2021年7月28日 9:57

MySQL

参考: https://www.bilibili.com/video/BV12b411K7Zu?from=search&seid=5468517210623443154

1.相关概念

（1）DB：数据库，保存一组有组织的数据的容器

（2）DBMS：数据库管理系统，数据库软件或产品

（3）SQL：结构化查询语言，用于和DBMS通讯的语言

几乎所有DBMS都支持SQL语言

2.数据库特点

3.DBMS分类

（1）基于共享文件系统的DBMS（Access）

（2）基于客户机——服务器的DBMS（MySQL、Oracle、SqlServer）

4.MySQL服务的启停

（1）net start mysql

（2）net stop mysql

5.服务端的登陆和退出

（1）登陆：mysql -h localhost(主机名) -P 3306(端口号) -u root(用户名) -p(密码)

如果访问本机，-h和-P可省略

（2）退出：exit/quit/ctrl+C

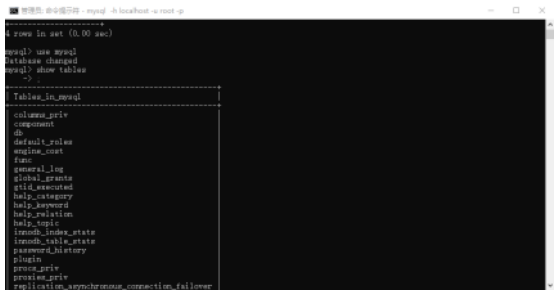
6.常见命令

（1）show databases —— 查看数据库

（2）use 库名 —— 进入数据库

use mysql(库名)

show tables



（3）show tables from mysql —— 查看某个数据库中的表

（4）select database() —— 查看所在的库

（5）create table 表名(

-> id int,

-> name varchar(20)) —— 建表

（6）desc 表名 —— 查看表的结构

（7）select * from 表名 —— 查看表中有哪些数据

（8）insert into 表名 value(1, 'john') —— 表中插入数据

（9）update 表名 set name='lilei' where id=1 —— 改表中数据

（10）delete from 表名 where id=1 —— 删除表中数据

7.查看服务端版本

（1）进入MySQL服务端输入select version()

（2）在cmd中输入mysql --version

8.语法规范

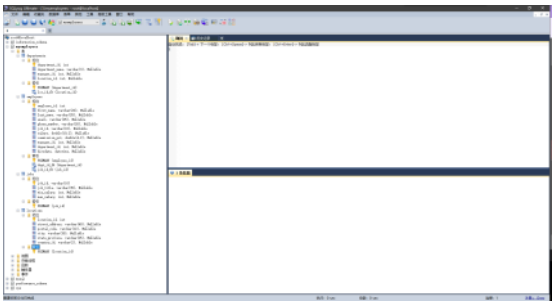
（1）不区分大小写，但建议关键字大写，表名列名小写

（2）每条命令建议用；结尾。\g也可以

（3）每条命令根据需要，可以进行缩进或换行

（4）注释：单行注释：#注释文字或者-- 注释文字；多行注释：/*注释文字*/

9.SQL语言



9.1查询（DQL语言）

9.1.1基础查询

select 查询列表 from 表名 #查询列表可以是：表中的字段，常量，表达式，函数；查询的结果是一个虚拟的表格

（1）查询表中的单个字段

select last_name from 表名

（2）查询表中的多个字段

select last_name, salary, email from 表名

（3）查询表中的所有字段

select * from 表名 #顺序与原始表一样

（4）查询常量值

select 100

select 'john' #统一用单引号

（5）查询表达式

select 100%98

（6）查询函数

select VERSION()

（7）起别名

方式一：使用as
select 100%98 as 结果
select last_name as 姓，first_name as 名 from 表名
作用：便于理解；如果要查询的字段有重名的情况，使用别名区分
方式二：使用空格
select last_name 姓，first_name 名 from 表名
（8）去重
select distinct 要查询的事物 from 表名
（9）+号作用
mysql中的+号只有一个功能：运算符
select 100+90
select '123'+90; #其中一方为字符型，试图将字符型数值转换为数值型。如果转换成功，则继续做加法运算；否则将字符型数值转换为0
select null+10 #只要其中一方为null，结果为null
（10）拼接
select concat('a', 'b', 'c') as 结果

9.1.2条件查询

select 查询列表 from 表名 where 筛选条件
方式一：按条件表达式筛选
条件运算符：> < = != <> >= <=
方式二：按逻辑表达式筛选
逻辑运算符：&& || ！ and or not
方式三：模糊查询
like，between and，in，is null
示例1：查询员工名中第三个字符为e，第五个字符为a的员工名和工资
select last_name, salary from employees where last_name like '__n_l%'
示例2：查询员工名中第二个字符为_的员工
select last_name from employees where last_name like '__%' #需要用到转义字符
或者 select last_name from employees where last_name like '_$_%' escape '$'
示例3：查询没有奖金的员工名和奖金率
select last_name, commission_pct from employees where commission_pct is null
或者 selete last_name, commission_pct from employees where commission_pct <=> null #安全等于<=>
示例4：查询员工号为176的员工的姓名，部门号和年薪
select last_name, department_id, salary*12*(1+ifnull(commission_pct, 0)) as 年薪 from employees

