## Day09 MySQL高级部分

## 内容简介

完整性约束

多表查询

多表连接

存储过程

触发器.....

## 完整性约束

### 1.作用:

保证用户输入到表中的数据是正确合法的.

在建表的时候进行完整性约束.

### 2.完整性约束的分类(重点)

①.实体完整性

②.域完整性

③.引用完整性(参照完整性)

#### 2.1 实体完整性

实体:指的是表中的每一行,一条记录=一个实体!

实体完整性作用:保证每一行是不重复的!

实体完整性常见的约束:

主键约束,自增长约束,唯一性约束

##### 2.1.1 唯一性约束(unique)

特点:该列数据不能重复!

注意:

唯一性约束的列可以插入空值null.

唯一性约束的列可以插入多个空值null.

案例一:





案例二:创建test表,给id添加unique



插入值:id=1



再插入值,id=1



##### 2.1.2 自增长约束(auto\_increment)

特点:自增长的列会自动给列值加1.列值只会变大,不会变小.

注意:

1.自增长auto\_increment不能单独使用,必须和primary key一起使用;

2.自增长的列数据类型必须是整型数字.

错误信息!



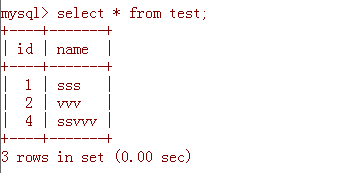
自增长约束的实现:



自增长测试.



注意:自增长的列号,只会增,不会减!



##### 2.1.3 主键约束

主键的特点:非空+唯一性.

注意:每个表中一般都有一个主键,但也不是一定的.

违反了非空约束.



违反了唯一性约束.





#### 2.2 域完整性

域针对的是表中的列,是从列这个角度来进行约束的.

##### 2.2.1 域完整性的约束种类

数据类型约束,非空约束(not null),默认值约束,

check(检查)约束:MySQL目前不支持.

##### 2.2.2 非空约束

特点:非空约束的列不能为null,必须赋值!

非空约束定义



违反非空约束



##### 2.2.3 默认值约束

特点:

1. 在没有赋值的时候,给某一列赋予默认值;
2. 如果赋值了,则采用自己所赋的值;
3. 默认值所在的列,是可以插入空值的.

默认值约束的定义



默认值所在的列,是可以插入空值的.



##### 2.2.4 检查约束check,MySQL不支持!

##### 2.2.5 数据类型约束

插入的值必须与定义的列的数据类型相匹配!

#### 2.3 引用(参照)完整性

主要是针对表的主键和外键来分析的.

约束种类:外键约束

外键约束的目的:为了约束两个表的数据.

使用外键约束的常见情景:

1. .部门表与员工表;
2. .课程表与学生表;
3. .个人信息表与身份证表

....

外键约束相关概念:

主表:被依赖的表,是主表

从表:依赖别人的表是从表.

外键:外键应该建立在从表上.

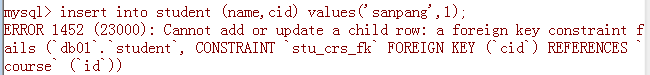
注意:

1. 当建立了主从表之间的主键和外键约束的时候,

此时应该先在主表中插入数据,然后在从表中插入数据.

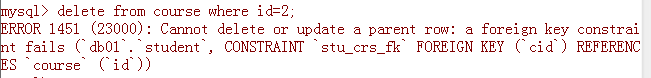
否则就会出现以下错误!

案例一:直接先往从表student中插入数据!



1. 当建立了主从表之间的主键和外键约束的时候,

此时如果直接去删除主表的数据,而该数据已经被从表引用了,会发生如下错误!

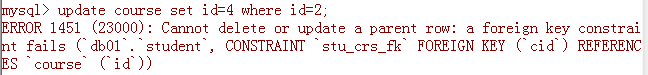


但是如果主表中的数据,没有被从表引用,是可以直接把主表的该行记录给删掉的.



1. 当建立了主从表之间的主键和外键约束的时候,

此时如果直接去修改主表的数据,而该数据已经被从表引用了,会发生如下错误!

