SQL优化

一、SQL优化目的：

**1.增加数据库的稳定性**（很多数据库问题都是低效查询引起的）

**2.避免出现页面访问错误**（慢查询出现页面无法加载，阻塞造成数据无法提交）

**3.优化用户体验**（加快页面的访问速度）

二、SQL语句优化

1.慢查询日志：

查询 show variables like ‘%log%’;

show variables like 'slow\_query\_log';//是否开启慢查询日志

set global log\_queries\_not\_using\_indexes = on //是否要把没有使用索引的SQL记录到慢查询日志中，以便后面进行优化

set global long\_query\_time = 1 //超过多少秒的查询记录到慢查询日志中

慢查询日志分析工具：(1)**mysqldumpslow** (2)**pt-query-digest**

**表索引，使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。  
　在关系数据库中，索引是一种与表有关的[数据库结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BA%93%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ydnjw9uAuhuHP9mhczm1KW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1bYPWmkPHbz" \t "https://zhidao.baidu.com/question/_blank)，它可以使对应于表的[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ydnjw9uAuhuHP9mhczm1KW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1bYPWmkPHbz" \t "https://zhidao.baidu.com/question/_blank)执行得更快。索引的作用相当于图书的目录，可以根据目录中的页码快速找到所需的内容。当表中有大量记录时，若要对表进行查询，第一种搜索信息方式是全表搜索，是将所有记录一一取出，和查询条件进行一一对比，然后返回满足条件的记录，这样做会消耗大量数据库系统时间，并造成大量磁盘[I/O操作](https://www.baidu.com/s?wd=I/O%E6%93%8D%E4%BD%9C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ydnjw9uAuhuHP9mhczm1KW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1bYPWmkPHbz" \t "https://zhidao.baidu.com/question/_blank)；第二种就是在表中建立索引，然后在索引中找到符合查询条件的索引值，最后通过保存在索引中的ROWID（相当于页码）快速找到表中对应的记录。  
　　索引是一个单独的、物理的[数据库结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BA%93%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ydnjw9uAuhuHP9mhczm1KW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1bYPWmkPHbz" \t "https://zhidao.baidu.com/question/_blank)，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单。  
　　索引提供指向存储在表的指定列中的数据值的指针，然后根据您指定的排序顺序对这些指针排序。数据库使用索引的方式与您使用书籍中的索引的方式很相似：它搜索索引以找到特定值，然后顺指针找到包含该值的行。  
　　在数据库关系图中，可以在选定表的“索引/键”属性页中创建、编辑或删除每个索引类型。当保存索引所附加到的表，或保存该表所在的关系图时，索引将保存在数据库中。**

