MySQL服务器里面有很多数据库，每个数据库里面又有很多张表。

**1.基本语句**

数据库语句一定要以分号;结尾

1. 查看当前服务器下有几个数据库：show databases；

2. 创建数据库： create database 数据库名；

3. 使用某个数据库：use 数据库名； (数据库名字不可以修改,表名可以修改)

4. 查询数据库中有什么表：show tables；

5. 删除某个数据库：drop 数据库名；

6. 查询表中列的信息，查看表描述：desc 表名；（desc就是description缩写）

7. 声明数据库字符集: set names gbk; （声明为gbk字符集）

**2.表操作**

1.创建表

create table表名(

    列名1 类型,

    列名1 类型

)；

create table person\_info
(
person\_id smallint(**5**) unsigned auto\_increment,
name varchar(**50**) not null comment 'person\_name',
country varchar(**60**) default 'china',
salary decimal(**10**,**2**) default **0.00** comment 'salary',
primary key (person\_id)
)engine=innodb default charset=utf8;

多个列名之间用逗号隔开，最后一个不需要逗号

2.修改表的名字

rename table oldName to newName；

3.删除表

drop table 表名；

4. 修改列

（1）增加列

alter table `user\_movement\_log` **add column** GatewayId int not null default 0 AFTER `Regionid` (在哪个字段后面添加)

（2）删除列

alter table `user\_movement\_log` **drop column** Gatewayid

（1）修改列类型等属性

ALTER TABLE student **MODIFY COLUMN** sname2 INT NOT NULL COMMENT 'newname123';

（2）修改列名,

ALTER TABLE student **CHANGE COLUMN NAME1 sname2** VARCHAR(200) COMMENT 'newname';

**3.增删改查语句**

1.1增加

（1）插入一行

insert into 表名(列名1，列名2，列名3)

values（12，'aaa','bbb'）;  /\*当插入所有列的时候，可以省略列名列举\*/

（2）连续插入多个行

insert into 表名(列名1，列名2，列名3)

values (....),(....),(.....);

1.2修改

update 表名

set 列名1 = 新值1, 列明2=新值2

where 列名1=某个值；

1.3删除

delete from 表名

where 列名=值；(删除条件)

注：delete语句中不能用数据库别名

1.4查询

select 列名,\* from 表名 where 条件；

**4.整型列**

(1)

整型范围从小到大又分为: Tinyint, Smallint, Mediumint, int, Bigint

SMALLINT 2字节, MEDIUMINT 3字节， INT 4字节 ，BIGINT 8字节

关于范围：

TINYINT 1字节,8位。无符号范围：0到2^8-1, 有符号：-2^7到2^7-1

INT 4字节，32位。 无符号范围： 0到2^32-1，有符号：-2^31到2^31-1

(2)

整型列属性： tinyint(M) unsigned **zerofill**

Unsigned: 表示该列为无符号整型,只能为正数，且正数范围会比有符号时候更大。 默认情况下整型都是有符号的，存储范围都是包含负数的。

M: 代表宽度，但在zerofill时才有意义。并不是限制的存储整型数的位数。更不会影响存储范围。

Zerofill:用零填充（如果某列是zerofill，则默认就是unsigned，无需再声明unsigned）。M宽度代表用零填充后的位数。如果M是5，则插入9会被填充为00009.

Not null代表不能为空，一般与default设置默认值一起使用。

alter table class add age unsigned not null deafult 0;加默认值的意义就是避免null值。

**5.浮点型与定点型**

float(M,D) M代表总位数(不包括小数点和正负号)  D代表小数点后位数

float(6,2)最大9999.99 最小-9999.99,当插入3位小数的浮点数以后，录入数据库时候会舍入为2位小数。 在Oracle里面是不能指定小数点后位数的。

decimal(M,D) 定点型更精确些。对应正好达到最大范围的数据，decimal比float更精确。

浮点和定点型也可也加unsigned，此时float(4,2)从0.00-99.99

**6.字符型**

utf-8:中文汉字一个字符占3字节。英文字符一个字符占1字节。

char(8):定长，8代表宽度，即可容纳8个字符，不是8个字节.