9.1.3排序查询

语法：
select 查询列表
from 表
where 筛选条件
order by 排序列表
示例1：查询员工信息，要求工资从高到低排序
select * from employees order by salary desc
示例2：按年薪的高低显示员工的信息和年薪（按表达式排序）

select *, salary*12*(1+ifnull(commission_pct, 0)) as 年薪 from employees order by salary*12*(1+ifnull(commission_pct, 0)) desc

示例3：按年薪的高低显示员工的信息和年薪（按别名排序）

select *, salary*12*(1+ifnull(commission_pct, 0)) as 年薪 from employees order by 年薪 desc

示例4：按姓名长度显示员工的姓名和工资（按函数排序）

select length(last_name) 字节长度, last_name, salary from employees order by length(last_name) desc

示例5：查询员工信息，要求先按工资排序，再按员工编号排序（按多个字段排序）

select * from employees order by salary asc, employee_id desc

9.2常见函数

分类：

单行函数，如concat，length，ifnull

分组函数，功能：做统计使用，又称为统计函数，聚合函数，组函数

9.2.1单行函数

（1）字符函数

length：

select length('john')

select length('我是john')

concat：

select concat(last_namem, '_', first_name) from employees

upper、lower：

select upper('john')

select lower('john')

substr、substring：

select substr('李莫愁爱上了陆展元', 6) out_put #注意：索引从1

select substr('李莫愁爱上了陆展元', 1, 3) out_put #截取从指定索引处的指定字符长度的字符

instr：

select instr('杨不悔爱上了殷六侠', '殷六侠') as out_put #返回子串第一次出现的索引，如果没有，返回0

trim：

select length(trim('　　张翠山　　')) as out_put #去前后空格

select trim('a' from 'aaaaaaaaa张aaaaaaa翠山aaaaaaaaaaaaaaaaaaaa') as out_put

lpad、rpad：

select lpad('殷素素', 10, '*') as out_put #左填充

replace：

select replace('张无忌爱上了周芷若', '周芷若', '赵敏') as out_put

（2）数学函数

round：

select round(1.65) #四舍五入

select round(1.567, 2) #小数点后保留两位

ceil：

select ceil(1.002) #向上取整

floor：

select floor(1.56) #向下取整

truncate

select truncate(1.65, 1) #小数第几位后截断
mod：
select 10%3 #取余
（3）日期函数
now：
select now() #返回当前系统日期+时间
curdate：
select curdate() #返回系统当前日期，不包含时间
curtime：
select curtime() #返回当前时间，不包含日期
获取指定的部分，年，月，日，小时，分钟，秒：
select year(now()) 年
select year('1998-1-1') 年

序号	格式符	功能
1	%Y	四位的年份
2	%y	2位的年份
3	%m	月份（01,02...11,12）
4	%c	月份（1,2,...11,12）
5	%d	日（01,02,...）
6	%H	小时（24小时制）
7	%h	小时（12小时制）
8	%i	分钟（00,01...59）
9	%s	秒（00,01,...59）

str_to_data：将字符通过指定的格式转换为日期
select str_to_data('1998-3-2', '%Y-%c-%d') as out_put
data_format：将日期转换成字符
select data_format(now(), '%y年%m月%d日') as out_put
（4）其他函数
select version()
select database()
select user()
（5）流程控制函数
if：
select if(10>5, '大', '小')
case：
使用一：
select salary 原始工资, department_id,
case department_id
when 30 then salary*1.1
when 40 then salary*1.2
when 50 then salary*1.3
else salary
end as 新工资
from employees
使用二：

```
select salary,
case
when salary>20000 then 'A'
when salary>15000 then 'B'
when salary>10000 then 'C'
else 'D'
end as 工资级别
from employees
```

9.2.2分组函数

功能：用作统计使用，又称为聚合函数，统计函数或组函数

分类：sum求和，avg平均，max最大值，min最小值，count计算个数

（1）简单使用

```
select sum(salary) from employees
select avg(salary) from employees
select max(salary) from employees
select min(salary) from employees
select count(salary) from employees
```

（2）参数支持的类型

（3）是否忽略null值

分组函数都忽略null值

（4）和distinct搭配

```
select sum(distinct salary), sum(salary) from employees
```

（5）count函数的详细介绍

```
select count(salary) from employees
select count(*) from employees
select count(1) from employees
```

（6）和分组函数一起查询的字段要求是group by后的字段

9.3分组查询

```
语法：
        select 分组函数,列（要求出现在group by的后面）
        from 表
        【where 筛选条件】
        group by 分组的列表
        【order by 子句】
注意：
        查询列表必须特殊，要求是分组函数和group by后出现的字段
```

案例1：查询每个工种的最高工资

```
select max(salary), job_id from employees group by job_id
```

案例2：查询每个位置上的部门个数

```
select count(*), location_id from employees group by location_id
```

案例3：查询邮箱中包含a字符的，每个部门的平均工资

```
select avg(salary), department from employees where email like '%a%' group by department_id
```

案例4：查询哪个部门的员工个数大于2

```
select count(*), department_id from employees group by department_id having count(*)>2
```

