# 初识Mysql

JavaEE:企业级java开发web

* 前端：页面展示数据
* 后台：连接数据库，给前端传递数据，控制页面视图跳转
* 数据库：存储数据

## 为什么学习数据库

时代的趋势：大数据时代，顺势而为！

数据库的作用：存储数据

数据库的存在意义：数据库是所有软件体系中最核心的存在

## 什么是数据库

数据库DataBase

概念：数据仓库，作为软件安装在操作系统上的软件。windows和linux等

作用：存储数据，管理数据

## 数据库分类

关系型数据库：通过表和表之间，行列之间的关系进行数据的存储

* Mysql
* Oracle
* SqlServer
* DB2

非关系型数据库：非关系型数据库，存储对象通过对象自身的属性来决定。

* Redis
* MongDB

## DBMS数据库管理系统

数据库的管理软件，可以科学有效的管理数据，维护和获取数据

## Mysql简介以及工具

Mysql是最好的关系型数据库管理系统应用软件之一，必须会

官网：<https://www.mysql.com/>

SQL图形化工具SQLyou

## 数据库命令基本操作

基本命令行操作：

* 命令行连接：
  + mysql -u root -p
* 查看数据库：
  + show databases;
* 选择数据库：
  + use database;
* 查询数据库中的表：
  + show tables;
* 查询表的结构信息：
  + describe table;
* 退出：
  + exit
* 强行终止：ctrl+c

### 数据库语言分类

核心CRUD增删改查

* DDL数据库定义语言defination
* DML数据库操作语言manipulation
* DQL数据库查询语言query
* DCL数据库控制语言control

# 操作数据库

操作数据库》操作数据库中的表》操作表中的数据

规范：表名，列名要用``括起来，分号表示sql语句的结束

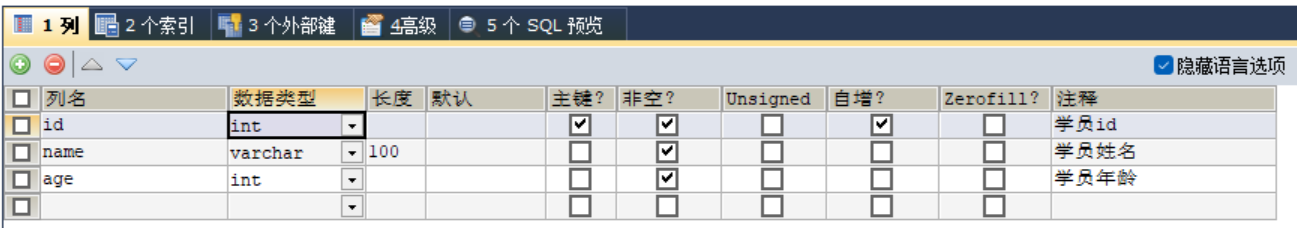
mysql关键字不区分大小写！

* 操作数据库
  + 创建数据库
    - CREATE DATABASE [IF NOT EXISTS] test0 CHARACTER SET utf8 ;
  + 删除数据库
    - DROP DATABASE [IF EXISTS] test0
  + 使用数据库
    - USE test0
  + 查看所有的数据库
    - SHOW DATABASES

## 数据库中的数据类型

* 数值
  + tinyint 1个字节
  + smallint 2个字节
  + int 4字节
  + bigint 8字节
  + float浮点数4字节
  + double双精度8字节
  + decimal(p,d)字符串形式浮点数，金融计算的时候一般使用decimal
    - p表示有效数字的精度，在1-65之间
    - d表示小数点后位数，范围0-30，d<=p
* 字符串
  + char 字符串固定大小0-255
  + varchar可变字符串0-65535常见的String
  + tinytext 微型文本2^8-1
  + text 文本串2^16-1 保存大文本
* 时间日期：对应java.util.Date
  + date YYYY-MM-DD日期格式
  + time HH-MM-SS时间格式
  + datetime YYYY-MM-DD HH-MM-SS最常用的时间格式
  + timestamp 时间戳全球同一
  + year年份表示
* null
  + 空值或者未知
  + 不要使用null进行运算，运算结果为null

## 数据库表的字段属性



* Unsigned
  + 无符号整数
  + 不能为负数
* zerofill
  + 0填充
  + 不足的位数用0填充
* 自增
  + 理解为自增，自动在上一条记录的基础上+1
  + 通常用来设计唯一的主键，必须是整数类型
  + 可以自定义设计主键自增的起始值和步长
* 非空
  + 数据不允许为空
* 默认
  + 设置数据的默认值

## 数据库表规范

规范表示每一个数据库表都必须存在的字段：

* id主键
* ‘version’乐观锁
* is\_delete 伪删除
* gmt\_create创建时间
* gmt\_update修改时间

## 创建数据库表



列名 数据类型 属性 属性 属性…

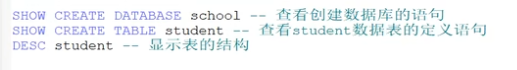
列名和数据类型必填，注意引擎设置和编码

mysql文件夹下的my.ini文件是mysql的配置文件，可以配置默认编码和端口等

default null:默认为空

comment:注释内容

常用命令：



## 数据表的类型

INNODB 默认使用

MYIASM 早期使用



物理空间上的存在位置：所有的数据库文件都存在data目录下，本质还是文件存储

Mysql引擎在物理文件上的区别：

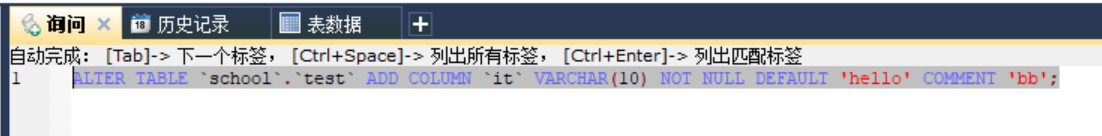
* Innodb在数据库表中只有一个\*frm文件，上级目录的ibdata1文件
* MYIASM对应的文件
  + \*.frm 表结构的定义文件
  + \*.MYD 数据文件data
  + \*.MYI索引文件index

