文索引机制，但是有要求，要求表的存储引擎必须是MyISAM，而且默认的全文索引支持英文，不支持中文。如果对中文进行全文检索，可以使用sphinx的中文版(coreseek)。

```
CREATE TABLE articles (
    id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    title VARCHAR(200),
    body TEXT,
    FULLTEXT (title,body)
)engine=MyISAM;
```

```
INSERT INTO articles (title,body) VALUES
    ('MySQL Tutorial','DBMS stands for DataBase ...'),
    ('How To Use MySQL Well','After you went through a ...'),
    ('Optimizing MySQL','In this tutorial we will show ...'),
    ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    ('MySQL vs. YourSQL','In the following database comparison ...'),
    ('MySQL Security','When configured properly, MySQL ...');
```

- 查询有没有database数据

如果使用如下查询方式，虽然查询出数据，但是没有使用到全文索引

```
mysql> select * from articles where body like '%database%';
+----+-------------------+----------------------------------------+
| id | title             | body                                   |
+----+-------------------+----------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...           |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
+----+-------------------+----------------------------------------+
```

可以用explain工具看一下，是否使用到索引

```
mysql> explain select * from articles where body like '%database%'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: articles
         type: ALL
possible_keys: NULL
          key: NULL  <== key为null表示没有用到索引
      key_len: NULL
          ref: NULL
         rows: 6
        Extra: Using where
1 row in set (0.00 sec)
```

- 如何使用全文索引呢?

```
mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body) AGAINST ('database');
    +----+-------------------+----------------------------------------+
    | id | title             | body                                   |
    +----+-------------------+----------------------------------------+
    |  5 | MySQL vs. YourSQL | In the following database comparison ... |
    |  1 | MySQL Tutorial    | DBMS stands for DataBase ...           |
    +----+-------------------+----------------------------------------+
```

通过explain来分析这个sql语句

```
mysql> explain SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('database')\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: articles
         type: fulltext
possible_keys: title
          key: title <= key用到了title
      key_len: 0
          ref:
         rows: 1
        Extra: Using where
```

10.3 查询索引

- 第一种方法：`show keys from 表名`

```
mysql> show keys from goods\G
    *********** 1. row ***********
           Table: goods    <= 表名
      Non_unique: 0        <= 0表示唯一索引
        Key_name: PRIMARY  <= 主键索引
    Seq_in_index: 1
     Column_name: goods_id <= 索引在哪列
       Collation: A
     Cardinality: 0
        Sub_part: NULL
          Packed: NULL
            Null:
      Index_type: BTREE    <= 以二叉树形式的索引
         Comment:
1 row in set (0.00 sec)
```

- 第二种方法：`show index from 表名;`
- 第三种方法（信息比较简略）：`desc 表名;`

10.4 删除索引

- 第一种方法-删除主键索引：`alter table 表名 drop primary key;`
- 第二种方法-其他索引的删除：`alter table 表名 drop index 索引名;` 索引名就是show keys from 表名中的 Key_name 字段

  `mysql> alter table user10 drop index idx_name;`
- 第三种方法方法： drop index 索引名 on 表名

  `mysql> drop index name on user8;`

10.4 索引创建原则

- 比较频繁作为查询条件的字段应该创建索引
- 唯一性太差的字段不适合单独创建索引，即使频繁作为查询条件

- 更新非常频繁的字段不适合作创建索引
- 不会出现在where子句中的字段不该创建索引