        对于定长，如果没有存到8个字符，在数据库里面存的仍是8个字符。char数据在数据库内部存的时候会在尾部用空格补齐，但在取出时，尾部的多余空格会被去掉。所以，定长当要保存尾部有空格的字符串时，取出时尾部空格会被去掉。

varchar(10):变长，10代表宽度，即可容纳10个字符.

        如果没有存到10个字符，那么存几个字符在数据库里面就占几个字符再加上1到2个字节用于存储varchar所存字符的信息。

\* varchar利用率永远小于100%，而char可能到100%。

\* 速度上定长char更快些

选择原则：当字符串宽度确定时，应该选择char

注册用户名一般考虑用char，因为浪费空间不大且速度更快。空间换时间。

而微博内容适合用varchar。

text：文本类型。2w字以上才会被用到。但缺点是不能全文索引，且不能设置default默认值。一般也不会选用。

**7.日期时间类型**

\* 存储时候用字符串即可, 除了整型和浮点型其他都应该用单引号引起来

Year类型: 用1个字节 范围：1901-2155年, 当输入不合法的year数据时，可能会被转为0000年。

如果输入2位数，‘00-69’对应 2000-2069  '70-99'对应1970-1999,不过一般都输入4位数。

Date类型: YYYY-MM-DD 范围1000年-2999年  mysql还可以自动识别无效的日期。

Time类型：hh:mm:ss  "-839:59:59" ~ "839:59:59"

DateTime类型：YYYY-MM-DD hh:mm:ss

时间开发中一般不用DateTime类型来表示时间，而是用时间戳。用整型数据来存储距 1970-01-01 00:00:00的描述。这样可以方便的算时间点之间的差距。用int来存储。

**8.Enum和Set**

enum:枚举类型 gender enum('男','女') 对于enum类型的列，只能插入枚举出来的值得数据,否则报错。 enum有悖于关系数据库设计原理，因为元素不是原子的，不是不可以再分的。

set: 是可以列举出更多的可能数据。但也不符合设计科设计原理。

建表案例：

create table student(

    id   int unsigned primary key auto\_increment,

    name char(3) not null default '',

    age  tinyint unsigned not null default 0,

   email varchar(30) not null default '',

  salary decinal(7,2) not null default 1300

)charset utf-8;

**9.增删改语句**

(1)insert into 表名(列名1，列名2，....列名3)

values（....）;

当列名不写时，默认插入所有列

id的自增长默认情况是在不给id指定值的时候才会自增长。

当需要一次性插入多列时：

insert into 表名(列名1，列名2，列名3)

values (....),(....),(.....);

（2）update 表名 set

     列1 = 值1，

     列2 = 值2

     where ...    <update语句一定要注意不能少where语句，不然就全改了>

**where 前面的最后一个set值语句末尾千万不要加符号**

（3）delete from 表名

      where ... <如果不加where则指定表中所有行都会被删除>

**10.GroupBy语句（一般group by都要和统计函数一起使用 min，max，sum，avg，count ）**

（1）select order\_id, max(final\_fee) from order\_info 并不是选出最大订单金额的订单号及金额。 对应order\_id列，被选出来的只是表中的第一行的order\_id，而查询结果的第二列为最大的final\_fee,只不过不与订单号对应。

（2）在groupby语句中，

select order\_id, res\_name, max(final\_fee) from order\_info group by res\_name; 会将每家餐馆最大金额的订单找出。这里可以直接用res\_name是因为group分组后，组中行的res\_name列值相同。但是对于order\_id，还是取得分组后第一行元素的值，所以order\_id没有意义。

group by就相当于按某个属性将行分隔成很多组，并对这些组单独做操作。例如求最大值，平均值等。group by 如果不跟统计函数直接select就是分组后的第一行元素。

