# 查询表的字段数

select count(1) from information\_schema.columns where table\_schema='库名' and table\_name= '表名';

# 数据初始化

Sql脚本：bjpowernode.sql 这个文件以sql结尾，这样的文件被称为“sql脚本”。什么是失去了脚本呢？

当一个文件的扩展名是.sql，并且该文件中编写了大量的sql语句，我们称这样的文件为sql脚本。

注意：直接使用source命令可以执行SQL脚本

\*\*\*.sql这种文件被称为sql脚本文件。

Sql脚本文件中编写了大量的sql语句

我们执行sql脚本文件的时候，该脚本文件中所有的sql语句会全部执行！

批量执行sql语句，可以使用sql脚本文件。

在mysql当中怎么执行脚本呢？

格式：source 文件储存位置；

mysql> source E:\vip.sql;

# 删除数据库

drop database 数据库名；

# 查看表结构（不看表中的数据）

desc 表名；

# 查看当前使用的数据库

select datsbase();.

# 结束一条语句

\c

# 查看mysql数据库的版本号

mysql> select version();

+-----------+

| version() |

+-----------+

| 8.0.20 |

+-----------+

1 row in set (0.00 sec)

# 查看当前使用的那个数据库

mysql> select database();

+------------+

| database() |

+------------+

| NULL |

+------------+

1 row in set (0.00 sec)

注意：mysql不见分号不执行，分号表示结束

mysql> show

-> databases

-> ;

+--------------------+

| Database |

+--------------------+

| bj |

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------+

5 rows in set (0.00 sec)

mysql>

mysql>

mysql>

mysql> show

->

-> \c

\c用来终止一条命令的输入

也可以用分号结束但可能会报错

# 查看创建表的语句

show create table 表名

# 简单查询

查询一个字段？

Select 字段名 from 表名；

其中注意：

Select和from都是关键字

字段名和表名都是标识符 关键字不可改变，系统定义 标识符可以改变

强调：

对于说有的sql语句来说是通用的

所有的sql语句以分号结尾

另外sql语句不区分大小写

查询多个字段？(可以插入数学运算)

使用逗号隔开：select 字段名1,字段名2 \*12 from 表名;

查询所有字段？

第一种方式：可以把每个字段都写上，用逗号隔开

Select a，b，c。。。from biaoming

第二种方法：可以使用\*

Select \* from 表名

这种方法的缺点：

1. 效率低：先转换成全部字段
2. 可读性差

在实际开发中不建议

可以在DOS命令窗口中想快速查看全表数据时采用这种方式

# 给查询的列起别名

Select 字段名 as 别名 from 表名

例：select name as deptname from dept；

使用as关键字起别名

注意：只是将显示查询结果列名显示为别名，原表列名还是原来的名字，不发生改变

记住：select语句永远都不会进行修改操作的。（因为select只负责查询）

As可以省略么？ 可以

As用空格代替，不能用逗号

例：select name deptname from dept；

假设起别名的时候，别名里有空格，怎么办？

例：select name dept name from dept；

DBMS看到这样的语句，进行SQL语句的编译，不符合语法，编译报错。

怎么解决？select name ‘dept name‘ from dept；也可以用双引号

注意：在所有的数据库中，字符串统一使用单引号括起来，单引号是标准，双引号在oracle数据库中是用不了。但在mysql中可以使用。

再次强调：数据库中的字符串都是用单引号括起来的，单引号是标准，双引号不标准

列参与数学运算：

乘法：select 字段名\*n from 表名；//结论：字段可以使用数学表达式！

但是显示出来的表列名也是表达式：字段名\*n

所以我们可以起别名：select 字段名\*n as 别名 from 表名；

Sql语言中使用的是中文（字符串），必须用单引号括起来（条件查询中的条件）

解释：该字符串表示对该字段名作出的限制（varchar）后的

# 条件查询

不是将表中所有数据都查出来。而是查询出来符合条件的

格式 ：

Select

字段1，字段2，字段3.。。。

From

表名

Where

条件；

条件种类：

=等于

查询薪资等于800的员工姓名和编号？

Select number，name from list where reward=800；

Number表示编号；name表示名字；list表示表名；reward表示薪资。

查询某人的编号和薪资

Select number，reward from list where name=‘某人‘

<>或！=不等于

<小于 <=小于等于

>大于 >=大于等于

Between。。。and。。。两个值之间，等同于>= and <=

查询薪资在2450和3000之间的员工信息

Select number，name from list where reward>=2450 and reward<=3000;

Select number，name from list where reward between 2450 and 3000;

注意：使用between。。。and。。。的时候，必须遵循左小右大的原则，且该原则是闭区间（包含两边的值）

Is null为null（is not null不为空） null：无效，空的

例：查询那些员工的津贴/补助为null？

Select number，name，reward from list where jintie = null 这是不对的

Select number，name，reward from list where jintie is null jintie表示津贴/补助

注意：在数据库中null不能使用等号进行衡量。需要使用is null。因为数据库中的null代表什么也没有，它不是一个值，所以不能使用等号衡量。

# And

And 并且：

查询工作岗位是MANAGER并且工资大于2500的员工信息？

Select number，name，job，reward from list where job=‘MANAGER’ and reward>2500;

# Or

Or或者：

查询工作岗位是MANAGER和SALENAME的员工？  
select number，name，job from list where job=‘MANAGER’；

select number，name，job from list where job=‘SALENAME’；

综合两条命令到一条：select number，name，job from list where job=‘MANAGER‘ or job=‘SALENAME’；

And和or同时出现的情况下，有优先级问题吗？

查询工资大于2500，并且部门编号为10或20部门的员工？

Select \* from list where reward>2500 and deptno =10 or deptno=20;

分析以上语句的问题？

And优先级比or高。

以上语句会先执行and，然后执行or。

以上语句表示什么含义？

找出工资大于2500并且部门编号为10的员工，或者20部门所有员工找出来。

Select \* from list where reward>2500 and （deptno =10 or deptno=20）;

And和or同时出现，and优先级较高。如果想让or先执行，需要加小括号

以后在开发中不确定优先级，加小括号就行了

in 包含：（相当于多个or（not in不在这个范围中））

查询工作岗位是MANAGER和SALENAME的员工？

select number，name，job from list where job=‘MANAGER‘ or job=‘SALENAME’；

select number，name，job from list where job in （‘MANAGER‘ ，‘SALENAME’）；

注意：in不是一个区间，in后面跟的具体的值

查询薪资为800和5000的员工信息？

Select \* from list where reward=800 or reward=5000；

Select \* from list where reward in （500，3000）；

Not in表示不在这几个之当中的

Select \* from list where reward not in （500，3000）；

Not是取反

# Like称为模糊查询

支持%或下划线匹配

%代表任意多个字符

下划线代表一个字符

（%是一个特殊的符号，\_也是一个特殊的符号）

例：找出名字中含有字母o的？

Select name from list where name like ‘%o%‘；#o前面后面的百分号表示名字前面后面可能出现任意个百分号

找出名字以T结尾的？

Select name from list where name like ‘%T’；#表示T前面可能会出现任意多字符，但T是最后一个字符

找出名字以K开头的？

Select name from list where name like ‘T%’；#表示T后面可能会出现任意多字符，但T第一个字符

找出第二个字母是A开头的？

Select name from list where name like ‘\_A%’;#\_指标是一个字符，所以总的表示A前面只有一个字符，后面有任意多个

找出第三个字母是A开头的？

Select name from list where name like ‘\_ \_A%’;#\_指标是一个字符，所以总的表示A前面只有两个字符，后面有任意多个

找出名字中含有“\_”的？

Select name from list where name like ‘%\_%’;//这样是不行的，结果会将所有name查出来，因为\_（下划线）有特殊含义：表任意一个字符，用\转义字符转换成普通下划线

Select name from list where name like ‘%\\_%’;// \转义字，将\_装换成普通的下划线;

# 排序

例：查询所有员工薪资 ，排序？order by

Select name，reward from list order by reward；//默认是升序-asc,desc表示降序

怎么降序？order by 。。。desc；

指定降序：

Select name，reward from list order by reward desc；

指定升序？order by 。。。asc；

Select name，reward from list order by reward asc;

可以两个字段排序吗？或者说按照多个字段排序？（当排序有相同时再按另一个排序规则排名）

查询员工名字和薪资，要求按照薪资升序，如果薪资一样的话，再按照名字升序排列（名字字母）

Select name,reward from list order by reward asc,name asc;//reward在前，起主导，只有reward相等的时候，才会考虑启用name排序

了解：根据字段位置也可以排序

Select name，reward from list order by 2；//2表示第二列，第二列是reward；按照查询结果的第二列reward排序

了解一下，不建议在开发中这样写，因为不健壮

因为列的顺序很容易发生改变，列数按需修改之后，2就废了

综合案例：

找出工资在1250到3000之间的员工信息，要求按照薪资兼续排列。

Select name,reward from list where reward between 1250 and 3000 order by reward desc;

关键字顺序不能变：select…from…where…order by…

以上语句的执行顺序必须掌握：第一步：from 第二步：where 第三步： select

第四步：order by（排序总是在最后执行）

# 单行处理函数：（数据处理函数）

单行处理函数的特点：一个输入对应一个输出。

和单行处理函数相对的是：多行处理函数。（多行处理函数特点：多个输入，对应1个输出）

重点：所有数据库都是这样规定的，只要有null参与的运算结果一定是null

Ifnull（）空处理函数？

Ifnull（可能为null的数据 ， 被当作什么处理）； 属于单行处理函数

单行处理函数常见的有哪些？

Lower 转换大小写：

Select lower（name）from list；

Select lower（name）as name from list；

多少个输入最后就有多少个输出

Upper 转换大写：

Select upper（name）from list；

Substr 取子串（substr（被截取的字符串，起始下标，截取长度））

注意：起始下标从1开始，没有0

Select substr(name,1,1) from list;

