# 数据库的介绍

## 1.什么是数据库

文件保存数据有以下几个缺点：

* 文件的安全性问题
* 文件不利于数据查询和管理
* 文件不利于存储海量数据
* 文件在程序中控制不方便

为了解决上述问题，专家们设计出更加利于管理数据的软件——数据库

## 2.数据库分类

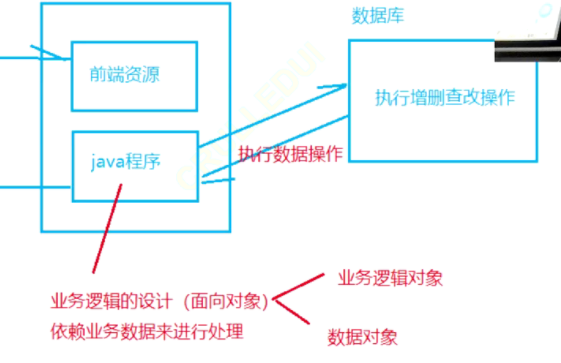
Ps:面试点：

不管哪门语言，重点是解决实际的业务问题，只是解决的方式不同，有以下两种方式：

* 面向对象编程：c
* 面向对象编程的：java,c++

### （1）.关系型数据库（RDBMS）

1.前言：java是面向对象编程，数据库中对应的关系和java对象中对应的关系很类似。



2.关系型数据库有一套统一的规范，而产品真正落地的时候，有它自己的实现。这个就像java里面的接口（关系型数据库的定义）和接口实现（落地的实现方法）很类似。因此，**Oracle,MySQL就像是对关系型数据库的一种实现，它们各自有自己的语法和规定等**

3.

是指采用了关系模型来组织数据的数据库。 简单来说，关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。

基于标准的SQL，只是内部一些实现有区别。常用的关系型数据库如：

1. Oracle：甲骨文产品，适合大型项目，适用于做复杂的业务逻辑，如ERP、OA等企业信息系

统。收费。

2. MySQL：属于甲骨文，不适合做复杂的业务。开源免费。

3. SQL Server：微软的产品，安装部署在windows server上，适用于中大型项目。收费。

### （2）.非关系型数据库

保存的是数据本身，并不保存它们之间的关系

1. 基于键值对（Key-Value）：如 memcached、**redis（开发的重点关注）**

2. 基于文档型：如 mongodb，如一整个网页的元素

3. 基于列族：如 hbase

4. 基于图型：如 neo4j

### （3）.两者的区别

SQL：指SQL语言，使用SQL语言来操作数据库，非关系一般不SQL



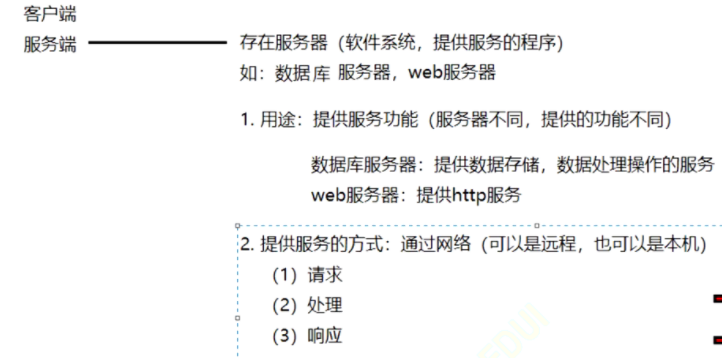
注：

OLTP（On-Line Transaction Processing）是指联机事务处理，必须很精准。比如去银行取钱，就需要使用关系型数据库。

OLAP（On-Line Analytical Processing）是指联机分析处理，可以不是很精准。比如统计分析喜欢游戏的男生占百分之几这样，类似统计分析这种，就需要使用菲关系型数据库

## 3.知识补充

### （1）.客户端和服务端



客户端向服务端发送请求，服务端给客户端进行响应。两个概念是相对的，一般来说，在某次网络请求/响应中，通过客户端调用服务端的服务。

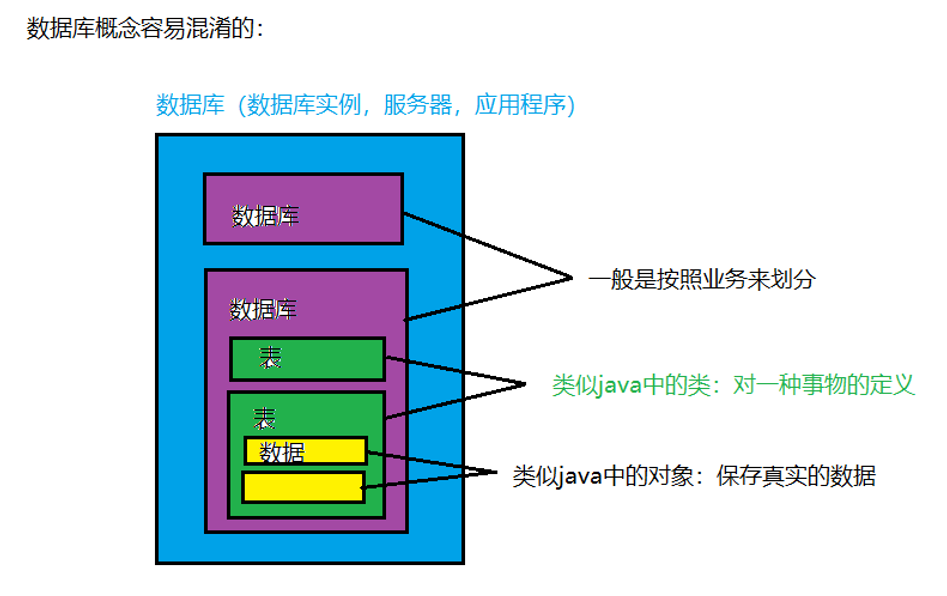
Ps:A相对于B是客户端，但是A相对于C可能就是服务端了

# 数据库的基础

## 0.前言

不同的地方说的数据库是不一样的，可能指的是MySQL数据库这个程序，也可能是MySQL应用程序中的一个数据库。

这里说的数据库操作，删除数据库等，都是指的是这个应用程序中的一个数据库



扩展：数据结构也可以保存数据，数据库使用到大量的数据结构

数据库学习的目标：数据库，表，数据进行增删查改（CRUD）的操作（SQL）

（CRUD:增加（Create）、读取查询（Retrieve）、更新（Update）和删除（Delete））

注意：命令行（黑窗口，cmd）使用时，最后以“；”结束，代表一条SQL语句

但是在java程序中，一般不使用

## 1.数据库的操作

数据库操作

### （1）查询，显示

show databases;

### （2）创建

-直接创建（如果已经存在该数据库，报错）

create database 数据库名称 character set utf8mb4;

-如果没有才创建（if 数据库不存在 创建）

create database if not exists 数据库名称 character set utf8mb4;

### （3）删除数据库

-直接删（如果不存在就报错）：

drop database 数据库名称;

-如果有才删

drop database if exists 数据库名称;

### （4）使用

（出现No database selected错误，要先进入/使用数据库，才能对表进行操作）

use 数据库名;

## 2.常用数据类型

### （1）. 数值型



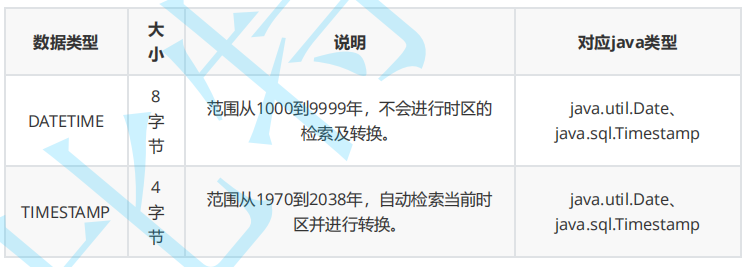
* BIT一般用来用来表示java里面的boolean类型
* DECIMAL里面的M表示指定长度，包括整数位和小数位