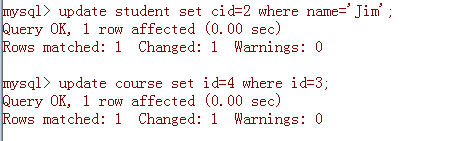


但是如果主表中的数据,没有被从表引用,是可以直接把主表的该行记录给修改掉的.



4.添加数据:.....先添加主表,再添加从表的数据...

5.修改数据:(当没有级联更新的情况下)如果要想修改具有主外键约束的表,应该先修改从表,再修改主表.



6.删除数据:(当没有级联删除的情况下).....先删除从表,再删除主表....

#### 2.4 主键的创建与删除方式

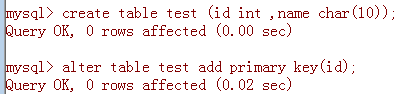
①.第一种方式:



②.第二种方式:



③.第三种方式:



④.联合主键的创建

传统的方式去创建主键,肯定是不行的!



联合主键的创建方式:



⑤.主键的删除(alter + drop)

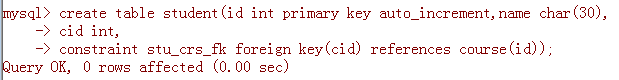


#### 2.5 外键的创建

**先建立主表:**



**再建立从表:**



constraint:是用来声明外键的关键字.

stu\_crs\_fk:外键的名称,随便写.

foreign key(cid):指明该表中哪个字段是外键.

references:建立引用参照关系,references后面一般要跟待关联的表名和字段名.

student表的外键是course的主键!

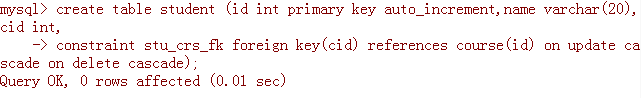
course表的主键是student表的外键!

#### 2.6 主从表之间的级联操作

级联操作的实现:

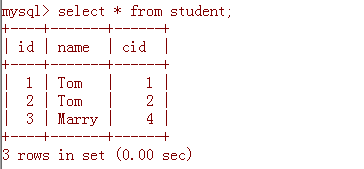
on update cascade:级联更新

on delete cascade:级联删除



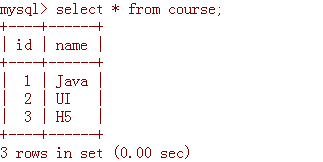
级联更新案例:

第一步:学生表,学生表中有学生选了课程编号是4的课程.

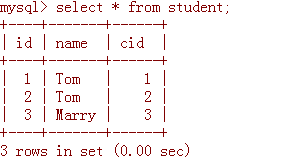


第2步:在主表中,将课程编号为4的课程编号,改为编号3.



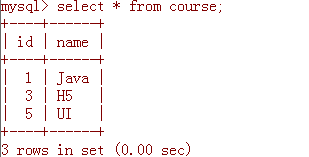


第三步:查询student表,结果如下:

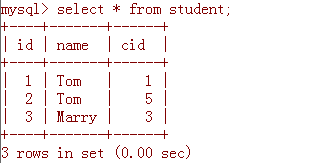


级联删除的实现:

第一步:课程表



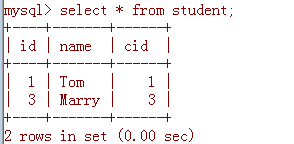
学生表:



第二步:删除课程表中id=5的课程.



第三步:查询学生表



## 多表查询

### 表与表之间的关系

1. .一对一:person表与身份证card表,或者person表与社保账号shebao表.
2. .一对多:人与他自己各种证件之间的关系,一个人可以有多张卡....
3. .多对多:学生和老师之间的关系,student和teacher表.

学生和课程之间.....

### 多表查询(重中之重!)

#### 2.1 多表查询的方式

1. .合并结果集:
2. .连接查询:

----交叉连接(没有实用意义):

----内连接:

----外连接:

---左外连接:

---右外连接:

---全外连接:

----自然连接:

1. .子查询.

