**初级SQL（包括中级SQL中的有关内容）**

**create table** r  **创建一个表**

**primary key (A,B…)将A、B等属性设置为主码**

**foreign key (A,B…) references s 将A、B等属性设置为s的外码,后面可以加：**

**on delete cascade 表示该变量允许级联删除**

**on update cascade 表示该变量允许级联更新**

**若用set null代替cascade，则表示将删除后不满足外码约束的属性设置为null**

**create table 子表名**

**(定义子表的数据)**

**inherits 父表名**

**上述表示表之间的继承**

**create table r1 like r2 表示创建一个表r1，他具有r2相同的模式**

**create table r1 as (嵌套子查询) with data 表示利用子查询结果创建一个新表，如果去掉with data，则只会创建表，表中无元组**

**A 域 not null 将A属性设置为不可为空**

**A 域 default 值 表示当A中有值为null时，将其值设置为default后的值**

**unique(A,B…)将A、B等属性设置唯一性**

**check(p) 设置某个属性必须满足P条件，如check(**semester **in (‘**FALL’**,’**Spring’**))，其中p同样可以使用嵌套子查询**

**constraint 约束名 约束 可以将一个约束命名**

**check和constraint可以单独成一句，也可以直接跟在变量定义的后面，可以同时跟在后面(中间不需要加逗号等分隔符)**

**drop table** r **删除一个表**

**delete from** r **删除表中所有的元组，但保留表**

**alter table** r **add** A  **r表中添加属性A**

**alter table** r **drop** A **r表中删除属性A**

**alter table** r **drop constraint 约束名 从表中删除该约束**

**set constraints 约束名 deferred 将一个约束设置为可延迟的**

**delete from r where p 表示在r中删除满足p的元组**

**insert into r(A,B…) values(A,B,…)表示在r中插入values后的元组,r后面的括号可加可不加，加了就表示后面的元组按属性赋值**

**update r set A=A\*1.05 表示更新r中的属性A，更新为A\*1.05**

**update** r

**set** A **= case**

**when p1 then result1**

**when p2 then result2**

**…**

**else result0**

**end**

**update中的分支结构，可以用于分支更新**

**create assertion 断言名 check p 设置一个断言约束，表示数据库必须时刻保持该约束**

**distinct** A **用于select中，表示去除重复 all显示的表示可以重复**

**as 比较灵活，重命名一个属性或关系，可以同时对一个关系和该关系中的属性重命名**

**like 用于字符串匹配**

**escape 设置转义符**

**order by 查询结果按某个属性排序**

**desc 降序 asc 升序 同时使用时，以先声明的为准，后一个为相同时的排序规则**

**union 并运算，可以放在两个查询结果之间 union自动去除重复，想重复可以用all**

**intersect 交运算**

**except 差运算**

**true>unknown>false not unknown = unknow 真值与空值的比较结果时unknown**

**avg (A) 计算A属性的平均值**

**count (A) 计算一个表或属性中元素的个数**

**其余的还有min max sum**

**group by A 与聚集函数搭配使用，表示利用某个属性来聚集结果**

**若使用了group by 则select中选择的属性必须时group by使用的，(聚集函数属性除外)**

**having 聚集函数(A) 表示利用聚集函数来作为筛选条件，与where的区别是，where发生在聚集前，而having发生在聚集后，where单独作用与from**

**in 和not in，多用于嵌套查询，也可以后面直接跟一个表的属性**

**some 表示“至少一个”，例如where A > some (B)表示A至少比B中的一个大**

**all 表示“所有”**

**exists 嵌套查询 表示判断嵌套查询结果是否存在，前可加not**

**unique A或r 表示判断后面的属性或元组有无重复的元素，前可加not，区别于创建表时定义的约束unique**

**lateral 嵌套查询 表示让from中的子查询可以使用外围的重命名，用于解除select和group by的限制(用另一种方式实现)**

**with r(A,B,…) as 嵌套查询 在select前使用，表示定义一种临时关系，通过逗号分隔可以定义多个临时关系**

**标量子查询：如果一个子查询返回的是单个值，则它可以出现在任何地方**

**中级SQL**

**r1 natural join r2 自然连接，即重名属性直接去除，不重名直接补充在后面，以重名的所有属性的组合为主码进行合并。