索引其实是一种特殊的数据结构，附加在表之外的数据集，为了加速读取操作。索引也叫键。

索引：简单索引(索引一个字段)，组合索引(索引多个字段上)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHOW INDEXES FROM students;

如果有时候查询很快可能是有缓存了；

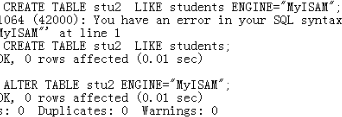
SHOW GLOBAL STATUS LIKE ‘%cache%’; 查看缓存命中等情况

flush query cache; 清空缓存

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

为了做实验证明，不同的存储引擎的索引不一样，所以再新建一个表



这里好像不能在建立表的时候指定存储引擎？

把别的表数据插进来：



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

索引类型：**在存储引擎级别实现**

1)BTREE：平衡树

2) HASH索引

3)RTREE地理索引

4) FULLTEXT索引

索引的物理结构：

MyISAM:

.frm: 表结构

.MYD: data数据

.MYI: 索引

InnoDB:

TableSpace: 表空间，数据和索引都在这里。 把**所有表的数据放到同一个表空间中**

.frm 表结构

也可以每张表使用单独的表空间文件： 便于管理，备份

innodb\_file\_per\_table = ON

innoDB的BTREE：是一种叫做聚簇索引BTREE；数据和索引在一块，普通的btree是索引存放的数据的指针。效率更高。这仅仅是innodB的一种索引。

每张表聚簇索引只能有一个，因为这个索引直接决定了页面的位置。还可以建立别的索引。其他索引再次指向局促索引，所以别的索引效率并不高。尽可能减小局促索引的大小。

INDEX(EXTRA), fdkfdkgu89329r8eajkdagjkalg

BTREE:也叫最左前缀索引，只从左到右的索引

WHERE EXTRA LIKE ‘1234%’;

WHERE EXTRA LIKE '%6789%';

生效的场合：

匹配全名

匹配最左前缀

匹配列前缀

匹配范围

如果不是通过最左侧查找的就没有意义没法查找

2)HASH索引：

只有Memory支持显式的hash索引；

键值对 key-value

HASH(NAME)

无法做范围查找和排序，只能做精确匹配和等值匹配

WHERE NAME='GuoXiaotian';

=, !=, IN

WHERE NAME LIKE 'Guo%';

CRC32()冲突还需要二次匹配

3)RTREE: 空间索引

地理位置

MyISAM, 必须使用内置函数来调用索引

4)FULLTEXT索引：

MyISAM, 不支持事务，要想既支持事物又支持全文索引的使用：

sphinx：全文索引服务器

lucene: 搜索引擎

###################################

创建索引：

CREATE INDEX index\_name ON table(col字段名(length),...)

[USING {BTREE|HASH}] 指定索引类型

COMMENT "string" 注释符

如果索引是text字段，必须指定length；非常长的varchar字段也需要指定length

样例：



CREATE TABLE tb (col1, col2, col3,...,

INDEX

CONSTRAINT 定义约束PRIMARY KEY

FULLTEXT

SPACIAL

CONSTRAINT FOREIGN KEY

CONSTRAINT UNIQUE



还可以：ALTER TABLE 直接增加索引 ADD INDEX



这样不需要名字

删除索引DROP INDEX



覆盖索引：索引也会存储对应索引字段的数据。一种索引的使用方式。

stu2(NAME)

SELECT Name WHERE NAME LIKE 'stu11%';