找出员工名字第一个字母是A的员工信息？

第一种：模糊查询

Select name from list where name like ‘A%’；

第二种：substr函数

Select name from list where substr(name,1,1)=’A’;

Concat函数进行字符串的拼接

Select concat(number,name) from list；

Number，name为字符串

将名字首字母大写：

Select name from list;

Select upper(substr(name,1,1)) from list;#将首字母大写

Select substr(name,2,length(name)-1) from list;#将后面的字母截取

拼接：select concat(upper(substr(name,1,1)),substr(name,2,length(name)-1) from list;#拼接

Length 去长度

Select length（name）from list;

Trim 去空格：去除前后空白

Select \* from list where name=’ king’;

该条命令式查询不到的，因为表中的king没有空格，所以不能与他对应

Select \* from list where name=trim(‘ king’);

Str\_to\_date 将字符串转换成日期

Date\_format格式化日期

Format 设置千分位

Case…when…then…when…then…else…end

注释：case 事件 when 条件1 then 做法1 when 条件2 then 做法2 else 做法3 end；

对于某一事件当满足条件1，该怎么做；满足条件2，该怎么做；其他情况下怎么做。

当员工的工作岗位是MANAGER的时候，工资上调10%，当工作岗位是SALESMAN的时候，工资上调50%，其他正常。（注意：不修改数据库，只是将查询结果显示为工资上调）

Select name，reward as oldreward，（Case job when ‘MANAGER‘ then reward\*1.1 when ’SALESMAN’ then reward\*1.5 else reward end ）as newreward from list；

Round 四舍五入

Select 字段 from 表名；

Select name from list；#查询的是list中name字段中的值

Select abc from list；“如果表中没有abc”

mysql>select abc from metrics;

ERROR 1054 (42S22): Unknown column 'abc' in 'field list'

肯定会报错，因为会把abc当作一个字段名字，去list表中找abc字段去了

Select ‘abc‘ from list；//select 后面直接跟字面值或字面量 加引号了即为字面量/字面值

mysql> select 'abc' from metrics;

+-----+

| abc |

+-----+

| abc |

| abc |

| abc |

| abc |

| abc |

| abc |

| abc |

| abc |

结论：select 后面可以跟某个表的字段名（可以等同看作变量名）,也可以跟字面量/字面值（数据）

Select round（1234.24356,0）as result from metrics;#round表四舍五入，括号中的零表示保留到整数。若为1以表示保留一位小数.若为-1则表示保留到十位

mysql> select round(123.2434,0) as result from metrics;

+--------+

| result |

+--------+

| 123 |

| 123 |

| 123 |

| 123 |

| 123 |

| 123 |

Rand() 生成随机数

Select rand（）from list；

Select rand（）\*100 from list；

Select round（rand（）\*100，0） from list；#100以内的随机数

Ifnull 可将null转换成一个具体值

Ifnull 是空处理函数，专门处理空的

在所有数据库当中，只要有NULL参与的数学运算，最终结果就是NULL

Select name，reward+buzhu as gongzi from list；

只要有空参与，结果就是空，为了避免这个现象，需要使用ifnull函数。

Ifnull函数用法：ifnull（数据，被当作那个值处理）

如果“数据”为NULL的时候把这个数据结构当作那个值。

上述函数中buzhu补助为NULL的时候，将补助当作0；

Select name ，reward+ifnull（buzhu，0）as result from list；

# 分组函数：（多行处理函数）

记住：所有的分组函数都是对“某一组”数据进行操作

多行处理函数的特点：输入多行，最终输出一行。分组函数自动忽略NULL（

求和时能出来结果，计数时null也被忽略）

5个：count 计数 sum 求和 avg 平均值 max 最大值 min 最小值

注意：

分组函数在使用的时候必须先进行分组，然后才能用。

如果没有对数据进行分组，整张表默认为一组。

找出最高工资？

Select max（reward）from list；

找出最低工资？

Select min（reward） from list；

计算工资和？

Select sum（reward） from list；

计算平均工资？

Select avg（reward） from list；

计算员工数量？

Select count（任意表中存在字段，如name，reward，number都可以）from list；

分组函数在使用时需要注意哪些？  
1.分组函数自动忽略NULL，你不需要提前对NULL进行处理。

2.分组函数中，count（\*）和count（具体字段）有什么区别？

Select count（reward） from list；

Select count（\*） from list；

Count（具体字段）：表示统计该字段下所有不为NULL的元素的总数

Count（\*）：统计表当中的行数。

因为每一行记录不可能都为NULL，一行数据中中要有一个数据不为NULL，则这一行数据就是有效的。

1. 分组函数不能直接使用在where子句中。

找出比最低工资高的员工信息：

Select name，reward from list where reward>min（reward）；

运行会出错的

为什么分组函数不能直接写在where后面？

因为分组函数在使用的时候必须进行分组之后才能使用。

Where在执行时，还没有分组，所以where后面不能出现分组函数

1. 所有的分组函数可以组合起来一起用。

Select sum(reward),min(reward),max(reward),avg(reward),count(\*) from list;

# 分组查询

（可以先分组，在进行计算）group by

在实际的应用中，可能有这样的需求，需要先进行分组，然后对每一组的数据进行操作，这个时候我们需要使用分组查询，怎么进行分组查询呢？

Select…from…group by （字段1）；

通过字段进行分组

计算每个部门的工资和？

计算每个工作岗位的平均薪资？

找出每个工作岗位的最高薪资？

将之前的关键字全部组合在一起，他们的执行顺序？  
select…from…where…group by…order by…

以上关键字的顺序不能颠倒，需要记忆

执行顺序：

1. from
2. where
3. group by 当语句中没有group by时不代表没有分组，只是将整张表分为一组
4. select
5. order by

为什么分组函数不能直接写在where后面？

因为分组函数在使用的时候必须进行分组（执行顺序）之后才能使用。

Where在执行时，还没有分组，所以where后面不能出现分组函数

# 多字段分组查询

找出每个工作岗位的工资和？

实现思路：按照工作岗位分组，然后对工资求和?(各个组的求和)

Select job,sum(reward) from list group by job;

以上这个语句的执行顺序？

先从list表中查询数据。

根据job字段进行分组。

然后对每一组的数据进行sum(reward)

Select name,job,sum(reward) from list group by job;

以上语句在mysql中可以执行，但毫无意义；在Oracle中执行报错。

重点结论：

在一条select语句当中，如果有group by语句的话，select后面只能跟：参与分组的字段，以及分组函数。其他的一律不能跟

找出每个部门的最高薪资？

实现思路？按照部门编号分组，求每一组的最大值。

Select number，max（reward）from list group by number；

\*找出每个部门不同工作岗位的最高薪资？

技巧：两个字段联合成一个字段看。（两个字段量和分组）

每个部门的最高工资：Select bumen max(reward) from list group by bumen;

每个岗位的最高工资：Select gangwei max(reward) from list group by gangwei;

每个部门不同岗位的最高工资：select bumen，gangwei,max(reward) from list group by bumen,gangwei;

多字段联合分组时，字段间用逗号隔开

# Having

使用having可以对分完组之后的数据进一步过滤。Having不能单独使用，having不能代替where，having必须和group by联合使用。

找出每个部门最高薪资，要求显示最高薪资大于3000的？

法1：

第一步：找出每个部门最高薪资：按部门分组，求每组最大值。

Select bumen，max（reward）from list group by bumen;

第二步：要求显示最高薪资大于3000.

Select bumen，max(reward) from list group by bumen having max(reward)>3000;

思考一个问题：以上的SQL语句执行效率是不是低？

比较低，实际上可以先将大于3000的都找出来，然后再分组

Select bumen,max(reward) from list where reward>3000 group by bumen;

优化策略：where和having，优先选择where，where实在完成不了，在选择having

Where没办法：

找出每个部门平均薪资，要求显示平均薪资高于2500的？

第一步：找出每个部门平均薪资

Select bumen,avg(reward) from list group by bumen;

第二步：要求显示平均薪资高于2500的

Select bumen ，avg(reward) from list group by bumen having avg(reward)>2500;

总结：

表单的查询：

Select。。。from。。。where。。。group by。。。having。。。order by。。。（语言编写时顺序不能变）

执行顺序：

1. from
2. where
3. group by
4. having
5. select
6. order by

先从某张表中查询数据，先经过where条件筛选出有价值的数据。

对这些数据进行分组，分组后可以使用having继续筛选。

Select查询出来。

最后排序

# 去除重复记录

（原表数据不会被修改，只是查询结果去重）  
去重需要使用一个关键字：distinct (只能出现在所有字段的最前方)

Select distinct job from list；

Select name，distinct job from list； 这样编写是错误的，语法错误。

//distinct只能出现在所有字段的最前方

Distinct出现在job，name两个字段之前，表示两个字段联合起来去重。

Select distinct job，name from list；

例：统计一下工作岗位的数量？

Select count（distinct gangwei）from list；

# 连接查询

从一张表中查询称为单表查询。

从两张不同的表联合起来查询数据，从一张表中查询一项，从另一张表中查询另一项，这种跨表查询，多张表联合起来查询数据，被称为连接查询。

根据表连接的方式分类：

1. 内连接：等值连接，非等值连接，自链接
2. 外连接：左外连接（左连接），右外连接（右链接）

2.3、当两张表进行连接查询时，没有任何条件的限制会发生什么现象？

案例：查询每个员工所在部门名称？

mysql> select ename,deptno from emp;

+--------+--------+

| ename | deptno |

+--------+--------+

| SMITH | 20 |

| ALLEN | 30 |

| WARD | 30 |

| JONES | 20 |

| MARTIN | 30 |

| BLAKE | 30 |

| CLARK | 10 |

| SCOTT | 20 |

| KING | 10 |

| TURNER | 30 |

| ADAMS | 20 |

| JAMES | 30 |

| FORD | 20 |

| MILLER | 10 |

+--------+--------+

mysql> select \* from dept;

+--------+------------+----------+

| DEPTNO | DNAME | LOC |

+--------+------------+----------+

| 10 | ACCOUNTING | NEW YORK |

| 20 | RESEARCH | DALLAS |

| 30 | SALES | CHICAGO |

| 40 | OPERATIONS | BOSTON |

+--------+------------+----------+

两张表连接没有任何条件限制：

select ename,dname from emp, dept;

+--------+------------+

| ename | dname |

+--------+------------+

| SMITH | ACCOUNTING |

| SMITH | RESEARCH |

| SMITH | SALES |

| SMITH | OPERATIONS |

| ALLEN | ACCOUNTING |

| ALLEN | RESEARCH |

| ALLEN | SALES |

| ALLEN | OPERATIONS |

...

