课程名称: 数据库系统

第8讲:(第3章)

基 本 SQL

一 基本的数据定义语言DDL

1.定义数据结构

```
create table instructor
    (ID          varchar (5),
    name         varchar (20) not null,
    dept_name    varchar (20),
    salary       numeric (8,2),
    primary key (ID),
    foreign key (dept_name) references department);
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),
    foreign key (course_id) references course);
create table teaches
    (ID          varchar (5),
    course_id    varchar (8),
    sec_id       varchar (8),
    semester     varchar (6),
    year         numeric (4,0),
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (course_id, sec_id, semester, year)
references section,
    foreign key (ID) references instructor);
```

①

②

③

④

```
create table department
    (dept_name      varchar(20),
    building        varchar (15),
    budget          numeric (12,2),
    primary key (dept_name));
create table course
    (course_id      varchar (7),
    title           varchar (50),
    dept_name       varchar (20),
    credits         numeric (2,0),
    primary key (course_id),
    foreign key (dept_name) references department);
```

图 3-1　大学数据库的部分 SQL 数据定义

一 基本的数据定义语言DDL

1.定义数据结构

2. 定义主键和外键约束(完整性约束)

键约束

外键约束(参照约束)

```
create table department
    (dept_name      varchar(20),
    building        varchar (15),
    budget          numeric (12,2),
    primary key (dept_name));
create table course
    (course_id      varchar (7),
    title           varchar (50),
    dept_name       varchar (20),
    credits         numeric (2,0),
    primary key (course_id),
    foreign key (dept_name) references department);
```

主键包含多个属性

包含多个外键

```
create table instructor
    (ID             varchar (5),
    name            varchar (20) not null,
    dept_name       varchar (20),
    salary          numeric (8,2),
    primary key (ID),
    foreign key (dept_name) references department);
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),
    foreign key (course_id) references course);
create table teaches
    (ID             varchar (5),
    course_id       varchar (8),
    sec_id          varchar (8),
    semester        varchar (6),
    year            numeric (4,0),
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (course_id, sec_id, semester, year)
references section,
    foreign key (ID) references instructor);
```

属性约束

图 3-1　大学数据库的部分 SQL 数据定义

3. 创建数据库的实例**(选讲-上机学习)

```
—CREATE  DATABASE  数据库名
—ON
—( NAME = 逻辑文件名,
—    FILENAME= ‘mdf数据文件路径’,
—        SIZE = 10 MB, /*数据文件初始大小*/
—        MAXSIZE = 20 MB, /*数据文件最大值*/
—        FILEGROWTH =2 MB), /*数据文件增长值*/

—LOG ON   /*创建日志文件，可省略*/
—( NAME = 日志文件名,
—FILENAME= ‘日志文件名’,
—        SIZE = 10 MB,
—        MAXSIZE = 20MB,
—        FILEGROWTH =10%)
—GO
```

命令格式

```
—CREATE  DATABASE  Bank
—ON
—( NAME = Bank_data1,
—    FILENAME= ‘D:\Bank_data1.mdf’,
—        SIZE = 20 MB,
—        MAXSIZE =100MB,
—        FILEGROWTH =2 MB)

—LOG ON
—( NAME = Bank_log1,
—FILENAME= ‘D:\Bank_log1.ndf’,
—        SIZE = 4 MB,
—        MAXSIZE = 25MB,
—        FILEGROWTH =1MB)
—GO
```

图2 一个实例

二 SQL查询语言QL&关系运算

• **(P.35-36 SQL语句)**

投影 Select name
From instructor
选择 Where dept_name = ‘Comp.Sci’ and salary > 70000;

Select name, instructor.dept_name, building
等值连接 From instructor, department
Where instructor.dept_name = department.dept_name;

(Select course_id
From section
Where semster = ‘Fall’ and year = 2009)
并 **Union** ⟷ p.44：差 **Except**，交 **Intersact**
(Select course_id
From section
Where semster = ‘Spring’ and year = 2010);

[图5] 关系模式上的数据查询例子

• **(补充案例)**

更名 Select name **as** '教师姓名' **/ as** instuctor_name
From instructor;

as可以于属性，也可用于表(参P.40)

Select name, instructor.dept_name, building
笛卡尔积 From instructor, department;

[图6] 更简单的数据查询例子

• **(P.35-36 SQL语句)**

```
Select name
From instructor
Where dept_name =  'Comp.Sci'  and salary > 70000;

Select name, instructor.dept_name, building
From instructor, department
Where instructor.dept_name = department.dept_name;

(Select course_id
From section
Where semster =  'Fall'  and year = 2009)
Union
(Select course_id
From section
Where semster =  'Spring'  and year = 2010);
```