## 修改表结构

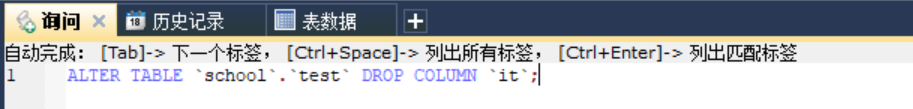
表结构的增删改：

alter table database.table\_name：

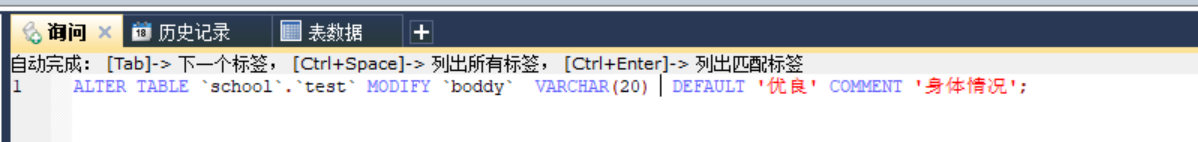
* add column <列名> <数据类型><完整性约束> after 列名 ：在列名之后添加新的列，没有after就默认在表的最后一列添加新的列。



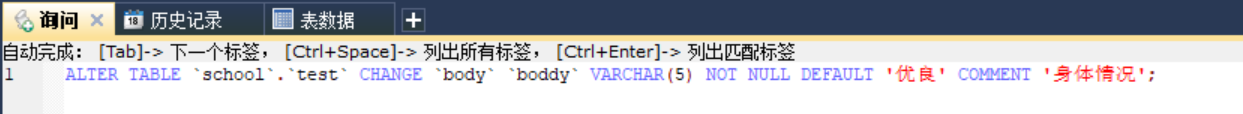
* drop <列名>：删除列



* modify <列名><列属性> ：modify只能改属性



* change<列名><新列名><新列名属性>：change属性列名都可以更改



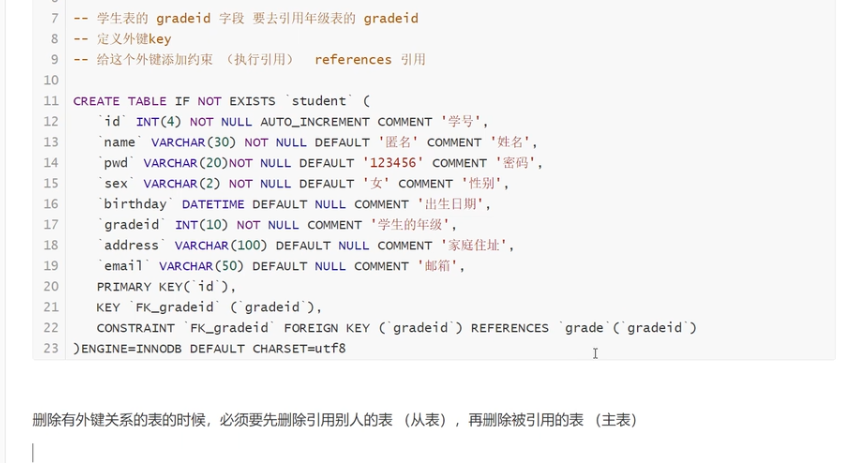
改表名：alter table table\_name rename to new\_table\_name;

删除表：drop table if exists `数据库名`.`表名`;

删除数据库：drop database `数据库名称`;

## MySQL数据管理

### 外键





从表设置外键引用主表！以上，student表为从表，grade表为主表。

从表设置的外键连接主表的主键！且外键和主表的主键数据类型要匹配上！

以上操作都是物理外键，数据库级别的外键。不建议使用！

最佳实践：

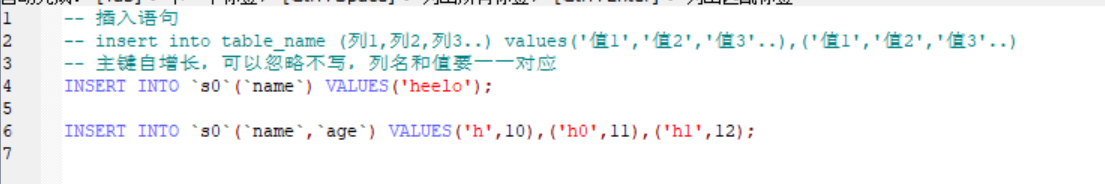
* 数据库就是单纯的表，只用来存数据，只有行和列！
* 我们想使用外键，通过程序去实现！