56 rows in set (0.00 sec)

14 \* 4 = 56

当两张表进行连接查询，没有任何条件限制的时候，最终查询结果条数，是

两张表条数的乘积，这种现象被称为：笛卡尔积现象。（笛卡尔发现的，这是

一个数学现象。）

2.4、怎么避免笛卡尔积现象？

连接时加条件，满足这个条件的记录被筛选出来！

select

ename,dname

from

emp, dept

where

emp.deptno = dept.deptno;

select

emp.ename,dept.dname

from

emp, dept

where

emp.deptno = dept.deptno;

加条件是起选择作用，也是为了数据的有效性

注意：加了条件不改变匹配次数，只是为了找出有效的组合记录显示出来

// 表起别名。很重要。效率问题。

select

e.ename,d.dname

from

emp e, dept d

where

e.deptno = d.deptno;

+--------+------------+

| ename | dname |

+--------+------------+

| CLARK | ACCOUNTING |

| KING | ACCOUNTING |

| MILLER | ACCOUNTING |

| SMITH | RESEARCH |

| JONES | RESEARCH |

| SCOTT | RESEARCH |

| ADAMS | RESEARCH |

| FORD | RESEARCH |

| ALLEN | SALES |

| WARD | SALES |

| MARTIN | SALES |

| BLAKE | SALES |

| TURNER | SALES |

| JAMES | SALES |

+--------+------------+

思考：最终查询的结果条数是14条，但是匹配的过程中，匹配的次数减少了吗？

还是56次，只不过进行了四选一。次数没有减少。

注意：通过笛卡尔积现象得出，表的连接次数越多效率越低，尽量避免表的

连接次数。

# 内连接之等值连接

特点：条件是等量关系

案例：查询每个员工的部门名称，要求显示员工名和部门名称。

Sql92(已经不用了，连接条件跟条件查询容易混淆)

select emp.ename,dept.dname from emp,dept where emp.deptno = dept.deptno;

select ename,dname from emp,dept where emp.deptno = dept.deptno;

sql99：（常用）

select emp.ename,dept.dname from emp join dept on emp.deptno = dept.deptno;

语法：select

…

from

A

join //这里省略了inner，带上inner可读性高，inner join

B

on

连接条件

Where

…

# 内连接之非等值连接

特点：连接条件中的关系是非等量关系

案例：找出每个员工的工资等级，要求显示员工名，工资，工资等级。

select ename,sal,grade from emp join salgrade on emp.sal between salgrade.losal and salgrade.hisal;

# 内连接之自连接

特点：一张表看作两张表，自己连接自己

案例:找出每个员工的上级领导，要求显示员工名和对应的领导名。

select a.ENAME as '员工',b.ENAME as '领导' from emp a inner join emp b on a.mgr = b.empno;

# 外连接

什么是外链接，和内连接的区别是什么

内连接：

假设A和B表进行连接，使用内连接的话，凡是A表和B表能够匹配上的记录查询出来，匹配不上就不查询，这就是内连接。AB两张表没有主副之分 ，两张表平等

外连接：

假设A和B表进行连接，使用外连接的话，AB两张表中有一张是主表，一张是副表，主要查询主表中的数据，捎带查询副表，当副表中的数据没有和主表中的数据匹配上，副表自动模拟出null与之匹配

所以：内连接查询会丢失数据

外连接的分类？

左连接：表示左边这张表是主表（关键字：left）

右连接：表示右边这张表是主表

案例：找出每个员工的上级领导领导。

select a.ENAME as '员工',b.ENAME as '领导' from emp a left join emp b on a.mgr = b.empno;

# 三个表连接

案例：找出每一个员工的部门名称一级工资等级。

select

emp.ename,dept.dname,salgrade.grade

from

emp

join

dept

on

emp.deptno = dept.deptno

join

salgrade

on

emp.sal

between

salgrade.losal and salgrade.hisal;

# 子查询

Select语句中嵌套select语句，被嵌套的select语句称为子查询。

子查询都可以出现在哪里呢？

Select

。。（select）。

From

。。（select）。

Where

。。（select）。

## Where子句中的子查询

案例：找出比最低工资高的员工姓名和工资？

Select

Name，reward

From

List

Where

Reward>min(reward);

不能实现。因为where子句中不能直接使用分组函数。

实现思路：

第一步：找出最低工资是多少

Select min（reward）from list；

查询结果为x

第二步:工资比最低工资高的

Select name，reward from list where reward>x；

第三步：合并

Select name，reward from list where reward>( Select min（reward）from list);

## From子句中的子查询

注意：from后面的子查询，可以将子查询的查询结果当作一张临时表；

案例：找出高于平均薪资的员工信息。

select \* from emp where sal > (select avg(sal) from emp);

案例：找出每个岗位的平均工资的薪资等级。

第一步：找出每个岗位的平均工资（按照岗位分组求平均值）

select job,avg(sal) from emp group by job;

+-----------+-------------+

| job | avgsal |

+-----------+-------------+

| ANALYST | 3000.000000 |

| CLERK | 1037.500000 |

| MANAGER | 2758.333333 |

| PRESIDENT | 5000.000000 |

| SALESMAN | 1400.000000 |

+-----------+-------------+t表

第二步：克服心理障碍，把以上的查询结果就当做一张真实存在的表t。

mysql> select \* from salgrade; s表

+-------+-------+-------+

| GRADE | LOSAL | HISAL |

+-------+-------+-------+

| 1 | 700 | 1200 |

| 2 | 1201 | 1400 |

| 3 | 1401 | 2000 |

| 4 | 2001 | 3000 |

| 5 | 3001 | 9999 |

+-------+-------+-------+

t表和s表进行表连接，条件：t表avg(sal) between s.losal and s.hisal;

select

t.\*, s.grade

from

(select job,avg(sal) as avgsal from emp group by job) t

join

salgrade s

on

t.avgsal between s.losal and s.hisal;

+-----------+-------------+-------+

| job | avgsal | grade |

+-----------+-------------+-------+

| CLERK | 1037.500000 | 1 |

| SALESMAN | 1400.000000 | 2 |

| ANALYST | 3000.000000 | 4 |

| MANAGER | 2758.333333 | 4 |

| PRESIDENT | 5000.000000 | 5 |

|  |  |  |
| --- | --- | --- |
|  |  |  |

## select子句中的子查询

案例： 找出每个员工坐在的部门名称，要求显示员工名和部门名。

# Union的用法

Union合并查询结果集（可以将查询结果集相加）

案例：查询工作岗位是MANAGER和SALESMAN的员工？

Select name ，job from list where job=’MANAGER’ or job=’SALSEMAN’：

Select name ，job from list where job in(’MANAGER’ , ’SALSEMAN’);

使用union：Select name ，job from list where job=’MANAGER’

Union

Select name ，job from list where job=’SALSEMAN’;

三种方法结果一样

优点：可以将两张不相关的表查询结果结合在一起（查询列数要一样）

Union的效率要高一些。对于表连接来说，每连接一次新表，则匹配的次数满足笛卡尔积，成倍的翻。。。

但是union可以减少匹配的次数，在减少匹配次数的情况下，还可以完成两个结果集的拼接。

a 连接 b 连接 c

a 10条记录

b 10条记录

c 10条记录

匹配次数是：1000

a 连接 b一个结果：10 \* 10 --> 100次

a 连接 c一个结果：10 \* 10 --> 100次

使用union的话是：100次 + 100次 = 200次。（union把乘法变成了加法运算）

Union在使用时注意事项：

1.Select name ，job from list where job=’MANAGER’

Union

Select name from list where job=’SALSEMAN’;

//运行后是错误的；union在进行结果集合并的时候，要求两个结果集的列数相同

2. Select name ，job from list where job=’MANAGER’

Union

Select name ，reward from list where job=’SALSEMAN’;