用于from句**

**ri natural join r2 using(A) 指定本次自然连接相等属性为A(避免属性之间的同名不同义)**

**on p 出现位置于using一样，用于表示连接条件p**

**left outer join 左外连接，保留左侧关系中多出来的元组**

**right outer join 右外连接**

**full outer join 全外连接**

**on与where on针对于外连接，会补充多出来的元组，但where会去除，因为p不可能为真**

**join = inner join = natural join = natural inner join**

**create vier** v(A,B,…) **as <查询表达式> 创建一个视图,其中属性名可以不指定**

**cast(A as 域) 表示将选择的A属性转化为指定域，多用于select语句中**

**coalesce(A,值) 多用于select中，表示将查找A结果中的null设置为指定的值**

**decode (A,要更改的值,更改后的值,A) 多用于select中，表示将查找A属性的结果中原来要更改的值变为更改后的值**

**以上对于A属性的查找，都可以在后面加as设定成一个新的属性名**

**与日期有关的数据类型：**

**date ‘2018-04-25’**

**time ‘09:30:00’**

**timestamp ’2018-04-25 10:29:01.45’**

**extract(**field **from** d**) 表示从域field中提取出d单独的域**

**一些关于日期的函数：**

**current\_date/time/timestamp 返回当前日期/时间/时间戳(带时区)**

**locatetime 返回当前的本地时间(不带时区)，其余可以自行组合**

**大数据类型：**

**A clob(大小，如10KB) 定义属性A为指定大小的字符数据**

**A blob(大小) 定义属性A为指定大小的二进制数据**

**自定义数据类型：**

**create type 自定义类型名 as 域 final 表示自定义一种变量类型（注意不是域）**

**drop/alter type 自定义类型名 自定义类型名更改或删除**

**create domain 自定义域名 as 域 域约束 自定义一种域**

**域和变量类型的区别：域可以设置约束或缺省值，但变量类型不可以**

**不同类型变量无法相互赋值，但自定义域只要基本类型相同，就可以赋值**

**面向对象的自定义数据类型：**

**create type自定义类型名**

**(定义类中的数据，其中需要定义一个主码，在主码数据定义后加一个primary key);**

**使用create table 对象名 of 自定义类型名 来创建一个对象**

**可以在定义时加一句 ref form(A) 表示该属性可以被其他类型或表引用做外码**

**则在定义其他表时，不需要像create table一样定义和约束分开，而是直接这么写：**

**A ref(要引用的自定义类型名) scope (自定义类型名定义的对象)**

**create 继承类名 under 父类名**

**(定义一些子类的特殊数据);**

**表示定义一个继承类**

**create table 对象名 of 自定义类名**

**under 父表名**

**(定义);**

**创建一个表对象，该对象是继承于指定父表的自定义类**

**A 域 generated always as identity 表示属性A的值是系统自动指定的一个唯一值**

**此时最好避免insert时指定了A的值，**

**by default替换always 表示可以选择指定自己挑选的值或系统自动生成的值**

**create (unique) 索引名 on r(A,B…) 表示在r的AB等属性上创建一个索引，加unique表示后面的属性一定时候选码，否则会报错**

**drop 索引名 删除一个索引**

**grant <权限列表，如select、insert等>**

**on <关系或视图>**

**to <用户/角色列表>**

**表示为一些用户对开放对指定关系或视图的指定权限，若权限列表有update权限，则需要在后面加括号，里面为一些属性名，表示对这些属性有更新权限，省略则说明对所有属性都有权限**

**用public表示用户/角色列表，表示是对所有的现有和将来的用户都增加权限**

**用revoke代替grant 表示收取权限**

**revoke grant option for <权限列表> 后同 表示收取授权的权限**

**在最后加 with grant option 表示用户可以把这次得到的权限授权给其他用户**

**在revoke 的最后加restrict 表示防止级联收权**

**create role 角色名 表示创建一个角色**

**grant 角色1 to 角色2 表示将角色1的权限授予角色2**

**对视图的授权：用于解决这种类型的问题，该角色只能看到历史系老师的工资**

**grant references (A) on r to 角色名 表示该用户在使用r创建表时，可以用A作为外码**

**其他的大胆猜测不考(后接p119行级授权)**

**高级SQL**

**函数：**

**create function 函数名(参数1 参数类型(域), 参数2 ………)**

**returns 