### DML语言

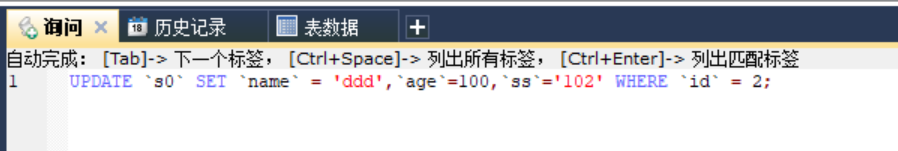
数据库意义：数据存储和数据管理

DML语言：数据操作语言

* 添加：如果字段设置非空，则插入数据时不能忽略



* 修改：update `表名` set `列名`=value，`列名` = value where 条件



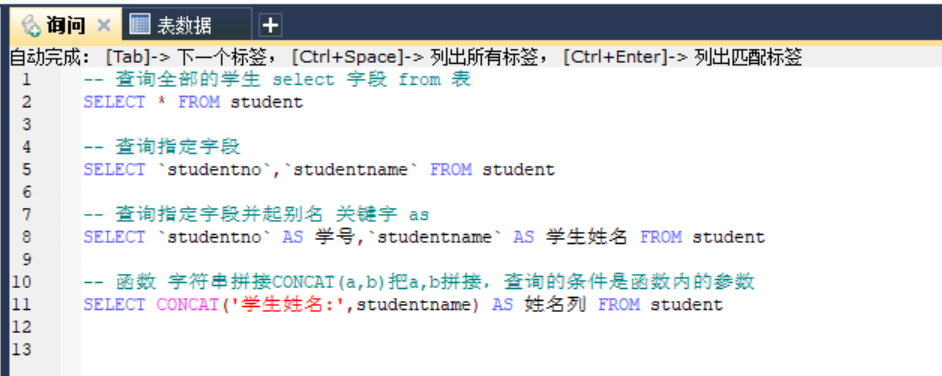
* + 条件where子句返回布尔值，true则执行，false不执行
    - 条件语句操作符
      * <> !=:不等于
      * >:大于
      * <:小于
      * between x and y:在x和y之间！
      * and:与运算
      * or:或运算
* 删除：delete from 表名 where 条件
  + 清空表：delete from 表名 truncate table 表名
    - truncate 清空表，自增量归0
    - delete清空表，自增量不变

delete删除数据的问题：

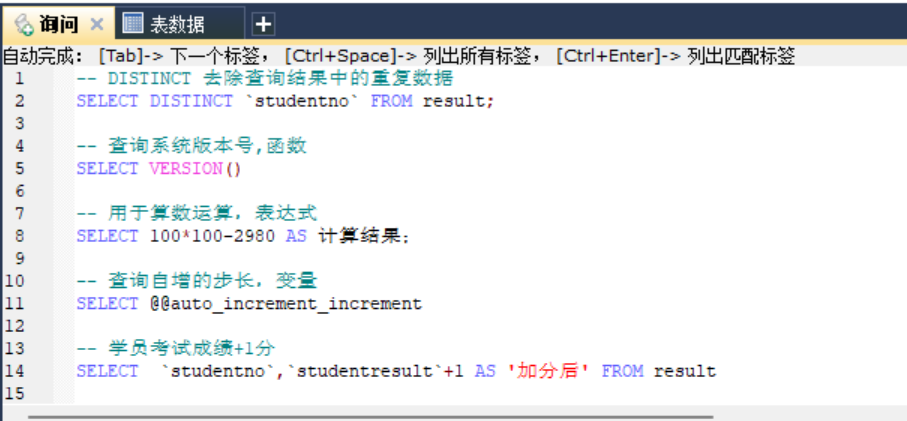
* 引擎是InnoDB,数据库重启则自增量从1开始(断电即失！)
* 引擎是MyISAM,数据库自增量不变！(存在文件中的)

### DQL数据查询语言\*\*\*

* 所有的查询操作都用它
* 简单查询，复杂查询
* 数据库中最核心的部分
* 使用频率最高的语言



数据库中的表达式：文本值，列，null，函数，计算表达式，系统变量等等

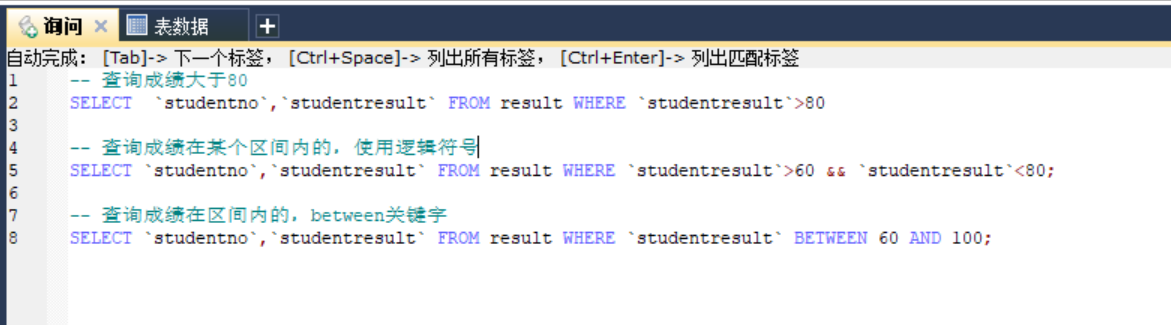


### where条件字句

作用：检索数据中符合条件的值

逻辑运算符：建议使用英文字母

* 与：and &&
* 或：or ||
* 非：not !



