# 2 SQL

## 2.1 建表

**语法：**

**create table** *r* (*A*1 *D*1, *A*2 *D*2, ..., *An Dn,* (integrity-constraint1),  
 ...,  
 (integrity-constraintk))

★例子：

**create table** *instructor* (

*ID* **char**(5),

*name* **varchar**(20) **not null,**

*dept\_name* **varchar**(20),

*salary* **numeric**(8,2),

**primary key** (*ID*),

**foreign key** *(dept\_name*) **references** *department);*

## 2.2 更新表结构

* + **alter table** *r* **add** *A D 添加属性*
  + **alter table** *r* **drop** *A 删除属性*
    - * + **drop table** *r 删除表*

## 2.3 更新表的内容

**Insert （增元组）**

**insert into** *instructor* **values** (‘10211’, ’Smith’, ’Biology’, 66000);

**Delete （删元组）**

**delete from** *student；*

**Update（修改元组）**

**Update instructor set salary= 70000；**

## 2.4 查询

**select** *A*1, *A*2, ..., *An*  
**from** *r*1, *r*2, ..., *rm*  
**where** *P*

*Ai* represents an attribute

*Ri* represents a relation

*P* is a predicate.

例子：

1. 从instructor表中输出dept\_name的信息，删除重复

**select distinct** *dept\_name* **from** *instructor*

1. 查询所有老师的工号、姓名和月薪

**select** *ID, name, salary/12* **from** *instructor*

**select** *ID, name, salary/12* **as** *monthly\_salary* **from** instructor

1. To find all instructors in Comp. Sci. dept with salary > 80000 （from子句）

**select** *name* **from** *instructor* **where** *dept\_name =* *‘*Comp. Sci.'**and** *salary* > 80000

1. Find the names of all instructors in the Art department who have taught some course and the course\_id （笛卡儿积）

**select** *name, course\_id***from** *instructor , teaches***where** *instructor.ID = teaches.ID* ***and*** *instructor. dept\_name =* ‘Art’

（5）Find the names of all instructors who have a higher salary than some instructor in ‘Comp. Sci’. （更名操作）

* + **select distinct** *T.name***from** *instructor* **as** *T, instructor* **as** *S***where** *T.salary > S.salary* **and** *S.dept\_name = ‘Comp. Sci.’*

## 2.5 字符串比较

通配符 %, 可以匹配任意一个字符串

占位符 \_，可以匹配任意一个字符。

★例子：

（1）Find the names of all instructors whose name includes the substring “dar”.

**se**le**ct** *name* **from** *instructor* **where***name* **like '**%dar%'

* SQL supports a variety of string operations such as
  + 连接字符串 (using concat(A,B))
  + 大写转换为小写 (and 小写变大写) LOWER,UPPER
  + 统计字符串长度, 截取字符串, etc.
  + LENGTH(STR), LEFT(STR,N),RIGHT(STR,N),SUBSTRING(STR,N,LEN),…

## 2.6 排序

**order by** *name* **desc （降序）**

**order by** *name* **asc （升序，默认是升序）**

★例子：

List in alphabetic order the names of all instructors

**select distinct** *name* **from** *instructor* **order by** *name*

## 2.7 集合操作

Set operations **union, intersect,** and **except**

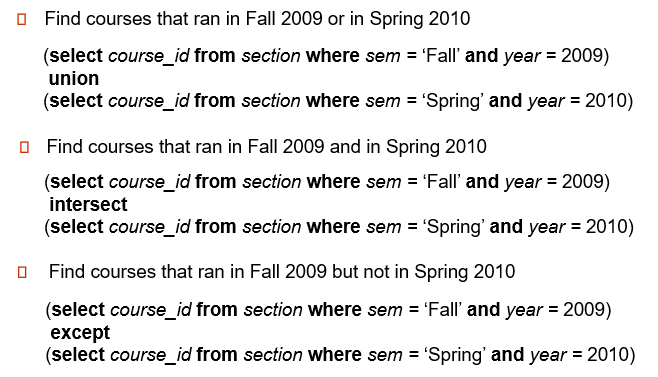
★例子：

找到instructor中的最高工资

（**select distinct** *salary* **from** *instructor*）

**except**

（**select distinct** *T.salary* **from** *instructor* **as** *T, instructor* **as** *S* **where** *T.salary < S.salary*）



