第四章　中级SQL

一、SQL的数据类型与模式

- SQL中的日期和时间类型：
  - **date**：日历日期，包括年(四位)、月和日
  - **time**：一天中的时间，包括小时、分和秒
    - 可以用**time**(p)来表示秒的小数点后的数字位数
  - **timestamp**：**date**和**time**的组合
    - 可以用**timestamp**(p)来表示秒的小数点后的数字位数
  - **interval**：一段时间
- 用户定义的类型：
  - creat type：定义数据类型，没有约束

    create type *Dollars* **as numeric (12,2) final**
  - creat domain：定义一个域，可以有约束

    create domain *person_name* **char**(20) **not null**
  - 强制类型转换：(cast r.A as Pounds) 把关系r中的A的数据类型强制转换为Pounds

- 大对象类型:
  - 用于存储可能很大(KB级)的属性
  - 分类:
    - 字符数据的大对象数据类型(**clob**)
    - 二进制数据的大对象数据类型(**blob**)

二、完整性约束

- 目的：防止对数据的意外破坏
- 单个关系上的约束：
  - **not null**
  - **unique**
  - **check**
- 多个关系上的约束：
  - 参照完整性

1.单个关系上的约束

- **not null约束**
  - 声明变量name非空

    name **varchar**(20) **not null**
- **unique约束**
  - **unique** (A_1, A_2, …, A_m)
  - **unique**声明指出属性A1,A2,…,Am形成了一个候选码;
    即在关系中没有两个元组能在所有列出的属性上取值相同;
    然而候选码属性可以为null，除非它们已被显式地声明为**not null**;
    空值不等于其他任何的值;

- **check**子句
  - **check** (P)
  - **check**(P)子句指定一个谓词P，关系中的每个元组都必须满足谓词P

```
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.多个关系上的约束

- 参照完整性
  - 保证在一个关系中给定属性集上的取值也在另一关系的特定属性集的取值中出现
  - 创建键
    - primary key
    - unique key
    - foreign key：某一个表中创建了一个外键，这个外键一定是另一个表中的主键

意思是在account创建branch_name外键与branch表关联(在branch表中branch_name是主键)

- 例子
  - customer表和branch表,
    此时两表没有关系

```
create table customer
    (customer_name    char(20),
    customer_street   char(30),
    customer_city     char(30),
    primary key (customer_name ))
create table branch
    (branch_name      char(15),
    branch_city       char(30),
    assets            numeric(12,2),
    primary key (branch_name ))
```

  - account表(表中的branch_name与branch表中的
    branch_name是一个东西，故创建一个外键进行关联)

```
create table account
    (account_number   char(10),
    branch_name       char(15),
    balance           integer,
    primary key (account_number),
    foreign key (branch_name) references branch )
```

- depositor表

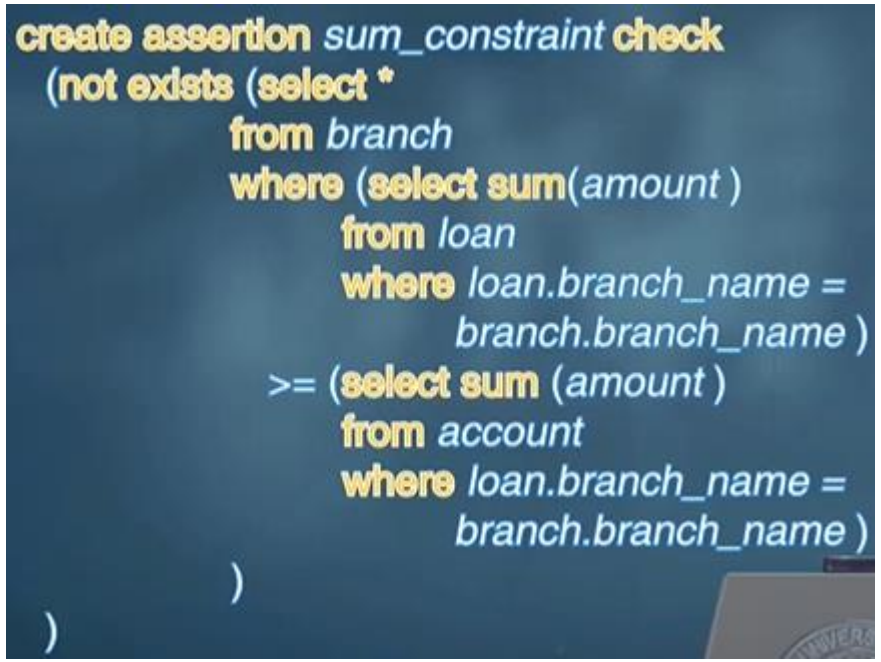
```
create table depositor
    (customer_name  char(20),
    account_number  char(10),
    primary key (customer_name, account_number),
    foreign key (account_number ) references
                account, (account_number )
    foreign key (customer_name ) references
                customer ) (customer_name )
```

3.断言

- check只能在单个关系中进行约束，断言可以在多个表间进行约束
- 使用断言会增大数据库的开销
- 格式：

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

- 例子：



题目规定：对于每个银行，它的总资产都要大于它的贷款总额

左边的意思是对于branch表中的每一条数据都查询是否有贷款总额大于等于总资产的情况

四、授权

- 我们可能会给一个用户在数据库的某些部分授予几种形式的权限。
- **权限**(privilege)：每种类型的授权都称为一个权限
- 在数据库中常用的权限：
  - 对于某一个表中数据的权限
    - 读操作(Read)
    - 插入操作(Insert)
    - 修改操作(Update)
    - 删除操作(Delete)
  - 对数据库当中的对象的权限
    - Index：可以创建和删除索引
    - Resources：将常用的权限归结在一起形成一个角色
      创建、修改、删除关系以及将它所创建的关系的插入修改删除的权限赋给其他的用户
    - Alteration：允许对关系当中的某一些属性进行修改
    - Drop：删除关系

1.权限的授予与收回

- SQL标准包括的权限:（主要是对于数据的操作）
  - **select**
  - **insert**
  - **update**
  - **delete**
  - **all privileges**（上面四个权限加起来并不等于**all privileges**）
- 权限授予(**grant**)
  - 格式：**grant** <privilege list>
    on <relation name or view name> **to** <user list>
- 权限收回(**revoke**)
  - 格式：**revoke** <privilege list>
    on <relation name or view name> **from** <user list>

五、角色

- 一个权限的集合
- 创建角色：
  - **create role** *instructor*;
- 角色被授予权限：
  - **grant select on** *takes* **to** *instructor*;
- 角色可以授予给用户，也可以授予给其他角色
- 一个用户或一个角色的权限包括：
  - 所有直接授予用户/角色的权限
  - 所有授予给用户/角色所拥有角色的权限