9.4连接查询

又称为多表查询，当查询的字段来自多个表时，会用到连接查询

等值连接：

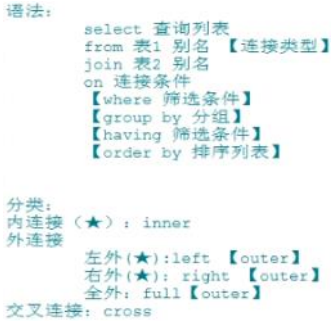
案例1：查询女神名对应的男神名

select name, boyName from beauty, boys where beauty.boyfriend_id=boys.id

非等值连接

自连接

内连接：



案例1：查询员工名和部门名

select last_name, department_name from employees e inner join departments d on e.department_id = d.department_id

案例2：查询名字中包含e的员工名和工种名

select last_name, job_title from employees e inner join jobs j on e.job_id=j.job_id where e.last_name like '%e%'

案例3：查询部门个数>3的城市名和部门个数

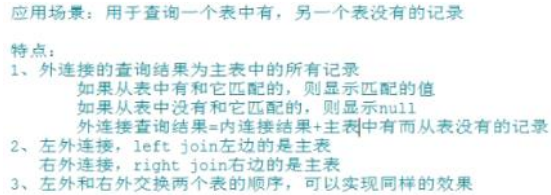
select city, count(*) from employees d inner join location l on d.location_id = l.location_id group by city having count(*)>3

案例4：查询员工名，部门名，工种名，并按部门名降序

select last_name, department_name, job_tilte from employees e inner join departments d on e.department_id = d.depertment_id inner join jobs j on e.job_id = j.job_id order by department_name desc

外连接：

用于查询一个表中有，另一个表中没有的情况



案例1：查询男朋友不在男神表的女神名

select b.name, bo.* from beauty b left outer join boys bo on b.boyfriend_id = bo.id where bo.id is null

案例2：查询哪个部门没有员工

select d.*, e.employee_id from departments d left outer join employees e on d.department_id = e.department_id where e.employee_id is null

全外连接：

等效于：内连接的结果+表1中有但表2没有的+表2中有但表1没有的

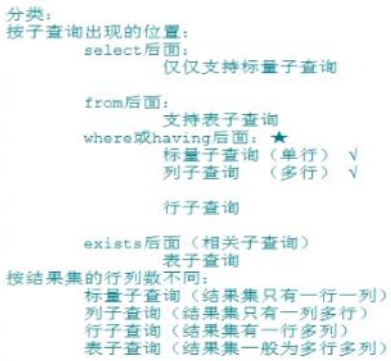
交叉连接：

笛卡尔乘积

select b.*, bo.* from beauty b cross join boys bo

9.5子查询

含义：出现在其他语句中的select语句，称为子查询或内查询；外部的查询语句，称为主查询或外查询



标量子查询：

案例1：谁的工资比Abel高

```
select * from employees where salary>(
        select salary
        from employees
        where last_name='Abel'
)
```

案例2：返回job_id与141号员工相同，salary比143员工多的员工的姓名，job_id和工资

```
select last_name, job_id, salary
from employees
where job_id=(
    select job_id
    from employees
    where employee_id = 141
) and salary>(
    select salary
    from employees
    where employee_id = 143
)
```

列子查询（多行子查询）：

案例1：返回location_id是1400或1700的部门中所有员工姓名

```
select last_name
from employees
where department_id in(
    select distinct department_id
    from departments
    where location_id in (1400, 1700)
)
```

案例2：返回其他部门中比job_id为'IT_PROG'部门任一工资低的员工的员工号，姓名，job_id以及salary

```
select last_name, employee_id, job_id, salary
from employees
where salary<any(
    select distinct salary
    from employees
    where job_id='IT_PROG'
```

) and job_id<>'IT_PROG'

行子查询：

案例1：查询员工编号最小并且工资最高的员工信息

```
select *
from employees
where (employee_id, salary)=(
    select min(employee_id), max(salary)
    from employees
)
```

exists后面（相关子查询）：

语法：

exists(完整的查询语句)

结果：1或0

案例1：查询有员工的部门名

```
select department_name
from departments d
where exists(
    select *
    from employees e
    where d.department_id=e.department_id
)
```

9.6分页查询

应用场景：当要显示的数据，一页显示不全，需要分页提交sql请求

```
select 查询列表
from 表
【join type join 表2
on 连接条件
where 筛选条件
group by 分组字段
having 分组后的筛选
order by 排序的字段】
limit 【offset,】size;

offset要显示条目的起始索引（起始索引从0开始）
size 要显示的条目个数
```

案例1：查询前五条员工信息

```
select * from employees limit 0, 5
```

案例2：查询第11条到第25条

```
select * from employees limit 10, 15
```

9.7联合查询

union：将多条查询语句的结果合并成一个结果

语法：

查询语句1 union 查询语句2 union ……

```
union 联合 合并：将多条查询语句的结果合并成一个结果

语法：
查询语句1
union
查询语句2
union
...

应用场景：
要查询的结果来自于多个表，且多个表没有直接的连接关系，但查询的信息一致时

特点：
1、要求多条查询语句的查询列数是一致的！
2、要求多条查询语句的查询的每一列的类型和顺序最好一致
3、union关键字默认去重，如果使用union all 可以包含重复项
```