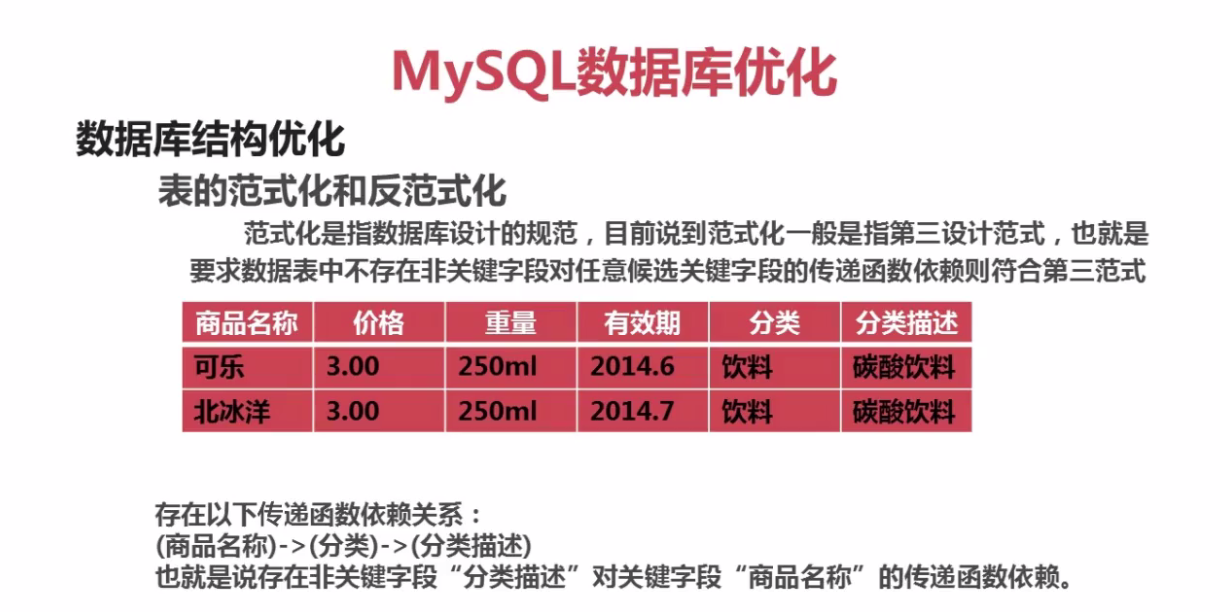
实例：

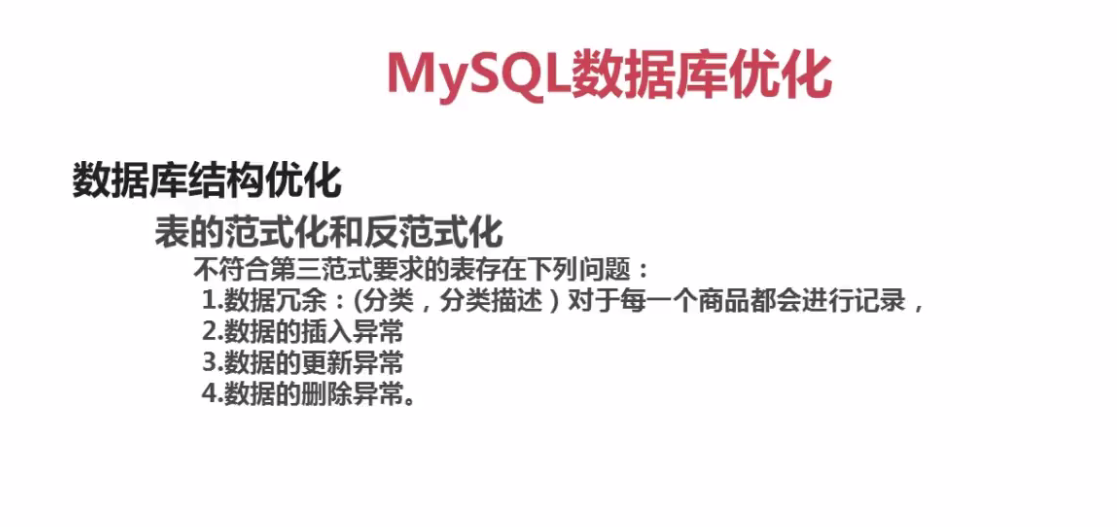
一、SQL语句优化：

1. Count()和Max()函数优化：
2. 子查询的优化：
   1. 将子查询改为join查询
3. 优化GROUP BY查询：
4. Limit查询的优化：
5. 索引优化：
6. 如何选择合适的列建立索引：
   1. 在where从句，group by从句，order by从句，on从句中出现的列。
   2. 索引字段越小越好
   3. 离散度大的列放在联合索引的前面
7. 索引优化SQL的方法：
   1. 重复及冗余索引：如：primary key 和unique（id）,primary key和key(name,id)
   2. 如何查找重复及冗余索引
   3. 索引维护的方法
8. 数据库结构优化：
9. **选择合适（重点）的数据类型**
   1. 使用可以存下你数据的最小存储方式
   2. 使用简单的数据类型。如int会比varchar类型在mysql处理上简单
   3. 尽可能的使用not null定义字段
   4. 尽量少使用text类型，非用不可时最好考虑分表

