2 SQL

2.1 建表

语法:

create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,

(integrity-constraint$_1$),

...,

(integrity-constraint$_k$))

★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 $A_1, A_2, \ldots, A_n$

from $r_1, r_2, \ldots, r_m$

where P

A_i represents an attribute

R_i 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 $\boxed{\textbf{distinct}}$ *dept_name* **from** *instructor*

（2） 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

（3） 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

（4） 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’

切值存在

（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”.

select *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*)

- Find courses that ran in Fall 2009 or in Spring 2010

  (**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
  union
  (**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

- Find courses that ran in Fall 2009 and in Spring 2010

  (**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
  intersect
  (**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

- Find courses that ran in Fall 2009 but not in Spring 2010

  (**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
  except
  (**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

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;

（2） 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*)
from *instructor*

Having 子句是在 group 之后的。
需要和 where 子句区分

```
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

The nesting can be done in the following SQL query

> **select** $A_1, A_2, \ldots, A_n$
> **from** $r_1, r_2, \ldots, r_m$
> **where** P

as follows:

- A_i can be replaced be a subquery that generates a single value.
- r_i can be replaced by any valid subquery
- P can be replaced with an expression of the form:

  B <operation> (subquery)

  Where B is an attribute and <operation> to be defined later.

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 $\boxed{\textbf{in 或者 not in}}$

★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*

且

```
                          from section
                          where semester = 'Spring'
                          and year= 2010);
```

（2） 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');
```

（3） 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;
```

提出来了

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

Join types
inner join
left outer join
right outer join
full outer join

Join Conditions
natural
on < predicate>
using $(A_1, A_1, \ldots, A_n)$

连接类型和连接条件可以任意两两联合使用。

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

course **natural left outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

course **natural right outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

course **natural full outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

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，若类型检查，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:

select: allows read access to relation,or the ability to query using the view

- Example: grant users U_1, U_2, and U_3 **select** authorization on the *instructor* relation:

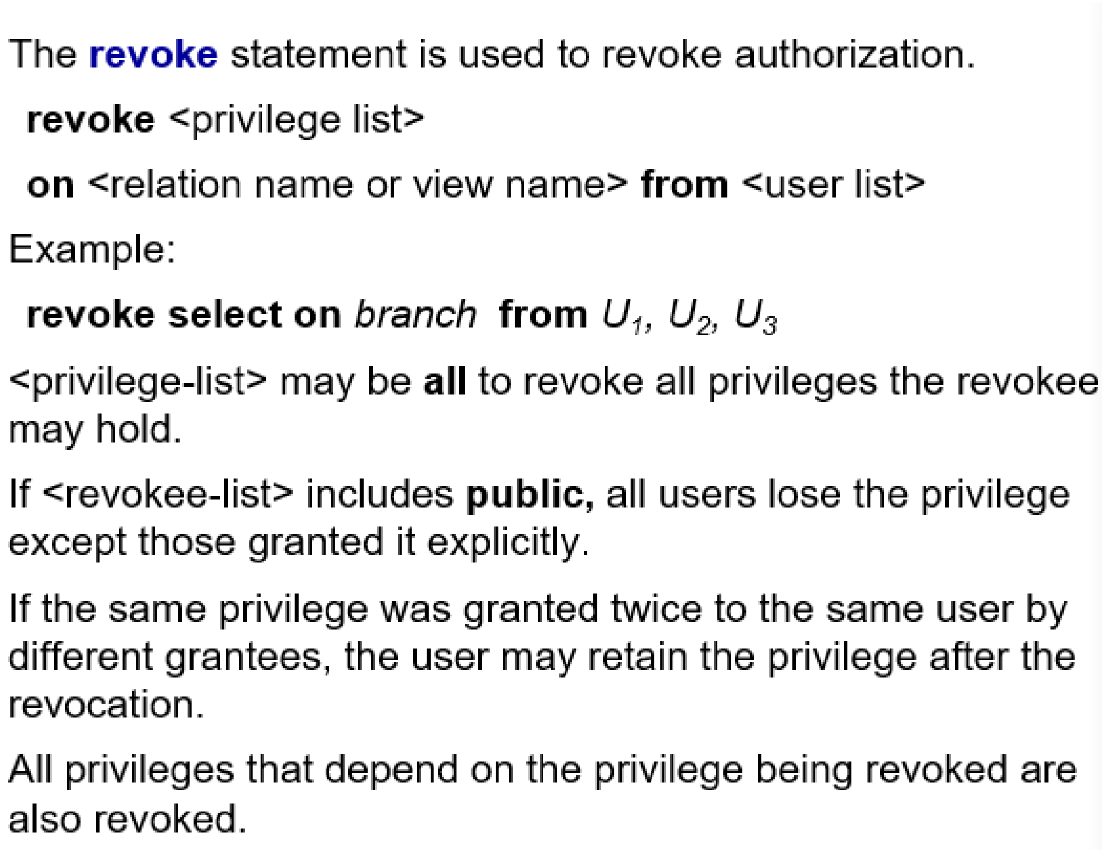
  grant select on *instructor* **to** U_1, U_2, U_3

insert: the ability to insert tuples

update: the ability to update using the SQL update statement

delete: the ability to delete tuples.

all privileges: used as a short form for all the allowable privileges



The **revoke** statement is used to revoke authorization.

revoke <privilege list>

on <relation name or view name> **from** <user list>

Example:

revoke select on *branch* **from** U_1, U_2, U_3

<privilege-list> may be **all** to revoke all privileges the revokee may hold.

If <revokee-list> includes **public,** all users lose the privilege except those granted it explicitly.

If the same privilege was granted twice to the same user by different grantees, the user may retain the privilege after the revocation.

All privileges that depend on the privilege being revoked are also revoked.

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。

- **create role** instructor;
- **grant** *instructor* **to Amit;**
- Privileges can be granted to roles:
  - **grant select on** *takes* **to** *instructor*;
- Roles can be granted to users, as well as to other roles
  - **create role** *teaching_assistant*
  - **grant** *teaching_assistant* **to** *instructor*;
    - *Instructor* inherits all privileges of *teaching_assistant*
- Chain of roles
  - **create role** *dean*;
  - **grant** *instructor* **to** *dean*;
  - **grant** *dean* **to** Satoshi;

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

触发器是由出发事件发生而自动执行的一段代码。出发时间可以是 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;
```