### （2）. 字符串类型



* VARCHAR对应的就是我们java里面使用的String类型的字符串
* TEXT用于存放比如一篇文章、博客、网站等这样的长文本数据

### （3）. 日期类型



* 如果你的时间是要包含时区的话，要使用TIMESTAMP

Java里面Date和Timestamp的使用



## 3.表的操作

### （1）. 查看表结构

语法：

desc 表名



### （2）. 创建表

语法：

CREATE TABLE table\_name (

field1 datatype,

field2 datatype,

field3 datatype

);

注意：

0.命名规则exam\_result，这种全小写下划线分割的命名格式

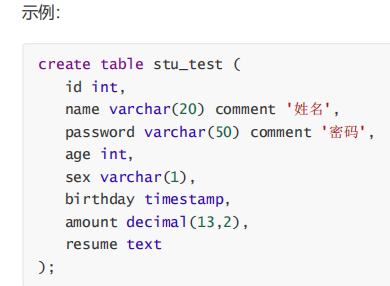
1.前面的field是字段名称，后面的是字段的数据类型

2.每条字段的最后用一个” , ”隔开，但是最后一个字段后面没有逗号

3.最后的大括号外面要有分号作为结尾

4.可以使用comment增加字段说明。类似java里面的注释，注意逗号在comment之后

5.报语法错误可能是因为中文符号引起的



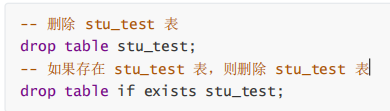
### 面试点：给一个场景，设计一个表，并给出他们的关系

### （3）. 删除表

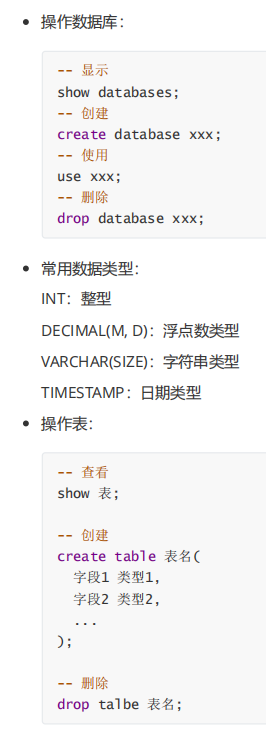
语法：

DROP [TEMPORARY] TABLE [IF EXISTS] tbl\_name [, tbl\_name] ...

eg:



## 4.总结



Show tables：查看所有表

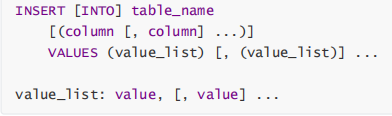
# MySQL表的增删改查（基础）

## 1.CRUD

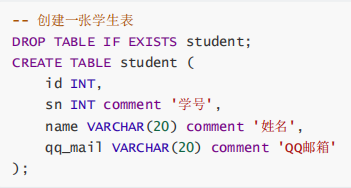
* 注释：在SQL中可以使用“**--空格**+描述”来表示注释说明
* CRUD 即增加(Create)、查询(Retrieve)、更新(Update)、删除(Delete)四个单词的首字母缩写。

## 2.新增（Create）

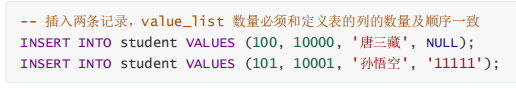
语法：



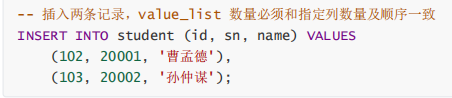
举例：



### （1）. 单行数据 + 全列插入（别用）



### （2）. 多行数据 + 指定列插入（强推）



1.数据库里面的字符串是单引号

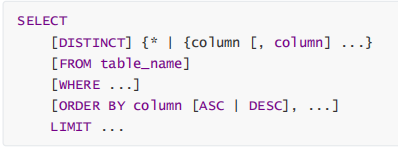
2.数据库一般要有唯一字段来进行表的标识，在java里面如果为了和数据库对应的话，那也需要有这么一个属性

**3.以后写，只使用指定列插入的写法**，因为如果你不指定列插入，以后你的表要是增加了一个字段，那之前的语句就会报错了

**4.注释语法：--空格**

## 3. 查询（Retrieve）

语法：





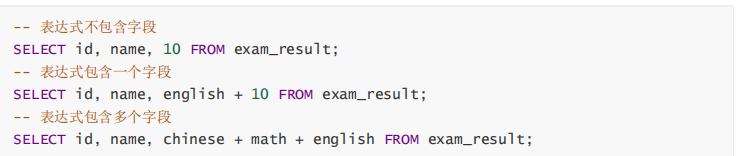
### （1）. 全列查询



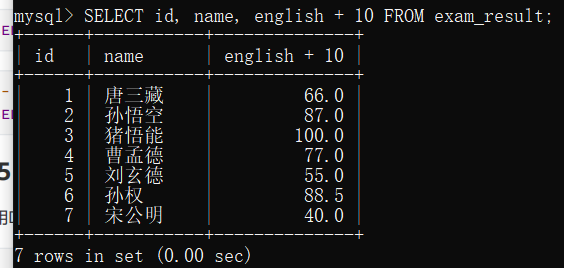
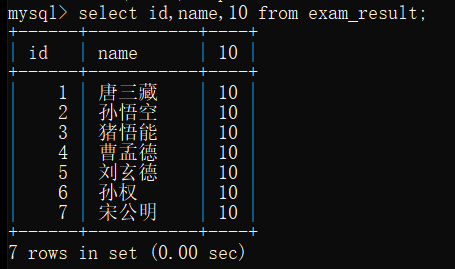
### （2）. 指定列查询