**（如用int来保存时间日期，用bigint来保存IP地址）**

1. **数据库表的范式化优化：**

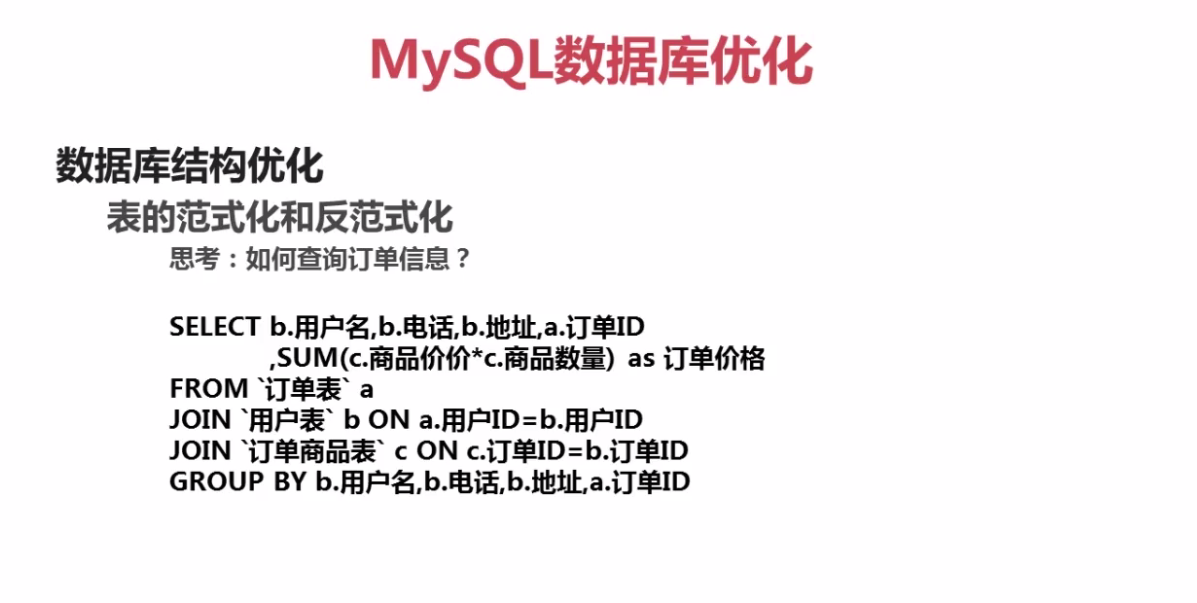




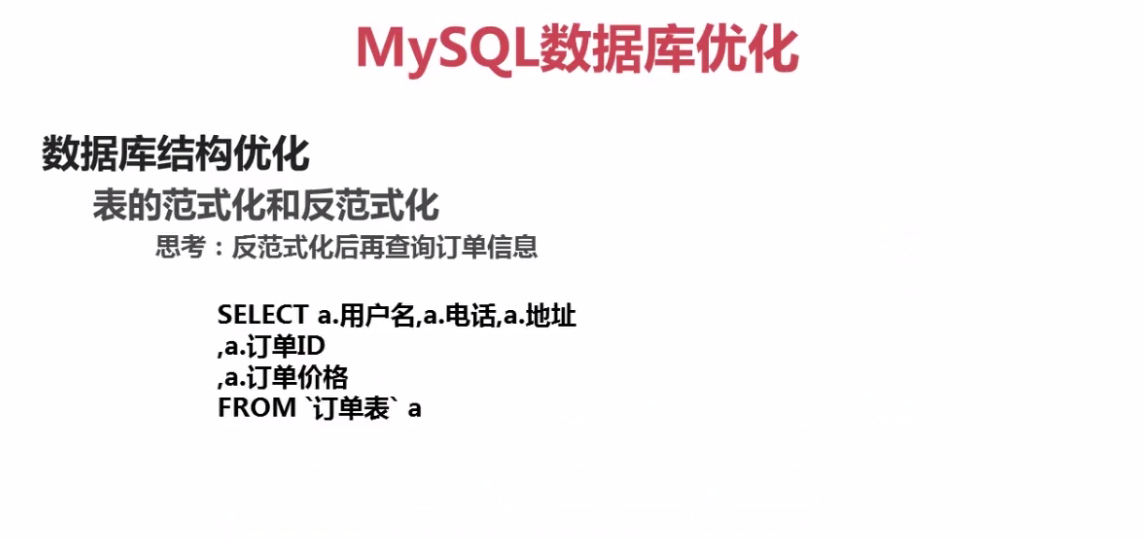


1. **数据库表的反范式优化：**









1. 数据库表的垂直拆分：

所谓的垂直拆分，就是把有很多列的表拆分成多个表，这解决的表的宽度问题。通常垂直拆分可以按以下原则进行：

1. 把不常用的字段单独放到一个表中
2. 把大字段的列独立放到一个表中
3. 把经常一起使用的字段放到一起
4. 数据表的水平拆分：
5. 数据库系统配置优化：

1.

**韩顺平mysql优化**

Mysql数据库优化的常见方法：

1. 表的设计要合理（满足3NF）3范式
2. 创建适当索引[主键索引||唯一索引||普通索引||全文索引||空间索引]
3. 对SQL语句优化----->定位慢查询（explain）
4. 使用分表技术（垂直分表，水平分表），分区技术（了解）
5. 读写分离（配置）