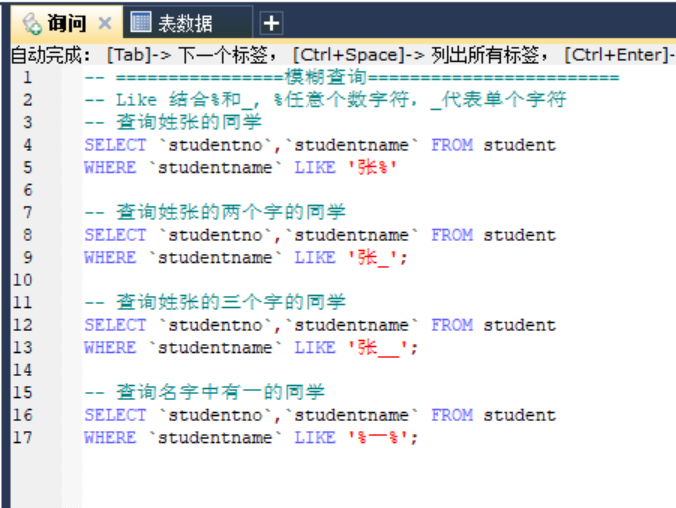
### 查询完整语法\*\*\*

各个关键字的上下位置不能调换，优先级都是确定的！



### 模糊查询

|  |  |  |
| --- | --- | --- |
| 运算符 | 语法 | 描述 |
| IS NULL | a is null | 如果a为null,结果为真 |
| IS NOT NULL | a is not null | 如果a不为空，结果为真 |
| Between and | X Between a and b | 如果x在a和b之间，为真 |
| Like | a like b | SQL匹配，a匹配b，为真 |
| In | A in (a1,a2,a3…) | A为括号内的某个值，为真 |

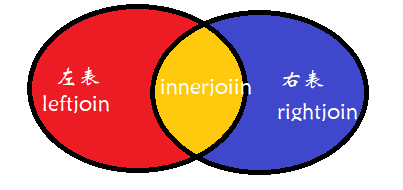
like关键字使用，通配符%和\_

in 关键字的使用

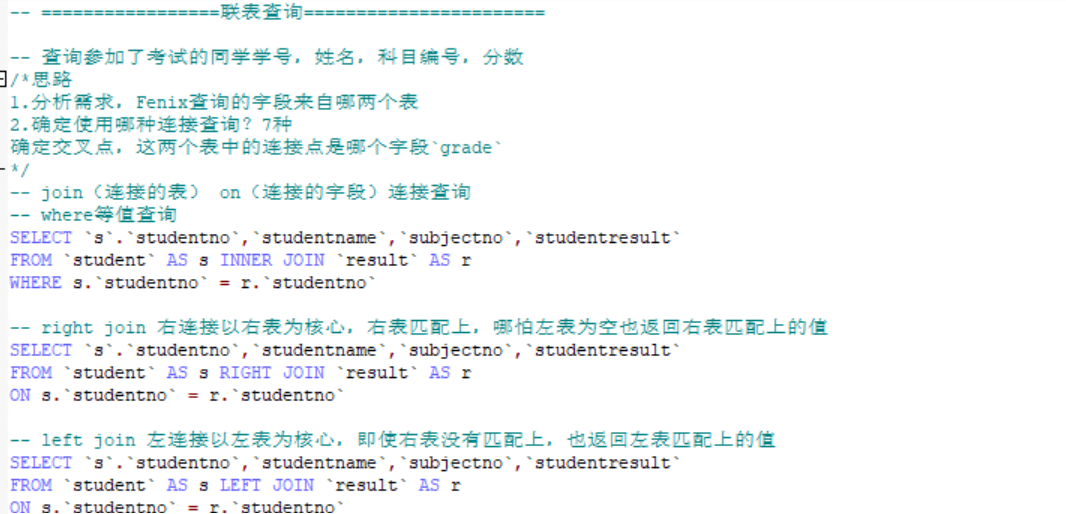
is null 和 is not null

### 连表查询

本质上是三种连接：左连接，右连接，中间连接



|  |  |
| --- | --- |
| 操作 | 描述 |
| Inner join | 如果表中至少有一个匹配，就返回行 |
| Left join | 从左表中返回所有的值，即使右表中没有匹配 |
| Right join | 会从右表中返回所有的值，即使左表中没有匹配 |