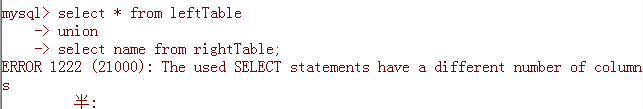
#### 2.2 合并结果集

**union关键字的使用**

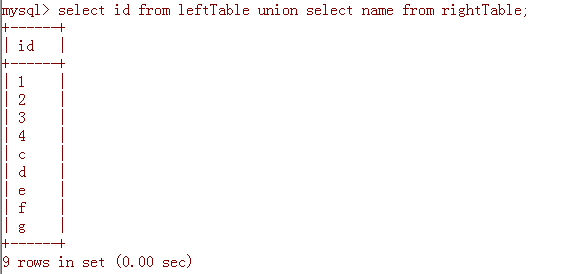
语法:将2个查询中间用一个union关键字连接.



注意:左右两侧查询的列的数量应该一致!如果不一致,错误如下.

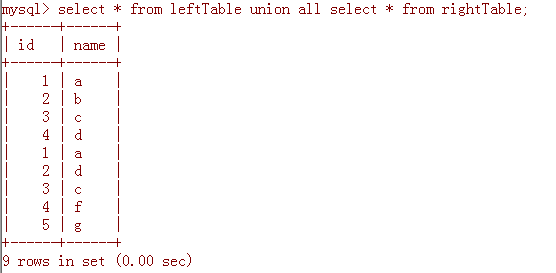


只要左右两侧的查询,列的数量一致即可,类型可以不同!



**union all关键字**

**语法: 两个select之间用union all连接.**



**union和union all的区别:**

**union会消除重复记录(两个表中的记录必须完全一致.)**

**union all 不会消除重复记录.**

#### 2.3 连接查询(很重要)

交叉连接

内连接

外连接

自然连接

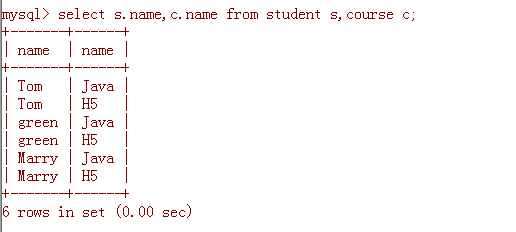
问题:列名重复!



##### **2.3.1 交叉连接:会产生笛卡尔积**.

也就是说在进行多表查询的时候,会把table1的行数乘以table2的行数,得到的这么一个结果,就是所谓的笛卡尔积.

所以交叉连接没有实际意义.



##### 2.3.2 内连接

需要考虑的问题:

1. .确定要查询的表有哪些?
2. .确定要查询的列有哪些?
3. .确定连接查询条件.

条件的数量至少,条件的数量=表的数量-1.

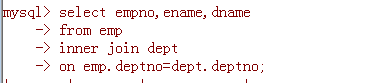
案例:查询员工所在的部门名称.

内连接的标准写法:

select...from 表1

inner join 表2

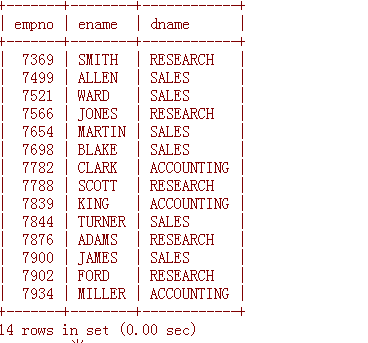
on 条件;



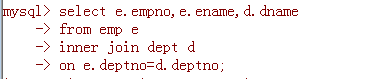
简化写法(常用):



结果如下:



别名的使用:



##### 2.3.3 外连接

外连接分类:

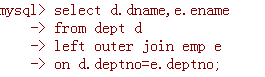
左外连接,右外连接,全外连接.

