**第二十六节知识点:**

1. **范式化和反范式化  
    1. 过分的反范式化为表建立太多的列**

**不要在一张表中建立太多的列**

**为什么?**

**答：我们在设计数据库的结构时，比较容易犯的一个错误，就是对表进行了过分的反范式化的设计，就会造成一个表中的列过多，虽然说Mysql允许为一个表建立了很多的列，这是由于Mysql的插件式架构的原因，Mysql服务器层和存储引擎层是分离的，Mysql存储引擎API，工作时需要把在服务器层和存储引擎层之间，通过缓冲格式来拷贝数据，然后在服务器层将缓冲中的内容解析成个列，操作过程的成本是非常高的，特别是对于MyISAM这种变长结构和InnoDB这种行结构，在进行解析时还必须进行转换。**

**这个转换的成本就依赖于列的数量，所以如果一个表的列太多，在使用这个表示就会带来额外的CPU的消耗，所以大家在进行表设计的时候一定要注意，不要把所有的相关的列全放到一个表中，而是要按照范式化适当的对表进行拆分**

**2．过分的范式化造成太多的表关联**

**不要把一张大表分成太多的表，或者将常用的的两个小表和并成一个大表**

**为什么?**

1. **表关联成本是很高的**

**扩展知识:mysql最多只能关联61个表，不过我们要把控在10个以内；**

**3．在OLTP环境中使用不恰当的分区表**

**分区表是个好东西，可以帮助我们把一个大表物理上分成多个小表，但是逻辑上没有分**

**分区表不同与分库 分表**

1. 分区表是在同一个数据库实例上所进行的
2. 分区表**物理储存上分，逻辑不分**

**注意的是:**

**使用分区表 ，分区键一定要选好**

**4．使用外键保证数据的完整性**

**外键约束**

**第二十七节知识点:**

1. **对mysql优化的顺序**

**总结:**

**问：按照对性能影响的严重程度，我们对MySQL服务器级进行优化时，应该按什么顺序来进行？**

**答：**

1. **数据库结构设计和SQL语句。**

**【这包括表结构的优化、索引的优化以及SQL语句的优化等内容，也是优化最容易看到成果的，另外这一部分优化并不是做一次就可以了，而是要随着应用的生命周期不停的进行。因为随着应用环境和程序的变更，这一部分的优化很可能会出现新的问题，所以需要我们持续的来关注】**