6. 创建适当存储过程，函数，触发器。
7. 对my.ini优化，优化配置
8. 软件硬件升级

**一、3NF**

1NF:

所谓1NF就是

1. 指表的属性（列）具有原子性，即表的列不能再分。
2. 不能有重复的列。

只要关系型数据库就天然满足1NF

2NF:

所谓2NF，就是指我们表中不能有完全一样的一条记录（行），一般情况下通过设置一个主键来搞定。

3NF：

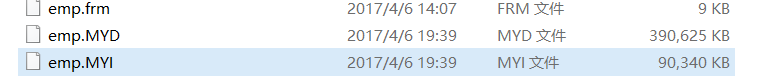
所谓3NF就是指，如果列的内容可以推导出，那么我们就不要单独的用一列存放。



反3NF：

通常情况下，我们的表设计要严格遵循3NF，但是也有例外的情况，这样反而会提高查询效率

1. **定义海量表，定位慢查询**
   1. 构建海量表（一千万条数据），这时我们要用存储过程完成该任务。
   2. 先使用索引来搞定。
      1. alter table emp add primery key(empno);

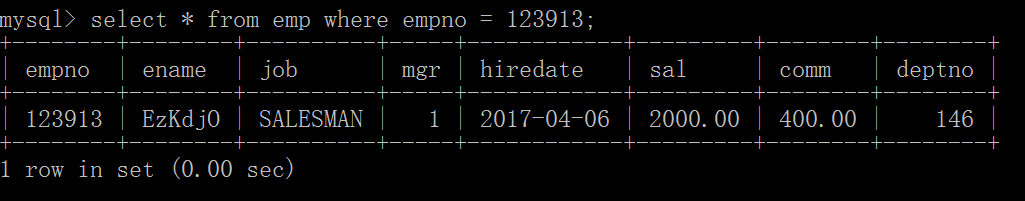
C)