要把列名当做是变量来看。可以对这些列名进行操作。

select goods\_id,goods\_name,good\_price-market\_price from goods;

给列取别名,用as

select goods\_id,good\_price-market\_price as AA from goods;

**11.Having语句**

用于对查询结果再进行查询。where仅是对于数据库中原始表进行查询。having是对查询结果的再次筛选。

（1）筛选第三类产品中高于市场价格200以上的商品

select cat\_id, sum(good\_price-market\_price) as AA from goods

where cat\_id=3

having AA>200;

此处不能直接在where中使用sum（）来筛选，因为where中不可以使用聚合函数。而实际上在having中是可以使用的。

（2）select name,avg(score)

from stu

group by name

having count(score<60)>2;

count()语句里面不能用条件判断语句。并不是count分数小于60的科目数。

Mysql因为没有boolean值，所以对于判断语句会返回0或者1的效果。

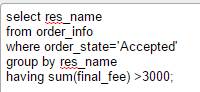
select name,score<60 from stu; 对于第二列值将会是0或者1。

ip_image002.jpeg

通过sum求除了分数小于60的科目总数，其实就是把所有行在挂科这列的数相加，巧用了sql里面的布尔值。

group by语句里面select的对应列很容易错误。因为对应不是聚合函数作用的列，默认就是取第一行的列。

**Having中可以使用聚合函数，而where中则不可以使用。**



**备注：**

having一般用于对group by后的分组的筛选。或者是当想在where中使用聚合函数时，可以通过having对select的聚合函数列再进行筛选，如：

筛选第三类产品中高于市场价格200以上的商品

select cat\_id, sum(good\_price-market\_price) as AA from goods

where cat\_id=3

having AA>200;

**12.Order by语句**

order by 列1 asc，列2 desc;

asc:升序 desc：降序

order by语句排序是放在having和group by，where 以后。

五种字句顺序： where， group by， having， order by， limit

默认是desc降序。

**Limit用法：**

（1）limit 3 取前三行记录

（2）limit 1,5

说明：1是偏移量，5是返回的记录行的最大数目,注：起始偏移量是0。

所以 limit 1,5 取的是查询结果中第2条记录至第6条记录。

（3）limit 96，-1 是取第97条记录到最后一条

归纳：limit n 等价于 limit 0 n

limit 10 返回的是第1到第9条数据。 而 limit 1,9 是返回第2条到第10条。

**13.Where子查询 ：使用 = 或者 in 后面接 （新的select查询语句）**

where中列的查询条件是通过另一个select查询的结果来指定的。

查询每个商家消费额度最大的订单：

select order\_id, cus\_name, max(final\_fee) from order\_info;是错的，因为order\_id,cus\_name不与max(final\_fee)对应，只是第一行的order\_id和cus\_name。

然而用

select order\_id, cus\_name, final\_fee from order\_info order by final\_fee desc; 可以实现，只是无效数据会很多，需要再次筛选。

最好的方法是子查询。

where子查询：将一次查询的结果作为另一个查询的条件。放在括号里面。

select order\_id, cus\_name, final\_fee from order\_info where final\_fee=(select max(final\_fee) from order\_info);

当select的结果有很多时，where 条件里面用in。例如查,每个商家的第一张订单信息。

select order\_id, cus\_name, final\_fee from order\_info where order\_id in (select min(order\_id) from order\_info group by res\_name);

**14.From子查询**

将内层查询结果作为外层查询的临时表

From子查询必须对内部查询出的临时表用as取别名

**14.Exists子查询**

把外层查询结果，拿到内层，看内层是否成立

查有商品的栏目：

select cat\_id,cat\_name from category where exists (select \* from goods where goods.cat\_id = category.cat\_id); 其实就是把外层查询结果拿到内层来做验证刷选看是否存在。