1. **数据库存储引擎的选择和MySQL参数配置**

**【就在于存储引擎的选择和缓冲池的配置，大家最好不要混合使用不同的存储引擎】**

1. **系统选择及优化**

**4．硬件升级**

**【最后两个对性能的影响是有限的，比如：我们的服务器只有64G的内存，在这种情况下，无论我们上面怎么优化，也不可能达到256G内存的服务器应有的性能】**

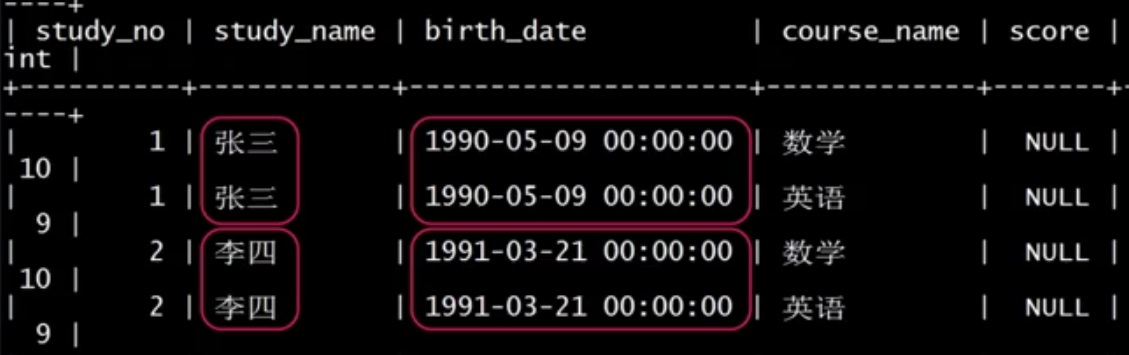
**数据库结构优化:**

1. **数据库优化的目的**

**第一节：**

**数据库结构优化的目的:**

1. **减少数据冗余**
   1. **什么是数据冗余?**
      1. **比如说数据库中的某列可以根据其他列计算得到**



**比如这个表数据，姓名和生日重复出现，冗余了**

**完全可以根据学生id （study\_no）获取到学生的这些东西**

1. **尽量避免数据维护中出现更新，添加，删除异常**
   1. **添加异常:**
   2. **更新异常:**
   3. **删除异常:**
2. **节约数据存储空间**
   1. **当数据没有冗余相对的所需要的空间也就小了**
3. **提高查询效率**

**第二节:**

1. **数据库从哪方面入手**

**数据库设计从哪方面入手:**

1. **首先要贯穿项目的始终。把握全局**
2. **需求分析:全面了解产品设计的存储需求(超级重要)  
   1) 存储需求 ， 要存什么类型的数据**

**2) 数据处理需求 ，是怎样处理数据的，**

**3) 数据的安全性和完整性，**

1. **逻辑设计:**
   1. **数据实体之间的逻辑关系，解决数据冗余和数据库处理异常。（就是遵从数据库范式。）**
2. **物理设计;根据数据库特点进行表结构设计(如引擎，)**
3. **维护优化:根据实际情况对索引，存储结构等优化**
4. **数据库三范式**

**数据库范式:**

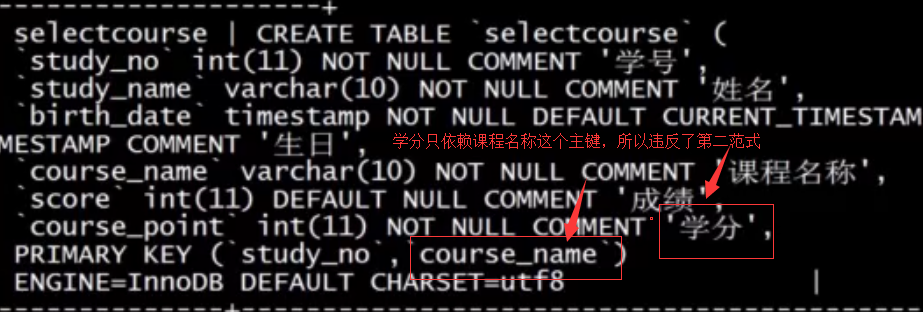
**在大部分情况下要根据范式来定**

**1范式:**

1. **字段不可再分**
2. **单一属性的列都是由基本的数据类型构成(string等)**
3. **都是二维表**

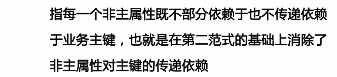
**2范式(在第一范式基础上)**

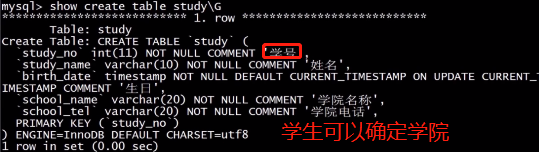
1. **要求一个表中只具有一个业务主键，也就是说符合第二范式的表中不能存在非主键列对只对部分主键的依赖关系**



**如果是复合主键，则就有可能不复合第二范式**

**3 范式**





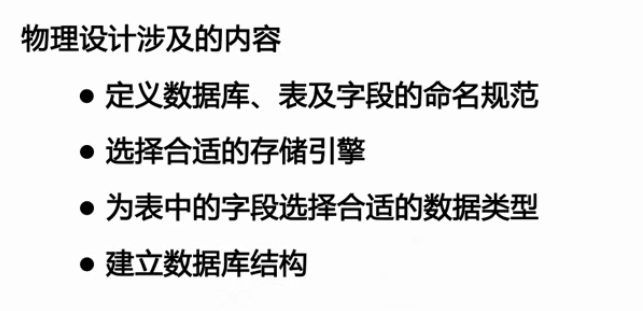
**这个表中，主键学号，决定了学院名称，而学院名称又决定了学院电话，充分的说明了依赖传递。违反了第三范式。**