9.8插入语句

9.8.1插入方式一

语法：insert into 表名(列名，……) values(值1，……)

（1）插入的值的类型要与列的类型一致或兼容

```
insert into beauty(id, name, sex, borndate, phone, photo, boyfriend_id)
values(13, '唐艺昕', '女', '1990-4-23', '189887787', NULL, 2)
```

（2）不可以为null的列必须插入值，可以为null的列如何插入

方式一：

```
insert into beauty(id, name, sex, borndate, phone, photo, boyfriend_id)
values(13, '唐艺昕', '女', '1990-4-23', '189887787', NULL, 2)
```

方式二：

```
insert into beauty(id, name, sex, phone)
values(15, '娜扎', '女', '2141251151')
```

（3）列的顺序可以调换

（4）列数和值的个数必须一致

（5）可以省略列名，默认所有列，而且列的顺序和表中列的顺序一致

9.8.2插入方式二

语法：insert into 表名 set 列名=值， 列名=值，……

9.8.3两种方式比较

（1）方式一支持多行插入

```
insert into beauty
values(13, '唐艺昕', '女', '1990-4-23', '189887787', NULL, 2)
, (13, '唐艺昕', '女', '1990-4-23', '189887787', NULL, 2)
, (13, '唐艺昕', '女', '1990-4-23', '189887787', NULL, 2)
```

（2）方式一支持子查询，方式二不支持

```
insert into beauty(id, name, phone)
select 26, '宋茜', '563424523'
```

9.9修改语句

9.9.1修改单表的记录

语法：update 表名 set 列=新值, 列=新值, …… where 筛选条件

案例1：修改beauty表中姓唐的女生电话为123224999

```
update beauty set phone='123224999' where name like '唐%'
```

9.9.2修改多表的记录

sql92语法：update 表1 别名, 表2 别名 set 列=值, …… where 连接条件 and 筛选条件

sql99语法：update 表1 别名 inner|left|right join 表2 别名 on 连接条件 set 列=值, …… where 筛选条件

案例1：修改张无忌的女朋友的手机号为114

```
update boys bo
inner join beauty b on bo.id = b.boyfriend_id
set b.phone = '114'
where bo.boyName = '张无忌'
```

9.10删除语句

9.10.1方式一

单表语法：delete from 表名 where 筛选条件
多表删除语法：sql92语法：delete 别名 from 表1 别名, 表2 别名 where 连接条件 and 筛选条件；sql99语法：delete 表1的别名, 表2的别名 from 表1 别名 inner|left|right join 表2 别名 on 连接条件 where 筛选条件
案例1：删除手机号以9结尾的女生的信息
delete from beauty where phone like '%9'
案例2：删除张无忌的女朋友的信息
delete b
from beauty b
inner join boys bo on b.boyfriend_id = bo.id'
where bo.boyname = '张无忌'

9.10.2方式二

语法：truncate table 表名
作用：全删除

9.11 DDL语言（数据定义语言）

库和表的管理

9.11.1库的管理

（1）库的创建
语法：create database 库名
（2）库的修改
rename database 库名 to 新库名
更改库的字符集：alter database 库名 character set gbk
（3）库的删除
drop database 库名
案例1：drop database is exists 库名

9.11.2表的管理

（1）表的创建
语法：create table 表名(
    列名 列的类型(长度) 约束,
    列名 列的类型(长度) 约束,
    列名 列的类型(长度) 约束,
    列名 列的类型(长度) 约束,
    ……
    列名 列的类型(长度) 约束
)
案例1：创建表Book
create if not exists table book(
    id int,
    bName varchar(20),
    price double,
    autherId int,
    publishData datetime
)
（2）表的修改

修改列名，修改列的类型或约束，添加新列，删除列，修改表名

案例1（修改列名）：alter table book change column publishData pubData datetime

案例2（修改列类型）：alter table book modify column pubData timestamp

案例3（添加新列）：alter table book add column annual double

案例4（删除列）：alter table book drop column annual

案例5（修改表名）：alter table book rename to books

（3）表的删除

drop table is exists book

show tables

（4）通用写法

drop database if exists 旧库名

create database 新库名

drop table if exists 旧表名

create table 新表名

（4）表的复制

仅仅复制表的结构：create table 新表名 like 要复制的表名

复制表的结构+数据：create table 新表名 select * from 要复制的表名

只复制部分数据：create table 新表名 select id, name from 要复制的表名 where nation='中国'

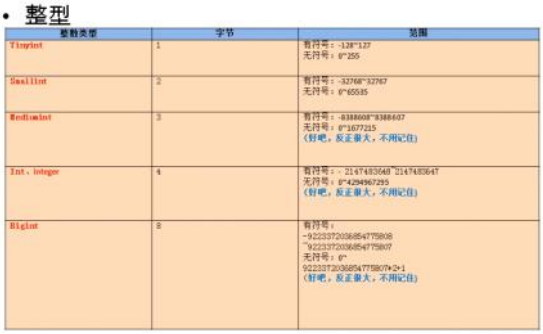
仅仅复制某些字段：create table 新表名 select id, name from 要复制的表名 where 0