//虽然可以运行，但是存在语法错误（MASQL可以运行，oracle语法严格，不可以运行，报错）。要求：结果集合并时列和列的数据类型也要一致。

# Limit

Limit作用：将查询结果集的一部分取出来。通常使用在分页查询中。

百度默认：一页显示10条记录。

分页的作用是为了提高用户的体验，因为一次全部查出来，用户体验差。可以一页一页翻着看。

Limit怎么用？

完整用法：limit startindex，length

Startindex是起始下标，length是长度 起始下标从0开始

缺省用法：limit 5；取前五

按照薪资降序，去除排名前5名的员工？

Select

name ，reward

from

list

order by

reward desc

limit 5；//取前5

Select

name ，reward

from

list

order by

reward desc

limit 0，5；

注意：mysql当中的limit在order by之后执行！（limit是sql语言中最后执行的）

案例：取出工资排名在【3-5】名的员工？

Select name，reward from list order by reward desc limit 2，3；

2表示其实为位置（第三个位置）

3表示长度为3

# 分页

每页显示3条记录

第1页：limit 0，3 【0 1 2】

第2页：limit 3，3 【3 4 5】

第3页：limit 6，3 【6 7 8】

第4页：limit 9，3 【9 10 11】

每页显示pagesize条记录

第pageNo页：limit（pageNo-1）\*pagesize，pagesize

Public static void main（string[ ] args）{

//用户提交过来一个页码，以及买也显示的记录条数

Int pageNo=5；//第5页

Int startindex=（pageNo-1）\*pagesize；

String sql=‘select。。。limit’ + startindex + ‘，’ + pagesize；

}

记公式：

Limit（pageNo-1）\*pagesize，pagesize

# 关于DQL语句的大总结

Select

。。。

From

。。。

Where

。。。

Group by

。。。

Having

。。。

Order by

。。。

Limit

。。。

执行顺序：from，where，group by，having，select，order by，limit

# 表的创建及数据类型

## 表的创建：（建表）

建表的语法格式：（建表属于DDL语句，DDL包括：create drop alter）

Create table 表名（字段名1 数据类型，字段名2 数据类型，字段名3 数据类型。。。）；

Create table 表名（

字段名1 数据类型，

字段名2 数据类型，

字段名3 数据类型

。。。

）；

表名：建议以t\_或者 tbl\_开始，可读性强，见名知意

字段名：见名知意

表名和字段名都属于标识符

## 关于mysql中的数据类型？

很多数据类型，我们只需要掌握一些常见的数据类型即可。

### Varchar

（最长255）：可变长度的字符串，比较智能，节省空间，会根据实际的数据长度动态分配空间。

优点：节省空间

缺点：需要动态分配空间，速度慢

### Char

（最长255）：定长字符串，不管实际的数据长度是多少，分配固定长度的空间去存储数据。使用不恰当的时候，可能会导致空间的浪费。

优点：不需要动态的分配空间，速度快

缺点：使用不当可能会导致空间的浪费

varchar和char我们应该怎么选择？

性别字段你选什么？因为性别是固定长度的字符串，所以选择char。

姓名字段你选什么？每一个人的名字长度不同，所以选择varchar。

### Int

(最长11)

数字中的整数型

### Bigint

数字中的长整型，相当于c++中的long

### Float

单精度浮点型

### Double

双精度浮点型

### Date

段日期类型

### Datetime

长日期类型

### Clob

字符大对象

最多可以存储4G的字符串

比如：存储一篇文章，储存一个说明

超过255个字符的都要采用CLOB字符大对象来储存

Character Large Object:CLOB

### Blob

二进制大对象

Binary Large OBject

专门用于存储图片，声音，视频等流媒体数据。

往BLOB类型的字段上插入数据的时候，例如插入一个图片，视频等，需要IO流才行

t\_movie 电影表（专门存储电影信息的）

编号 名字 故事情节 上映日期 时长 海报 类型

no(bigint) name(varchar) history(clob) playtime(date) time(double) image(blob) type(char)

------------------------------------------------------------------------------------------------------------------

10000 哪吒 ........... 2019-10-11 2.5 .... '1'

10001 林正英之娘娘 ........... 2019-11-11 1.5 .... '2'

....

创建一个学生表？

学号、姓名、年龄、性别、邮箱地址

create table t\_student(

no int,

name varchar(32),

sex char(1),

age int(3),

email varchar(255)

);

# 删除表

drop table t\_student; // 当这张表不存在的时候会报错！

// 如果这张表存在的话，删除

drop table if exists t\_student;

# 插入数据insert（DML）

### 一次插入一条

语法格式：

Insert into 表名（字段名1，字段名2，字段名3.。。。） values（值1，值2，值3.。。。）；

注意：字段名和值要一一对应，什么是一一对应？数量要对应，数据类型要对应。

Insert into t\_student(no,name,sex,age,email) values(1,'zhangsan','m',20,'zhangsan@123.com');

### 可以颠倒字段顺序

insert into t\_student(email,name,sex,age,no) values('lisi@123.com','lisi','f',20,2);

### 只插入部分字段

注意：不代表不给其他部分插入值，而是插入NULL

insert into t\_student(no) values(3);

select \* from t\_student;

+------+----------+------+------+------------------+

| no | name | sex | age | email |

+------+----------+------+------+------------------+

| 1 | zhangsan | m | 20 | zhangsan@123.com |

| 2 | lisi | f | 20 | lisi@123.com |

| 3 | NULL | NULL | NULL | NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

insert into t\_student(name) values('wangwu');

+------+----------+------+------+------------------+

| no | name | sex | age | email |

+------+----------+------+------+------------------+

| 1 | zhangsan | m | 20 | zhangsan@123.com |

| 2 | lisi | f | 20 | lisi@123.com |

| 3 | NULL | NULL | NULL | NULL |

| NULL | wangwu | NULL | NULL | NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

注意：insert语句但凡执行成功了，那么必然会多一条记录。

没有给其他字段指定值的话，默认为NULL。后期也没有办法再执行insert语句插入数据了，只能使用update进行更新了

### 默认值

drop table if exists t\_student;

create table t\_student(

no int,

name varchar(32),

sex char(1) default 'm', //建表时的默认值，建表创建默认值，当插入数据时没有该字段的值输入时，表中存放的就是默认值，若输入其他值时，则默认值更改为输入值

age int(3),

email varchar(255)

);

+-------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+--------------+------+-----+---------+-------+

| no | int(11) | YES | | NULL | |

| name | varchar(32) | YES | | NULL | |

| sex | char(1) | YES | | m | |

| age | int(3) | YES | | NULL | |

| email | varchar(255) | YES | | NULL | |

+-------+--------------+------+-----+---------+-------+

insert into t\_student(no) values(1);

mysql> select \* from t\_student;

+------+------+------+------+-------+

| no | name | sex | age | email |

+------+------+------+------+-------+

| 1 | NULL | m | NULL | NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

## Insert语句中的‘字段名‘可以省略吗？可以

Insert into t\_student values(2);//错误的

//注意：前面的字段名省略的话，等于都写上了！所以只也要都写上！（插入数量和数据类型都要跟字段名数量和数据类型匹配）

//insert into t\_student values（2，‘lisi‘，’f’，20，‘lisi@1233.com’）；

+------+------+------+------+--------------+

| no | name | sex | age | email |

+------+------+------+------+--------------+

| 1 | NULL | m | NULL | NULL |

| 2 | lisi | f | 20 | lisi@123.com |

## Insert一次可以插入多条记录吗？可以

语法：

Insert into 表名（字段名1，字段名2.。。。）values（值1，值2.。。。），（值1‘，值2’。。。）。。。

Insert into t\_user(id,name,birth,create\_time) values(1,’zs’,’2000-01-01’,now()),

(2,’ls’,’2000-01-01’,now()),

(3,’ww’,’2000-01-01’,now());

## Insert插入日期

Str\_to\_date:将字符串varchar类型转换成date类型

Dete\_format:将date类型转换成具有一定格式的varchar类型。

Drop table if exists t\_user;

Create table t\_user(

Id int,

Name varchar(32),

Birth date//生日也可以使用date类型

）；

Create table t\_user(

Id int,

Name varchar(32),

Birth char(10)//生日可以使用字符串

）；

生日：1995-11-14（10个字符）

注意：数据库中的有一条命名规范：

所有的标识符（表名和字段名）都是全部小写，单词和单词之间使用下划线进行衔接。

mysql> desc t\_user;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| Id | int | YES | | NULL | |

| Name | varchar(32) | YES | | NULL | |

| Birth | date | YES | | NULL | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

## 插入数据？

Insert into t\_user(id,name,birth) values(1,’zhangsan‘，‘1995-11-14’)；

出问题了：原因是类型不匹配。数据库birth是date类型，这里给了一个字符串varchar。

怎么办？可以使用str\_to\_date函数进行类型转换。

Str\_to\_date函数可以将字符串转换成日期类型date。

语法格式：

Str\_to\_date(‘字符串日期‘，‘日期格式’)

MYSQL的日期格式：

%Y 年 %m 月 %d 日 %h 时 %i 分 %s 秒

insert into t\_user(id,name,birth) values(1, 'zhangsan', str\_to\_date('01-10-1990','%d-%m-%Y'));

str to date函数可以把字符串varchar转换成日期date类型数据，

通常使用在插入insert方面，因为插入时需要一个日期类型数据，

需要通过该函数将字符串转换成date。

好消息？

如果你提供的日期字符串是这个格式，str\_to\_date函数就不需要了！！！

%y-%m-%d

insert into t\_user(id,name,birth) values(1, 'zhangsan', '1990-01-01');

+------+----------+------------+

| Id | Name | Birth |

+------+----------+------------+

| 1 | zhangsan | 1990-10-01 |

| 1 | zhangsan | 1990-01-01 |

+------+----------+------------+

Date\_format日期格式化：  
查询的时候可以以某个特定的日期格式展示吗？

Date\_format-日期格式化

这个函数可以将日期类型转换成特定格式的字符串。

Select id,name,date\_format(birth,'%m/%d/%y') as birth from t\_user;

mysql> Select id,name,date\_format(birth,'%m/%d/%y') as birth from t\_user;

+------+----------+----------+

| id | name | birth |

+------+----------+----------+

| 1 | zhangsan | 10/01/90 |

| 1 | zhangsan | 01/01/90 |

| 1 | 张三 | 01/01/90 |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Date\_format函数怎么用？

Dete\_format(日期使用数据类型，‘日期格式’)

这个函数通常使用在查询日期方面。设置展示的日期格式。

Select id,name,birth from t\_user;

+------+----------+------------+

| id | name | birth |

+------+----------+------------+

| 1 | zhangsan | 1990-10-01 |

| 1 | zhangsan | 1990-01-01 |

| 1 | 张三 | 1990-01-01 |

|  |  |  |
| --- | --- | --- |
|  |  |  |

以上的 SQL语句实际上是进行默认的日期格式化，

自动将数据库库中的date类型转换成varchar类型

并且采用的格式是mysql默认日期格式：’%y-%m-%d’

Select id,name,date\_format(birth,’%y-%m-%d’) as birth from t\_user;

Date和datetime两个类型的区别？

Date是短日期：只包括年月日信息

Datetime是长日期：包括年月日时分秒信息

Drop table if exists t\_user;

Create table t\_user(

Id int,

Name varchar(32),

Birth date,

Create\_time datetime

);

