# 数据库第二天

2014年10月13日 星期一　　上午8:48

1.外键约束

1> 建立两张表的联系

2> 以微博为例,每个数据模型都建立一张表，微博表和用户表

3> 微博表应该有个字段描述这条微博是哪个用户发的

4> 微博添加一个用户id字段，但是这个字段不严谨，可以为空，而且id可以是用户表中不存在的id.

5> 应该用外键建立两个表之间的联系，外键是某个表的字段是另外一个表的主键。

6> 外键加在一对多的表，学生和班级，一个班级对应多个学生，在学生表添加外键

7> sql语句创建学生表

8> 建立外键的表，不能轻易删除

2.表连接查询

1> 同时查询两个表中的数据，同时查学生姓名和班级名称

- 必须取别名，来区分相同名字的字段
- 需要精确查找，只查找有联系的数据，用where过滤。

2> 查询某个班级的所有学生

- 直接通过id查询
- 嵌套查询，先通过班级名称查询id，在根据id查询学生
- 表连接查询，利用两个表之间的联系，查询数据

3.FMDB：处理数据库的框架

好处：

1> 用OC开发，避免c语言，简化开发。

2> FMDB的更新包含增删改

- 插入数据的占位符是?，这个是数据库的占位符，不能用%d,%@,这些，数据库是不识别的。
- FMDB的占位符只能是对象。
- 用占位符?就不需要给text类型字段的值添加"单引号，会自动添加.
- 用占位符?进行模糊查询，%不需要转义，因为%在stringWithFormat才有特殊函数，在普通字符串常量没有特殊含义，就不需要转义.

3> FMDB有专门做查询的方法，会返回一个结果集

4> 这个结果集是一个指针，默认不指向任何结果，只有调用next才会去指向下一条记录

5> 注意数据库多线程安全问题，可能会导致同一时间同时操作一个数

据，为了避免这个问题，一般都使用多线程安全的FMDB数据库实例。

6> 不涉及修改数据，就不需要考虑多线程，离线缓存就不需要用多线程

4.数据库多线程和事物：

- FMDB会帮我们创建一个线程安全的数据库实例，并且自动打开，查看源码。
- GCD里面是线程安全的，在里面访问数据会自动加锁。

1> 银行取钱存钱例子，同时取钱和存钱，操作一个数据，可能会造成数据不对。

2> 这时候需要通过多线程管理，给线程加锁。

3> 事务：把有联系的业务逻辑划分到一个事务里，比如转账，比如必须两个同时修改成功，才能提交。

- 注意：即使不提交事务，也会修改成功，需要判断操作有没有执行对，如果发现事务里有一个操作不对，就主动回滚。
- 注意把操作放在事务里，并不会自动有一个操作失败就会主动回滚，需要我们自己处理，否则就会有有些执行，有些没执行。

5.微博离线缓存

- 把微博数据缓存起来，下次就不用重新加载。

1> 离线缓存业务逻辑，先从数据库取，取不到再去发送请求。

2> 在微博工具类做判断，微博工具类有个专门提供微博数据的方法，在这里先从数据库里取，没有取到就去加载。

3> 搞个专门处理数据库的业务类。

- 好处：业务逻辑清晰
- 以后换框架了，直接改这个类。
- 在initial方法创建数据库，不需要用多线程。
- 数据库文件存储到cache文件夹。

4> 创建表格，分析字段

- id 主键
- idStr 微博id 对应请求参数，根据他才能准确查找微博数据,字段约束不为空，唯一。
- accessToken 获取令牌-保证一个用户对应一个客户端 对应请求参数，能找到对应用户的微博数据。
- dict 微博数据
- 数据库字段技巧：用请求参数，请求结果，作为字段，保证通过某个字段能查到结果。

5> 数据库业务逻辑

- 提供保存微博数据的方法
- 提供获取微博数据的方法。
- 获取微博数据的方法，需要判断sinceID和maxID有没有值，有值才需要根据id去查找

需要根据idstr去查找。

6> 在获取到新的数据，记得存储到数据库

6.清空图片缓存和计算缓存尺寸

- 用SDwebImage有个管理缓存的单例
- 清空图片缓存
- `[[SDImageCache sharedImageCache] clearDisk];`
- 计算缓存尺寸
- `[[SDImageCache sharedImageCache] getSize]`

7.计算缓存文件尺寸的底层实现

- 文件夹获取的文件尺寸不准确，需要遍历所有文件，把文件夹里所有文件的尺寸加起来算出文件夹尺寸。
- 判断路径是否存在
- 判断路径是否是文件夹
- 如果是文件夹，就遍历所有的子路径，把所有子路径的尺寸加起来