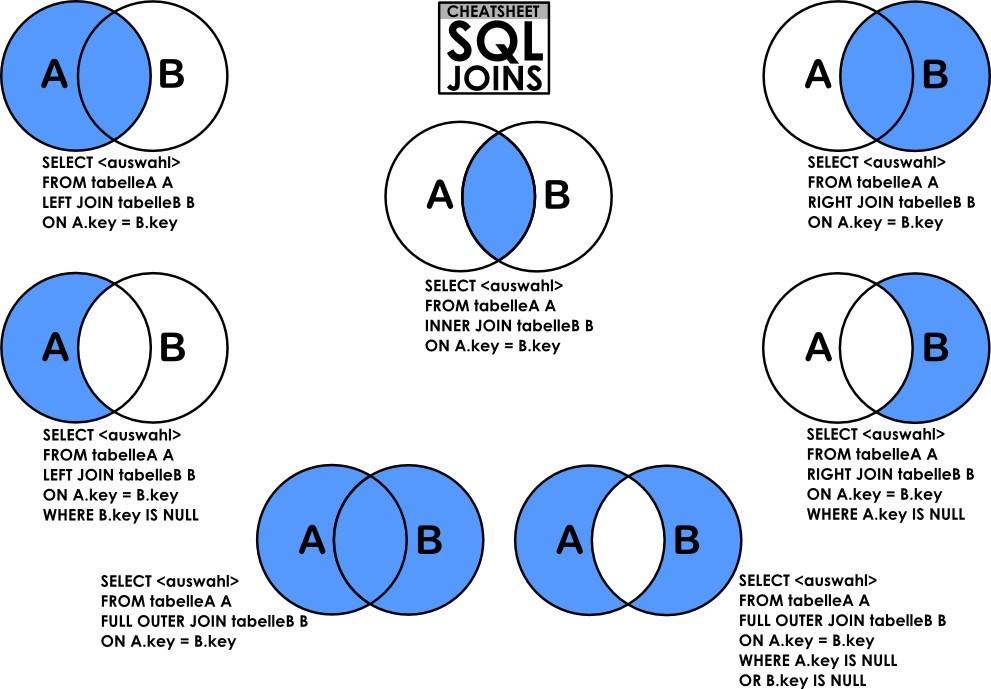
思路：

确定自己要查询哪些字段，这些字段来自哪些表。

根据想要的结果确定哪张表为核心（确定是left join,right join,innerjoin以及连接字段），

如果是多表连接，则先连接两张表，得出想要的结果在继续连接下一张表！

三种连接查询细分为七种连接：



多表连接查询：

select a.xx b.xx c.xx ……

from a

left join b

on a.xx = b.xx

left join c

on b.xx = c.xx

left join d

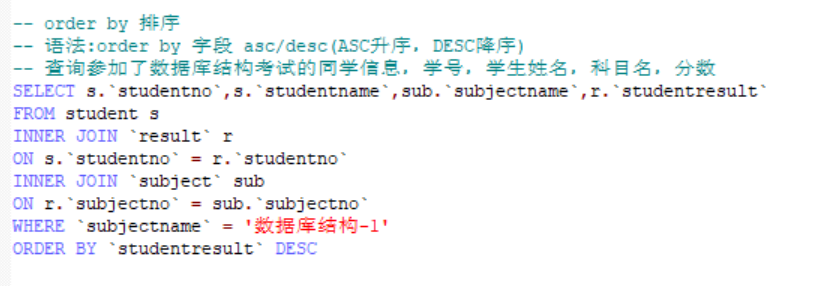
…

### 自连接

把一张表看做两张一模一样的表进行自连接，因为表名一致，所以有起别名的必要性

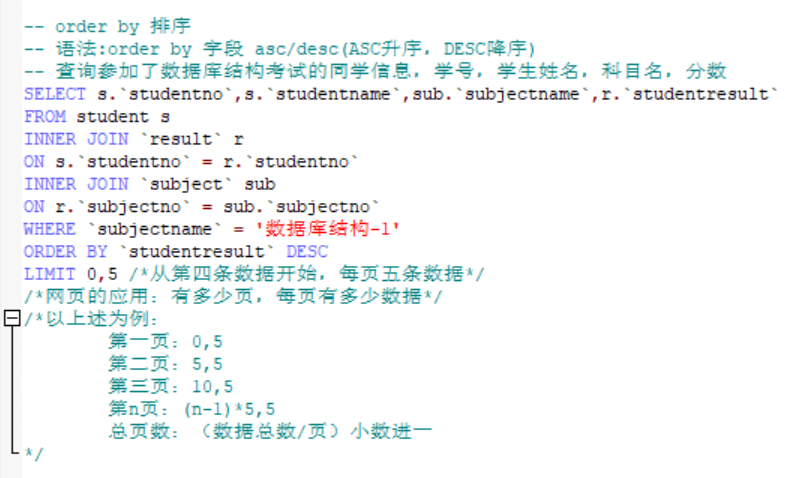
### 分页limit和排序order by

* 排序：order by 字段名称 asc/desc



* 分页limit：数据起始值位置索引(从0开始)，一页的数据条数据

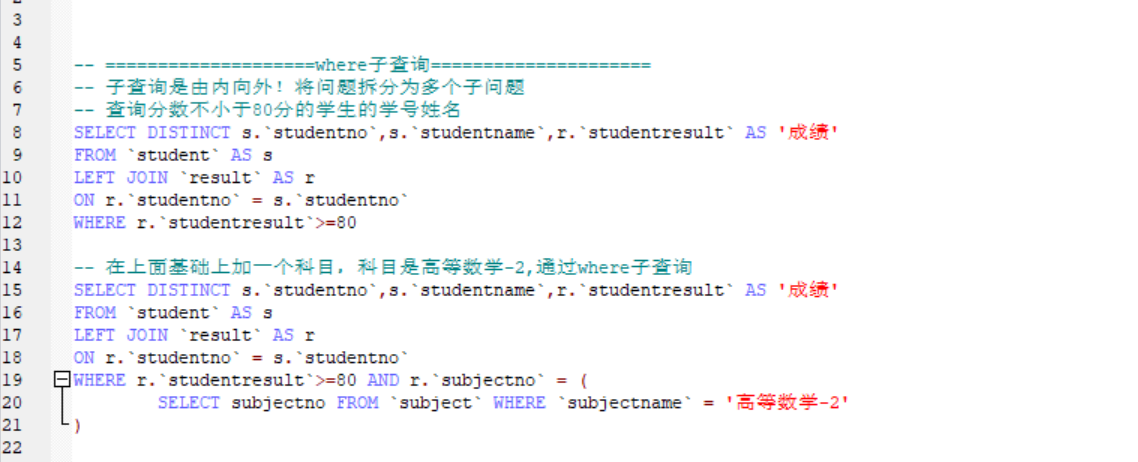
为什么要分页：缓解数据库压力，给人的体验更好！另外一种功能瀑布流！

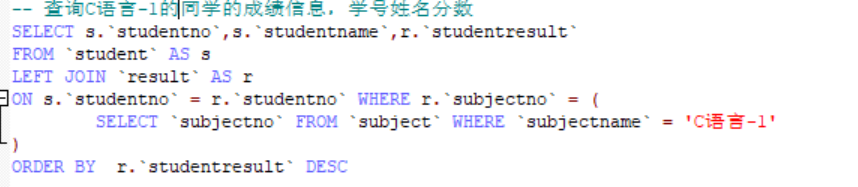