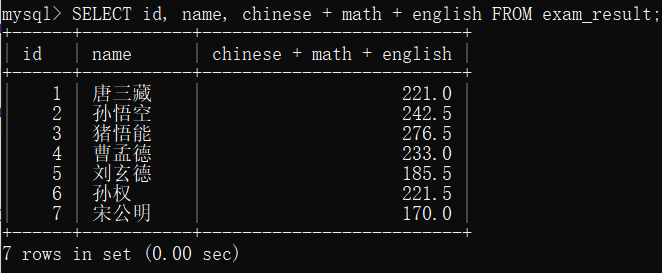


### （3）. 查询字段为表达式



通过这三组的结果我们可以知道：查询的字段为表达式时，他会查询以后，自己进行计算再输出

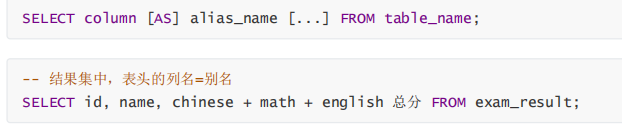


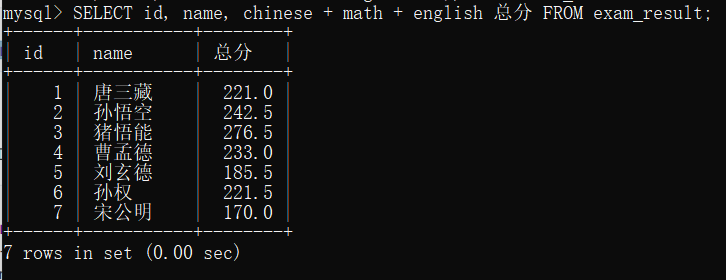


### （4）. 别名

为查询结果中的列指定别名，表示返回的结果集中，以别名作为该列的名称。

语法：

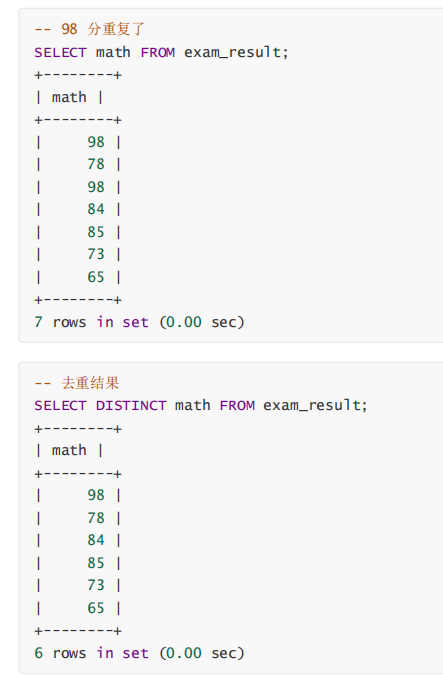




* Tips：别名“总分”前面可以加上as
* 注：这里可以看到，我们这里显示的数据是数据库返回给我们的结果集。相当于我们这里的黑框框是客户端，数据库是服务端

### （5）. 去重：DISTINCT

使用DISTINCT关键字对某列数据进行去重：



* distinct放在需要指定的字段前面，且只能加上一个字段

### 面试题：你对集合框架的理解：

1.关系图：Collection

List Set Map

每一种实现类

2.每种结构的特性，及互相之间的区别

### 面试题：排序场景题：

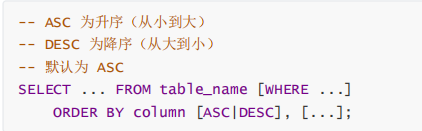
很多人，人里面有姓名，年龄等，然后根据人的年龄排序。

使用Comparable+Comparator接口（会代码并且表达出来）

### （6）. 排序：ORDER BY

Java排序：（1）算法排序 （2）java比较接口进行排序

语法：

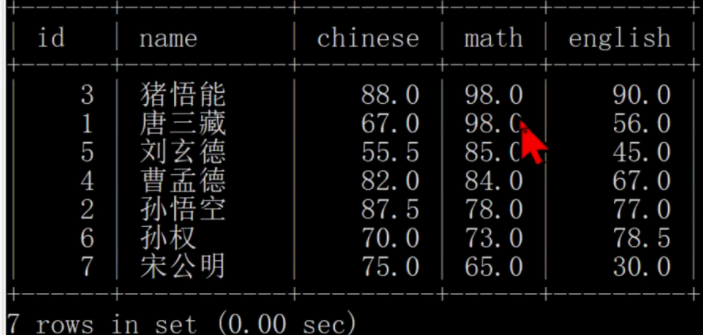


1. 没有 ORDER BY 子句的查询，返回的顺序是未定义的，永远不要依赖这个顺序

2. NULL 数据排序，视为比任何值都小，升序出现在最上面，降序出现在最下面

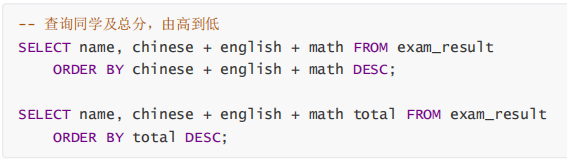




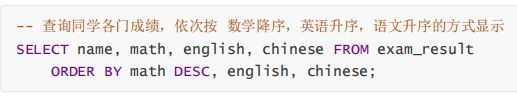


* Select \* from exam\_result order by math desc,chinese desc;
* 这里就是先根据数学降序排序，然后再根据语文降序排序（默认是升序排序）

3. 使用表达式及别名排序



4. 可以对多个字段进行排序，排序优先级随书写顺序

5.

5.order by 一般和limit一起使用，用来选取前三的元素。2

### （7）. （最重要）条件查询：WHERE

**比较运算符：**



这里面的比较运算符和java里面的比较运算符不太一样

* = 是比较运算符了，表示判断相等，不再是java里面的==了
* 等于只要知道 = 就可以，不等于只要知道 != 就行

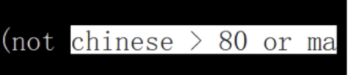
**逻辑运算符：**



1. WHERE条件可以使用表达式，但不能使用别名。

2.AND的优先级高于OR，在同时使用时，需要使用小括号()包裹优先执行的部分

3.NOT可以写到任意的逻辑运算的结果之前，如：



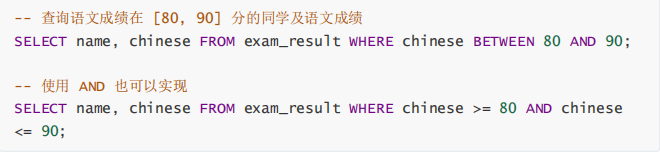
##### 1.基本查询



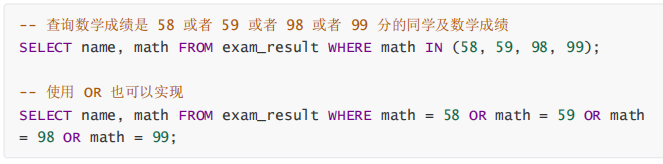
* 注意WHERE条件可以使用表达式，但不能使用别名。

##### 2.范围查询

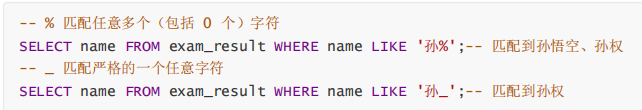
--BETWEEN ... AND ...



--IN

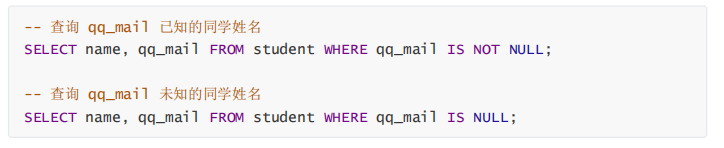


##### 3.模糊查询：LIKE



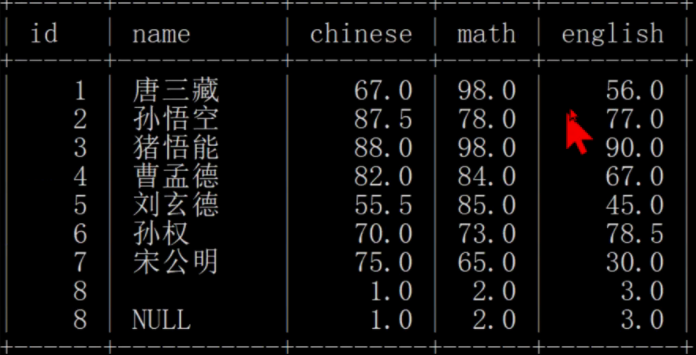
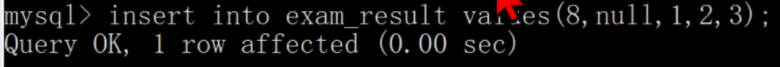
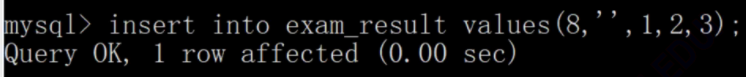
* % 匹配的是0个或多个字符，比如只有一个‘孙’，那只有‘孙%’能匹配出来
* \_ 匹配的是一个字符