**15.Union 把2次或多次查询结果合并在一起**

使用union的要求：两次查询的列数需要一致。一般建议列类型也要对应。

然而union的两张表的结构不需要一致，且表不需要是同一张。

select user\_name,user\_email,msg\_content from feedback where msg\_state=1 union

select name,email,content from comments where state=1;

虽然两次select查询的列名不同，但列名会取第一个select查询的列名为准。

面试题：

ta，tb两张表中都只有2列，id，num ，要求把ta，tb合并到一张表并对ta，tb表中重复的id的num求和。 即当ta中有（'a',10）tb中有（'a',5）时，合并表中为('a',15)；

select id,sum(num) from (select id, num from ta union select id, num from tb) as tmp group by id; 这里用到了union和from子查询。from子查询对于临时表需要用as来取别名。即使id是主键也可以union

如果不同的语句中取出的行有完全相同的，即每个列的值都相同。此时相同的行将会合并成一行。所以上面那到面试题还是有缺陷，当ta，tb表中的某行完全相同，则不会被叠加。 但是可以通过在union 后面加上all来避免去重复。

select id,sum(num) from (select id, num from ta union all select id, num from tb) as tmp group by id;

如果子句中有order by或者limit则需要在union两边的select上加上小括号即可 （select...）union (select... ) 并在最终合并后的结果上运用order by排序。 除非在union的子select语句中同时运用order by和limit。

**16.左连接，右连接，内连接**

（1）左右连接

select goods\_id, goods.cat\_id, cat\_name, goods\_name

from

goods left join category

on goods.cat\_id = category.cat\_id;

此时goods表和category表被连接起来了，是左连接。

左连接：

select 列1，列2...列n from

tableA left join tableB

on tableA.列 (关系) tableB.列;

左连接以后形成一张大表，这就是一张普通的表，后面可以照常写 where，group by, having 子句

右连接语法上就是把 left join换成right join

内连接语法上是把left join 换成 inner join

左连接：是以左表为准，去右表找匹配，找不到的话就右表属性以null来匹配。

右连接：是以右表为准，去左表找匹配，找不到的话就左表属性以null来匹配。

A left join B 就相当于 B right join A， 左右连接可以相互转化，并建议使用左连接来代替右连接。

（2）内连接

内连接：查询左右连接后左右中都有数据的行，即不要左右中null的那一部分。

内连接是左右连接的交集。

外连接就是求并集，只不过mysql不支持。

当需要链接多张表的时候，可以多次left join ... on

如：

select goods\_id, goods.cat\_id, cat\_name, goods.brand\_id, brand\_name

from

goods left join category

on goods.cat\_id = category.cat\_id

left join brand

on goods.brand\_id = brand.brand\_id;

就是把第一次左连接的结果再作为一张吧再进行一次左连接。

例题2：



select t1.teamName as hname, matchResult, t2.teamName as gname, matchTime

from match left join team as t1

on match.hostTeamID = t1.teamID

left join team as t2

on match.guestID = t2.teamID

where matchTime between '2014-01-01' and '2014-01-31';

多表连接查询时，如果要重复查同一张表两次，可以将这张表起别名就可以了。

**17.表增加，修改，删除列**

(1)增加列

alter table表名 add 新列名 列类型 其他信息（not null, unsigned, default等）;

\* 增加的列默认是在表的最后一列。但是可以通过after来声明新增的列在哪列后面。

alter table表名 add 列声明 after 现有列名;

\* 增加的列如果想放在第一列，可以使用first

alter table表名 add 列声明 first;

(2)修改列

alter table表名 change 待修改列名 列声明;

alter table boy change height height smallint not null default 180;

列名要写2次，一次是指定要修改的列，一次是定义新的列声明。

(3)仅修改列属性

ALTER TABLE student MODIFY COLUMN sname2 INT NOT NULL COMMENT 'newname123';

(4)删除列

alter table表名 drop 列名;

**18.视图**

如果某个查询结果出现的非常频繁，也就是拿这个结果进行子查询非常的频繁。

视图：是由查询结果形成的一张虚拟表

（1）视图创建语法：create view 视图名 as select语句;这样以后，一张视图就会被创建在数据库中，就当表一样看。show tables可以看到表和视图。