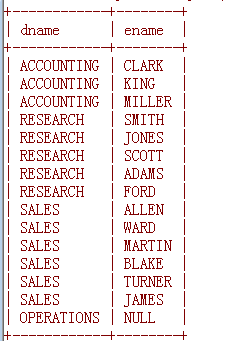
**左外连接:**

**左外连接,使用左边表的数据去匹配右边表的数据,如果符合连接条件,则显示;否则,显示null.**

**特点:左边表的数据一定是完整显示的!**

案例:查询每个部门的员工姓名:



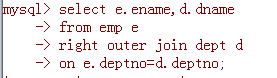


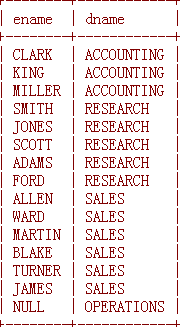
**右外连接:**

**右外连接,使用右边表的数据去匹配左边表的数据,如果符合连接条件,则显示;否则,显示null.**

**特点:右边表的数据一定是完整显示的!**

**案例:用右外连接实现**查询每个部门的员工姓名.



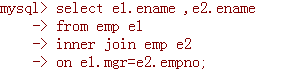


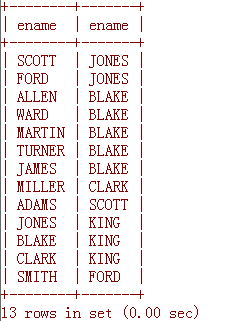
##### 2.3.4 自连接

自连接就是自己这个表连接自己这个表!

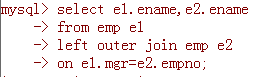
案例:查询员工的领导姓名.

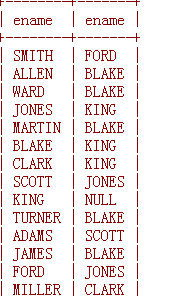
1. .以内连接来实现.





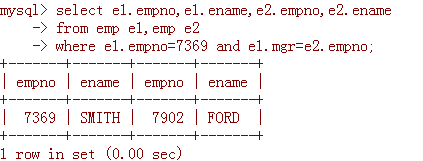
1. .以左外连接来实现:





案例二:

查询员工编号为7369的员工编号,姓名,及其直接领导的编号和姓名.(要求:以where语句的方式来实现).



#### 2.4 子查询(非常重要)

一个select里面包含另一个完整的select,就被称为子查询.

案例一:查询工资高于SMITH的员工姓名.

1. .查询SMITH的sal;

select sal from emp where ename=’SMITH’;

1. .以第一步的查询结果作为结果,查询比该结果大的记录.

select \* from emp where sal>①.sal;

具体实现:



注意:

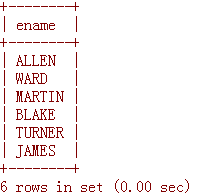
>,>=,<,<=等这样的操作符属于”一对一”的操作.

也就是说操作符的前后两部分都应是单一的.



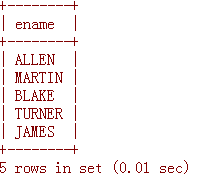
案例二:查询与WARD属于同一个部门的员工.





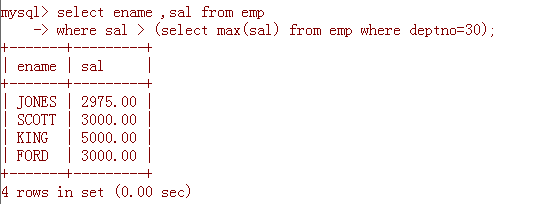
查询与WARD属于同一个部门的员工,且查询结果中不包含自身.



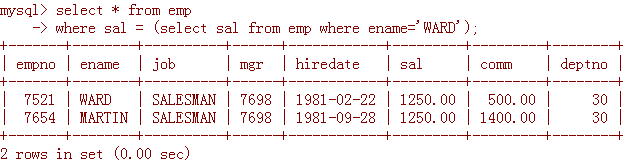


案例三:

查询工资高于30号部门所有人的员工信息.



案例四:查询工资与WARD薪资相等的员工信息.



案例五:查询有2个直接下属的员工信息.

1. .select distinct mgr from emp ;

查询所有的领导信息.

等同于如下语句:

select mgr from emp group by mgr;

查询所有的领导信息.

1. .在查询出来的6个领导中,进行过滤.看哪个领导mgr这个字段出现的次数大于等于2.

select mgr from emp group by mgr having count(mgr) >=2;

1. .select \* from emp where empno in (.......);