##### 4.NULL 的查询：IS [NOT] NULL

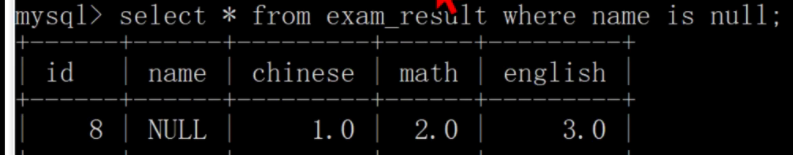
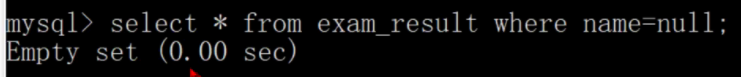


* **空字符串和null不一样，**

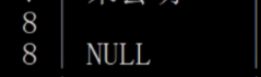
像这里，插入一个空字符串和null



* 这个时候你要是想找null的值，不能使用name = null，必须使用name is null

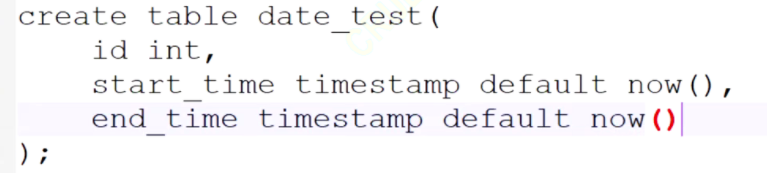


* **一个是空字符串‘’，一个是NULL**



### （8）. 日期的使用和查询

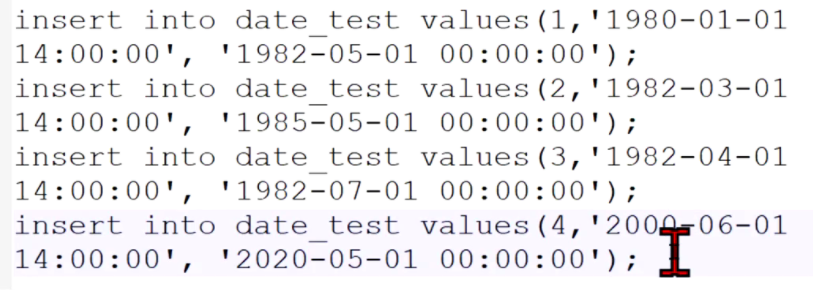
1.创建含有日期的表



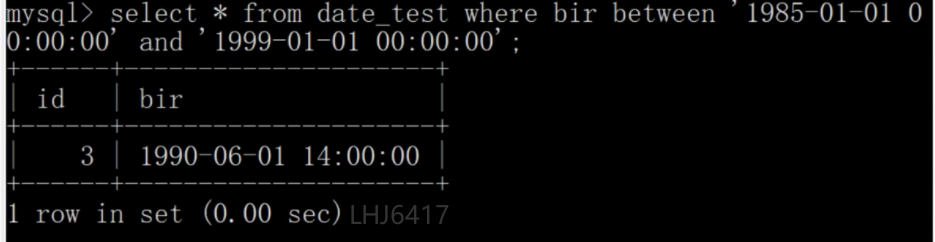
* 这里要给日期设置一个默认值，不过我自己那里不用设置默认值好像也可以。
* 设置默认值就设置为null就可以了

2.mysql默认的日期格式：yyyy-MM-dd HH:mm:ss

插入含日期的数据

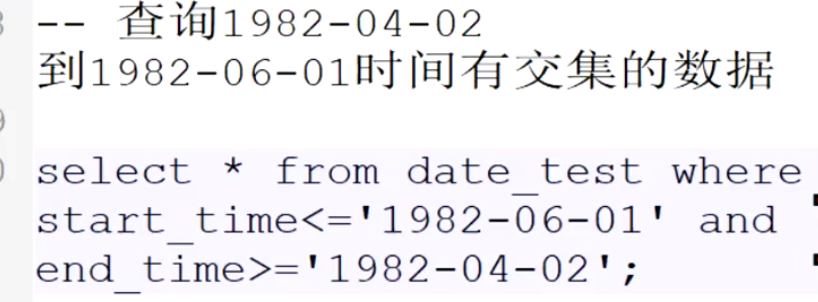


3.查询日期字段



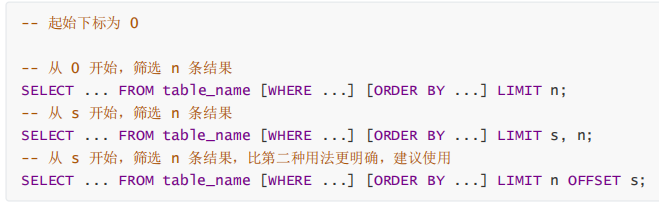
* 和其他字段一样，查询的时候也可以between-and。

4.



### （9）. 分页查询：LIMIT（MySQL专用）

语法：



注意：offset：偏移，推迟；

outset: 开始，开端。

* s表示的是偏移量（就是先偏移几个），n表示的是筛选结果的条数
* 第二个LIMIT s,n表示筛选n条结果，从第s条数据开始
* 第三个LIMIT n OFFSET s表示筛选n条结果，从第s条数据开始
* 这里也不是要建议第三种，第二种也可以，看你哪个用的比较顺，一个是先s后n，另一个是先n后s,**本人建议第三种，因为没有s的时候就是LIMIT n**
* **Order by一般和limit一起使用**

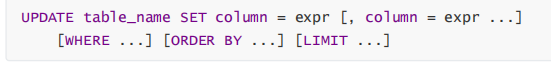
### 面试题：用分页查询选出语文成绩第二的同学

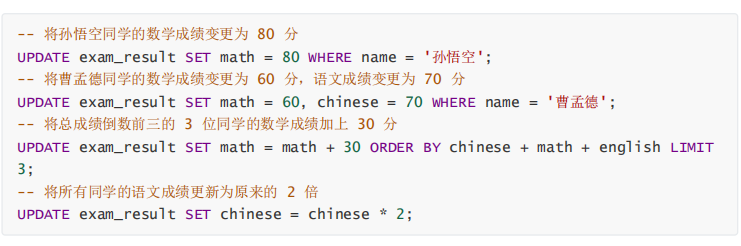
select \* from exam\_result order by chinese desc limit 1 offset 1

* 注意：这里的降序不能忘记
* 要注意这里where--->order by-->limit这样的顺序

## 4. 修改（Update）

语法：





* 在set的时候，如果值的长度超过了建表时给定的长度，会报错。
* 比如最后的set chinese = chinese \* 2;就可能会报错
* 如果不加上where,order by等筛选条件，那就是整列元素的值全部修改

## 5. 删除（Delete）

语法：

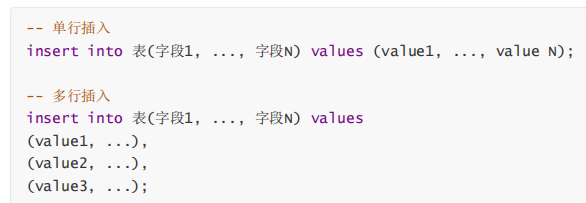




注意delete from table\_name，后面什么都不加，那就是整张表删除了，一定要注意。要加上where,order by,limit等------------------------------------（删库跑路doge）