并可以直接对视图进行操作，例如select from 视图名查询。视图中的数据是根据创建视图的select语句动态生成的,当表的数据改变了以后视图中的数据也会修改。

（2）删除视图语法：drop view 视图名;

视图的作用：

a. 方便子查询

b. 权限管理：通过提取表中部分列信息来创建view，从而隐藏重要的列数据。

c. 大数据分表:例如当表中行数超过200万行时，可以通过把一张表的数据拆成4张来存。 通过Id%4来实现。

 当要查询这4张散开的表的时候，可以把他们union起来然后放在视图里面，之后查询view即可。

（3）视图的修改

Alter view as select .... 相当于新建了一个视图来代替之前的了。

（4）视图与表的关系

视图是表的查询结果，表的数据改变了，视图中的结果也会受到影响。

视图的增删改也会影响到表，但视图并不总是可以增删改的。当视图中数据与表中数据一一对应时候才可以修改。

对应视图insert时，视图中必须包含表中没有默认的列。否则表中某些列不知道应该insert什么值。

**19.触发器 Trigger**

（1）

触发器作用：监视某种情况并触发某种操作。

当对order表增加一条订单后，将goods表中库存减少。此时监视的是order表的insert操作，触发的是goods表的update操作。

触发器能监视：增删改， 能触发的操作也是增删改

（2）触发器语法

create trigger triggerName

after/before insert/update/delete on 表名

For each row

Begin

    触发的Sql语句，一句或者多句，但范围只能是insert，update，delete

    中的。

End$

对于MySQL  For each row这句是固定的，但在Oracle中可以是其他的，不局限于 row.

对于触发器而言，需要先改变数据库结束标识符，delinirer $

这样语句结束的标志不是分号;而是美元符$

如何在触发器引用行的值？

对于insert而言，新增的行是new，行中的列 new.列名

（1）添加订单库存减少

create trigger t2

after insert on order

for each row

begin

    update goods set num = num-new.amount, id = new.gid;

end$

(2)删除一个订单时，库存增加

old代表旧行，就是之前有现在没有的，用old来引用

create trigger t3

after delete on order

for each row

begin

    update goods set num=num+old.amount, id = old.gid;

end$

（3）修改订单时，库存相应改变

old代表修改前的值，new代表被修改后的值

create trigger t4

after update on order

for each row

begin

    update goods set num=num+ old.amount ,id = old.id;

    update goods set num=num- new.amount ,id = new.id;

end$

(3)触发器里面after和before的区别

after：是先完成数据的增删改，然后再触发。所以触发中的语句晚于增删改，所以我们无法影响前面的增删改。

before：是先完成触发再增删改。触发的语句先于监视的增删改发生，所以我们有机会审查判断修改即将发生的操作。

案例：

对所下订单，如果订单数量大于5，就认为是恶意订单，并强制将所订商品数量改为5。

create trigger t3

before insert into o

for each row

begin

    if new.amount>5 then

       set new.amount = 5;

    end if;

    update goods set num=num+ old.amount ,id = old.id;

    update goods set num=num- new.amount ,id = new.id;

end$

如何查看触发器

show triggers;

**20.事务**

数据库中对于同样的数据，有着不同的存储方式和管理方式，在MySQL中，这个称为存储引擎。

常用表的引擎: Myisam, InnoDB



注：InnoDB以及也支持全文索引了

事务：ACID

（1）原子性（Atomicity）：一组操作，要么都成功执行，要么不执行。

（2）一致性(Consistency)：事务发生前后，数据的总额依然匹配。

（3）隔离性(Isolation)：一组操作里面，在其他操作没有完毕之前，中间的改变过程不可见。

（4）持久性(Durability)：事务产生的影响不能够被撤销MySQL服务器里面有很多数据库，每个数据库里面又有很多张表。