Myisam表和innodb表的区别：

Myisam表不支持事务处理等高级处理，而innodb表支持

Myisam表强调的是性能，其执行数度比innodb快，但是不提供事务支持

Innodb提供事务支持以及外部键等高级的数据库功能。

两种类型最主要的差别就是Innodb 支持事务处理与外键和行级锁

表结构修改：

添加新collumn: alter table company add deleted bit default 0;

修改已存在collumn: alter table company change email email char(100) default null;

修改已存在的collumn（不改变名字）：alter table company modify site varchar(50) default null;

删除已存在collumn: alter table company drop cc

如果设置了0填充zerofill则默认为无符号类型

Bit ------------------- 0或1

Tinnyint ------------------- 0~255

Smallint------------------- 0~65535

Mediumint

Int

Bigint

浮点类型：float double

格式:float(M,D) unsigned\zerofill;

金钱类型：decimal

year YYYY 范围:1901~2155. 可输入值2位和4位(如98,2012)

日期时间类型 date YYYY-MM-DD 如:2010-03-14

time HH:MM:SS 如:19:26:32

datetime YYYY-MM-DD HH:MM:SS 如:2010-03-14 19:26:32

timestamp YYYY-MM-DD HH:MM:SS 特性:不用赋值,该列会为自己赋当前的具体时间

内联查询好于where条件筛选即内联查询优于子查询

组合查询筛选条件：having相当于是对分组查询的结果进一步的筛选，因此having筛选的字段必须是分组查询出来的。

查看当前字符集：show variables like ‘%char%’

查看关于当前数据库的某些信息都可以使用show variables 匹配关键词

show variables like '%char%';

show variables like '%connect%';

开启事务：

①开启事务 start transaction

②运行sql;

③提交,同时生效\回滚 commit\rollback

Trigger

索引类型

primary key主键索引

index 普通索引

unique index 唯一性索引

fulltext index 全文索引

-----联合索引

----------------------------

**[MySQL中锁详解（行锁、表锁、页锁、悲观锁、乐观锁等）](http://blog.csdn.net/tanga842428/article/details/52748531)**

锁是为了保证数据库并发访问的一致性、有效性

Mysql中不同的存储引擎支持不同的锁机制

|  |  |  |  |
| --- | --- | --- | --- |
|  | 行锁 | 表锁 | 页锁 |
| MyISAM |  | √ |  |
| BDB |  | √ | √ |
| InnoDB | √ | √ |  |

开销、加锁速度、死锁、粒度、并发性能

* 表锁：开销小，加锁快；不会出现死锁；锁定力度大，发生锁冲突概率高，并发度最低
* 行锁：开销大，加锁慢；会出现死锁；锁定粒度小，发生锁冲突的概率低，并发度高
* 页锁：开销和加锁速度介于表锁和行锁之间；会出现死锁；锁定粒度介于表锁和行锁之间，并发度一般

表锁更适用于以查询为主，只有少量按索引条件更新数据的应用；行锁更适用于有大量按索引条件并发更新少量不同数据，同时又有并发查询的应用

**MyISAM表锁**

Myisam只支持表锁

可以通过检查table\_locks\_waited和table\_locks\_immediate状态变量来分析系统上的表锁定争夺：

**[html]** [view plain](http://blog.csdn.net/tanga842428/article/details/52748531) [copy](http://blog.csdn.net/tanga842428/article/details/52748531)

1. mysql**>** show status like 'table%';
2. +-----------------------+-------+
3. | Variable\_name         | Value |
4. +-----------------------+-------+
5. | Table\_locks\_immediate | 2979  |
6. | Table\_locks\_waited    | 0     |
7. +-----------------------+-------+

可见，对MyISAM表的读操作，不会阻塞其他用户对同一表的读请求，但会阻塞对同一表的写请求；对 MyISAM表的写操作，则会阻塞其他用户对同一表的读和写操作；

获得表film\_text的WRITE锁定

mysql> lock table film\_text write;

释放锁：

mysql> unlock tables;

当一个线程获得锁后，其他的线程需要等待才能执行查询

MyISAM在执行查询语句（SELECT）前，会自动给涉及的所有表加读锁，在执行更新操作（UPDATE、DELETE、INSERT等）前，会自动给涉及的表加写锁

-----------------------

Mysql查询优化之使用索引

首先，使用索引来获取最大的性能提高，接着在看其他的技术是否有用。

对于MyISAM数据表，该表的数据行保存在一个数据文件中，索引值保存在索引文件中。一个数据表上可能有多个索引，但是它们都被存储在同一个索引 文件中。索引文件中的每个索引都包含一个排序的键记录（它用于快速地访问数据文件）数组。

MySQL有几种使用索引的方式：  
  
　　· 如上所述，索引被用于提高WHERE条件的数据行匹配或者执行联结操作时匹配其它表的数据行的搜索速度。  
  
　　· 对于使用了MIN()或MAX()函数的查询，索引数据列中最小或最大值可以很快地找到，不用检查每个数据行。  
  
　　· MySQL利用索引来快速地执行ORDER BY和GROUP BY语句的排序和分组操作。

索引的缺点

首先，索引加快了检索的速度，但是减慢了插入和删除的速 度，同时还减慢了更新被索引的数据列中的值的速度。主要的原因是在进行一条数据的写入时不得不维护索引。

其次，索引会花费磁盘空间，

选择索引：

用于索引的最好的备选数据列是那些出现在WHERE子句、join子句、 ORDER BY或GROUP BY子句中的列。

如果列的值越分散则建立索引越好，如果数据列的值较集中则建立索引并没有什么效果

因为查询优化器发现某个值出现在表的数据行中的百分比很高的时候，它一般会忽略索引，进行全 表扫描。惯用的百分比界线是"30%"

索引较短的值。尽可能地使用较小的数据类型

索 引字符串值的前缀（prefixe）

使用最左（leftmost）前缀。建立多列复合索引的时候，（如果查询where语句中不包含最左边的列时，查询可能不会使用索引）

不要过多而使用索引，因为额外的索引会给查询优化器带来外的更多的工作量

B树索引也可以用于LIKE模式匹配，前提是该模式以文字串而不是通配符开头，如果使用了通配符的话就不会使用索引

可以在建立索引的时候选择索引的类型：散列索引或者是B树索引