**Mysql数据库约束：**

主键约束；非空约束；唯一约束；外键约束检查约束

* **主键约束**

可以建表的时候就创建主键：

Create table st5(

Id int primary key,

Name varchar(50)

);

也可以建表之后再创建主键：

假设已经建表完成了，执行alter table 表名add primary key（字段）即可。

主键的特性：唯一非空

不要选择业务字段作为主键；主键一般设置为自动增长，这个时候插入数据的时候就不用写id的值了,id会自动添加并自动增加。

Create table st5(

Id int primary key,

Name varchar(50)

);

Insert into st5(name) values(‘张三’);

注意：如果delete from st5,把里面所有的数据删除，如果再插入insert数据，那么他的id是从原先断掉的id开始继续往下走，如果以前是2条数据，那么新插入的id是从3开始的。

如果使用truncate table st5,再去插入数据，这个时候id就是从1开始的。

* **唯一约束**

**Create table st7(**

**Id int,**

**Name varchar(50) unique**

**);**

这个时候，是不能插入重复的值的，但是null除外

* **非空约束**

**Create table st7(**

**Id int,**

**Name varchar(50) not null**

**);**

注意是值不能为null, 但是’’ 还是可以插入的

* **默认值：**

**Create table st8(**

**Id int,**

**Name varchar(50),**

**Address Varchar(100) default ‘北京’**

**);**

* **外键约束**

insert into emp(name,age,dep\_name,dep\_location) values('张三',20,'研发部','广州');

insert into emp(name,age,dep\_name,dep\_location) values('李四',21,'研发部','广州');

insert into emp(name,age,dep\_name,dep\_location) values('王五',20,'研发部','广州');

insert into emp(name,age,dep\_name,dep\_location) values('老王',20,'销售部','深圳');

insert into emp(name,age,dep\_name,dep\_location) values('大王',22,'销售部','深圳');

insert into emp(name,age,dep\_name,dep\_location) values('小王',18,'销售部','深圳');

mysql> create table emp(

-> id int primary key auto\_increment,name varchar(50),

-> age int,

-> dep\_name varchar(30),dep\_location varchar(30)

-> );

上面的表其实可以通过拆分成两张表，通过外键dep\_id连接起来，这就是外键。

create table employee(

id int primary key auto\_increment,

name varchar(20),

age int,

dep\_id int

);

create table department(

id int primary key auto\_increment,

dep\_name varchar(30),

dep\_location varchar(20)

);

insert into department values(null,'研发部','广州'),(null,'销售部','深圳');

insert into employee(name,age,dep\_id) values('李四',21,1);

insert into employee(name,age,dep\_id) values('王五',20,1);

insert into employee(name,age,dep\_id) values('大王',20,2);

insert into employee(name,age,dep\_id) values('老王',22,2);

insert into employee(name,age,dep\_id) values('小王',18,2);

外键约束是为了防止insert 数据错误的时候，采取的有效措施，

比如 inser into employee(name,age,dep\_id) values(‘小红’,35,6);当不设置外键约束的时候，他是可以执行成功的，但是部门表里面又没有6号部门，这就冲突了，所以需要外约束。

* 主表：主键所在的表
* 从表：外键所指向的表

Drop table employee;

Create table employee(

Id int primary key auto\_increment,

Name varchar(50),

Age int,

Dep\_id int,

Constraint emp\_depid\_fk foreign key(dep\_id) references department(id)

);

这个时候在执行，insert into table employee(name,age,dep\_id) values(‘小红’,35,6);

就不会插入成功了。

**如何删除外键约束**：

Alter table employee drop foreign key emp\_depid\_fk;

**怎么在建表成功后去添加外键约束：**

首先要把不符合外键约束的数据删除掉，才能添加外键约束

再添加外键约束，alter table employee add constraint emp\_depid\_fk foreign key(dep\_id) references department(id);

**级联更新和级联删除**

如果这个时候执行update department set id =10 where id =1;会报错，因为一旦修改成id10 ，那么employee里面的id为1的员工就无法找到归属部门了。这就是级联更新。同理，delete这些操作也不能执行。

那么如何执行这种操作呢？执行级联更新即可。

其实级联就是外键关联导致的。

Drop table employeee;

Create table employee(

Id int primary key auto\_increment,

Name varchar(50),

Age int,

Dep\_id int,

Constraint emp\_depid\_fk foreign key(dep\_id) references department(id) on update cascade on delete cascade  
);

insert into employee(name,age,dep\_id) values('李四',21,1);

insert into employee(name,age,dep\_id) values('王五',20,1);

insert into employee(name,age,dep\_id) values('大王',20,2);

insert into employee(name,age,dep\_id) values('老王',22,2);

insert into employee(name,age,dep\_id) values('小王',18,2);

这个时候，继续执行update department set id =10 where id =1;就能成功。两边的表都会改变。Delte那一句也一样。

**表和表之间的关系**

* **一对多：**

比如部门对员工，即通过主外键关系来表示。

* **多对多的关系：**

学生和课程，学生和老师

通过中间表来体现多对多的关系,中间表至少两个字段，这两个字段分别指向各自一方的 主键。

Create table students(

Sid int primary key auto\_increment not null,

Sname varchar(50)