### 子查询

在where语句中嵌套一个子查询！

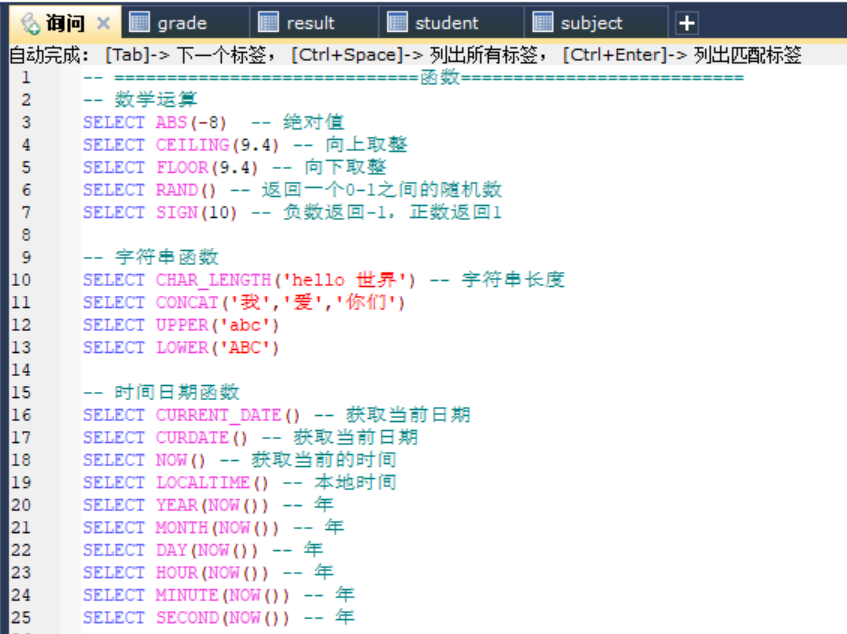
核心思想：把目标问题拆分为多个易解决的子问题，通过嵌套查询解决所有子问题也就解决了目标问题！查询的顺序是由内而外！



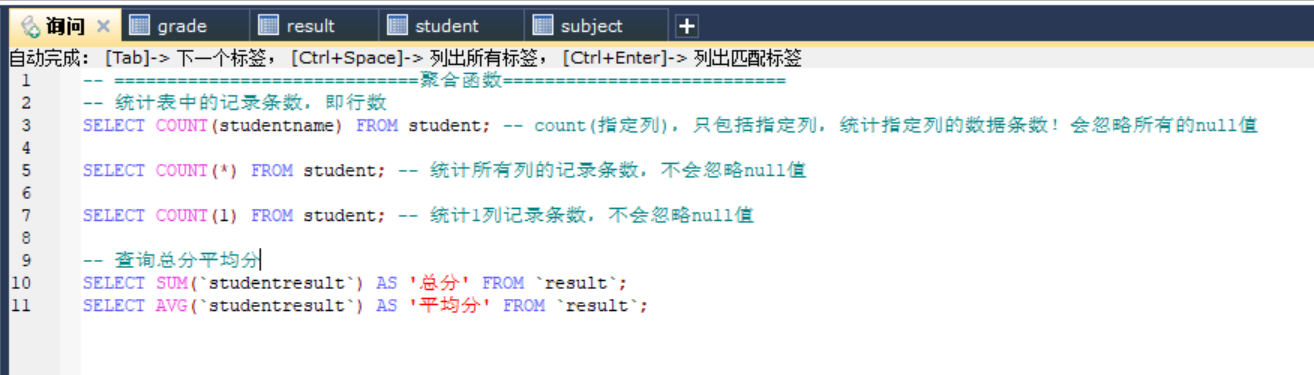


### MYSQL函数

* 常用函数



* 聚合函数
  + count()：计数
  + sum()：求和
  + avg()：平均数
  + max()：最大值
  + min()：最小值

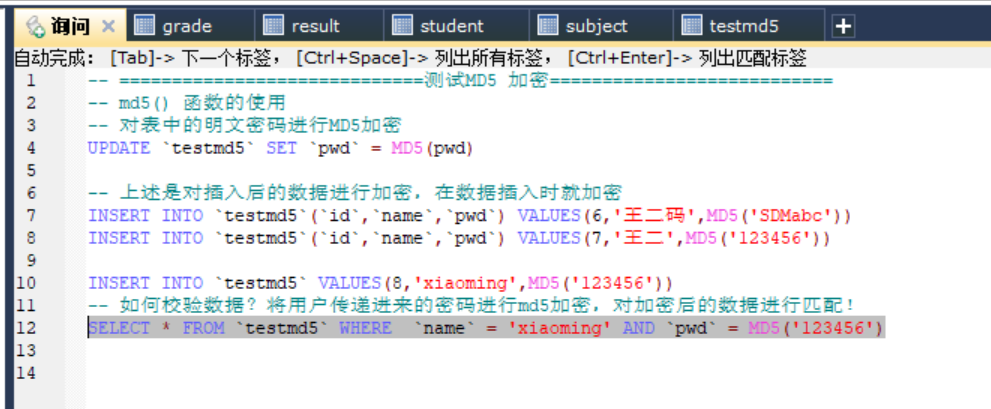


分组过滤：group by … having …

### 数据库级别的MD5加密

MD5：增强算法复杂度和不可逆性，MD5加密不可逆！

数据库几倍MD5加密，利用md5函数



### 事务

什么是事务：要么全部成功，要么全部失败！将一组SQL放在一个批次中去执行！

事务原则：ACID原则！原子性，一致性，隔离性，持久性！

* + 原子性：一组操作不可隔离单独操作，要么全部成功，要么全部失败
  + 一致性：事务前后的数据完整性保持一致
  + 持久性：事务一旦提交则不可逆转，被持久化到数据库中
  + 隔离性：多个用户并发访问数据库，数据库为每个用户开启的事务之间相互隔离