9.12数据类型

（1）数值型

整型：

- 整型



整数类型	字节	范围
Tinyint	1	有符号：-128~127 无符号：0~255
Smallint	2	有符号：-32768~32767 无符号：0~65535
Mediumint	3	有符号：-8388608~8388607 无符号：0~1677215 （好吧，反正很大，不用记住）
Int、integer	4	有符号：-2147483648~2147483647 无符号：0~4294967295 （好吧，反正很大，不用记住）
Bigint	8	有符号： -9223372036854775808 ~9223372036854775807 无符号：0~ 9223372036854775807*2+1 （好吧，反正很大，不用记住）

特点：默认有符号，int(7) unsigned表示无符号；如果插入的数值超出整型的范围，发出警告“out of range value“并且插入的是临界值；如果不设置长度，会有默认长度，长度表示显示的最大宽度，如果不够用0在左边填充，但必须搭配zerofill，如int(7) zerofill

浮点型：

- 小数

浮点数类型	字节	范围
float	4	±1.75494351E-38~±3.40282346E+38 （好吧，反正很大，不用记住）
double	8	±2.225073858507201E-308~ ±1.7976931348623157E+308 （好吧，反正很大，不用记住）
定点数类型	字节	范围
DEC(M,D) DECIMAL(M,D)	M+2	最大取值范围与double相同，给定decimal的有效取值范围由M和D决定

float(M, D)

double(M, D)

dec(M, D)

decimal(M, D)

特点：M表示整数部位+小数部位， D表示小数部位的位数；M和D都可以省略，如果是decimal，则M默认为10，D默认为0，如果是float和double，则会根据插入的数值的精度来决定精度；定点型的精度比较高，如果要插入数值的精度要求较高如货币运算等则考虑使用

（2）字符型

较短的文本：char，varchar

字符串类型	最多字符数	描述及存储需求
char(M)	M	M为0~255之间的整数
varchar(M)	M	M为0~65535之间的整数

写法	M的意思	特点	空间的耗费	效率
char	char(M)	最大的字符数，可以省略，默认为1	固定长度的字符	比较耗费	高
varchar	varchar(M)	最大的字符数，不可以省略	可变长度的字符	比较节省	低

较长的文本：text，blob(较长的二进制数据)

binary，varbinary：与char和varchar类似，不同的是它们包含二进制字符串而不包含非二进制字符串

枚举类型：enum

set类型：和enum类似，里面可以保存0~64个成员。与enum类型最大的区别是：set类型一次可以选取多个成员，而enum只能选一个，根据成员数不同，所占字节也不同

成员数	字节数
1~8	1
9~16	2
17~24	3
25~32	4
33~64	8

（3）日期型

日期和时间类型	字节	最小值	最大值
date	4	1000-01-01	9999-12-31
datetime	8	1000-01-01 00:00:00	9999-12-31 23:59:59
timestamp	4	19700101080001	2038年的某个时刻
time	3	-838:59:59	838:59:59
year	1	1901	2155

1、Timestamp支持的时间范围较小，取值范围：19700101080001——2038年的某个时间
Datetime的取值范围：1000-1-1 ——9999—12-31

2、timestamp和实际时区有关，更能反映实际的日期，而datetime则只能反映出插入时的当地时区

3、timestamp的属性受Mysql版本和SQLMode的影响很大

1、Timestamp支持的时间范围较小，取值范围：19700101080001——2038年的某个时间
Datetime的取值范围：1000-1-1 ——9999—12-31

2、timestamp和实际时区有关，更能反映实际的日期，而datetime则只能反映出插入时的当地时区

3、timestamp的属性受Mysql版本和SQLMode的影响很大

9.13常见约束

含义：一种限制，用于限制表中的数据，为了保证表中数据的准确和可靠性

分类：六大约束：not null（非空，用于保证该字段的值不能为空，比如姓名，学号等）；default（默认，用于保证该字

段有默认值）；primary key（主键，用于保证该字段的值具有唯一性，并且非空，比如学号等）；unique（唯一，用于保证该字段的值具有唯一性，可以为空）；check（检查约束，mysql中不支持，加约束）；foreign key（外键，用于限制两个表的关系，用于保证该字段的值必须来自于主表的关联列的值，在从表添加外健约束，用于引用主表中某列的值）

添加约束的时机：创建表时；修改表时

约束的添加分类：列级约束（六大约束都可以写，但是外键约束没有效果）；表级约束（除了非空，默认，其他都支持）

（1）添加列级约束

语法：直接在字段名和类型后面追加约束类型即可，可加多个，用空格隔开即可；只支持：默认、非空、主键、唯一

```
create table stuinfo(
    id int primary key, #主键
    stuName varchar(20) not null, #非空
    gender char(1) check(gender='男' or gender='女'), #检查（mysql中不支持）
    seat int unique, #唯一
    age int default 18, #默认
    majorId int foreign key references major(id) #外键（无效果）
)
create table major(
    id int primary key,
    majorName varchar(20)
)
desc stuinfo
show index from stuinfo #查看stuinfo表中所有的索引，包括主键、外键、唯一
```

（2）添加表级约束

语法：在各个字段的最下面：constraint 约束名 约束类型(字段名)

```
create table stuinfo(
    id int,
    stuName varchar(20),
    gender char(1),
    seat int,
    age int,
    majorid int,
    constraint pk primary key(id),
    constraint uq unique(seat),
     constraint ck check(gender='男' or gender='女'),
    constraint fk_stuinfo_major foreign key(majorid) reference major(id)
)
```