## 6. 内容重点总结

新增：



查询：



修改：



删除：



# MySQL表的增删改查（进阶）

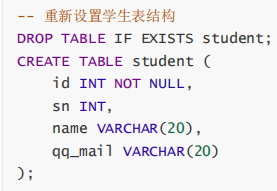
## 1. 数据库约束

### （1）. 约束类型

* NOT NULL - 指示某列不能存储 NULL 值。
* UNIQUE - 保证这一列的每个数据都不能重复。
* DEFAULT - 规定没有给列赋值时的默认值。如果给某一列设置了默认值。插入数据的时候，没有给这个列插入数值的话，就会把默认值作为数值插入进去
* PRIMARY KEY（主键） - NOT NULL 和 UNIQUE 的结合。确保某列（或两个列多个列的结合）有唯一标识，有助于更容易更快速地找到表中的一个特定的记录。注：并不一定只能有一个主键（如一个学校的学生-主键为学号，多个学校的学生-主键为学校和学号）
* FOREIGN KEY （外键）- 保证一个表中的数据匹配另一个表中的值的参照完整性。
* CHECK - 保证列中的值符合指定的条件。对于MySQL数据库，对CHECK子句进行分析，但是忽略CHECK子句。（一般是将数据的值指定在一个范围以内，超出这个范围就会报错）

### （2）.NULL约束

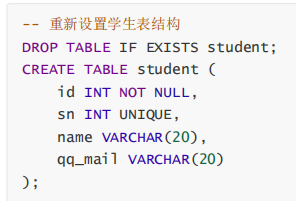
创建表时，可以指定某列不为空：



* 如果指定某列不为空，插入数据的时候没有给那一列插入数据，会报错
* 可以在NOT NULL后面加上default默认值，那就不会报错

### （3）.UNIQUE：唯一约束

指定sn列为唯一的、不重复的：



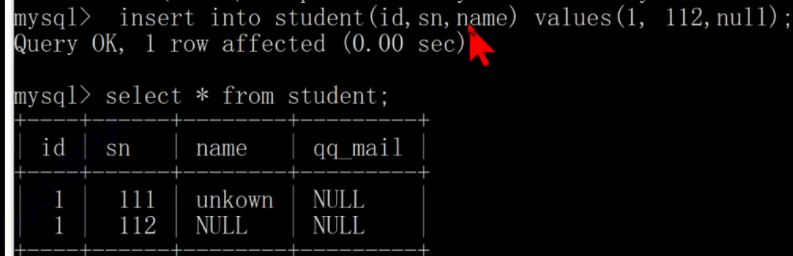
* 插入重复的列，会报错

### （4）.DEFAULT：默认值约束

指定插入数据时，name列为空，默认值unkown：

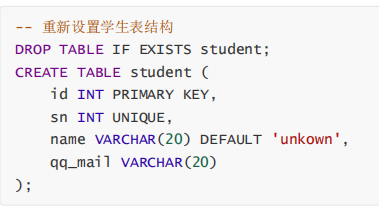


Default默认值执行的条件是不显示插入任何数值，包括NULL；如果你插入了NULL的时候给定的值是NULL，那就不会执行default默认值了，会直接显示NULL。

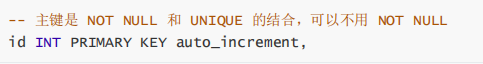


### （5）.PRIMARY KEY：主键约束（auto）

指定id列为主键：



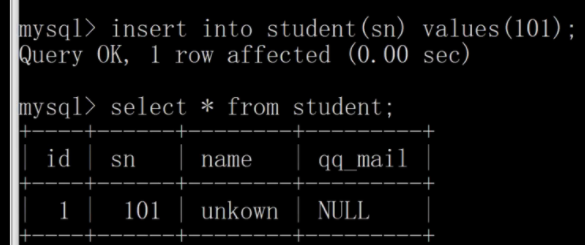
* PRIMARY KEY就是NOT NULL和UNIQUE的组合
* 主键可以运用在多列上，表示复合主键



* 对于整数类型的主键，常配搭自增长auto\_increment来使用。插入数据对应字段不给值时，使用最大值+1。相当于就是给了一个默认值，然后不显示插入的话，这个默认值会+1

我们设计数据库时就可以不显示地给id插入数值，让它自增就行，这是一种常用的方法。

如下图，没有给主键id赋值，但是并没有报错，他会自动赋值并且自增。



### （6）.FOREIGN KEY：外键约束

##### 1.数据库的四大特性：

原子性，一致性，隔离性，持久性。

具体看：[数据库四大特性博客](https://blog.csdn.net/sinat_35322593/article/details/81040479?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164173281016780264058370%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164173281016780264058370&biz_id=0&utm_m)

##### 2.数据库的三大范式：

* 第一范式（1NF）：列不可再分

1.每一列属性都是不可再分的属性值，确保每一列的原子性（像学校和学号，必须分开来，不能自己定义一个模糊的概念，学校\_学号这样）

2.两列的属性相近或相似或一样，尽量合并属性一样的列，确保不产生冗余数据

* 第二范式（2NF）：属性完全依赖于主键（消除部分依赖）

第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。

所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。简而言之，第二范式就是属性完全依赖于主键。

Eg:同一个订单中可能包含不同的产品，因此主键必须是“订单号”和“产品号”联合组成，但可以发现，产品数量、产品折扣、产品价格与“订单号”和“产品号”都相关，但是订单金额和订单时间仅与“订单号”相关，与“产品号”无关，这样就不满足第二范式的要求。

Zongjie:也就是说在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。

* 第三范式（3NF）：属性不依赖于其它非主属性，属性直接依赖于主键（消除传递依赖）

数据不能存在传递关系，即每个属性都跟主键有直接关系而不是间接关系。像：a-->b-->c 属性之间含有这样的关系，是不符合第三范式的

详情请看[数据库三大范式博客1](https://blog.csdn.net/qq_40899182/article/details/81706253)

[数据库三大范式博客2](https://wangandh.blog.csdn.net/article/details/105656952?spm=1001.2101.3001.6650.5&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-5.pc_relevant_paycolumn_v2&depth_1-utm_source=distribute.pc_relevant.none-task-)

[数据库三大范式博客3](https://blog.csdn.net/m0_48930261/article/details/108831319?spm=1001.2101.3001.6650.17&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-17.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-)