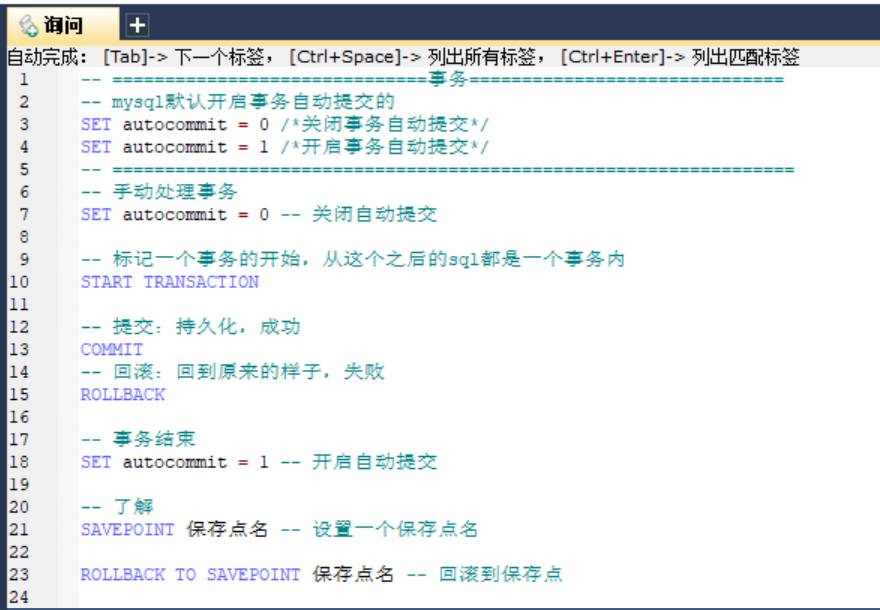
隔离导致的一些问题：

脏读：一个事务读取了另外一个事务未提交的数据

不可重复读：在一个事务中读取表的一行数据，多次读取结果不同

虚读：一个事务内读取到别的事务插入的数据导致读取前后不一致

mysql模拟事务，在java中，事务被封装成一个方法！

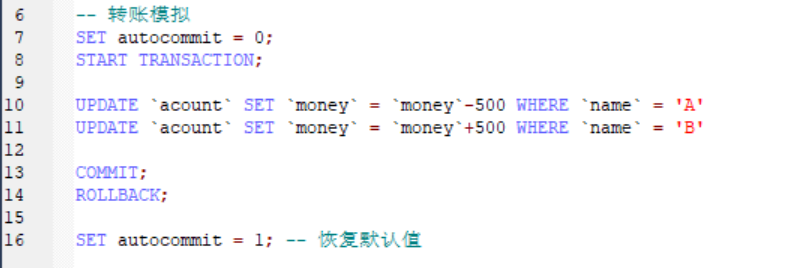


开始事务步骤：

* + 关闭自动提交
  + 开启事务
  + 事务内的sql语句
    - 成功：提交，开启自动提交，事务结束
    - 失败：回滚

事务一旦提交则持久化完成，不可逆！

测试用例：



### 索引\*\*\*

索引：是帮助Mysql高效获取数据的数据结构，针对于大数据流的数据库，创建索引能明显改善查询效率！

* 主键索引PRIMARY KEY
  + 唯一标识，主键不可重复，只能有一个列作为主键
* 唯一索引UNIQUE KEY
  + 避免重复的列出现，可以重复
* 常规索引KEY/INDEX
  + 默认的，index,key关键字来设置
  + 语法：create index 索引名 on 表(字段)
* 全文索引FULLTEXT INDEX
  + 快速定位数据
  + 在特点的数据库引擎下才有，MyISAM