## 2.8 聚合函数

**avg:** average value  
**min:** minimum value  
**max:** maximum value  
**sum:** sum of values  
**count:** number of values

★例子：

1. Find the total number of instructors who teach a course in the Spring 2010 semester

**select count** (**distinct** *ID*)  
**from** *teaches***where** *semester* = ’Spring’ **and** *year* = 2010;

1. Find the number of tuples in the course relation

**select count** (\*)  
**from** *course*;

## 2.9 分组聚合

★例子：

输出每个系的老师的平均年薪

**select** *dept\_name*, **avg** (*salary*) **as** *avg\_salary*  
**from** *instructor* **group by** *dept\_name*;

## 2.10 Having子句（分组后的限定条件）

★例子：

输出学院内老师的平均年薪大于40000的学院和他们的平均工资

**select** *dept\_name*, **avg** (*salary*)

Having子句是在group之后的。

需要和where子句区分

**from** *instructor*

**group by** *dept\_name*

**having avg** (*salary*) > 42000;

## 2.11 聚合函数和NULL

所有的聚合函数（除了count（\*））都忽略NULL

**select sum** (*salary* ) **from** *instructor*

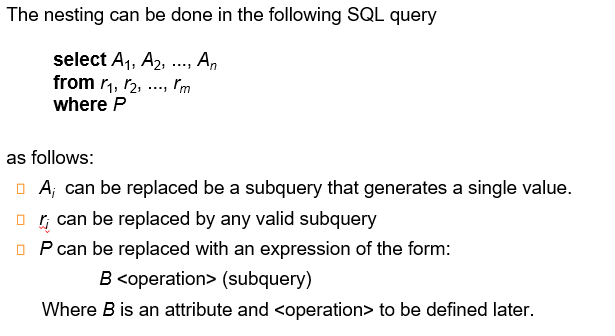
在计算sum的时候，会忽略null值，计算非null值的和。如果全部是null，那么最终这个查询返回的是null。

同理，max(salary), min(salary), avg(salary)也是忽略null的；当所有值都是null，则返回null。

count(salary)，若salary全都是null，则返回0。否则返回非null的数量。

count（\*）计算的是元组的数量，null值不忽略。

## 2.12 嵌套子查询



Select、from、where子句都可以再嵌套一个查询子句。只是要求select子句中嵌套的必须是标量查询，即只能返回一个值。

## 2.13 where子句

Where子句一般处理三类问题：（1）集合成员判断；（2）集合比较；（3）集合基数测试（是否是空集，是否存在重复的元组等）

1. 集合成员判断

用到的关键字是**in**或者**not in**

★例子：

Find courses offered in Fall 2009 and in Spring 2010

**select distinct** *course\_id*

**from** *section*

**where** *semester* = ’Fall’ **and** *year*= 2009 **and** *course\_id* **in** (**select** *course\_id*

**from** *section*

**where** *semester* = ’Spring’

**and** *year*= 2010);

1. 集合比较

用<, ≤, >, =, ≠等符号表示大小关系。

还可以使用some或者all表示“一些”和“所有”

★例子：

找到比生物学院所有老师工资都高的老师的姓名

**select** *name*

**from** *instructor*

**where** *salary* > **all** (**select** *salary*

**from** *instructor*

**where** *dept name* = ’Biology’);

1. 集合基数测试

用exists判断是否不为空集?若不是空集，则返回true，否则false。

Not exists与exists正好相反。

Unique判断是否存在重复。如果没有重复，则返回true，否则false。

★例子：

（1）找到选了生物学院所有开设的课程的学生。

**select distinct** *S*.*ID*, *S*.*name*

**from** *student* **as** *S*

**where not exists** ( (**select** *course\_id*

**from** *course*

**where** *dept\_name* = ’Biology’)

**except**

(**select** *T*.*course\_id*

**from** *takes* **as** *T*

**where** *S*.*ID* = *T*.*ID*));

（2）找到在2009年最多只上一次课的课程。

**select** *T*.*course\_id***from** *course* **as** *T***where unique** (**select** *R*.*course\_id* **from** *section* **as** *R* **where** *T*.*course\_id*= *R*.*course\_id* **and** *R*.*year* = 2009);