选择

等值连接

注:两关系连接时可以使用大于、小于等比较符号

1)关系记录的筛选
2)两关系间的连接

等效**p.38**

注:自然连接**Natural join**与笛卡尔积X两点最大不同(比较p. 36图36p. 38图38):
1）仅包含符合连接条件的元组
2）连接属性仅出现一次

[图5] 关系模式上的数据查询例子

• **(补充案例)**

```
Select name
From instuctor;

Select name, instructor.dept_name, building
From instructor, department;
```

[图6] 更简单的数据查询例子

三（QL中）的聚集函数与嵌套子查询

1. 聚集函数

SQL查询能力很强！
1）实现了基本代数运算
2）灵活的表间连接方式
3）实现了代数运算复合(下面的嵌套子查询)
4）灵活的where条件
5）聚集函数等常用函数
6）嵌入式和动态SQL

(教材P.46)
平均值avg
最小值min
值大值max
总和sum
计数count

```
select avg (salary)                          仅计算一个系的平均工资
from instructor
where dept_name= 'Comp. Sci';

select count (distinct ID)                   计数前先去除重复元组
from teaches
where semester ='Spring' and
year=2010;

select count (*)                             *代表计算所有元组个数
from course;

select dept_name, avg (salary) as
avg_salary                                   第1个为平均工资显示部门名
from instructor                              第2个用于指定计算范围(分组)
group by dept_name;

select dept_name, avg (salary)
from instructor    限定输出哪些平均工资(结果筛选)
group by dept_name
having avg (salary) > 42000;
```

2. 嵌套子查询

嵌套子句可以多种方式用在where子句中

(P.49&P.24)找出在2009年秋季和2010年春季同时开课的所有课程

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
    course_id  not in (select course_id
                       from section
                       where semester = 'Spring'
                       and year= 2010);
```

(集合成员资格)

(p.50)查出这些老师的姓名，他的工资要比Biology系某教师工资高

```
select name
from instructor
where salary > some (select salary
                     from instructor
                     where dept_name ='Biology');
```

(集合的比较)

(P.50&P.24)找出在2009年秋季和2010年春季同时开课的所有课程

```
select course_id
from section as S
where semester= ‘Fall' and year=2009 and
exists (select *
            from section as T
            where semester='Sring’
            and year=2010
            and S.course_id=T.course_id);
```

(空关系测试)

2. 嵌套子查询

嵌套子句可用在**having**子句中-输出结果筛选

(p.50)找出平均工资最高的系

```
select name
from instructor
group by dept_name
having avg(salary) >= all(select avg(salary)    (集合的比较)
                          from instructor
                          group by dept_name);
```

嵌套子句可用在**from**子句中-生成中间关系

(p.52)找出‘系平均工资超过42000美元的那些系'的教师平均工资

```
select dept_name, avg_salary
from (select dept_name, avg(salary) as avg_slary (属性的别名)
        from instructor
        group by dept_name)
where avg_salary >42000;
```

嵌套子句可用在select子句中–生成标量值

(p.54)列出所有系以及它们拥有的教师数

```
select dept_name,
        (select count(*)
          from instructor
          where department.dept_name=instructor.dept_name)
          as num_instructors (表的别名)
 from department;
```

四 SQL数据操作语言DML

删除数据(可利用嵌套子句)

(P.54)

```
delete from instructor
where dept_name= ’Finance’;


delete from instructor
where dept_name in (select dept_name
                    from department
                    where building = ’Watson’);
```

插入数据(三种常用方式)

P.56~57

insert into *course* (*course_id*, *title*, *dept_name*, *credits*)
values (’ CS-437’ , ’ Database Systems’ , ’ Comp. Sci.’ , 4);

insert into *student*
values (’ 3003’ , ’ Green’ , ’ Finance’ , *null*);

insert into instructor
select *ID, name, dept_name, 18000*
from *student*
where *dept_name = ‘Music’ and tot_cred > 144;*

更新数据

P.56~57给工资超过100000美元的教师涨3%的工资，其余教师涨5%。

```
update instructor
set salary = salary * 1.03
where salary > 100000;
update instructor
set salary = salary * 1.05
where salary < =100000;
```

如果改变两条语句的顺序，
有可能工资低于**100000**的员工涨**8%**

```
update instructor
set salary = case
      when salary <= 100000 then salary * 1.05
       else salary * 1.03
       end;
```