一个表（存储引擎是myISAM），对应三个文件：

Xx.frm 表结构文件

Xx.MYD 表数据文件

Xx.MYI 表索引文件

D)通过测试看效果：

E)问题来了，你怎么知道这个查询很慢（如何定位慢查询）

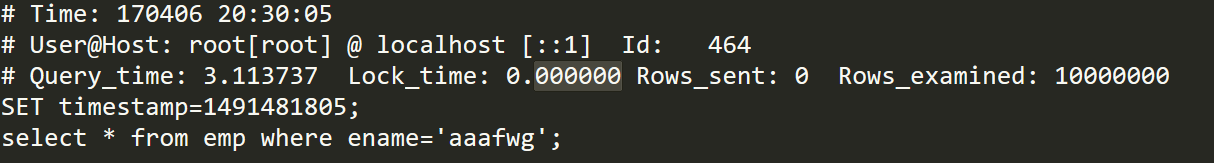
默认情况下，mysql是不会记录慢查询的

查询 show variables like ‘%log%’;

show variables like 'slow\_query\_log';//是否开启慢查询日志

set global log\_queries\_not\_using\_indexes = on //是否要把没有使用索引的SQL记录到慢查询日志中，以便后面进行优化

set global long\_query\_time = 1 //超过多少秒的查询记录到慢查询日志中



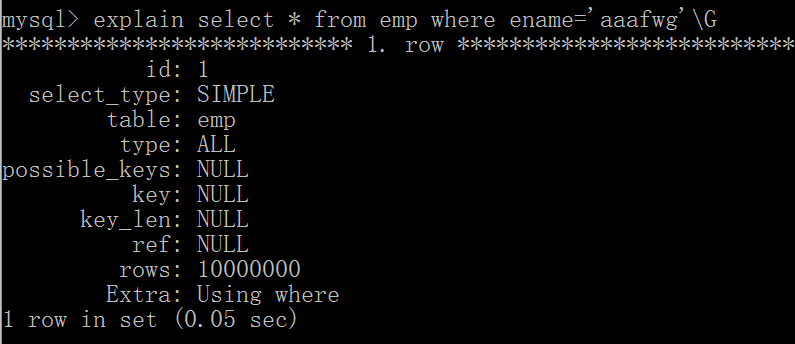
说明： Query\_time：是查询时间，数据量大的问题 优化

Lock\_time: 是锁时间，锁的时间过长，代表有一个连接过来的， 但是老是得不到查询，假设一个表只有十条记录，也可能慢，同时几千万人在查询，等了很久，查询时间很短，等了很久，

如果时间长，我们要解决的是数据量的吞吐问题，要用读写分离来 分担数据库的查询压力。（等待的时间，排队时间，多人同时查询）， 读写分离。银行例子

F)如何分析我们的sql语句问题在哪些地方（慢在哪里）--explain工具：

基本用法： explain sql\G



Select\_type 查询类型,simple简单查询

Table 查询的表

Type 显示连接使用的类型，all表示全表查询，效率低

Possible\_keys 查询可能用到的索引（有什么索引可以用）

Key 实际上用到的索引

Rows 估算或扫描的行数

优化，添加索引。

分表

1. **索引的详解：**

* 索引的创建

1.主键索引的创建 ：

* 在创建表时指定某列或者某几列为主键
* 在创建表后，再指定主键索引。

ALTER TABLE 表名 ADD PRIMARY KEY (列名1，列名2, ...)

* + - * **主键索引的特点**

1.一个表最多只有一个索引

2.一个主键可以指向多个列（符合索引）

3.主键索引的效率是最高的，因此我们一般给个id，id一般是自增的。

4.主键索引是不能重复的，也不能为空

1. 唯一索引的创建：

在创建表的时候，指定某列或者某几列为唯一索引。

把表创建好后，再指定某列或者某几列为唯一索引。

Create unique index 索引名 on 表名 (列名);

Alter table 表名 add unique [索引名] (列名);

唯一索引的特点：

1.一张表可以有多个索引

2.唯一索引不能重复，但是如果你没有指定not null，唯一

索引列可以为null值，且可以有多个。

3.什么时候使用唯一索引，当某列数据不会重复，才能使用。

4.唯一索引效率也很高，可以优先考虑使用。

3.普通索引的创建：

把表创建好后，再指定某列或者某几列为普通索引

Create index 索引名 on 表名(列名);