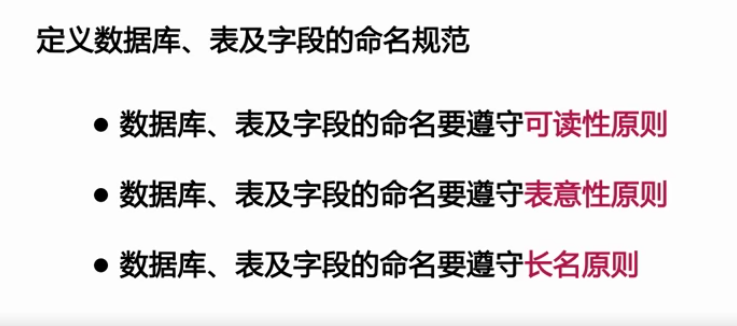
**总结:**

结论：第三范式该造后不会有数据冗余和数据维护异常。

**第三节:**

**根据范式化之后在根据业务来看进行反范式化**







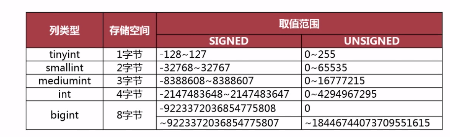
1. **数据库存储类型优化**

**第七节课:**

1. **数据页: mysql存储数据是以页为单位的，存储的数据量是一定的（innodb是16k）列的长度越小等于在数据页中容纳的行数越多，这样在加载同样的数据的时候，加载宽度较小的类型要比宽度较大的类型的数据页要少，这样减少了空间的使用，和IO的次数**
2. **为表中字段选择类型。**
   1. **当一个列可以选择多个数据类型的时候**

**数字 > 日期与二进制 > 字符型**

**原因:**

1. **对于排序，数字直接就能比较大小。而字符则要按照规定的排序规则；**
2. **对于所占空间大小:**
3. **对于**
4. **整数类型**
   1. 
   2. **根据上图所知，不管设置多少，其空间都是为存储空间所写那样。如 int(8) 和 int(11)存储空间都为4字节，所以我们根据所需选择不同的类型**
5. **实数类型**
6. **Varchar 和 char**

**（以字符为单位，1字符根据字符集不同需要不同字节，urf8 1字符= 3字节）**

**Varchar 类型:**

1. **可变，只占用必要的空间**
2. **<255 需要一个字节记录字符串的长度**
3. **>255 需要两个字节记录字符串的长度**
4. **如何确定宽度**
5. **使用最小的符合需求的长度**
6. **虽然varchar(10) 和varchar(100) 存储mysql**

