4 Intermediate SQL

1. Join
   1. inner join中，on, 可以用where替代，但用on来专门指定join条件，其它条件写在where中可读性更好。
   2. outer join  
      对于连接字段为空的元祖，使用inner join时不会列出，outer join却不一样，left /right join会保留左（右）侧的连接字段为空的元祖。left join的过程可以理解为先执行inner join，然后再向结果集中添加左侧关系中剩余的元祖。  
      在outer join中，on和where就不能互换了
2. 视图  
   前面学习过程中接触的表都是在数据库中实际存在的，但有时用户并不需要一张表的全部信息；甚至有时候不想让用户拿到额外的信息，比如有时只允许用户查阅instructor表的id, name；或者有一些复杂的查询我们不希望每次都去重复写一遍。这时可以使用虚拟的表(virtual relation)，即视图(view)，视图不属于数据库的表，但可以被用户访问到。
   1. 视图的定义  
      写法为create view v as <query expression>;其中query expression可以是任何查询语句。比如之前的instructor视图可以这样定义：  
      create view faculty as   
      select ID, name   
      from instructor;  
      然后只为用户提供访问faculty视图的权限，就可以达成目的。
   2. Using views in SQL Queries  
      视图被定义后，就可以像真实的表一样被使用，视图还可以嵌套视图。在使用到视图的时候，数据库会执行视图的定义语句来生成结果集，所以视图的内容是实时的。
   3. 物化视图(Materialized Views)  
      视图一般在被使用时才会实时查询出结果，这种方式虽然能保证数据是最新的，但开销却较大，在一些对计算资源有限制的场合、或者视图被频繁使用的时候、以及对查询速度有要求时，需要视图结果集能被保存起来，这种视图称为物化视图。
   4. 视图的更新  
      视图可以为查询带来方便，但视图的更新却涉及到多个问题，造成这些问题的根源是对视图的更改需要反映到真实表上。由于允许视图出现在真实表可以出现的任意位置，所以对于之前的faculty视图，就可以插入数据：  
      insert into faculty values (30765, ‘Green’);  
      但instructor.salary只能插入null了，instructor.salary规定如果不允许为空，则操作会因违反一致性而被阻止。此外再考虑视图关联多张表、视图的嵌套等情况，会使得问题变得非常复杂，所以通常对视图的更新是不被允许的。
3. 事务  
   SQL标准规定了事务形式为：包含查询、更新语句，必须用commit work或rollback work来结束且”work”可省略。  
   如果SQL语句可能出错，则可以包含在事务中，commit类似于保存编辑好的文档，而rollback相当于撤销对文档所做的修改。但是事务一旦被commit，就无法再被rollback。事务是原子性的，事务包含的操作要么全部成功，要么全部失败。如果在SQL语句执行过程中发生系统奔溃、断电等事故，只要没有执行过commit，则在系统重启后会首先回滚。  
   在许多数据库产品中，默认每条单独的SQL就是一个事务，执行结束后自动commit。
4. 一致性  
   not null  
   unique  
   check
   1. Not null  
      name varchar(20) not null
   2. Unique  
      如果A1, A2...等构成了候选键，可以用unique(A1, A2...)来保证其唯一性，但这些字段仍然可为空，而为空值与任何值都不相等。
   3. Chack  
      限制semester的值：  
      check (semester in (’Fall’, ’Winter’, ’Spring’, ’Summer’)
   4. Referential Integrity参照完整性  
      如果course.dept\_name作为外键引用department表，为了保证course.dept\_name的取值都存在于department.dept\_name，添加参照完整性约束为：
   5. foreign key (dept name) references department
5. 数据类型和Schemas
   1. Date和time  
      SQL标准规定的time相关类型有：  
      date ’2001-04-25’   
      time ’09:30:00’   
      timestamp ’2001-04-25 10:29:01.45’  
      字符串形式的日期可以使用cast e as t的方式来转换；也可以使用extract year/month/day/hour/minute/second from d的方式来单独提取年月日等数据；  
      还有current\_day, current\_timestamp(包含时区), localtimestamp(不包含时区的本地时间)；  
      interval类型表示时间的差
   2. Default value  
      create table student   
      (ID varchar (5),   
      name varchar (20) not null,   
      dept name varchar (20),   
      tot\_cred numeric (3,0) default 0,   
      primary key (ID));  
      这里设置了tot\_cred的默认值为0
   3. 创建索引  
      create index studentID index on student(ID)表示创建了名为studentID的索引，有的数据库产品又进一步区分了聚集索引(clustered)与非聚集索引(nonclustered)
   4. 大对象类型Large-Object Type  
      如果要存储声音、图像等数据，数据量可能为KB甚至MB, GB级别，为此SQL提供了两种大对象类型 clob(character large object)和blob(binary...)。不同数据库的具体实现会有区别，而且实际使用中不推荐使用这些类型，而往往将数据保存在文件系统，并在数据库保存其存放位置。
   5. 用户自定义类型  
      允许基于现有的类型来自定义数据类型，比如：  
      create type Dollars as numeric(12,2);   
      create type Pounds as numeric(12,2);  
      自定义类型Dollars和Pounds虽然都是numeric(12,2)类型，但在业务上被认为是不同的数据类型。  
      还有一种定义方式为：  
      create domain DDollars as numeric(12,2) not null;  
      type和domain的细微的区别在于domain可以同时添加约束如not null,；而且domain也不是完全的强类型，只要值是兼容的，就可以赋值给domain定义的类型，而type却不行。
   6. Create table的扩展
      1. create table temp instructor like instructor;创建了一个与sinstructor有相同结构的表
      2. 在编写SQL时，有时会创建临时表并存入数据，这时可以用简化写法：  
         create table t1 as   
         (select \*   
         from instructor   
         where dept name= ’Music’)   
         with data;  
         t1表结构与查询结果集相同，如果去掉with data，则只创建schema而不插入数据。
   7. 授权  
      权限控制可以针对用户或角色进行数据操纵、schema更新等的控制。
      1. 分配、撤销授权  
         分配权限的语法为：  
         grant <privilege list>  
         on <relation name or view name>  
         to <user/role list>;  
         privilege list包括select, insert, update, delete  
         对于update，可以设定允许更新某些属性：  
         grant update (budget) on department to Amit, Satoshi;  
         类似地，撤销授权语法为：  
         revoke <privilege list>  
         on <relation name or view name>  
         to <user/role list>;
      2. 角色  
         基于角色的权限控制不是SQL的专利，很多共享型应用都采用这种授权方式。  
         create role instructor;  
         grant select on takes to instructor;  
         grant dean to Amit;  
         前面的语句创建了角色instructor，为期分配select from takes权限，然后将Amit归入instructor角色。在Amit执行查询前，SQL根据它所属角色具有的权限来做控制。
      3. 关于schema的授权，外键  
         因为外键会影响后续的更新、删除等操作，所以有必要为外键的创建做权限控制：  
         grant references (dept name) on department to Mariano;