## 2.14 from子句

★例子：

输出学院内老师的平均年薪大于42000的学院和他们的平均工资

**select** *dept\_name*, *avg\_salary***from** (**select** *dept\_name*, **avg** (*salary*) **as** *avg\_salary* **from** *instructor* **group by** *dept\_name*)  
**where** *avg\_salary* > 42000;

## 2.15 with子句

★例子：

找到最大的学院预算。 **with** *max\_budget* (*value*) **as** (**select max**(*budget*)  
 **from** *department*)  
 **select** *department.name* **from** *department*, *max\_budget* **where** *department*.*budget* = *max\_budget.value*;

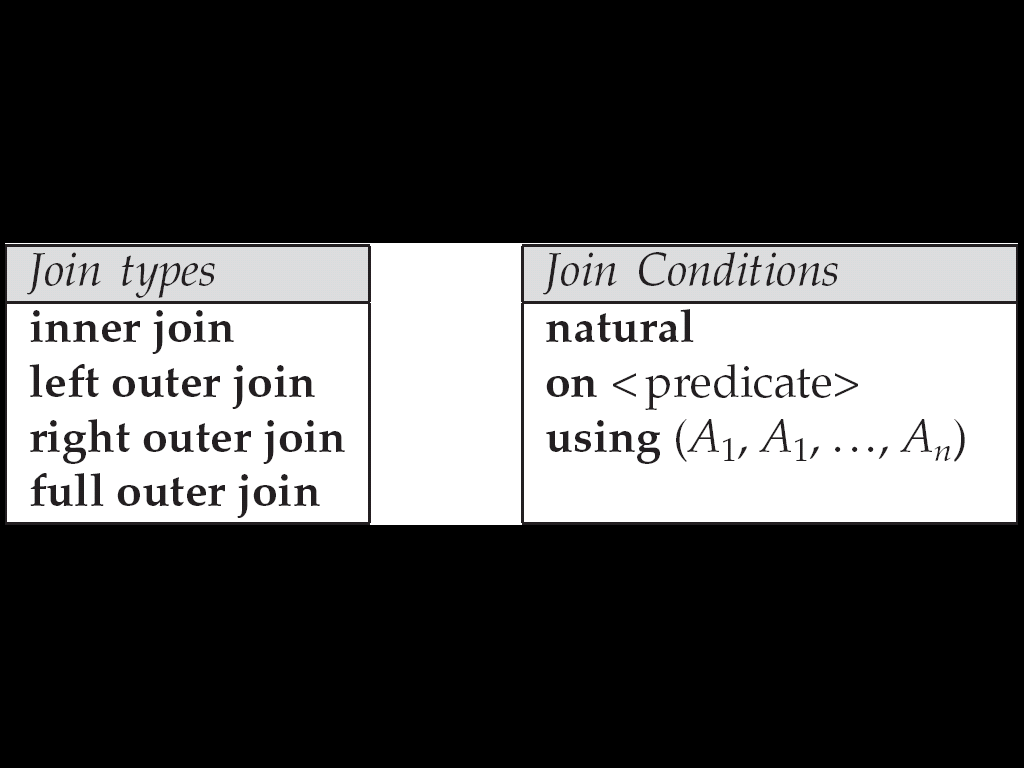
## 2.16 select子句

★例子：

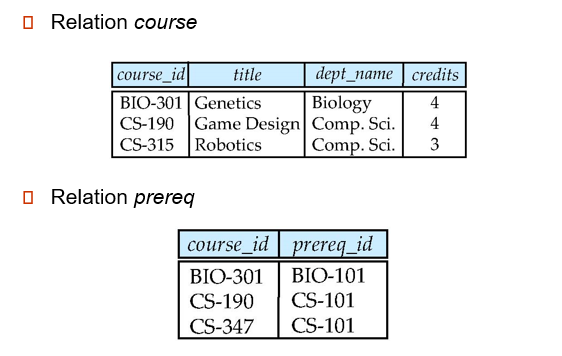
统计各个学院内老师的数量。

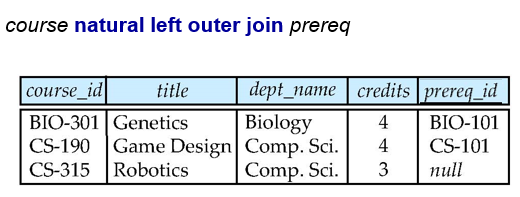
**select** *dept\_name*,   
 (**select count**(\*)   
 **from** *instructor* **where** *department*.*dept\_name* = *instructor*.*dept\_name*)  
 **as** *num\_instructors***from** *department*;

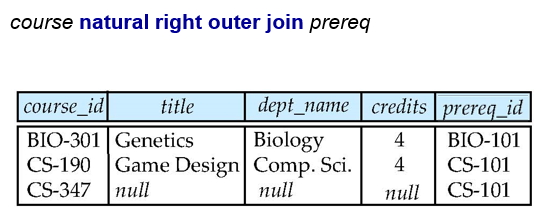
## 2.17 连接

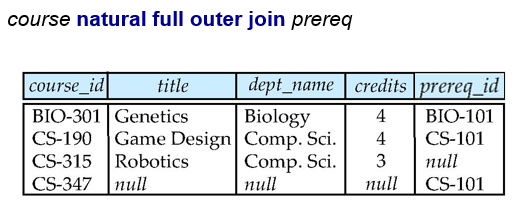


连接类型和连接条件可以任意两两联合使用。









## 2.18 View Definition

**create view** *v* **as** *<* query expression >

where <query expression> is any legal SQL expression. The view name is represented by *v.*