（3）主键和唯一的对比（面试题）

	保证唯一性	是否允许为空	一个表中可以有多少个	是否允许组合
主键	√	×	至多有1个	√，但不推荐
唯一	√	√	可以有多个	√，但不推荐

（4）外键的特点

要求在从表设置外键关系；从表的外键列的类型和主表的关联列的类型要求一致或兼容，名称无要求；要求主表中的关联列必须是一个key（一般是主键、唯一）；插入数据时，先插入主表，再插入从表；删除数据时先删除从表，再删除主表

（5）修改表时添加约束

添加非空：alter table stuinfo modify column stuname varchar(20) not null

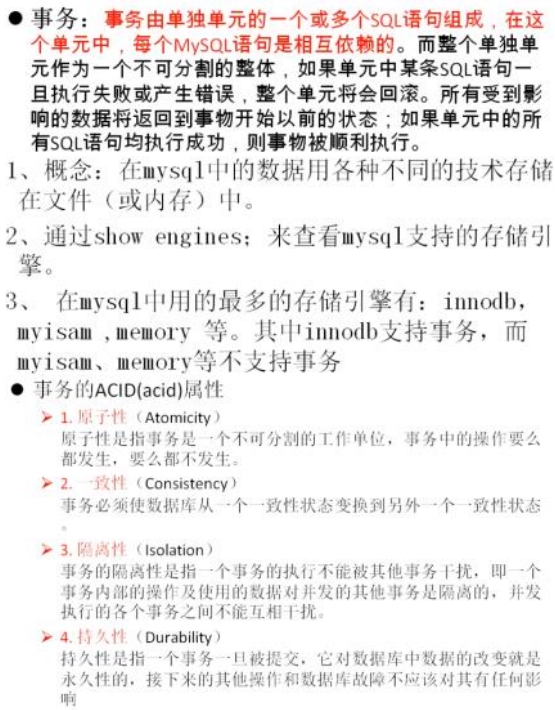
添加默认：alter table stuinfo modify column age int default 18

添加主键：列级约束：alter table stuinfo modify column id int primary key 或 表级约束：alter table stuinfo add primary key(id)
添加唯一键：列级约束：alter table stuinfo modify column seat int unique 或 表级约束：alter table stuinfo add unique(seat)
添加外键：alter table stuinfo add constraint fk_stuinfo_major foreign key(majorid) reference major(id)
（6）修改表时删除约束
删除非空约束：alter table stuinfo modify column stuname varchar(20) null
删除默认约束：alter table stuinfo modify column age int
删除主键：alter table stuinfo drop primary key
删除唯一：alter table stuinfo drop index seat
删除外键：alter table stuinfo frop foreign key fk_stuinfo_major
（7）标识列
又称为自增长列，含义：可以不用手动的插入值，系统提供默认的序列值
特点：标识列不一定必须和主键搭配，但要求是一个key；一个表中至多一个标识列；标识列的类型只能是数值型；标识列可以通过set auto_increment_increment = 3设置步长，也可以通过手动插入值设置起始值

```
create table tab_identity(
    id int primary key auto_increment,
    name varchar(20)
)
```

9.14 TCL语言（事务控制语言Transaction Control Language)

事务：一个或一组sql语句组成一个执行单元，这个执行单元要么全部执行，要么全部不执行



（1）事务的创建
隐式事务：事务没有明显的开启和结束标记，比如Insert，update，delete语句
显示事务：事务具有明显的开启和结束标记，前提：先设置自动提交功能禁用set autocommit=0
步骤1：开启事务

```
set autocommit=0
start transaction（可选）
```

步骤2：编写事务中的sql语句（select，insert，update，delete）

语句1
语句2
……
步骤3：结束事务
commit #提交事务
rollback #回滚事务
案例1：
set autocommit=0
start transaction
update account set balance=500 where username='张无忌'
update account set balance=1500 where username='赵敏'
commit
（2）事务并发问题

- 对于同时运行的多个事务，当这些事务访问数据库中相同的数据时，如果没有采取必要的隔离机制，就会导致各种并发问题：
  - 脏读：对于两个事务T1, T2, T1读取了已经被T2更新但还没有被提交的字段. 之后，若T2回滚，T1读取的内容就是临时且无效的.
  - 不可重复读：对于两个事务T1, T2, T1读取了一个字段，然后T2更新了该字段. 之后，T1再次读取同一个字段，值就不同了.
  - 幻读：对于两个事务T1, T2, T1从一个表中读取了一个字段，然后T2在该表中插入了一些新的行. 之后，如果T1再次读取同一个表，就会多出几行.
- **数据库事务的隔离性：**数据库系统必须具有隔离并发运行各个事务的能力，使它们不会相互影响，避免各种并发问题.
- 一个事务与其他事务隔离的程度称为隔离级别. 数据库规定了多种事务隔离级别，不同隔离级别对应不同的干扰程度，隔离级别越高，数据一致性就越好，但并发性越弱.
- 数据库提供的4种事务隔离级别：