##### 3.外键

如图，根据学生表的外键，可以定位到学校表中的数据，**极度类似于java中的引用，class里面套class**

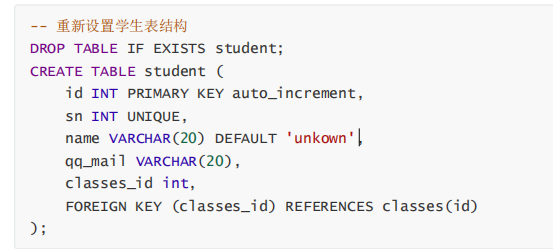


外键用于关联其他表的主键或唯一键，语法：



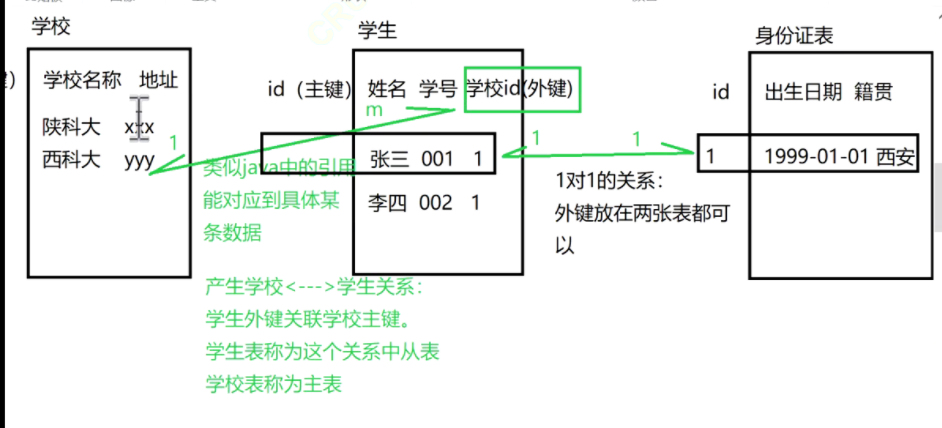


* 使用Mysql关键字作为字段时，需要使用``来进行标识



* 班级id是主表，学生外键关联班级主键，所以外键的定义放在学生表里面

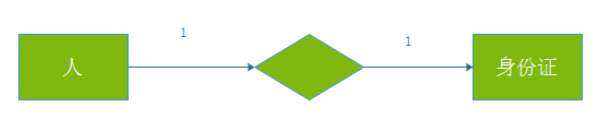
##### 4.表之间的关系



* 1对1的关系：可以把身份证的外键设置为学生表中的id,或者把学生表的外键设置为身份证表中的id
* 1对多的关系：主表到从表：1对多。主表到从表：1对1（有方向了，不能乱设置外键）

## 2. 表的设计

### （1）. 一对一



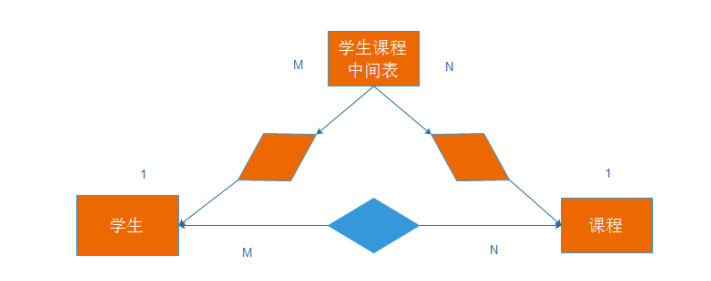
* 可以把身份证的外键设置为学生表中的id,或者把学生表的外键设置为身份证表中的id
* Java代码设计：人这个类中绑定一个身份证对象，身份证类中绑定一个人的对象（Java设计中必须双向都设计）

### （2）. 一对多

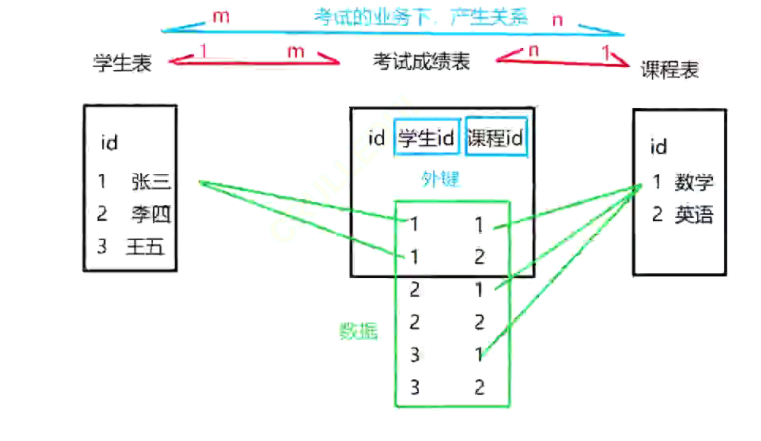


* 主表到从表：1对多。主表到从表：1对1（有方向了，不能乱设置外键，必须把学生表的外键设置为班级的id）
* Java代码设计：班级这个类中绑定一个对象的集合（如多个对象的list），学生类中绑定一个班级的对象（Java设计中必须双向都设计）

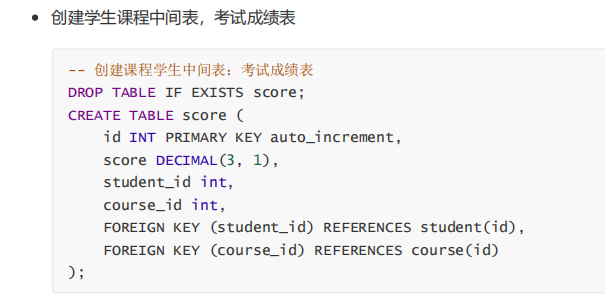
### （3）. （重要）多对多（注意看代码）



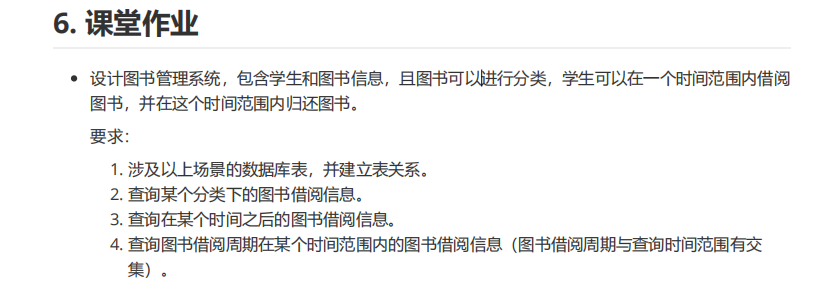
* Java设计与一对多类似，创建一个中间类，实现两个1对多即可
* 一旦这种多对多的情况出现，就不能直接使用两张主表上创建外键的方式。**需要一个学生课程中间表，设计两个外键分别关联两张主表的主键**
* 学生表和中间表是1对多的关系，课程表和中间表之间是1对多的关系，**用两个1对多构成了一个多对多**
* 中间的表相当于是关系数据，保存的仅仅是两张表之间的关系，没有保存两张表里面的细节信息，也可以保存其他的业务数据。







### （4）. （重要）图书管理系统数据库设计



学生表，图书表肯定是要的，然后就是一张借阅表，借阅表作为中间表，借阅时间等信息。

因为图书要有类别，不同的图书，类别可能会大幅度的相同，从而产生冗余，因此，我们可以将类别单独拿出来，弄成一张新的表

学生表：

create table students(

id int primary key auto\_increment,

name varchar(20) not null

);

图书表：

create table books(

id int primary key auto\_increment,

name varchar(20),

author varchar(20),

book\_type\_id,

foreign key (book\_type\_id) references book\_type(id),

);

借阅表：

id int primary key auto\_increment,

start\_time timestamp default now(),

end\_time timestamp default now(),

book\_id int,

student\_id int,

foreign key(book\_id) references books(id),

foreign key(student\_id) references students(id)

);