Id是整数 name是字符串 birth是短日期 creat\_time是创建这条记录的创建时间：长日期类型

Mysql短日期默认格式：%y-%m-%d

Mysql长日期默认格式：%y-%m-%d %h:%i:%s

insert into t\_user(id,name,birth,create\_time) values(1,'zhangsan','1990-01-01','2020-11-26 10:01:01');

mysql> select \* from t\_user;

+------+----------+------------+---------------------+

| Id | Name | Birth | Create\_time |

+------+----------+------------+---------------------+

| 1 | zhangsan | 1990-01-01 | 2020-11-26 10:01:01 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

在MSYSQL当中怎么获取系统当前时间？

Now（）函数，并且获取当前的时间带有时分秒信息！是datetime类型的

insert into t\_user(id,name,birth,create\_time) values(1,'zhangsan','1990-01-01',now());

# 表的复制及批量插入

## 表的复制

语法：create table 表名 as select语句

将查询结果当作表创建出来

例如：create table emp2 as select \* from emp；

## 将查询结果插入到一张表中

Insert into 表名 select语句

例如：insert into dept1 select \* from dept；

# update修改（DML）

语法格式：  
update 表名 set 字段名1=值1，字段名2=值2，字段名3=值3.。。。where 条件；

注意：没有条件限制会导致所有数据全部更新。

Update t\_user set name=’jack’,birth=’2000-10-11’,create\_time=now() where id=2;

# Delete删除数据（DML）

语法格式：

delete from 表名 where 条件；

注意：没有条件，整张表的数据会全部删除！

delete from t\_user where id=2;

删除表中所有记录

Delete from表名；

快速删除表中的数据？

//删除det\_bak表中的数据，不删除表

Delete from det\_bak;//这种删除数据的方式比较慢

Delete语句删除数据的原理？

表中的数据被删除了，但这个数据在硬盘上的真实储存空间不会被释放！！

这种删除的缺点：删除效率比较低。

这种删除的优点：支持回滚，后悔了可以恢复数据！

格式：delete from 表名；

## 回滚（恢复）：直接输入rollback；

## 删除大表

只删除表中数据

Truncate语句删除数据的原理？  
这种删除效率比较高，表被一次截断（除了字段名保留，其他数据一次截断），物理删除。

缺点：不支持回滚（删除）

优点：快速

语法格式：  
Truncate table 表名；

使用rollback恢复不了，将会是一个空表格

大表非常大，上亿条记录？？

删除的时候，使用delete，也许需要执行1个小时才能删除完！效率低。

可以选择使用truncate删除表中的数据。只需要不到一秒钟就删除结束，效率较高。

但是用truncate之前再三确认是否删除。

# 快速创建表？

Create table emp2 as select \*from emp;

原理：

将一个查询结果当作一张表新建！

这个可以完成表的快速复制！

表创建出来，同时表中的数据也存在了！

Create table mytable as select number,name from list where job=’MANAGER’;

意思是查询出来的信息当作一张表新建出来

将查询结果插入到一张表中？insert相关的！！

查询出来后新创建：Create table dep\_bak as select \* from dept;

查询出来后插入一个列表：insert into det\_bak select \* from dept;#查询到的字段数和要插入表的字段数一致

删除表操作？

Drop table 表名；//这不是删除表中数据，这是把表删除。

# 表结构的增删改

什么是对表结构的修改？  
添加一个字段，删除一个字段，修改一个字段！

由于修改表结构的操作很少，所以我们不需要掌握。

修改表结构的代码不会出现在代码中

# 创建表加入约束：（\*\*\*\*\*\*）

什么是约束？

在创建表的是时候，我们可以给表中的字段加上一些约束，来保证这个表中数据的完整性，有效性！

约束的作用：为了保证表中数据有效！

常见的约束：

1. 非空约束：not null：表示该字段默认不能为空（只有列级约束，没有表级约束）
2. 唯一性约束：unique：表示该字段不能出现重复数据，但可以为NULL
3. 主键约束：primary key（简称PK）：表示该字段既不能为NULL，也不能重复
4. 外键约束：foreign key（简称FK）
5. 检查约束：check（mysql不支持，oracle支持）

我们重点学习：1，2，3，4

## 非空约束not null

约束的字段不能为NULL。

Drop table if exists t\_vip;

Create table t\_vip(

Id int,

Name varchar(255) not null

);

Insert into t\_vip(id,name) values(1,’zhangsan’);

Insert into t\_vip(id,name) values(2,’lisi’);

Insert into t\_vip(id) values(3);

## 唯一性约束:unique

唯一性约束unique约束的字段不能重复，但可以为NULL。

Drop table if exists t\_vip;

Create table t\_vip(

Id int,

Name varchar(255) unique

);

Insert into t\_vip(id,name) values(1,’zhangsan’);

Insert into t\_vip(id,name) values(2,’zhangsan’);

ERROR 1062 (23000): Duplicate entry 'zhangsan' for key 't\_vip.Name'(重复)

如果某个字段被unique约束了，但可以为NULL（NULL可以重复，NULL并非一个值）

## 两个字段联合唯一

## 列级约束

新需求：name和email两个字段联合起来具有唯一性！

Drop table if exists t\_vip;

Create table t\_vip(

Id int,

Name varchar(255) unique,//约束直接添加到列后面的，叫做列级约束

Email varchar(255) unique

);//这是各自唯一，这样创建是不符合我以上“新需求”的。

这样创建表示：name具有唯一性，email具有唯一性。各自唯一。

以下这样的数据是符合我“新需求”的。

但如果采用以上方式创建表的话，肯定创建失败，因为'zhangsan'和'zhangsan'重复了。

insert into t\_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');

insert into t\_vip(id,name,email) values(2,'zhangsan','zhangsan@sina.com');

怎样创建这样的表，才符合新需求呢？

## 表级约束

Drop table if exists t\_vip;

Create table t\_vip(

Id int,

Name varchar(255)，

Email varchar(255)，

Unique（name，email）//约束没有添加在列的后面，这种约束被称为表级约束

);

insert into t\_vip(id,name,email) values(1,'zhangsan','zhangsan@123.com');

insert into t\_vip(id,name,email) values(2,'zhangsan','zhangsan@sina.com');

select \* from t\_vip;

Name和email两个字段联合起来唯一

insert into t\_vip(id,name,email) values(3,'zhangsan','zhangsan@sina.com');

ERROR 1062 (23000): Duplicate entry 'zhangsan-zhangsan@sina.com' for key 'name'

什么时候使用表级约束呢？

需要给多个字段联合起来添加某一个约束的时候，需要使用表级约束

Unique和not null可以联合吗？

Drop table if exists t\_vip;

Create table t\_vip(

Id int,

Name varchar(255) not null unique

);

mysql> desc t\_vip;

+-------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+--------------+------+-----+---------+-------+

| id | int(11) | YES | | NULL | |

| name | varchar(255) | NO | PRI | NULL | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

在MySQL当中，如果一个字段同时被not null和unique约束的话，

该字段自动变成主键字段（PRI）。（注意：oracle中不一样）

Insert into t\_vip(id,name) values(1,’zhangsan’);

Insert into t\_vip(id,name) values(2,’zhangsan’);//错误了：name不能重复

Insert into t\_vip(id) values(3);//错误了：name不能为NULL

## 主键约束（primary key,简称PK）

主键约束的相关术语？

主键约束：就是一种约束。

主键字段：该字段上添加了主键约束，这样的字段叫做：主键字段

主键值：主键字段中的每一个值都叫做：主键值

什么是主键？有啥用？

主键值是每一行记录的唯一标识

主键值是每一行记录的身份证号！不可重复

记住：每一张表都应该有主键，没有主键，表无效！！

主键的特征：not null+unique（主键值不能是NULL，同时也不能重复）

怎么给一张表添加主键约束呢？

//1个字段做主键，叫做单一主键

drop table if exists t\_vip;

create table t\_vip(

id int primary key,//列级约束

name varchar(255)

);

insert into t\_vip(id,name) values(1,'zhangsan');

insert into t\_vip(id,name) values(2,'lisi');

insert into t\_vip(id,name) values(1,'wangwu');//错误：不能重复

ERROR 1062 (23000): Duplicate entry '1' for key 't\_vip.PRIMARY'

insert into t\_vip(name) values('zhaoliu');//错误：不能为NULL

ERROR 1364 (HY000): Field 'id' doesn't have a default value

可以这样添加主键吗？使用表级约束？

drop table if exists t\_vip;

create table t\_vip(

id int，

name varchar(255)

primary key（id）//表级约束

);

表级约束主要是给多个字段联合起来添加约束

drop table if exists t\_vip;

//id和name联合起来做主键：复合主键！！！

create table t\_vip(

id int，

name varchar(255)

Email varchar（255）

primary key（id，name）

）；

Insert into t\_vip(id,name,email) values(1,’lisi’,’lisi@123.com);

Insert into t\_vip(in,name,email) values(1,’zhangsan’,’zhangsan@123.com’);

Insert into t\_vip(id,name,email) values(1,’lisi’,’lisi@123.com’);//错误：不能重复

一个表中主键约束能加两个吗？

drop table if exists t\_vip;

create table t\_vip(

id int primary key，

name varchar(255) primary key，

）；//错误

## 主键的分类

根据主键字段的字段数量来划分：

单一主键：（推荐的，常用的）

复合主键：（多个字段联合起来添加一个主键约束）（复合主键不建议使用，因为复合主键违背三范式）

根据主键性质分：

自然主键：主键之最好就是一个和业务没有任何关系的的自然数。

业务主键：主键值和系统的业务挂钩，例如：拿着银行卡的卡号做主键，拿着身份证号码做主键。（不推荐使用）最好不要拿着和业务挂钩的字段作为主键。因为以后的业务一旦发生改变的时候，主键值可能也需要随着发生变化，但有的时候没有被变法变化，因为变化可能会导致主键值重复。

