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。