Alter table 表名 add index [索引名] (列名);

普通索引特点：

1.一张表中可以有多个普通索引，一个普通索引可以指向多

列。

2.普通索引列的数据可以重复。

3.效率相对而言较低。

1. 全文索引创建：

概述：全文索引主要是针对对文章，汉字，英文，数据量比较大的检索，可以快速的检索到文章中的某个关键字。

使用：

Select \* from 表名 where match(字段名...) against(‘关键字’);

特点是：

1.mysql默认的全文索引，只对myisam存储引擎

2.mysql默认的全文索引，只支持英文。

3.停止词：对于特别普通的字母，不会建立全文索引。

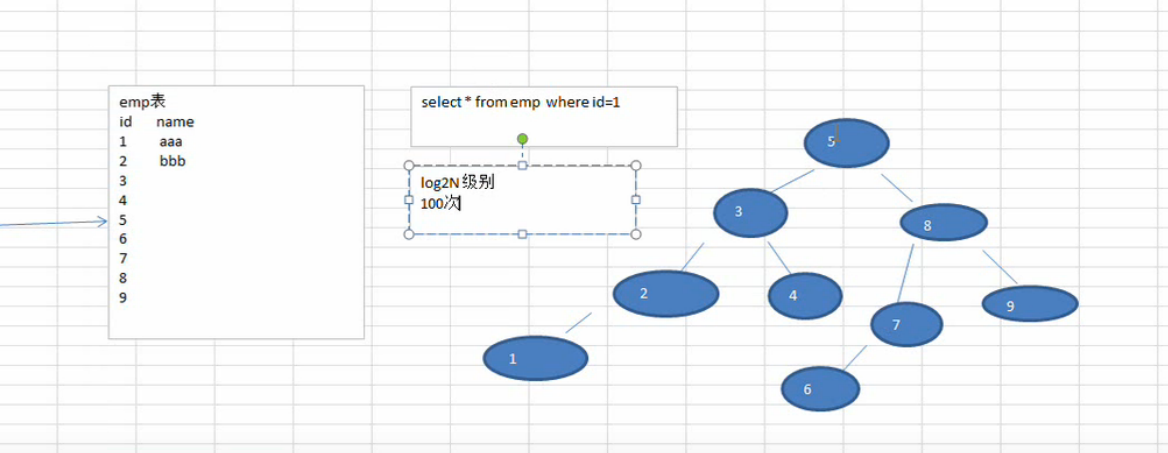
4. 匹配度： 全文索引是按一定概率来匹配的。

如何解决mysql全文索引不支持中文问题？

1.使用mysql的一个中文检索插件mysqlcft,具体看文档

2.可以使用专门的中文检索引擎sphinx中文版（coreseek）

* 索引的查询
  + 1. Desc 表名
    2. Show key from 表名
    3. Show index from 表名
    4. Show indexes from 表名
* 索引的修改
  + - 先删除，再添加
* 索引的删除
  + DROP INDEX 索引名 ON 表名;
  + ALTER TABLE 表名 DROP INDEX 索引名;
* 索引的原理



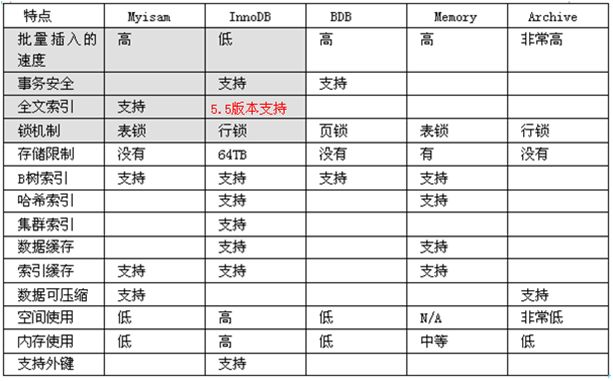
为什么添加的时候会慢呢，要维护这个二叉树。

* 索引的注意事项
  + - 比较频繁作为查询条件的字段应该建立索引
    - 唯一性太差的字段不适合单独建立索引，即使它频繁作为查询条件（性别）
    - 更新太频繁的字段不适合创建索引（登录次数，用户状态）
    - 不会出现在where字句中的字段不该创建索引