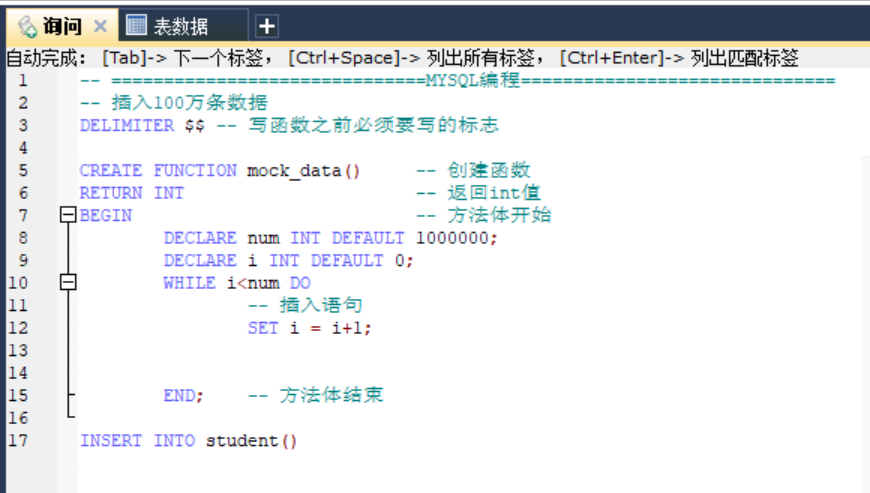
explain sql语句：分析该sql语句

#### 索引原则

* 索引不是越多越好
* 不要对经常变动的数据加索引
* 小数据量不要加索引
* 索引一般加在常用来查询的字段上

索引的数据结构：hash类型索引

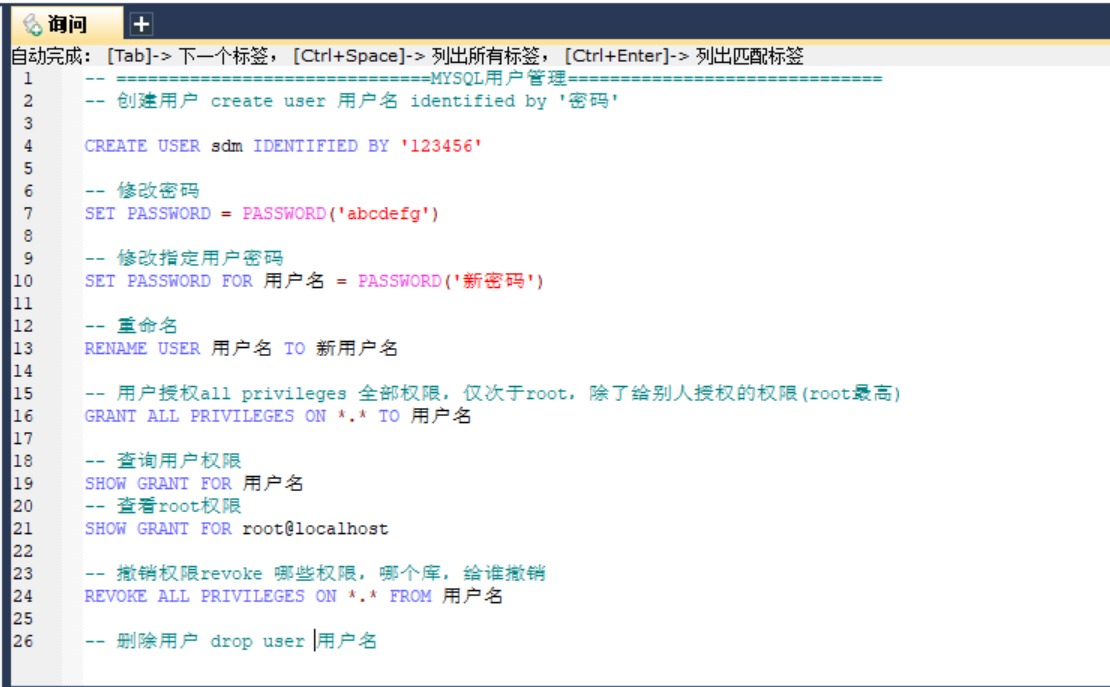
### MYSQL编程



### SQL权限管理

SQL命令操作：

用户表：mysql.user，该表存放所有用户表！



### MYSQL备份

为什么备份：

* 保证重要的数据不丢失
* 做数据转移

Mysql数据库备份的方式：

* 拷贝物理文件，mysql目录下的data文件夹
* 在sqlyog可视化工具中手动导出
* 使用命令行导出 mysqldump 命令行使用

命令行实例：补充mysql命令行登录:mysql -u用户 -p密码

数据导出语法：mysqldump -h 主机名 -u 用户名 -p 密码 数据库 表名(可忽略)>物理磁盘位置(带文件名)

C:\Users\ASUS>mysqldump -hlocalhost -uroot -pSDM4444NOG school>D:\school.sql

mysqldump: [Warning] Using a password on the command line interface can be insecure.

数据导入：在登录的情况下，导入数据命令：source 资源路径

如果资源是表，需要指定数据库，如果是数据库则不需要指定数据库！

# 设计数据库\*\*\*

数据库如果比较复杂，就需要合理设计数据库

粗制滥造的数据库：

* 数据冗余
* 数据插入和删除都麻烦，异常。
* 程序性能差

优良的数据库：

* 结束内存空间
* 保证数据库的完整性，屏蔽使用外键
* 方便开发系统

## 数据库设计

1. 分析需求：分析业务和需要处理的数据库的需求
2. 概要设计：设计关系图E-R图

补充:数据库不区分大小写，所以字段命名不用驼峰命名，使用下划线命名

设计步骤：(个人博客)

* 收集信息，分析需求
  + 用户表：用户登录注销，用户的个人信息，创建分类
  + 分类表：文章分类，谁创建的
  + 文章表：文章的信息
  + 友链表：友链信息
  + 自定义表
* 标识实体，把需求落地到每个字段

## 数据库三大范式

设计出的数据库表要符合三范式！以规范数据库！

为什么要数据规范化？

* 信息重复
* 更新异常
* 插入异常
* 删除异常

三大范式：

1. 第一范式：要求数据库表中的每一列都是不可分割的原子数据项
   * 1. 原子性：保证每一列不可再分
2. 第二范式：前提是满足第一范式
   * 1. 保证数据库每一列都完全依赖于主键，不能只和主键某一部分相关，保证每张表只描述一件事情！
3. 第三范式：前提是满足第一第二范式
   * 1. 确保数据表中的每一列数据都和主键直接相关而不能间接相关，消除依赖的传递性

规范和性能问题：数据库设计严格遵照规范往往会损失数据库的行文，这就和性能产生了一定的矛盾！为达到平衡有了以下条件！

* 约定条件：关联查询的表不能超过三张！
* 考虑商业化需求和目标，成本和用户体验！数据库性能更重要！
  + 故意给表增加一些冗余的字段（多表查询变成单表查询）以增加性能
  + 故意增加一些计算列，降低数据量！（索引）