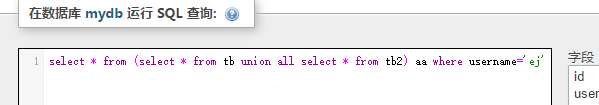
1. 在数据库中有个小坑，如果你使用了union all查出出来后的结果，然后再进行查询的话，一定要给表一个别名，不然会报错；union 和 union all 两个的本质都是从两个表的结果集后在汇总到一张表，但列数行数要一样，而且如果列名不设置，那他会默认第一表的列名，他们两个的区别就是union 会自动去除重复的，浪费磁盘资源，union all 会比较快的查询出结果出来。(注意可以union 的两个表可以括号起来查询出结果集之后再来union的，这样就怎样去判断条件都是没事的了)



1. 如果两个列的行数要合并起来，比如：每个列各3行，合起来共3行的，可以用concat 或者 group\_concat ，然后如果合起来共6行的，那就是用union 或者 union all



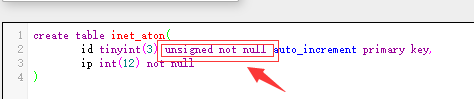
1. 在创建表的时候可以采用 : create table tb\_name engine=archive as select \* from my\_table; 然后就会把my\_table 的字段和内容都弄到了 tb\_name ；
2. 引擎 MyISAM、Archive、InnoDB 这三个引擎中，MyISAM 是用于非常频繁的查询 ；Archive是用于存档，因为其压缩了比MyISAM少73%空间，比InnoDB少86%空间，Archive一般都是用于存储日志和历史资料的查询；InnoDB 一般用于更新和并发高的查询，支持事物和行。
3. 在创建索引的时候要注意付出的代价，如果查询的数据时候，索引会自动 进行判断，如果索引找到匹配后，然后就会进行代价判断，如果代价估计数据大于本表的15%以上，就不使用索引了，此时索引的代价就更大。 索引使用总共有 查询的时候mysql自动索引 ；还有另外一个可以强制使用：select \* from tb force index(index\_email)
4. **查询优化：**如果你使用了 name like “%s”; 这个时候是不会使用到索引的，因为%是通配符，进行全局匹配搜索。
5. 如果在数据过大的时候，比如超过10G，可以采用分区：partition,可以在创建表的时候：

**Create table user( id tinyint(3) not null primary key, name char(30) not null comment ‘姓名’)**

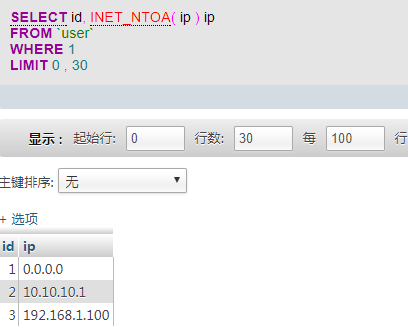
**partition by range(id)( partition id01 values less than (10), partition id02 values less than (20), partition id03 values less than maxvalue);**  或者你可以用list来创建也可以，或者你也可以采用 Hash() 函数来随机分配，固定分区，创建分区，如： **create table tb( a int(10), b int(19)) partition by hash(id); partitions 4; hash会自动创建4个分区，然后会取模%来随机分配(注意 分区4 的partitions 有个s);**

当然你也可以add partition 增加分区，也可以drop partition了。

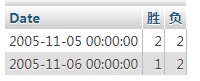
1. 查询分区的内容使用 select \* from tb\_name partition (id01); 其中id01 是自定义的分区名；(**注意这里的mysql版本要求是 5.6以上才可以**).
2. unsigned 要在 not null 前面才行：

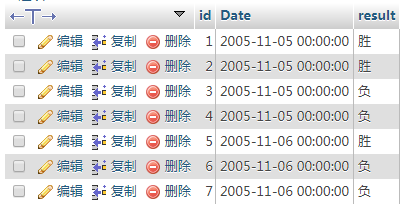


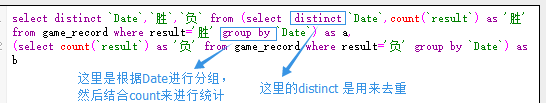
10、在设置要存储IP特殊字段值时要采用，如：ip int(12) unsigned not null,这个时候一定要有 unsigned 不然，128以上的ip段不能正常保存，出现值不一样，insert 数据的时候要用 inet\_aton(‘192.168.1.1’)这样来保存，然后select的时候用 inet\_ntoa(ip) 来读取。



1. 如果遇到下列情况，而且是一张表，或者多张表，可以用 迪尔卡集 来操作：

结果：

表：

使用sql语句: 

1. 行锁(Innodb)，表锁(myisam,memory)，页锁(BDB)浅析：

1、行锁优点：(可以并发处理，一般行锁还是用的比较多)

储存在内存中，运行速度快；  
 能进行针对行的数据进行锁，并出现异常时，可以小数据进行回滚；

行锁缺点：

消耗内存，占据的内存比较大，不适应做那些经常查询的，读操作；

I/O需要的量多，而且容易造成死锁；

2、表锁相对其他两个优点：(可以并发处理，用concurrent\_insert 设置)

现在的myisam的表锁都是默认状态2 (就是可以在读的同时，并发向尾部写入)但在涉及到金额的时候，要加上lock table table\_name write(或者read) 来确保数据的完整性，不然一个表完成了数据操作，另外一个表可能失败，导致数据错误；

除了命令执行SET LOW\_PRIORITY\_UPDATES=1，降低写(更新)的优先级，还可以降低insert、delete的优先级，还可以使用系统提供的 max\_write\_lock\_count设置一个合适的值，来降低写的优先级，让一些读操作有机会获得锁；实在复杂的查询最好安排在闲时，比如夜间；

表锁一般用于读操作，而且严格条件下可以进行更新操作；

不会造成死锁，加锁速度快；

3、BDB页锁是基于这两个锁之间，反应最快的还是表锁。

加锁 Lock table table\_name write(或者 read);

释放锁 Unlock tables;

4、在行锁InnoDB中，行锁的引用是基于索引id的，在数据库中要begin 启用事务,commit 结束事务，其中也可以利用rollback;在提交commit前进行回滚；在行锁中可以使用

Select \* from tb where id=1 for update;来进行排他锁

Select \* from tb where id=1 lock in share mode;来进行共享锁

但记得，都基于事务的开始和id索引

但是在要对表进行复杂操作或者太多行数据更新的话，InnoDB才采用表锁，太多这种操作就建议使用MyISAM表了。

表锁可以采用：

SET AUTOCOMMIT=0;

LOCK TABLES tb WRITE;

Insert into table\_name …..;

UNLOCK TABLES;

5、间隙锁：就是进行范围查询的时候不匹配的值，成为“间隙(GAP)”,mysql的 InnoDB 会给其加“间隙锁”；如果是在并发的情况下，“间隙锁”会容易造成等待锁太久，所以在业务逻辑尽量用相等的来查询；