索引

一、基本介绍

1、定义

是数据库中专门用于帮助用户快速查询数据的一种**数据结构（排好序的快速查找数据结构）**

类似于字典中的目录，查找字典内容时可以根据目录查找到数据的存放位置，然后直接获取即可。

2、数据结构

MySQL索引使用的数据结构主要有**B+Tree索引** 、BTree、 **哈希索引**

在绝大多数需求为单条记录查询的时候，可以选择哈希索引，查询性能最快；

其他情况一般使用B+Tree索引

3、优点

1）大大加快数据的**检索速度**（大大减少的检索的数据量）

2）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**

3）加速表和表之间的连接

4、缺点

1）**占用物理存储空间**：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

2）**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

3）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**。

二、分类

1、从物理存储角度

- **聚集索引（clustered index)**
- **非聚集索引（non-clustered index)**

2、从逻辑角度

- **普通索引（单列索引)**
- **唯一索引**
- **主键索引**
- **组合索引（符合索引)**
- **全文索引**

三、使用索引的注意事项

1、使用场景

- 在经常需要**查询**的列上，可以加快搜索的速度；
- 在经常使用在**WHERE子句**中的列上面创建索引，加快条件的判断速度。
- 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；
- 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；
- 表的主键、外键必须有索引；外键是唯一的，而且经常会用来查询

2、注意事项

- 对于中到大型表索引都是非常有效的，但是特大型表的话维护开销会很大，不适合建索引
- 避免 where 子句中对字段施加函数，这会造成无法命中索引。

- 在使用InnoDB时使用与业务无关的自增主键作为主键，即使用逻辑主键，而不要使用业务主键。
- 将打算加索引的列设置为 NOT NULL ，否则将导致引擎放弃使用索引而进行全表扫描
- 删除长期未使用的索引，不用的索引的存在会造成不必要的性能损耗 MySQL 5.7 可以通过查询 sys 库的 chema_unused_indexes 视图来查询哪些索引从未被使用
- 在使用 limit offset 查询缓慢时，可以借助索引来提高性能

3、索引的选取

- **越小的数据类型通常更好：越小的数据类型通常在磁盘、内存和CPU缓存中都需要更少的空间，处理起来更快。**
- **简单的数据类型更好：整型数据比起字符，处理开销更小，因为字符串的比较更复杂。**
- 尽量避免**NULL**：应该指定列为NOT nuLL,在MySQL中， 含有空值的列很难进行查询优化，因为它们使得索引、索引的统计信息以及比较运算更加复杂