索引创建的几种方式：

1. CREATE TABLE table1 (

P\_id int,

name varchar(15),

created\_at datetime,

INDEX(P\_id, created\_at)

)

1. ALTER TABLE table1 ADD INDEX index\_name (p\_id, created\_at)
2. CREATE INDEX index\_name ON table1 (p\_id, created\_at)

删除索引：

1. DROP INDEX index\_name ON table1
2. ALTER TABLE table1 DROP INDEX index\_name

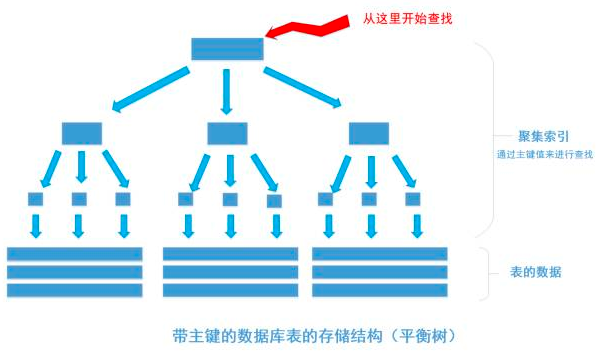
索引的创建比较简单，但索引本身不简单，对于只有1万条数据的数据库而言，加不加索引影响不大，但对于大量数据的数据库，影响就比较大。

PRIMARY KEY主键本身就会自带索引。

一般的数据库都是使用B+树来作为索引(极少数使用哈希表)，它是一棵平衡树(但不是二叉的)，但它基本操作(增删改查)的时间复杂度与平衡二叉树类似，平衡二叉树为，默认是以2为底的，而B+树是，其中m就树的结点分叉数。

数据库在创建表时，一般都默认有一个主键，其自带索引(一般称为聚合索引)，某些数据库如果不指定主键，就无法创建表。如果一张表没有索引，就类似于平时见到的表，数据无序的存储在磁盘上，一行一行的比较整齐，当添加了主键后，表的存储结构就变成了树状结构。主键是聚集索引，整张表就变成了一个索引。

这是Mysql中InnoDB引擎的数据库表。



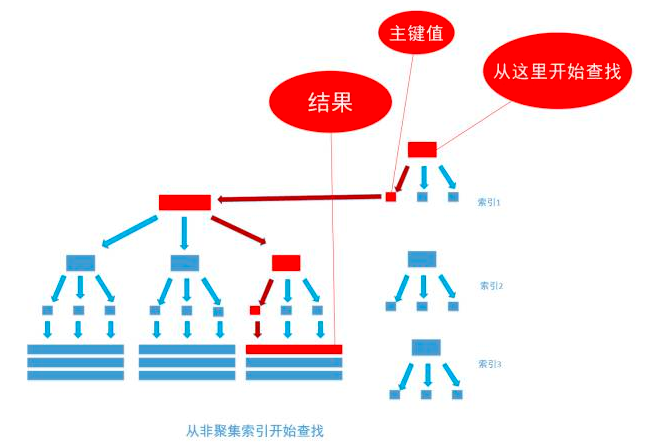
其中，数的非叶子结点存储的数据是主键id(主键字段的值)，只有叶子结点存储的是实际的数据。

执行一条SQL语句，如：select \* from tweets where id = 1256;

假设tweets表中有一亿条记录，如果没有主键索引的情况下，需要逐条匹配，时间复杂度为O(n)，最坏情况下可能会匹配一亿次，需要的时间可能是几个月，而有了主键索引，假设结点的分叉数为10，匹配的次数就是,只需要匹配9次就能够找到数据。

索引能大幅度提高数据库的查询速度，但写入(增加、修改、删除)速度会下降，原因就是索引必须维护成一个B+树，当写入一条记录时，数据库系统需要调整索引以保持为一棵平衡树。

非聚集索引：一般就是我们创建的普通索引，它的结构和聚集索引类似，也是B+树，不同的是它的非叶子结点存储的是建立索引的字段值，例如对username建立索引，那么非叶子结点存储的就是username的值，而叶子结点存储的则是主键值，通过非聚集索引查询时，会首先查找到记录对应的主键值，然后通过这主键值再去原始表单中查找最终的记录，因此需要两次查询，故而叫非聚集索引。



非聚集索引可以创建多个，互相之间不影响，但不建议多创建，因为会影响数据库的性能。

数据库的联合索引支持最左匹配原则，例如index\_together=((‘user’, ‘created\_at’))。这个联合索引支持3种方式：

1. 查找user在某个范围内(联合索引是有效的，虽然没有用的索引的全部能力)
2. 查找user=XX and created\_at在某个范围内(最标准的用法)
3. 查找user=XX and created\_at==YY