* A view of instructors without their salary  
   **create view** *faculty* **as   
   select** *ID*, *name*, *dept\_name* **from** *instructor*
* Find all instructors in the Biology department  
   **select** *name* **from** *faculty* **where** *dept\_name =* ‘Biology’

2.19 约束

（1）单一关系上的约束

* **not null 非空**
* **primary key 主键**
* **unique 唯一约束**
* **check** (P), where P is a predicate

**create table** *section* (

*course\_id* **varchar** (8),

*sec\_id* **varchar** (8),

*semester* **varchar** (6),

*year* **numeric** (4,0),

*building* **varchar** (15),

*room\_number* **varchar** (7),

*time slot id* **varchar** (4),

**primary key** (*course\_id*, *sec\_id*, *semester*, *year*),

**check** (*semester* **in** (’Fall’, ’Winter’, ’Spring’, ’Summer’))  
);

（2）参照完整性约束

**create table** *course* ( *course\_id*  **char**(5) **primary key**,  
 *title* **varchar**(20),  
  *dept\_name* **varchar**(20) **references** *department*)

2.20 用户定义的类型和域

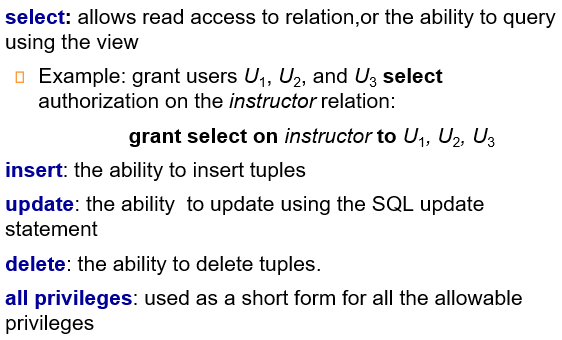
**create type** *Dollars* **as numeric (12,2) final，强类型检查，不能加约束**