隔离级别	描述
READ UNCOMMITTED (读未提交数据)	允许事务读取未被其他事物提交的变更. 脏读，不可重复读和幻读的问题都会出现
READ COMMITED (读已提交数据)	只允许事务读取已经被其它事务提交的变更. 可以避免脏读，但不可重复读和幻读问题仍然可能出现
REPEATABLE READ (可重复读)	确保事务可以多次从一个字段中读取相同的值. 在这个事务持续期间，禁止其他事物对这个字段进行更新. 可以避免脏读和不可重复读，但幻读的问题仍然存在
SERIALIZABLE(串行化)	确保事务可以从一个表中读取相同的行. 在这个事务持续期间，禁止其他事务对该表执行插入，更新和删除操作. 所有并发问题都可以避免，但性能十分低下

- Oracle支持的2种事务隔离级别：**READ COMMITED**, SERIALIZABLE。 Oracle默认的事务隔离级别为: READ COMMITED
- Mysql支持4种事务隔离级别. Mysql默认的事务隔离级别为: **REPEATABLE READ**
- 每启动一个mysql程序，就会获得一个单独的数据库连接. 每个数据库连接都有一个全局变量@@tx_isolation，表示当前的事务隔离级别.
- 查看当前的隔离级别: SELECT @@tx_isolation;
- 设置当前mySQL连接的隔离级别:
  - set transaction isolation level read committed;
- 设置数据库系统的全局的隔离级别:
  - set global transaction isolation level read committed;

9.15视图

含义：虚拟表，和普通表一样使用，mysql5.1版本出现的新特性，是通过表动态生成的数据，只保存sql逻辑，不保存查询结果
应用场景：多个地方用到同样的查询结果；该查询结果使用的sql语句较复杂
案例1：查询姓张的学生名和专业名
原来写法：
select stuname, majorname
from stuinfo s
inner join major m on s.majorid=m.id
where s.stuname like '张%'
利用视图写法：
create view v1
as

select stuname, majorname
from stuinfo s
inner join major m on s.majorid=m.id
select * from v1 where stuname like '张%'
（1）视图创建
（2）视图修改
方式一：create or replace view 视图名 as
方式二：alter view 视图名 as
（3）视图删除
drop view 视图名, 视图名, ……
（4）查看视图
方式一：desc 视图名
方式二：show create view 视图名
（5）视图的更新
create or replace view v1
as
select last_name, email,
from employees
插入：
insert into v1 values("张飞", 'ssafa@qq.com')
修改：
update v1 set last_name='张无忌' where last_name='张飞'
删除：
delete from v1 where last_name='张无忌'
具备以下特点的视图不允许更新：分组函数，distinct，group by，having，union或者union all，常量视图不能更新，select中包含子查询不能更新，join不能更新，from一个不能更新的视图，where子句的子查询引用了from子句中的表
（6）视图与表对比

	创建语法的关键字	是否实际占用物理空间	使用
视图	create view	只是保存了sql逻辑	增删改查，一般不能增删改
表	create table	保存了数据	增删改查

10.变量
（1）系统变量
说明：变量由系统提供，不是用户定义，属于服务器层面
查看系统变量：
show variables #查看所有系统变量
show global variables
show session variables
查看满足条件的部分系统变量：
show variables like '%char%'
查看指定的某个系统变量的值：
select @@系统变量名
为某个系统变量名赋值：
set 系统变量名 = 值
注意：如果是全局级别，则需要加global，如果是会话级别，则需要加session，如果不写，则默认session
（2）自定义变量

用户变量：针对于当前会话（连接）有效，同于会话变量的作用域，应用在任何地方

Ⅰ.声明并初始化

set @用户变量名=值

set @用户变量名:=值

select @用户变量名:=值

Ⅱ.赋值，更新用户变量的值

方式一：通过set或select

set @用户变量名=新值

set @用户变量名:=新值

select @用户变量名:=新值

方式二：通过select into

select 字段 into 变量名 from 表

Ⅲ.查看用户变量的值

select @用户变量名

局部变量：仅仅在定义它的begin end中有效，应用在begin end中的第一句话

Ⅰ.声明

declare 变量名 类型

declare 变量名 类型 default 值

Ⅱ.赋值

方式一：通过set或select

set @局部变量名=新值

set @局部变量名:=新值

select @局部变量名:=新值

方式二：通过select into

select 字段 into 局部变量名 from 表

Ⅲ.使用

select 局部变量名

对比用户变量和局部变量：

	作用域	定义和使用的位置	语法
用户变量	当前会话	会话中的任何地方	必须加@符号，不用限定类型
局部变量	BEGIN END中	只能在BEGIN END中，且为第一句话	一般不用加@符号，需要限定类型

11.存储过程和函数

说明：类似于java中的方法，好处：提高代码重用性，简化操作

11.1存储过程

存储过程：一组预先编译好的SQL语句的集合，理解成批处理语句

（1）创建语法

create procedure 存储过程名(参数列表)

begin

　　存储过程体（一组合法的SQL语句）

end

注意：

Ⅰ.参数列表包含三部分：参数模式，参数名，参数类型

参数模式：

in：该参数可以作为输入，也就是该参数需要调用方传入值

out：该参数可以作为输出，也就是该参数可以作为返回值

inout：该参数既可以作为输入又可以作为输出，也就是该参数既需要传入值，又可以返回值

Ⅱ.如果存储过程体仅仅只有一句话，begin end可以省略

存储过程体中的每条SQL语句的结尾必须加分号

存储过程的结尾可以使用delimiter重新设置：语法：delimiter 结束标记

（2）调用语法

call 存储过程名(实参列表)

