课程名称: 数据库系统

第9讲:(第4章)

中级SQL

一 SQL支持的表间连接方式 P. 63

1.（自然/等值）连接的不同方式 P. 63

（方式**1**） (p.39)

```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID;
```

查询结果表虽然相同，但**Where**允许按指定属性**(**可不同名**)**连接，且在连接表中连接属性都会出现**(ID**两次**)** 。

（方式2） (p.39)

```
select name, course_id
from instructor natural join teaches;
```

虽然都是按相同属性连接，但**using**允许按指定属性、而非两表所有同名属性连接

（方式3）(p.40)

```
select name, title
from (instructor natural join teaches)
join course using(course_id);
```

```
select *
from student, takes
where student.ID = takes.ID;
```

作用及查询结果表都相同，

在连接表中连接属性都会出现**(ID**重复出现两次**)**。

（方式4）(P.63)

```
select *
from student join takes on student.id=takes.id;
```

2. 外连接的不同方式

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

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

这**4**种连接类型和**3**种条件可任意组合

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

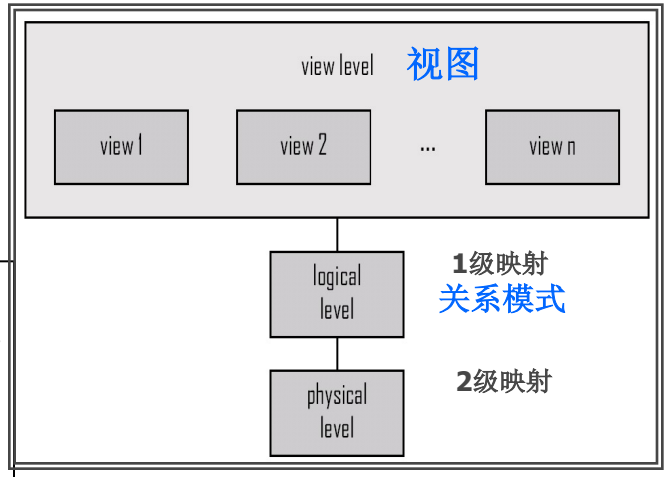
Join Conditions
natural
on <predicate>
using $(A_1, A_1, \ldots, A_n)$

二 视图

视图采用**create view**语句定义，
可以定义为任何一个**SQL**语句，
无实际数据，'虚表'，有利于数据一致性！
视图上可以在定义新的视图！

```
create view faculty as
select ID, name, dept_name
from instructor ;        (P.68)
```

1）可以在任何**QL**语句中像表一样的被使用！
2）增强查询能力且方便**(**用户**/**程序员**)**使用！
3）还可以提供数据访问安全控制**(**隐藏数据**)**！
4）作为外模式**(1**级映射**)**有利于应用独立性！



仅在少数简单视图**(updatable)**上可以更新数据！**P.72**

```
create view physics_fall_2009 as                    (P.68)
select course.course_id, sec_id, building, room_number
from course, section
where course.course_id = section.course_id
        and course.dept_name = ’ Physics’
        and section.semester = ’ Fall’
        and section.year = ’ 2009’ ;
```

```
create view physics_fall_2009_watson as
select course_id, room_number
from physics_fall_2009
where building= ’ Watson’ ;        (P.68)
```

三 事务p. 73

案例：通过银行卡为校园一卡通充值的处理过程：

1）输入充值金额100元→2）银行卡下账100元→3）一卡通上账100元

⇕

异常现象:若此时操作失败(因停电、网断等)，导致银行卡少100元，而一卡通钱仍未增加！

解决方法

事务： **SQL**允许将多个数据操作捆绑为一个逻辑单元

要么这些操作全部完成，要么一个也未执行！

例子：(银行卡下账100元，一卡通上账100元)两事务操作-1个事务

SQL提供的相关命令为：

Commit work---提交事务；

Rollback work---撤销事务；

Begin atomic---事务(操作语句)开始；

……

End---事务(操作语句)结束；

四 完整性约束p. 72-73

1. 键完整性约束（主码/主键）
   关系(模式)必需有一个主码，来区分不同元组！
   SQL采用primary key…来定义！

2.参照完整性约束（外码/外键）
   用另一关系的主码，来约束属性取值的有效性！
   SQL采用foreign key … references …来定义！

3.其它数据完整性约束：

属性(非空)完整性约束
属性(唯一)完整性约束
属性(范围)完整性约束
键完整性约束
参照完整性约束

```
create table instructor
    (ID          varchar (5),
     name        varchar (20) not null  unique,
     dept_name   varchar (20),
     salary      numeric (8,2) Check (salary>10000),
     primary key (ID),
     foreign key (dept_name) references department);
```

1. 复杂条件(标量集合限定取值范围)完整性约束
2. 复杂条件(来自他表**select**结果限定取值范围)完整性约束

```
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'))
); check(time_slot_id in(select time_slot_id from time_slot)));
```