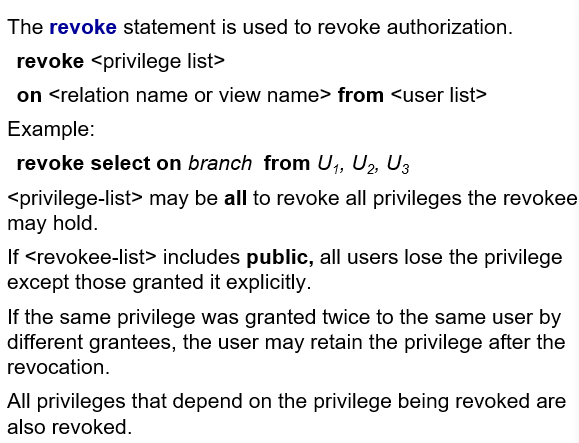
**create domain** *person\_name* **char**(20) **not null，弱类型检查，可以加约束**

**create domain** *degree\_level* **varchar**(10)  
**constraint** *degree\_level\_test***check** (**value in** (’Bachelors’, ’Masters’, ’Doctorate’));

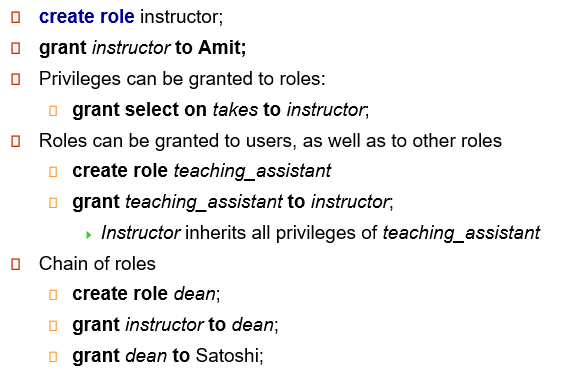
2.21 权限与角色

可以对用户进行授予/撤销权限操作：





角色可以看作是权限的集合，当某个用户属于某个角色，那么这个角色所对应的权限自然就赋予给了这个用户。



2.22 函数与过程

函数和过程都是存储在数据库中的元数据。

函数和过程都可以被调用。二者的区别在于，函数有显式的返回值，而过程没有显式的返回值。但是二者实际上都可以有返回值还可以不止一个。

**create function** *dept\_count* (*dept\_name* **varchar**(20))

**returns integer**

**begin**

**declare** *d\_count* **integer;  
 select count** (*\** ) **into** *d\_count* **from** *instructor* **where** *instructor.dept\_name = dept\_name* **return** *d\_count;* **end**

**调用函数**

**select** *dept\_name, budget* **from** *department* **where** *dept\_*count (*dept\_name* ) > 12

**表函数**

**create** **function** *instructor\_of* (*dept\_name* **char**(20))

**returns** **table** (

*ID* **varchar**(5),  
 *name* **varchar**(20),  
 *dept\_name* **varchar**(20),  
 *salary* **numeric**(8,2))

**return** **table**  
 (**select** *ID, name, dept\_name, salary*  
 **from** *instructor* **where** *instructor.dept\_name = instructor\_of.dept\_name*)

**select \* from table** (*instructor\_of* (‘Music’))

**过程**

**create procedure** *dept\_count\_proc* (**in** *dept\_name* **varchar**(20), **out** *d\_count* **integer)  
 begin**

**select count**(*\**) **into** *d\_count* **from** *instructor* **where** *instructor.dept\_name = dept\_count\_proc.dept\_name*

**end**

**过程调用**

**declare** *d\_count* **integer**;

**call** *dept\_count\_proc*( ‘Physics’, *d\_count*);

2.23 触发器 trigger

触发器是由触发事件发生而自动执行的一段代码。触发事件可以是insert, delete，update。不可以是select。

**create trigger** *setnull\_trigger* **before update of** *takes* **referencing new row as** *nrow* **for each row  
 when (***nrow.grade* = ‘ ‘)  
 **begin atomic** **set** *nrow.grade* = **null;  
 end;**