);

Create table course(

cid int primary key auto\_increment not null,

cname varchar(50) not null

);

Create table stu\_course(

Sid int not null,

cid int not null

);

Insert into students values(‘张三’),(‘李四’);

Insert into course(cname) values(‘语文’),(‘java’);

Insert into stu\_course(sid,cid) values(1,1),(1,2),(2,1),(2,2);

一对一太简单，不说了，可以放一个表里头了。

**数据库三大范式：**

第一范式：

原子性， 表中各个记录不能再次拆分。

第二范式：

在第一范式的基础上，每一个字段都完全依赖于主键。

如：

表 学生证号；学生证名称；学生证办理时间；借书证号；借书证名称；借书证办理时间

学生号是不依赖于借书证号的，所以可以拆分成两张表

学生证号；学生证名称；学生证办理时间 >>>>学生证表

借书证号；借书证名称；借书证办理时间>>>>借书证表

第三范式：

在满足第二范式的前提下，表中每一列都直接依赖于主键，而不是通过其他的列来间接依赖于主键。

比如

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 年龄 | 所在学院 | 学院地点 |

所在学院是依赖于学号的，学院地点依赖于所在学院再依赖于学号，这样就是不满足第三范式的。

如何修改呢？

拆分成两张表：

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 年龄 | 所在学院的编号（外键） |

|  |  |  |
| --- | --- | --- |
| 学院编号 | 所在学院 | 学院地点 |

学号>>学院编号>>所在学院>>学院地点

还有其他范式，但是一般只需要满足这三个范式即可。

**多表查询和内连接：**

Create table dept(

Id int primary key auto\_increment,

Name varchar(20)

);

Insert into dept(name) values(‘开发部’),(‘市场部’),(‘财务部’),(‘销售部’);

Create table emp(

Id int primary key auto\_increment,

Name varchar(10),

Gender char(1),

Salary double,

Join\_date datetime,

Dept\_id int,

Foreign key (dept\_id) references dept(id)  
);

Insert into emp(name,gender,salary,join\_date,dept\_id) values('孙悟空','男',7200,'2020-01-01',1);

Insert into emp(name,gender,salary,join\_date,dept\_id) values('猪八戒','男',4200,'2011-02-24',1);

Insert into emp(name,gender,salary,join\_date,dept\_id) values('唐僧','男',3200,'2019-02-24',2);

Insert into emp(name,gender,salary,join\_date,dept\_id) values('白骨精','女',4200,'2016-02-24',3);

Insert into emp(name,gender,salary,join\_date,dept\_id) values('蜘蛛精','女',4200,'2015-02-24',3);

**查询孙悟空的名字，和部门名字**

Select \* from emp,dept; 这个查出来是一个笛卡尔积，是不对的

Select \* from emp,dept where emp.dept\_id=dept.id; 内连接查询,属于隐式内连接。

Select \* from emp inner join dept on emp.dept\_id=dept.id; 显式内连接，跟上面的结果是一样的。只是语法跟完整。

**查询唐僧的姓名，性别，工资，所在部门**

Select emp.name,gender,salary,dept.name from emp,dept where emp.dept\_id =dept.id and emp.name='唐僧';

Select emp.name,gender,salary,dept.name from emp inner join dept on emp.dept\_id=dept.id and emp.name='唐僧';

**外连接：**

**左外连接：**

Select \* from dept left join emp on emp.dept\_id=dept.id;

那个表距离select近，或者在前面，哪个表就是主表，就显示全部内容，右边表没匹配上就显示null

**右外链接**

在员工表里面再加入1条数据，

Insert into emp(name,gender,salary,join\_date,dept\_id) values('沙僧','男',6600,'2021-02-24',null);

Select \* from dept left join emp on emp.dept\_id=dept.id;

**子查询**

查询开发部中有哪些员工

Select \* from emp where dept\_id =(Select id from dept where name=”开发部”);

**子查询的三种情况：**

* 结果是一个值：

如查询工资最高的是谁：

Select \* from emp where salary=(select max(salary) from emp);

查询工资小于平均工资的有哪些？

Select \* from emp where salary<(select avg(salary) from emp);

* 查询结果是多行单列

如查询工资大于5000的员工，来自于哪些部门名字

Select name from dept where id in (select dept\_id from emp where salary >5000);

查询开发部和财务部所有员工的信息

Select id from dept where name in (‘开发部’,’财务部’);

Select \* from emp where dept\_id in (Select id from dept where name in ('开发部','财务部'));

* 查询结果是多行多列

查询出2011年以后入职的员工信息，包括部门

Select \* from emp where join\_date>'2016-01-01 00:00:00';

Select \* from dept,(Select \* from emp where join\_date>'2016-01-01 00:00:00') as e where dept.id=e.dept\_id;

这个子查询语句相当于作为1个表来用的。

也可以用表连接来做：

Select \* from emp,dept where emp.dept\_id=dept.id and join\_date>'2016-01-01 00:00:00';

**一般子查询可以做，表连接也可以做，优势是表连接的性能要优于子查询。**