结论：一张表，主键约束只能添加1个（主键只能一个）

主键值建议使用：int bigint char等类型

不建议使用：varchar来做主键。主键值一般都是数字，一般都是定长的

主键除了：单一主键和复合主键之外，还可以这样进行分类？

自然主键：主键值是一个自然数，和业务没有关系。

业务主键：主键值和业务紧密关联，例如银行卡账号做主键值，这就是业务主键！

# 主键值自增 auto\_increment

在mysql当中，有一种机制可以帮我们自动维护一个主键值？

设置了主键的字段，该字段自动维护和一个自增的数字，从1开始，以1递增

Drop table if exists t\_vip;

Create table t\_vip(

Id int primary key auto\_increment,//auto\_increment表示自增，从一开始，以1递增

Name varchar（255）

);

Insert into t\_vip(name) values(‘zhangsan’);

Insert into t\_vip(name) values(‘zhangsan’);

Insert into t\_vip(name) values(‘zhangsan’);

Insert into t\_vip(name) values(‘zhangsan’);

Insert into t\_vip(name) values(‘zhangsan’);

select \* from t\_vip;

+----+----------+

| id | name |

+----+----------+

| 1 | zhagnsan |

| 2 | zhagnsan |

| 3 | zhagnsan |

| 4 | zhagnsan |

| 5 | zhagnsan |

|  |  |
| --- | --- |
|  |  |

# 外键约束（foreign key,简称FK）

外键约束涉及到的相关术语：

外键约束：一种约束——foreign key

外键字段：添加有外键约束的字段

外键值：外键字段当中的每一个值

业务背景：

请设计数据库表，来描述“班级和学生“的信息？

第一种方案：班级和学生存储在一张表中？

NO（pk） name classno classname

1 jack 100 机械1班

2 rose 100 机械1班

3 lilin 100 机械1班

4 lisi 100 机械1班

5 wangwu 101 机械2班

6 zhaoliu 101 机械2班

7 zhangsan 101 机械2班

8 jucy 101 机械2班

分析以上方案缺点：

数据冗长，浪费空间！！设计比较失败

第二种方案：班级一张表，学生一张表？

t\_class 班级表

classno(pk) classname

100 机械1班

101 机械2班

t\_student 学生表

no（pk） name cno（FK引用t\_class这张表的classno）

1 jack 100

2 rose 100

3 lilin 100

4 lisi 100

5 wangwu 101

6 zhaoliu 101

7 zhangsan 101

8 jucy 101

子表中：

当cno字段没有任何约束的时候，可能会导致数据无效。可能出现一个102，但是102班级不存在，所以为了保证cno字段的值都是100和101，需要给cno字段添加外键约束。

那么：cno字段就是外键字段。Cno字段中的每一个值都是外键值。

注意：t\_class是父表 t\_student是子表

删除表的顺序：先删子，再删父。

创建表的顺序：先创建父，再创建子。

删除数据的顺序：先删子，再删父。

插入数据的顺序：先插入父，再插入子。

drop table if exists t\_student;

drop table if exists t\_class;

create table t\_class(

classno int primary key,//必须具有唯一性才能被引用

classname varchar(255)

);

create table t\_student(

no int primary key auto\_increment,

name varchar(255),

cno int,

foreign key(cno) references t\_class(classno)//外键：cno reference引用 表t\_class中的字段classno

);

insert into t\_class(classno,classname) values(100,'机械1班');

insert into t\_class(classno,classname) values(101,'机械2班');

insert into t\_student(name,cno) values('jack',100);

insert into t\_student(name,cno) values('rose',100);

insert into t\_student(name,cno) values('lilin',100);

insert into t\_student(name,cno) values('lisi',100);

insert into t\_student(name,cno) values('wangwu',101);

insert into t\_student(name,cno) values('zhaoliu',101);

insert into t\_student(name,cno) values('zhangsan',101);

insert into t\_student(name,cno) values('jucy',101);

select \* from t\_class;

select \* from t\_student;

mysql> select \* from t\_class;

+---------+-----------+

| classno | classname |

+---------+-----------+

| 100 | 机械1班 |

| 101 | 机械2班 |

+---------+-----------+

2 rows in set (0.00 sec)

mysql> select \* from t\_student;

+----+----------+------+

| no | name | cno |

+----+----------+------+

| 1 | jack | 100 |

| 2 | rose | 100 |

| 3 | lilin | 100 |

| 4 | lisi | 100 |

| 5 | wangwu | 101 |

| 6 | zhaoliu | 101 |

| 7 | zhangsan | 101 |

| 8 | jucy | 101 |

+----+----------+------+

Cno中只能插入100或101或者NULL

思考：子表中的外键引用的父表中的某个字段，被引用的这个字段必须是主键吗？

不一定是主键，但至少具有unique约束（唯一性）。

测试：外键可以为NULL吗？  
外键值可以为NULL

# 存储引擎：（了解内容）

什么是存储引擎？有什么用？

存储引擎是MYSQL中特有的一个术语，其他数据库中没有。

实际上存储引擎是一个表储存/组织数据的方式。

不同的储存引擎，表存储数据的方式不同。

怎么给表添加/指定‘储存引擎’呢？

Show create table t\_student;

可以在建表的时候给表指定存储引擎。

CREATE TABLE `t\_student` (

`no` int NOT NULL AUTO\_INCREMENT,

`name` varchar(255) DEFAULT NULL,

`cno` int DEFAULT NULL,

PRIMARY KEY (`no`),

KEY `cno` (`cno`),

CONSTRAINT `t\_student\_ibfk\_1` FOREIGN KEY (`cno`) REFERENCES `t\_class` (`classno`)

)ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci |

在建表的时候可以在最后小括号的“）“的右边使用：

ENGINE来指定存储引擎。

CHARSET来指定这张表的字符编码方式。

结论：

MySQL默认的存储引擎是：InnoDB

MySQL默认的字符编码方式是：utf8

怎么查看MySQL支持那些存储引擎？

Show engine-g；

查看MySQL版本？

Select version（）；

、

# 事务(transaction)

什么是事务？

一个事务其实就是一个完整的业务逻辑。

是一个最小的工作单元。不可再分。

什么是一个完整的业务逻辑？

假设转账，从A账户向B账户中转账10000。

将A账户的钱减去10000（update语句）

将B账户的钱加上10000（update语句）

这就是一个完整的业务逻辑

以上的操作是一个最小的工作单元，要么同时成功，要么同时失败，不可再分。这两个update语句要求必须同时成功或同时失败，这样才能保证钱的正确

只有DML语句才会有事务这一说，其他语句和事务无关！

Insert delete update

只有以上的三个语句和事务有关系，其他语句和事务无关！

因为 只有以上的三个语句是数据库表中数据进行增，删，改的。

只要你的操作一旦涉及到数据的增删改，那么就一定要考虑到安全问题。

假设所有的业务，只要一条DML语句就能完成，还有必要存在事务机制吗？

正是因为做某件事的时候，需要多条DML语句共同联合起来才能完成，

所以需要事物的存在。如果任何一件复杂的事情都能一条DML语句搞定，

那么事务则没有存在的价值了。

到底什么是事务呢？\*\*\*\*\*

说到底，说到本质上，一个事务其实就是多条DML语句同时成功，或者同时失败！

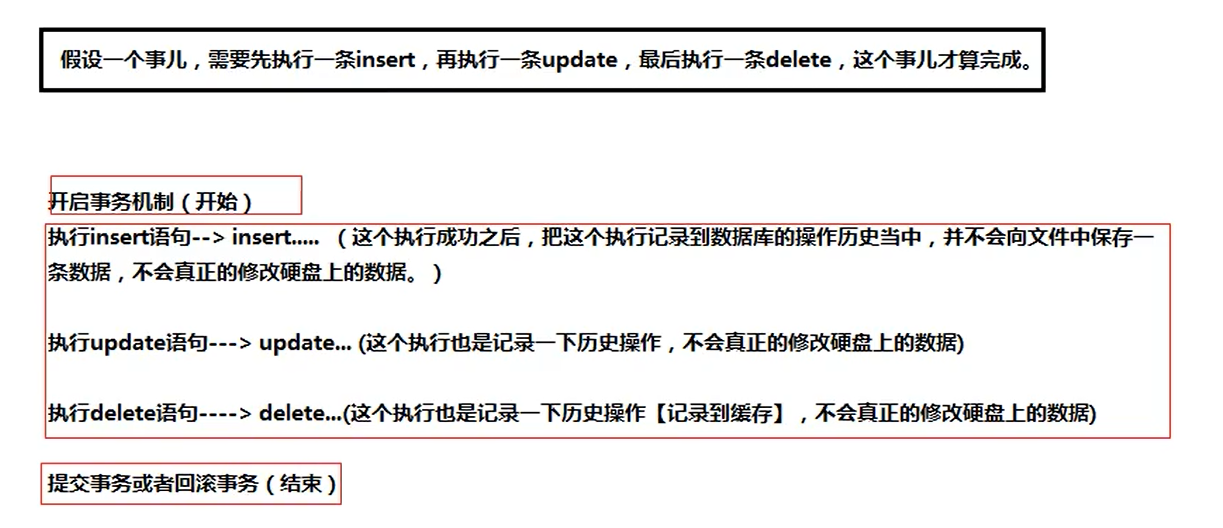
## 事务：

就是批量的DML语句同时成功，或者同时失败！！

事务是怎么做到多条DML语句同时成功和同时失败的呢

InnoDB存储引擎：提供一组用来记录事务性活动的日志文件

## 事务的原理



事务开启了：

Insert

Insert

Delete

Update

Update

Update

事务结束了！

在事务的执行过程中，每一条DML的操作都会记录到‘事务性活动的日志文件’中。

在事务的执行过程中，我们可以提交事务，也可以回滚事务。

提交事务？

清空实务性活动的日志文件，将数据全部彻底持久化到数据库表中

提交事务标志着，事务的结束。并且是一种全部成功的结束。

回滚事务？

将之前所有的DML操作全部撤销，并且清空事务性活动的日志文件

回滚事务标志着，事务的结束。并且是一种全部失败的结束。

怎么提交事务，怎么回滚事务？

提交事务：commit；语句

回滚事务：rollback；语句（回滚永远都是只能回滚到上一次的提交点！）只要提交了，则不能回滚。

事务对应的英语单词是：transaction

测试一下，在MySQL当中默认的事物行为是怎样的？

MySQL默认情况下是支持自动提交事务的。（自动提交）

## 什么是自动提交？

每执行一条DML语句，则提交一次！insert，delete，update

这种自动提交实际上不符合我们的开发习惯，因为一个业务通常是需要多条DML语句共同执行才能完成的，为了保证数据的安全，必须要求同时成功之后再提交，所以不能执行一条就提交一条。