1. mysql语句的优化和正确使用索引

* 如何正确使用索引？
  + 对于创建多列（复合）索引，只要查询条件使用了最左边的列，索引一般都会被使用。
  + 对于like的查询，查询如果是’%aaa’或者是’\_aaa’，不会使用到索引，其他的’aa%a’,’aaa%’,’aa\_a’,’aaa\_’则会使用索引。
  + 下列的表将不使用索引：
    - 如果条件中有or，则要求or中所有列都要有索引，否则不使用索引。
    - 对于多列索引，不是使用的第一部分，则不会使用索引。
    - Like查询是以%，\_开头。
    - 如果列类型是字符串，那一定要在条件中将数据用引号引起来。否则不使用索引。
    - 如果mysql估计使用全表扫描比使用索引快，则不使用索引。
    - 优化group by语句：默认情况下mysql对所有group by col1,col2进行排序，这与在查询中使用order by col1,col2类似，如果查询中使用group by但是想避免排序结果的损耗，则可以使用order by null禁止排序。
    - 在可以的情况下，尽量使用连接（join）查询来代替子查询，因为在使用join的情况下，mysql不需要在内存中创建临时表。

1. 选择合适的存储引擎：



* MyIAM:默认的mysql存储引擎，如果是以读操作和插入操作为主，只有很少的更新操作和删除操作，并且对事务的完整性要求不是很高。其优势是访问速度快（适合论坛的帖子/信息表/新闻表/商品表）
* InnoDB:提供了具有提交，回滚，和崩溃恢复能力的事务安全，但是对比起MyISAM,写的处理效率会差一点并且会占用更多的磁盘空间（如果对安全要求相对高，建议用InnoDB）。
* Memory：一些访问频繁，变化频繁，又没有必要入库的数据，如用户在线状态。
  + **特别说明memory存储引擎的使用**：

create table memory(

id int primary key ,

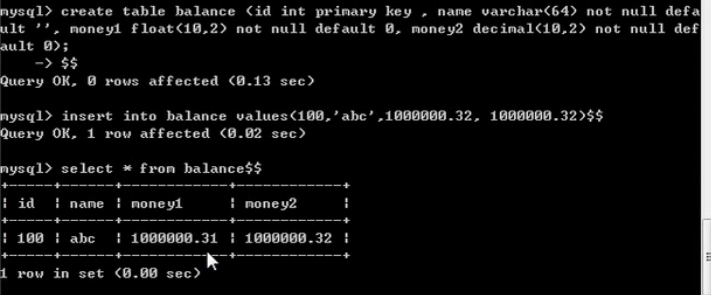
name varchar(32) not null default '',

status tinyint not null default 0

)engine=memory charset=utf8;

1. 数据本身存在内存中，因此操作速度很快
2. 当mysql重启后，数据丢失，但是表的结构还在。
3. **如何选择正确的数据类型**

* **在精度要求高的应用中，建议用定点数（decimal）来存储数值，而不用(float),以保证结果的准确性.**
  + **举例：**



这里我们看，float(10,2) 和 decimal(10,2),decimal(10,2)更加精准。

* 对存储引擎是MyISAM的表，要定时碎片管理
  + 举例说明：当我们在users表中有大量数据时，我们delete数据后，发现磁盘空间没有回收，因此我们要定时的进行碎片整理：
    - optimize table 表名;
    - 主从复制，自我复制（insert into 表名 select \* from 表名;）
* 如何解决2k38问题