图书种类表：

create table book\_type(

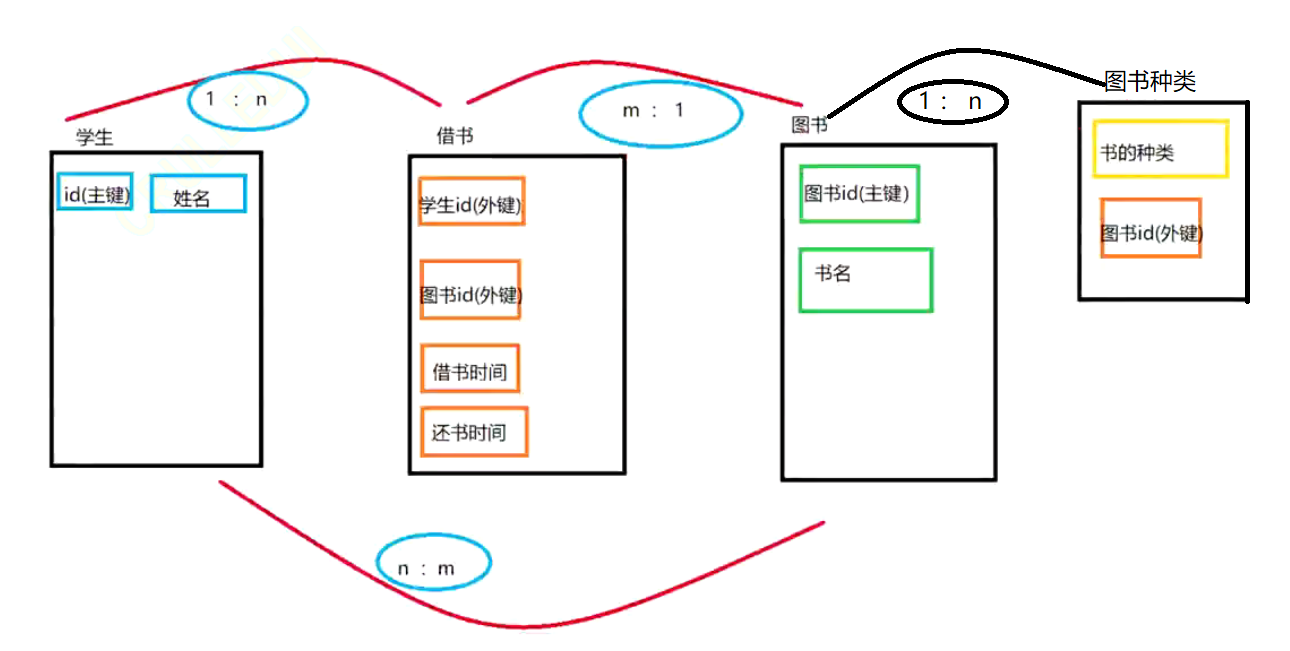
id int primary key auto\_increment,

type varchar(20),

);

* 这里我们图书种类表和图书表是一对多的关系，那么图书种类表是主表，图书表是从表（**1的为主表，多的为从表，外键一般设置在从表上**），所以外键要设置在图书表上。
* 要是把外键设置在图书种类上面，要存放一个种类，多种图书，那图书种类这个表里面就会有很多一样的种类，从而造成分类表冗余。
* 注意：外键可以重复，其他字段尽量不要重复。

大致示意图：



## 3. 新增

插入查询结果

语法：



案例：创建一张用户表，设计有name姓名、email邮箱、sex性别、mobile手机号字段。需要把已有的学生数据复制进来，可以复制的字段为name、qq\_mail



* 注意，表的列的数量和类型一定要统一，不然会报错
* Select选取数据的时候，也可以where，order by，limit一下，比如把名字排序最靠前的三个人插到指定表的指定列上。

## 4. （进阶重点难点）查询

### （1）. 聚合查询

##### 1.聚合函数（count，sum，average...）

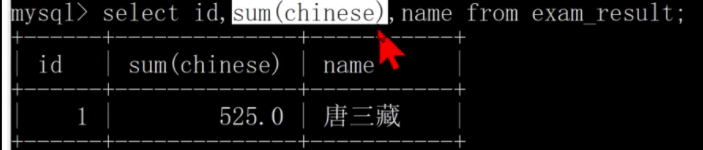
常见的统计总数、计算平局值等操作，可以使用聚合函数来实现，常见的聚合函数有：





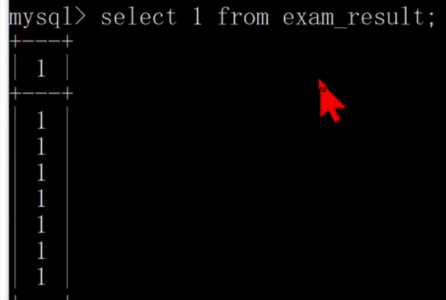
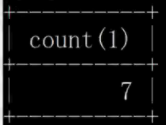


* 注意最后面都是可以加上where,order by这些语句的，这些聚合函数会根据这些语句的结果集来进行查询
* 后面最好不要加上limit，因为MySQL的limit和别的数据库的limit不一样
* 有使用统计列的聚合函数的时候，避免再写上其他列，不然会出现这种无意义显示第一列的情况



**注意点：**

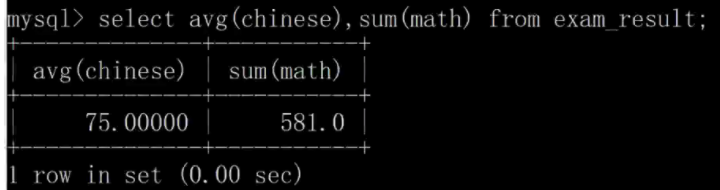
1.count：可以使用count（\*），count（0），count（1）这样，说白了实际上和select 1 from整个表是一样的道理，这个count里面的0，1只是作为参数传入而已，先select 1,然后再统计count的值



##### 面试题：count（\*），count（0），count（1）的执行效率有没有什么问题

2.sum，max，min，avg都不可以传入\*,必须传入字段或者表达式使用

3.avg可以和sum结合起来用，聚合函数都可以多个一起用



##### 2.GROUP BY子句（对我来说难点）

SELECT 中使用 GROUP BY 子句可以对指定列进行分组查询。需要满足：使用 GROUP BY 进行分组查询时，SELECT 指定的字段必须是“分组依据字段”，其他字段若想出现在SELECT 中则必须包含在聚合函数中。