域**

**begin**

**declare 要返回的值名 域;(此处有分号)**

**函数体**

**return 要返回的值名;(此处有分号)**

**end**

**其中在函数体里给返回值赋值不用as而是用into，我也不知道为啥**

**返回值也可以自定义一个table，这样可以直接在return中使用嵌套子查询**

**过程：**

**create procedure 过程名(in 参数，参数类型，out 参数，参数类型)**

**begin**

**过程体**

**end**

**表示创建一个过程，in和out表示要输入的参数和可以输出的参数(和c++指针参数有点像)**

**call 过程名(需要的参数) 可以调用定义的过程**

**循环：**

**while 布尔表达式 do**

**语句序列;（有分号）**

**end while**

**repeat**

**语句序列;（有分号）**

**until 布尔表达式**

**end repeat**

**for r as**

**语句序列(一般是一个子查询，查出来一个单独的属性，好让r遍历)**

**do**

**语句序列(例如set n = n-r.budget)**

**end for**

**分支：**

**if 布尔表达式**

**then 语句或复合语句**

**elseif 布尔表达式**

**then 语句或复合语句**

**else 语句或复合语句**

**end if**

**异常：**

**declare 异常名 condition**

**declare exit handler for 异常名**

**begin**

**语句序列;(分号)**

**end**

**定义一个异常，表示出发异常后，采取以上语句序列**

**可以在程序中加上signal 异常名 来触发这个异常**

**其他的大胆假设不考(后接P141触发器)**

**第七章**

**函数依赖：**

**自反：若b∈a，则a->b成立(特异性高的能推出来特异性低的)**

**增补：若a->b，且y为一个属性集，则ay->by**

**传递：略**

**合并：若a->b且a->y成立，则a->by**

**分解：若a->by成立，则a->b,a->y**

**伪传递：若a->b且by->d,则ay->d**

**BCNF:**

**·a->b 是一个平凡的函数依赖**

**·a是R的超码**

**3NF(a->b b-a中的所有属性都被某个R的候选码包含)**

**无关属性：考虑一个a->b**

**从左侧移除：如果A∈a,并且F逻辑蕴含(F-{a->b})∪{(a-A)->b},则A是无关属性**

**从右侧移除：如果A∈b，并且(F-{a->b})∪{a->(b-A)}逻辑蕴含F,则A是无关属性**

**因为从左侧移除，函数会变强，右移除会变弱，如果弱的能蕴含强的，说明是无关属性**

**正则覆盖：**

**对于一个函数集F，F的正则覆盖Fc满足：**

**·Fc中的任何函数依赖都不包括无关属性**

**·Fc中的每个函数依赖的左侧都是唯一的**

**BCNF无损分解流程：**

**1、计算F的正则覆盖Fc**

**2、检查每一个分解出的子表的每一个函数，如果有一个子表R不满足函数依赖(假设为a->b)，则用以下两个模式去取代R：**

**·(a∪b)**

**·(R-(b-a))**

**3、重复2步骤，直到所有函数都满足BCNF**

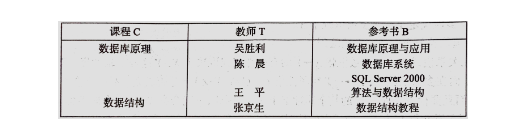
**检查分解出来的所有函数集的并集是否逻辑蕴含原来的Fc，若蕴含，则是保持依赖的分解**

**3NF的保持依赖分解流程：(注意3NF是可以保持依赖的，但BCNF不一定可以)**

1. **首先计算出F的正则覆盖Fc**
2. **对于Fc中的每一个依赖a->b,都创建出一个Ri=ab**
3. **如果没有一个Ri包含R的一个候选码，则加一个Rj，该Rj的属性是R的一个候选码**
4. **遍历所有的Ri，如果有一个Ri被另一个Rj包含，则删除Ri**
5. **直到没有Ri被其他Rj包含，得出3NF分解**

**多值依赖：**

**考虑R上的三个属性集abc(a∪b∪c = R)，如果a确定下来(一个)，对于不同的b(多个)总能确定出一个相同的c(多个，数量与b的无关),则说明a->->c，类似于下图关系**



**说明C->->B，C->->T**

**·任何函数依赖也是多值依赖**

**·若a->->b,则a->->R-a-b**

**第四范式：**

**·a->->b是一个平凡的函数依赖**

**·a是R的超码**

**4NF分解：**

**考虑一个D+，D+是F+中多加了所有的多值依赖，他在Ri上的限定Di含义为：**

**·只含Ri中的所有函数依赖**

**·所有形如a->->b∩Ri 的多值依赖(意思是后面的也在Ri内部)，其中a∈Ri且a->->b属于D+，相当于原本的Fi中多了Ri中的所有的多值依赖**

**分解步骤：**

1. **计算每个Ri的Di**
2. **如果有一个Ri不属于4NF，则找出一个a->->b为Ri上成立的一个非平凡多值依赖(a∩b为空集)，使a->Ri不再属于Di**

**也就是说，将Ri转化为(Ri-b)和(a,b)(注意，BCNF是Ri-(b-a))**

1. **重复2，直到所有Ri都属于4NF**

**支持度：a->b的支持度为ab同时发生占所有行为的比例**

**置信度：a->b的支持度为有a行为后，也有b行为占总a行为的比例**