**这个字符的空间大小都是一样大的 但是还是要用最小的符合需求的长度 ，因为这样能够节省内存优化查询。内存中的长度都是固定的。**

1. **字符串的最大宽度与平均宽度差不了多少的**

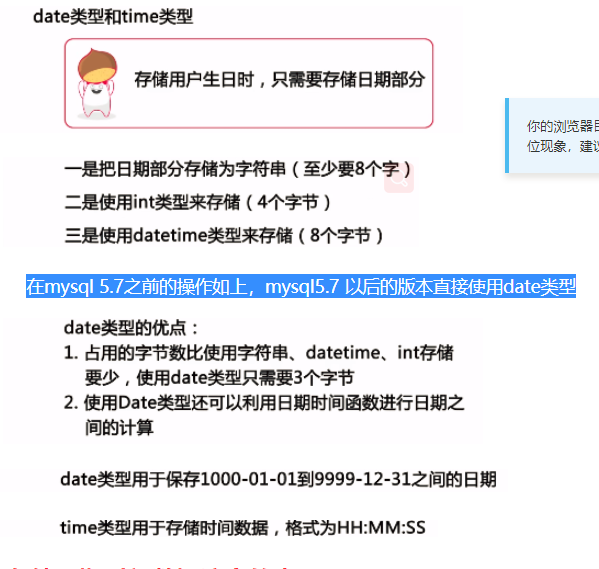
**（也就是 宽度差距比较大的）**

1. **字符串列很少被更新**
2. **Char类型是定长的**
3. **存储时char类型的数据后面的空格会自动删除**
4. **最大宽度255**
5. **适合存储字符串长度差别不大的 如MD5 手机号，，，，，**
6. **还有短小的字符串如 varchar(1) 和char(1)**
7. **适合存储经常被更新的-**

**时间存储:**

**四个选择:**

1. **Datetime**
   1. **八个字节**
   2. **不跟着时区变化**
2. **Timestarp**
   1. **4个字节**
   2. **跟着时区变化**
   3. **修改数据可以自动修改（只有第一个timestarp属性列有这个属性）**
3. **Date**
   1. **三个字节**
   2. **存年月日**
4. **Time**
   1. **存时分秒、**
   2. **三个字节**



1. **高可用架构设计：**

**Mysql复制功能:**

1. **实现在不同服务器上的数据分布**
   1. **利用二进制日志实现**
   2. **基于语句复制 ，不许太多带宽**
   3. **基于行的复制进行批量更改，会造成带宽压力**
2. **数据读取的负载均衡(读取分离)**
   1. **需要其他组件配合完成(DNS轮询等)**
3. **增加数据安全性**
   1. **利用备库的备份来减少主库负载**
   2. **复制不代表备份。（如主库误删数据后，由于主从延迟小。从库可能也跟着删了。那么数据只能靠备份恢复。）**
4. **实现数据库的高可用和故障切换**
5. **实现数据库在线升级，（用高版本的mysql从库）**

**Mysql 二进制日志：**

1. **作用**
   1. **数据库的复制，用于主从**
   2. **数据库的恢复，如果数据库在两次备份之间宕机，那么就要用于二进制日志进行基于时间点的恢复了。**
   3. **能够没事看两下，看看都做了哪些操作。**
2. **二进制日志的格式**

**Mysql 中 DDL 都是基于段的格式**

* 1. **基于段的格式（存sql语句）statement**

**优点:**

* + - 1. **存的都是sql语句 。日志量小。减少磁盘I/O，降低带宽压力 （在修改的行多的情况下。如果只修改一条row格式比段格式还小呢）**
      2. **不要求从库中的列和主库上的列一致。还能够对类型的兼容。因为是sql语句。对这些没要求；**
      3. **基于行更灵活。想看就看**