## 怎么将MySQL的自动提交机制关掉呢？

先执行这个命令：start transaction；（开始事务）

事务的开始表示自动提交的结束，提交表示事务结束

演示事务：

---------------------------------回滚事务--------------------------------------

mysql> use bjpowernode;

Database changed

mysql> select \* from dept\_bak;

Empty set (0.00 sec)

mysql> start transaction;#关闭自动提交；（开始事务）

Query OK, 0 rows affected (0.00 sec)

mysql> insert into dept\_bak values(10,'abc', 'tj');#插入

Query OK, 1 row affected (0.00 sec)

mysql> insert into dept\_bak values(10,'abc', 'tj');#插入

Query OK, 1 row affected (0.00 sec)

mysql> select \* from dept\_bak;#查询

+--------+-------+------+

| DEPTNO | DNAME | LOC |

+--------+-------+------+

| 10 | abc | tj |

| 10 | abc | tj |

+--------+-------+------+

2 rows in set (0.00 sec)

mysql> rollback;#回滚

Query OK, 0 rows affected (0.00 sec)#变空了

mysql> select \* from dept\_bak;

Empty set (0.00 sec)

## 演示回滚和提交：

一．验证mysql默认自动提交

1. 先准备一张表（parent），select \* from parent;
2. 向其中插入数据 insert into parent(number,class) values(103,'机械三班');
3. select \* from parent;查看后表中该数据已经存在
4. rollback；回滚
5. select \* from parent;该数据依然存在

结论：rollback后插入数据以让存在，回滚该数据失败

二．演示关闭自动提交和成功回滚，提交事务

1. 先准备一张表（parent），select \* from parent;

2. start transaction;开启事务（即关闭自动提交）

3.向表中插入数据，insert into parent(number,class) values(103,'机械三班');

4. select \* from parent;查看后表中该数据已经存在

5. rollback；回滚

6. select \* from parent;查看后表中该数据不存在，表明回滚将数据缓存清除

7.再次向表中插入数据，insert into parent(number,class) values(103,'机械三班');

8.commit，提交

9. select \* from parent;查看后表中该数据已经存在

10. rollback；回滚

11.select \* from parent;查看后表中该数据存在，该数据已经提交保存至硬盘，，回滚不能成功

结论：

## 事务包括四个特性？

A: 原子性

说明事物是最小的工作单元，不可再分。

C：一致性

所有事务要求，在同一个事物当中，所有的操作必须同时成功，或者同时会失败，以保证数据的一致性。

I：隔离性

A事务和B事务之间具有一定的隔离。

教室A和教室B之间有一道墙，这道墙就是隔离性

A事务在操作一张表的时候，另一个事务B也操作这张表会怎么样？  
D: 持久性

事务最终结束的一个保障。事务提交，就相当于将没有保存到硬盘上的数据保存到硬盘上！

重点研究一下事务的隔离性

A教室和B教室中间有一道墙，这道墙很厚，也可以很薄。这就是事务的隔离级别。

这道墙越厚，表示隔离级别就越高。

## 事务和事务之间的隔离级别有哪些？4个级别

1.读未提交：read uncommitted（最低的隔离级别）（没有提交就读到了）

什么是读未提交？

事务A可以读取到事务B未提交的数据。

这种隔离级别存在的问题就是：脏读现象（dirty read），我们称读到了脏数据。

这种隔离级别一般都是理论上的，大多数的数据库隔离级别都是二档起步！

2.读已提交：read committed（提交之后才能读到 ）

什么是读已提交？

事务A只能读取到事务B提交之后的数据。

这种隔离级别解决了什么问题？

解决了读脏的现象

这种隔离级别存在什么问题？

不可重复读取数据。

什么是不可重复读取数据？

在事务开启之后，第一次读到3条，当前事务还没结束，可能第二次再读取的时候，读到的数据是4条，3不等于4，称为不可重复读取。

这种隔离级别是比较真实的数据，每一次读到的数据是绝对的真实。

3.可重复读：repeatable read（提交之后也都不到，永远读取的都是刚开启十五时的数据）

什么是可重复读取？

事务A开启之后，不管过多久，每一次在事务A中读到的数据都是一致的，即使事务B将数据已经修改，并且提交了，事务A读到的数据还是没有发生改变，这就是可重复读。

可重复读可以解决什么问题？

解决了不可重复读取数据。

可重复读存在的问题是什么？

可能会出现幻影读。

每一次读到的数据都是幻想，不够真实！

MySQL中默认的事务隔离级别就是这个！！

4.序列化/串行化：serializable（最高隔离级别）

只是最高隔离级别，效率最低。解决了所有的问题。

这种隔离级别表示事务排队，不能并发

每一次读到的数据都是最真实的，并且效率是最低的

例如：如果用到该级别。银行清理账务时，别人都不能存取钱。

验证各种隔离级别：

## 查看隔离级别：

select @@transaction\_isolation;

## 设置隔离级别：

set global transaction isolation level 隔离级别;

## 演示各个隔离级别

被测试的表：t\_user

### 验证：read uncommitted

mysql> set global transaction isolation level read uncommitted;

事务A 事务B

Use sys；

Start transaction;

Use sys;

Start transaction;

Insert into t\_user(name) values(‘zhangsan’);

Select \* from t\_user;

### 验证：read committed

mysql> set global transaction isolation level read committed;

事务A 事务B

Use sys;

Start transaction;

Use sys;

Start transaction;

Insert into t\_user(name) values(‘zhangsan’)

Select \* from t\_user;#表中没有数据

Commit;

Select \* from t\_user;#表中显示插入数据

### 验证：repeatable read

mysql> set global transaction isolation level repeatable read;

事务A 事务B

use sys;

use sys;

start transaction;

start transaction;

select \* from t\_user;

insert into t\_user values('lisi');

insert into t\_user values('wangwu');

commit;

select \* from t\_user;

、

### 验证：serializable

mysql> set global transaction isolation level serializable;

事务A 事务B

use sys;

use sys;

start transaction;

start transaction;

select \* from t\_user;

insert into t\_user values('abc');

select \* from t\_user;

commit；

事务A提交后出现结果

# 索引

## 什么是索引？有什么用？

索引就相当于一本书的目录，通过目录可以快速地找到对应的资源。

在数据库方面，查询一张表的时候有两种检索方式：

第一种方式：全表扫描

第二种：根据索引检索（效率高）

索引为什么可以提高检索效率呢？

其实最根本的原理是缩小了扫描范围。

索引虽然可以提高检索效率，但不能随意添加索引，因为索引也是数据库中的对象，也需要数据库不断的维护。是有维护成本的。比如，表中的数据经常被修改这样就不适合添加索引，因为一旦被修改，索引需要重新排序，进行维护。

添加索引是给某个字段，或者说某些字段添加索引

Select ename , sal from emp where ename = ‘SMITH’;

当字段上没有添加索引时，以上sql语句会进行全表扫描，扫描enamel字段中所有的值。

当字段上添加索引时，以上sql语句会根据索引扫描，快速定位。

## 怎么创建索引对象？怎么删除索引对象？

创建索引对象

Create index 索引名称 on 表名（字段名）;

删除索引对象

Drop index 索引名称 on 表名；

## 什么时考虑给字段添加索引？（满足什么条件）

1. 数据庞大。（根据客户的需求，根据线上环境）
2. 该字段很少DML操作。（因为字段进行修改操作，索引也需要进行维护）
3. 该字段经常出现在where子句中。（经常根据哪个字段查询）

注意：主键和具有unique约束的字段自动会添加索引。

根据主键查询效率高，尽量根据主键检索

## 查看sql语句的执行计划explain

语法：explain sql语句；

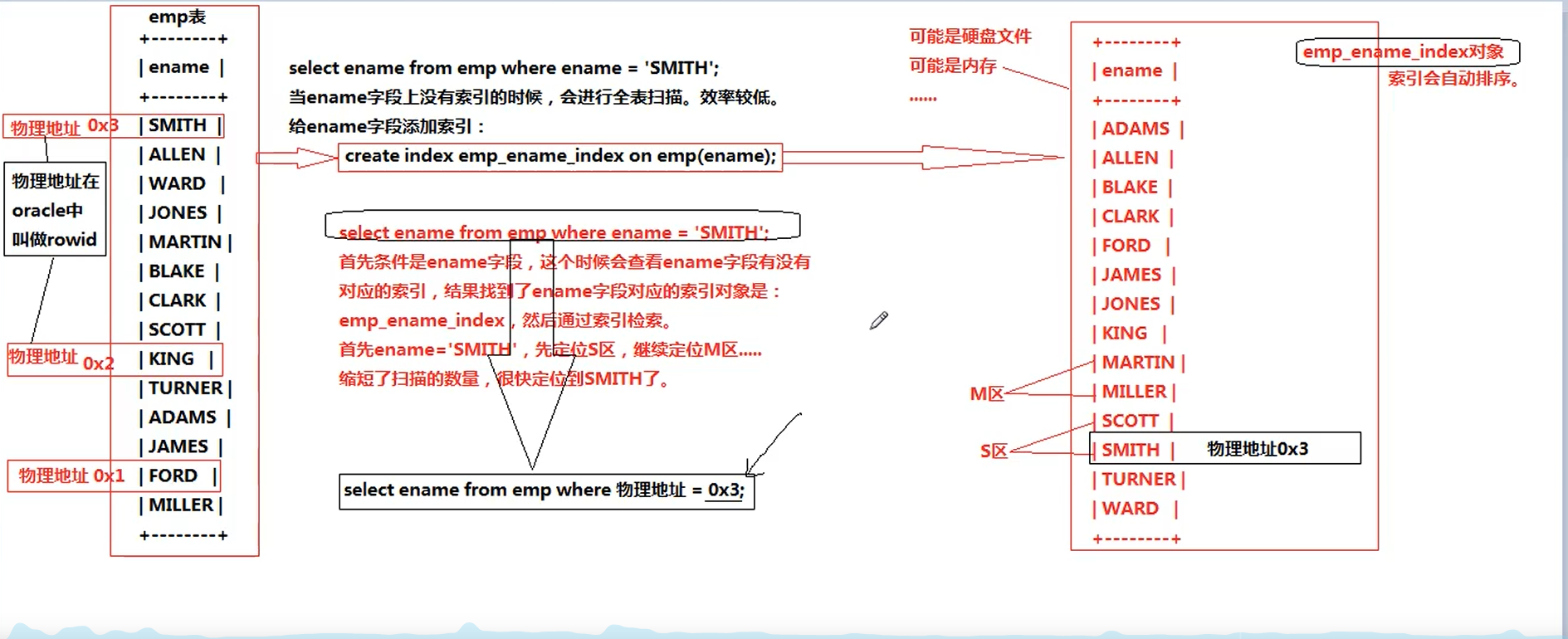
explain select ename , sal from emp where sal = 5000;