2. 外键约束方式

三种参照约束方式：

```
create table course (                                P.73-74
…
dept_name varchar(20),          1)不写时拒绝删除
foreign key (dept_name) references department
        on delete cascade       ---2)连带删除(修改)
        on update cascade,
…);     set null/set defaul     ---3)设置为空/默认值
```

3. 断言

例：约束要求：**student**每个元组的**tot_cred(**学生的总学分**)**取值应等于该生所修完课程的学分总和**(**关系**takes**∞**course**的**credits)**

create assertion <assertion-name> **check** <predicate>;

```
create assertion credits_earned_constraint check
    (not exists (select ID
        from student
        where tot_cred <> (select sum(credits)
                from takes natural join course
                where student.ID = takes.ID
                and grade is not null and grade <> 'F' ));
```

当数据更新时，保持谓词为真。(否则拒绝更新)
但作用有利有弊

图 4-9　一个断言的例子

五 授权 p.81

1. 表(关系)上的授权

在开放环境中通过授权限制用户对数据的合法访问！

只有授权用户才能查看(/插入/修改/删除)相关表中的数据.

注:表的创建者, 自然拥有表上的一切权限.

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

select
insert
update
delete
all privileges

public ? 所有用户

将instructor表上的查看(插入/…)权授予用户。

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

select
…
all

用户在**branch**表上的查看**(…)**权被收回。

作用及好处? 简化权限管理

create role instructor; 角色名

grant *instructor* **to Amit;** 用户名

grant select on *takes* **to** *instructor*;

可以建立角色role(用户群)。
当将某权限授予角色时，
该用户群均有该使用权限。

create role *teaching_assistant*;

grant *teaching_assistant* **to** *instructor*;

还可以创建子角色

3. 授权转移

- **grant select on** *department* **to** Amit **with grant option**;
- **revoke select on** *department* **from** Amit, Satoshi **cascade**;
- **revoke select on** *department* **from** Amit, Satoshi **restrict**;

允许转授权限

级联回收权限(默认值)

防止级联回收权限

变化示例1：
U_2执行语句:
grant update on teaches To U_6;

变化示例2：
U_1执行语句：
revoke update on teaches from U_5;
DBA执行语句：
revoke update on teaches from U_2;

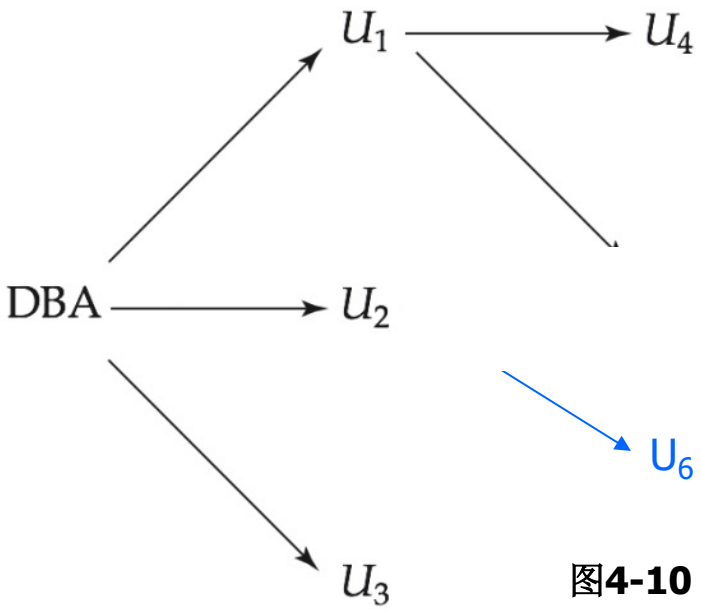


图4-10 示例：表**teaches**上的**update**更新权

作用：
1）描述在一张表上某种授权的当前状态，便于系统动态管理授权；
2）当DBA或具有权限的用户(树上节点)进行授权时，树扩展(生长)；
3）当DBA或具有权限的用户(树上节点)回收权限时，树收缩(枯萎)；

2. 视图(虚关系)上的授权P.82

视图上也可以授权(查看/修改/删除数据)

```
create view geo_instructor as
    (select *
     from instructor
     where dept_name='Geology');
```

在表instructor上创建一个视图*geo_instructor*

```
grant select on geo_instructor to geo_staff
```

(role角色)

将视图上的查看权授予一个角色*geo_staff*

```
select *
from geo_instructor;
```

如果该用户在instructor上没有获得select授权，则他仍然**看不到**数据！

(p.83) 用户可以函数与过程**(&5.2)**，
并可对其他用户授予**execute**执行权。

注：**1**）视图的创建者，自然拥有该视图上的所有权限！
　　2）函数与过程的创建者，自然拥有其上所有权限！