**缺点:**

**1. 用特定的函数如 UUID() USER()等非确定函数是无法复制的。可能造成主从数据不一致**

**2. 数据不一致容易主从链接断开**

**3. 在主库上执行的时候锁定了多少行数据在从库上也要锁定多少行数据，因为sql语句又执行了一遍。 所以没有基于行的高效**

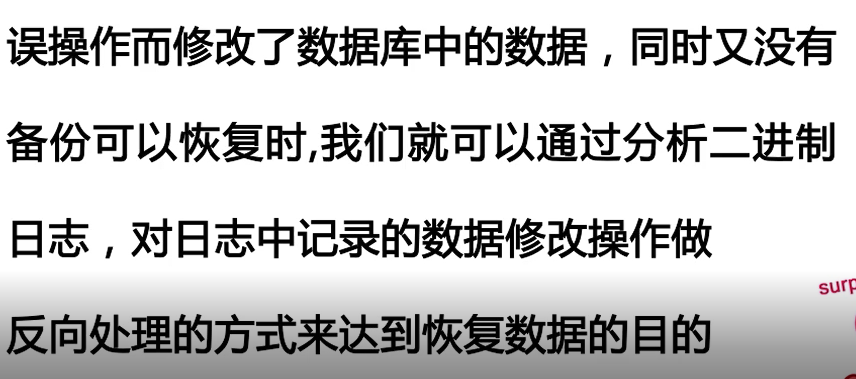
* 1. **基于行的格式（存修改后的数据）row**

**优点:**

**1. 安全。传到从上面的都是数据。直接在从库数据上面修改就行了。保证数据一致**

**2. 对每一行的数据比基于段的还高效，如update demo set name =‘aa’这个在基于段的时候会同样锁定整个表。在基于行的时候不会锁表**

**3. 还会有点小惊喜:**



**缺点:**

**1. 记录量大,如基于段的格式在修改1000条数据只存了个sql语句。但是在基于行的日志格式就存了对1000条数据的修改。增加了磁盘I/O。不过其实还可以接受。因为日志都是顺序写入。**

**针对这个缺点。Mysql官方给了个参数来安慰一下:**

**Binlog\_row\_image = [FULL|MINIMAL|NOBLOB]**

**FULL : 数据修改记录所有列的数据。不管你修改多少列;**

**MINIMAL: 只记录修改的列的数据**

**NOBLOB: 和FULL差不多，只不多不修改text列则不记录text列**

**2. 要求数据库的表结构相同，否则可能会中段复制**

**3. 无法在从上单独执行触发器**

* 1. **混合日志格式MIXED**

**根据sql语句由系统选择基于行和段**

**数据量大小由所运行的sql决定**

**建议使用 混合格式 和 ROW格式**

1. **Mysql 主从工作方式**
   * 1. **主 将变更写入二进制日志 ，并开启一个转储线程（记得开二进制日志）**
     2. **然后 从 通过i/o线程去请求 主 的二进制。并写入relay log 日志中。**
     3. **从 通过sql线程重放relay log日志**

**MySQL复制进行**

[**https://blog.csdn.net/min903245074/article/details/78081052**](https://blog.csdn.net/min903245074/article/details/78081052) **详细步骤网址。**

* + - 1. **基于日志点的复制**

1. **建立复制账号**

**create user 'xusong'@'123.56.16.17'identified by '123456';**

1. **授权该账号**

**GRANT REPLICATION SLAVE ON \*.\* TO ‘xusong’@'123.56.16.17';**

1. **配置主数据库服务器**
   1. **开启二进制日志 bin\_log = mysql-bin**

**这里指定了二进制的名字mysql-bin**

* 1. **Server\_id = 100**

1. **配置从数据库服务器**
   1. **Bin\_log = mysql-bin**
   2. **Server\_id = 101**
   3. **Relay\_log = mysql-relay-bin**

**对于relay\_log这个参数默认会启动，但是它的默认名字是主机名字，当我们修改了主机名，这时候再重新启动主从复制的时候会找不到，所以我们指定一个名字；**

* 1. **Log\_slave\_update = on [可选]**

**是否将sql线程存放的中继日志，是否放在二进制日志中，用来做从为其他主的时候**

* 1. **Read\_only = on [可选]**

**安全配置参数，禁止没有super权限的用户对这个服务器进行写操作。建议开启**

**初始化从服务器数据**

1. **Mysqldump （会锁表，会导致阻塞）**
2. **Xtrabackup –slave-info（对innodb不锁表）**

**启动复制连路（从服务器上进行）**

**mysql>change master to master\_host ='master\_host\_ip', aster\_user='repl',master\_password='password',master\_log\_file='mysql\_log\_file\_name',master\_log\_pos = 4;**

**#master\_log\_pos 日志偏移量，从什么节点进行复制**

**日志偏移量可以在主上面:**

**show master status 查看**

**等配置好了之后。重启mysql就完事了**