案例1：插入到admin表中五条记录

```
delimiter $
create procedure myp1()
begin
    insert into admin(username, password)
    values('john1', '0000') , ('john2', '0000') , ('john3', '0000') , ('john4', '0000') , ('john5', '0000')
end $
call myp1() $
```

案例2：创建存储过程实现，根据女神名，查询对应的男神信息

```
create procedure myp2(in beautyName varchar(20))
begin
    select bo.*
    from boys bo
    right join beauty b on bo.id = b.boyfriend_id
end $
```

案例3：根据女神名，返回对应的男神名

```
create procedure myp3(in beautyName varchar(20), out boyName varchar(20))
begin
    select bo.boyName into boyName
    from boys bo
    inner join beauty b on bo.id = b.boyfriend_id
    where b.name = beautyName
end $
call myp3('小昭', @bname) $
select @bname
```

案例4：根据女神名，返回对应的男神名和男神魅力值

```
CREATE PROCEDURE myp6(IN beautyName VARCHAR(20),OUT boyName VARCHAR(20),OUT userCP INT)
BEGIN
        SELECT bo.boyName ,bo.userCP INTO boyName,userCP
        FROM boys bo
        INNER JOIN beauty b ON bo.id = b.boyfriend_id
        WHERE b.name=beautyName;
END $
```

（3）删除存储过程

语法：drop procedure 存储过程名

（4）查看存储过程

show create procedure 存储过程名

11.2函数

含义：一组预先编译好的SQL语句的集合，好处：提高代码的重用性；简化操作；减少了编译次数并且减少了和数据库服务器的连接次数，提高了效率

与存储过程区别：存储过程可以有0个返回，也可以有多个返回；函数只能有一个返回

（1）创建语法

```
create function 函数名(参数列表) returns 返回类型
begin
    函数体
end
```

注意：

Ⅰ.参数列表包含两部分：参数名和参数类型

Ⅱ.函数体肯定会有return语句，如果没有会报错；如果return语句没有放在函数体的最后也不报错，但不建议

Ⅲ.函数体中仅有一句话，则可以省略begin end语句

Ⅳ.使用delimiter语句设置结束标记

（2）调用语法

```
select 函数名(参数列表)
```

案例1：返回公司的员工个数

```
create function myf1() returns int
begin
    declare c int default 0 #定义局部变量
    select count(*) into c
    from employees
    return c
end $
select myf1() $
```

（3）查看函数

```
show create function 函数名
```

（4）删除函数

```
drop function 函数名
```

12.流程控制结构

分类：顺序结构，分支结构，循环结构

（1）分支结构

Ⅰ.if函数

功能：实现简单的双分支

语法：if(表达式1, 表达式2, 表达式3)

应用：任何地方

Ⅱ.case结构

情况1：类似于java中的switch语句，一般用于实现等值判断

语法：

```
case 变量|表达式|字段
when 要判断的值 then 返回的值1或语句1
when 要判断的值 then 返回的值2或语句2
……
else 要返回的值n或语句n
end
```

情况2：类似于java中的多重if语句，一般用于实现区间判断

语法：

```
case
```

when 要判断的条件1 then 返回的值1或语句1
when 要判断的条件2 then 返回的值2或语句2
……
else 要返回的值n或语句n
end
特点：可以作为表达式，嵌套在其他语句中使用，可以放在任何地方，begin end中或begin end的外面；可以作为独立的语句使用，只能放在begin end中；如果when中的值满足或条件成立，则执行对应的then后面语句，并且结束case，如果都不满足，则执行else中的语句或值；else可以省略，如果else省略，并且所有的when都不满足，则返回null
Ⅲ.if结构
功能：实现多重分支
语法：
if 条件1 then 语句1
elseif 条件2 then 语句2
……
else 语句n
end if
应用：begin end语句中
（2）循环结构
分类：while、loop、repeat
循环控制：iterate类似于continue，继续，结束本次循环，继续下一次；leave类似于break，跳出，结束当前所在循环
Ⅰ.while
语法：（标签可省略）
标签:while 循环条件 do
    循环体
end while 标签
Ⅱ.loop
语法：
标签:loop
    循环体
end loop 标签
可以用来模拟简单的死循环
Ⅲ.repeat
语法：
标签:repeat
    循环体
until 结束循环的条件
end repeat 标签
案例1：批量插入，根据次数插入到admin表中多条记录

```
create procedure pro_while1(in insertCount int)
begin
    declare i int default 1
    while i<=insertCount do
        insert into admin(username, password) values(concat('Rose', i), '666')
```

```
        set i=i+1
    end while
end $
```

案例2：批量插入，根据次数插入到admin表中多条记录，如果次数大于20停止

```
create procedure pro_while1(in insertCount int)
begin
    declare i int default 1
    a:while i<=insertCount do
        insert into admin(username, password) values(concat('Rose', i), '666')
        if i>=20 then leave a
        end if
        set i=i+1
    end while a
end $
```