* Select后面的聚合函数是等Group by执行完分组以后才进行的
* Group by语句的实质就是分组啊啊啊，常配合聚合查询使用

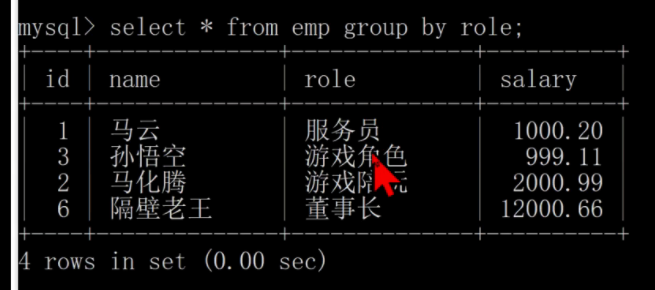
语法：



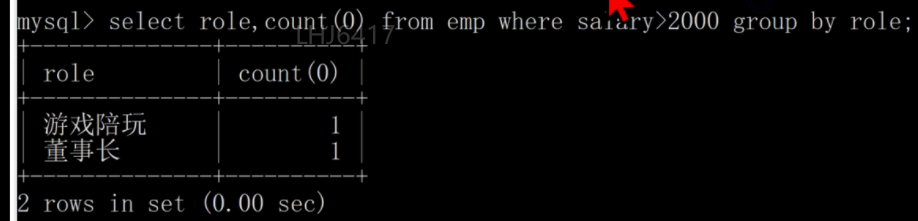


* Select指定的字段必须是group by后面有的，没有的只能出现在聚合函数中，不然会出问题

比如下面，你想按role分类，直接select\*，role为游戏角色的只有孙悟空了，剩下的两个数据被吞掉了

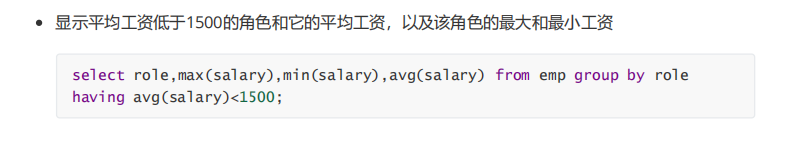


* Group by 语句之前是可以加上where语句的，而且where语句会先于group by执行，如下图，就是在薪水<2000的数据里面进行分组的



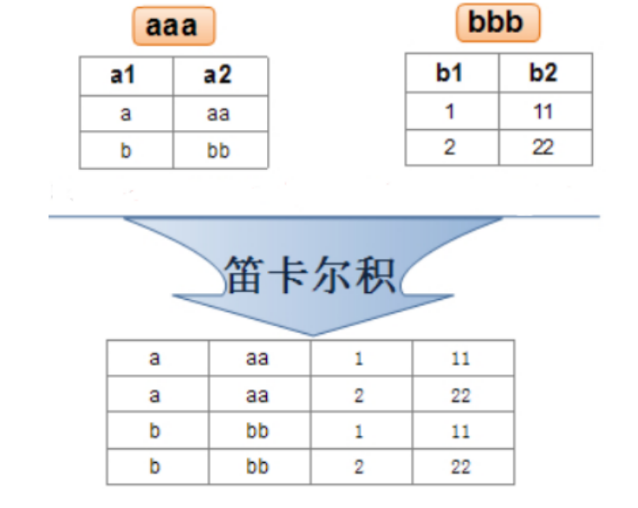
##### 3.HAVING

GROUP BY 子句进行分组以后，需要对分组结果再进行条件过滤时，不能使用 WHERE 语句，而需要用HAVING。HAVING是在GROUP BY 后面执行的



### （2）. 联合查询（（重点）多表）

实际开发中往往数据来自不同的表，所以需要多表联合查询。多表查询是对多张表的数据取笛卡尔积：



注意：关联查询可以对关联表使用别名。

初始化数据总共是4张表：classes,student,course,score

-- 创建班级表，有使用MySQL关键字作为字段时，需要使用``来标识

DROP TABLE IF EXISTS classes;

CREATE TABLE classes (

id INT PRIMARY KEY auto\_increment,

name VARCHAR(20),

`desc` VARCHAR(100)

);

-- 创建学生表

DROP TABLE IF EXISTS student;

CREATE TABLE student (

id INT PRIMARY KEY auto\_increment,

sn INT UNIQUE,

name VARCHAR(20) DEFAULT 'unkown',

qq\_mail VARCHAR(20),

classes\_id int,

FOREIGN KEY (classes\_id) REFERENCES classes(id)

);

-- 创建课程表

DROP TABLE IF EXISTS course;

CREATE TABLE course (

id INT PRIMARY KEY auto\_increment,

name VARCHAR(20)

);

-- 创建课程学生中间表：考试成绩表

DROP TABLE IF EXISTS score;

CREATE TABLE score (

id INT PRIMARY KEY auto\_increment,

score DECIMAL(3, 1),

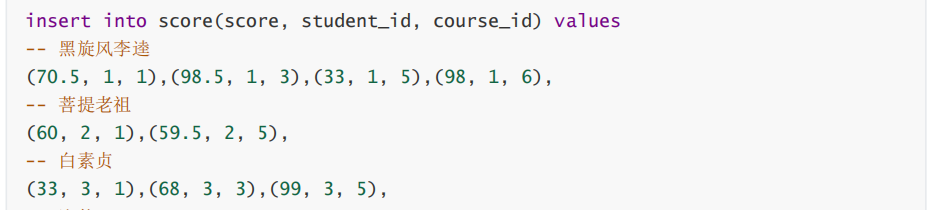
student\_id int,

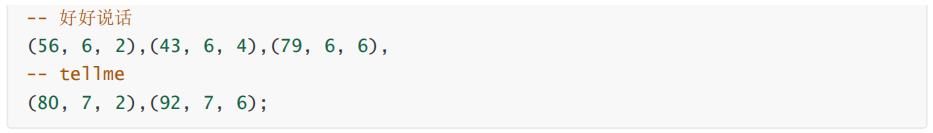
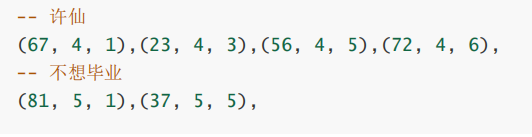
course\_id int,

FOREIGN KEY (student\_id) REFERENCES student(id),

FOREIGN KEY (course\_id) REFERENCES course(id)

);





##### 1.内连接

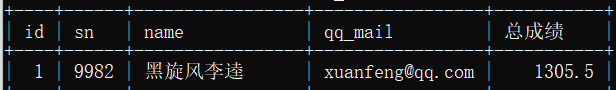
语法：



* 这里第二个方法的话，你可能不知道哪个是连接条件，哪个是筛选结果集的条件。用哪种都行，还是优先用第一种吧，为了和后面的外连接对应。
* 内连接就相当于是在得到的笛卡尔积上给了连接条件



* 这里要是不group by进行分组的话，就会像下面这样只有一行，相当于是把所有学生的所有成绩全部加起来了。但是我们这里需要的是单个学生的总成绩，所以我们要先按学生id进行分组。







* 这里order by的作用是把让一个学生的成绩到一起去，这样好看一点，同样也可以使用group by stu.id,course.id
* 为什么这里不需要分组呢，--因为没有使用聚合函数
* 为什么要group by 两个呢--因为如果单单按stu.id进行分组的话，就把每个学生的所有成绩折叠到一起了，每个学生只显示一行，如下图。



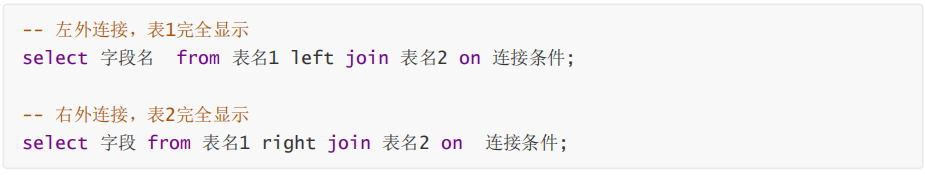
##### 2.外连接

外连接分为左外连接和右外连接。如果联合查询，左侧的表完全显示我们就说是左外连接；右侧的表完全显示我们就说是右外连接。

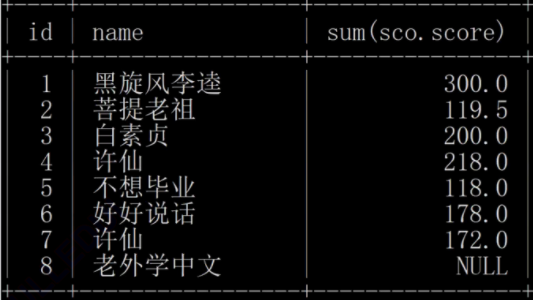
注意点：

* 左/右连接：左表/右表的数据不会依据**连接条件**过滤而全部显示，其他条件还是可以过滤，比如后面再加上where等条件

语法：







同样的查询方法，这次能显示出第8名成绩为空的学生的信息，左表也就是student表中的数据会全部显示出来，不会受到连接条件stu.id = sco.student\_id这一连接条件的影响，如果是之前的**内连接**，就**显示不出老外学中文这一学生的信息**，因为在sco表中根本就没有老外学中文这一学生的id。

##### 面试题：内连接和外连接的区别

答：内连接必须严格满足连接条件的条件。而外连接，左外连接的左表，或者右外连接的右表，它的数据会全部显示，不会受到**连接条件**的影响，虽然他还是会受到其他条件的影响，比如加上where，但是不会像内连接那样受到连接条件的影响。

##### 3.自连接

自连接是指在同一张表连接自身进行查询。

使用场景：同一张表，多行进行比较。