给薪资sal字段添加索引：

Create index emp\_sal\_index on emp(sal);

explain select ename , sal from emp where sal = 5000;

## 索引实现原理



## 索引的分类

单一索引：给单个字段添加索引

复合索引：给多个字段联合起来添加一个索引

主键索引：主键上会自动添加索引

唯一索引：有unique约束的字段上会自动添加索引

…

## 索引什么时候失效？

模糊查询的时候，第一个通配符使用的是%，这时候索引失效

# 视图（view）

## 什么是视图？

站在不同角度去看到数据。（同一张表的数据，通过不同的角度去看待）

## 怎么创建试图？怎么删除视图？

Create view 视图名 as select语句

Drop view 视图名

注意：只有DQL语句才能以视图对象的方式创建出来。

对试图进行增删改查，会影响到原表数据。（通过视图影响原表数据的，不是直接操作的操作）

## 创建视图步骤

1. create table emp1 as select \* from emp;//复制表
2. create view eye as select \* from emp1;//创建视图
3. select \* from eye;//操作试图
4. update eye set ename = 'hehe' , sal = 1 where empno = 7369;
5. select \* from eye;
6. select \* from emp1;

结论：操作eye视图，emp1也发生改变

## 视图的作用

视图可以隐藏表的实现细节，保密级别较高的系统，数据库之对外提供相关的试图，只对视图操作。

# 数据库数据的导入导出

## 将数据库当中的数据导出

在windows的dos命令窗口中执行：（导出整个库）

Mysqldump 数据库名>路径\文件名.sql -mysql用户名root -密码

在windows的dos命令窗口中执行：（导出指定数据库中的指定表）

Mysqldump 数据库名 表名>路径\文件名.sql -mysql用户名root -密码

## 导入数据

Create databases 数据库名；

Use 数据库名；

Source 路径\文件名.sql

# 数据库设计三范式

## 什么是数据库设计范式？

数据库表的设计依据。教你怎么进行数据库表的设计。

## 数据库设计范式共有？

3个。

第一范式：要求任何一张表必须有主键，每一个字段原子性不可再分。

第二范式：建立在第一范式的基础之上，要求所有非主键字段完全依赖主键，

不要产生部分依赖。

第三范式：建立在第二范式的基础之上，要求所有非主键字段直接依赖主键，

不要产生传递依赖。

声明：三范式是面试官经常问的，所以一定要熟记在心！

设计数据库表的时候，按照以上的范式进行，可以避免表中数据的冗余，空间的浪费。

## 第一范式

最核心，最重要的范式，所有表的设计都需要满足。

必须有主键，并且每一个字段都是原子性不可再分。

学生编号 学生姓名 联系方式

------------------------------------------

1001 张三 zs@gmail.com,1359999999

1002 李四 ls@gmail.com,13699999999

1001 王五 ww@163.net,13488888888

以上是学生表，满足第一范式吗？

不满足，第一：没有主键。第二：联系方式可以分为邮箱地址和电话

学生编号(pk) 学生姓名 邮箱地址 联系电话

----------------------------------------------------

1001 张三 zs@gmail.com 1359999999

1002 李四 ls@gmail.com 13699999999

1003 王五 ww@163.net 13488888888

## 第二范式

建立在第一范式的基础之上，

要求所有非主键字段必须完全依赖主键，不要产生部分依赖。

学生编号 学生姓名 教师编号 教师姓名

----------------------------------------------------

1001 张三 001 王老师

1002 李四 002 赵老师

1003 王五 001 王老师

1001 张三 002 赵老师

这张表描述了学生和老师的关系：（1个学生可能有多个老师，1个老师有多个学生）

这是非常典型的：多对多关系！

分析以上的表是否满足第一范式？

不满足第一范式。

怎么满足第一范式呢？修改

学生编号+教师编号(pk) 学生姓名 教师姓名

----------------------------------------------------

1001 001 张三 王老师

1002 002 李四 赵老师

1003 001 王五 王老师

1001 002 张三 赵老师

学生编号 教师编号，两个字段联合做主键，复合主键（PK: 学生编号+教师编号）

经过修改之后，以上的表满足了第一范式。但是满足第二范式吗？

不满足，“张三”依赖1001，“王老师”依赖001，显然产生了部分依赖。

产生部分依赖有什么缺点？

数据冗余了。空间浪费了。“张三”重复了，“王老师”重复了。

为了让以上的表满足第二范式，你需要这样设计：

使用三张表来表示多对多的关系！！！！

学生表

学生编号(pk) 学生名字

------------------------------------

1001 张三

1002 李四

1003 王五

教师表

教师编号(pk) 教师姓名

--------------------------------------

001 王老师

002 赵老师

学生教师关系表

id(pk) 学生编号(fk) 教师编号(fk)

------------------------------------------------------

1 1001 001

2 1002 002

3 1003 001

4 1001 002

背口诀：

多对多怎么设计？

多对多，三张表，关系表两个外键！！！！！！！！！！！！！！！

## 第三范式

第三范式建立在第二范式的基础之上

要求所有非主键字典必须直接依赖主键，不要产生传递依赖。

学生编号（PK） 学生姓名 班级编号 班级名称

---------------------------------------------------------

1001 张三 01 一年一班

1002 李四 02 一年二班

1003 王五 03 一年三班

1004 赵六 03 一年三班

以上表的设计是描述：班级和学生的关系。很显然是1对多关系！

一个教室中有多个学生。

分析以上表是否满足第一范式？

满足第一范式，有主键。

分析以上表是否满足第二范式？

满足第二范式，因为主键不是复合主键，没有产生部分依赖。主键是单一主键。

分析以上表是否满足第三范式？

第三范式要求：不要产生传递依赖！

一年一班依赖01，01依赖1001，产生了传递依赖。

不符合第三范式的要求。产生了数据的冗余。

那么应该怎么设计一对多呢？

班级表：一

班级编号(pk) 班级名称

----------------------------------------

01 一年一班

02 一年二班

03 一年三班

学生表：多

学生编号（PK） 学生姓名 班级编号(fk)

-------------------------------------------

1001 张三 01

1002 李四 02

1003 王五 03

1004 赵六 03

背口诀：

一对多，两张表，多的表加外键！！！！！！！！！！！！

## 总结表的设计？

一对多：

一对多，两张表，多的表加外键！！！！！！！！！！！！

多对多：

多对多，三张表，关系表两个外键！！！！！！！！！！！！！！！

一对一：

一对一放到一张表中不就行了吗？为啥还要拆分表？

在实际的开发中，可能存在一张表字段太多，太庞大。这个时候要拆分表。

一对一怎么设计？

没有拆分表之前：一张表

t\_user

id login\_name login\_pwd real\_name email address........

---------------------------------------------------------------------------

1 zhangsan 123 张三 zhangsan@xxx

2 lisi 123 李四 lisi@xxx

...

这种庞大的表建议拆分为两张：

t\_login 登录信息表

id(pk) login\_name login\_pwd

---------------------------------

1 zhangsan 123

2 lisi 123

t\_user 用户详细信息表

id(pk) real\_name email address........ login\_id(fk+unique)

-----------------------------------------------------------------------------------------

100 张三 zhangsan@xxx 1

200 李四 lisi@xxx 2

口诀：一对一，外键唯一！！！！！！！！！！

## 嘱咐一句话

数据库设计三范式是理论上的。

实践和理论有的时候有偏差。

最终的目的都是为了满足客户的需求，有的时候会拿冗余换执行速度。

因为在sql当中，表和表之间连接次数越多，效率越低。（笛卡尔积）

有的时候可能会存在冗余，但是为了减少表的连接次数，这样做也是合理的，

并且对于开发人员来说，sql语句的编写难度也会降低。

面试的时候把这句话说上：他就不会认为你是初级程序员了！

# 一对一设计

即设计在一张表中太过冗长，进而设计到两张表中

1.一对一设计有两种方案：主键共享

T\_user\_logon 用户登录表

Id(pk) username password

1 zs 123

2 ls 456

T\_user\_detail 用户详细信息表

Id（pk + fk） realname tel

1 张三 111111111111

2 李四 222222222222

2. 一对一设计有两种方案：外键唯一

T\_user\_logon 用户登录表

Id(pk) username password

1 zs 123

2 ls 456

T\_user\_detail 用户详细信息表

Id（pk） realname tel userid(pk+unique)

1 张三 111111111111 2

2 李四 222222222222 1