数据库

0. 概论

0.1 关系规则

1. 第一范式规则: 不允许多值属性
2. 第二范式规则: 只能基于内容存取行
3. 第三范式规则: 行唯一

0.2 键

超键(super key): 在关系中能唯一标识元组的属性集称为关系模式的超键

候选键(candidate key): 不含有多余属性的超键称为候选键. 也就是在候选键中, 若再删除属性, 就不是键了.

主键(primary key): 用户选作元组标识的一个候选键程序主键

外键(foreign key): 如果关系模式R中属性K是其它模式的主键, 那么k在模式R中称为外键.

例如:

学生信息 (学号 身份证号 性别 年龄 身高 体重 宿舍号) 和 宿舍信息 (宿舍号 楼号)

超键: 只要含有 "学号" 或者 "身份证号" 两个属性的集合就叫超键, 例如R1 (学号 性别) , R2 (身份证号 身高) , R3 (学号 身份证号) 等等都可以称为超键.

候选键: 不含有多余的属性的超键, 也简称为键, 比如 (学号) , (身份证号) 都是候选键, 又比如R1中学号这一个属性就可以唯一标识元组了, 而有没有性别这一属性对是否唯一标识元组没有任何的影响.

主键: 就是用户从很多候选键选出来的一个键就是主键, 比如你要求学号是主键, 那么身份证号就不可以是主键了.

外键: 宿舍号就是学生信息表的外键.

1. 关系代数

1.1 基本运算

并 (∪) , 差 (-) , 笛卡尔积 (×) , 投影 (π) 和选择 (σ)

其他运算 (交, 自然连接, θ 连接, 除) 都可以通过这五个基本运算表示出来.

1.2 运算具体表示

投影运算的表示方法: CN := CUSTOMERS[cname]

选择运算的表示方法: L := AGENTS where percent >= 6

连接运算的表示方法: T := R ⋈ S

θ 连接就像笛卡尔积加选择.

还可以进行重命名: R(cno) := P[pno]

除法:

第一步: 因为 R÷S 所得到的属性值是包含于 R, 但是 S 不包含的属性, 所以 R÷S 得到的属性列有 (A, B), S在 (C,D) 属性上的投影为 {(c1,d1),(c2,d2)}
第二步: 关系R中, AB属性可以取值为= ｛(a1,b1), (a2,b2), (a3,b3)｝
第三步: 求象集
(a1,b1) = {(c1,d1),(c2,d2),(c3,d3)}
(a2,b2) = {(c2,d2)}
(a3,b3) = {(c1,d1),(c2,d2)}
第四步: 从第三步中可以发现, 象集 (a1,b1) 和 (a3,b3) 包含了 S 在 (C,D) 属性上的投影, 所以 R÷S= ｛(a1,b1) , (a3,b3)

正确使用案例: (C ⋈ O)[aid, city] ÷ C[city]

2. SQL 语言

2.1 SELECT 语句

基本结构:

```
select * from customers;
```

```
select pid from orders;
```

distinct 关键字:

```
select distinct pid from orders;
```

笛卡尔积, 别名与选择:

```
select distinct c.cname, a.aname
    from customers as c, orders as o, agents as a
    where c.cid = o.cid and o.aid = a.aid;
```

通配符:

- '%' : 替代 0 个或多个字符
- '_' : 替代一个字符
- '[char]' : 字符列中的任何单一字符
- '[^char]' 或 [!char] : 不在字符列中的任何单一字符

```
SELECT * FROM Websites
WHERE url LIKE 'http%';
```

子查询与 in 谓词:

检索符合下述条件的顾客的编号: 在2021年没有购买过商品;

子查询能够从外层中接受数据.

```
select cid from customers
    where cid not in
        (select cid from orders where year(orddate) = '2021');
```

量化比较谓词:

检索享有最高销售提成比例的供应商 (请分别写出使用统计函数和不使用统计函数的两种不同表示方法);

```
select aname from Agent
    where per == (select max(per) from Agent);
```

```
select aname from Agent
    where per >=all (select per from Agent);
```

集合是否为空:

检索符合下述条件的商品的编号: 在所有有客户的城市中都被销售过;

```
select pid from Product p
    where not exists (select city from Customer c
        where city not in (select city from Order o
            where p.pid == o.pid));
```

集合函数:

包含 `count`，`sum`，`avg`，`max` 和 `min`.

```
select sum(dollars) as total_dollars from orders;
```

空值:

```
select * from customers where discount is null;
```

分组:

具体步骤为

1. 对 from 子句进行笛卡尔积.
2. 删除不满足 where 的行.
3. 根据 group by 进行分组.
4. 求出选择列表中的表达式的值.

```
select pid, sum(qty) as total from orders
    group by pid;
```

having 语句:

检索只购买过一次商品的顾客的编号 (请写出三种不同类型的查询语句);

```
select o1.cid from Order o1
    where o1.cid not in (select o2.cid from Order o2, Order o3
        where o2.cid == o3.cid and o2.ordno < o3.ordno);


select cid from Customer c
    where 1 == (select count(ordno) from Order o
        where c.cid == o.cid);


select cid from Order
    group by cid
    having count(ordno) == 1;
```

排序语句:

```
select a.aid, a.aname, year(o.ordno) as yea, sum(o.dols) as casales, count(o.ordno) as
    from Agent a, Order o
    where o.aid == a.aid
    group by a.aid, yea
    order by yea asc, casales desc;
```

2.2 其他语句

插入语句:

```
insert into orders (ordno, cid, aid)
    values (1107, 'c006', 'a04')
```

更新语句:

```
update customers set discount = 1.1 * discount where cid = 'c006';
```

删除语句:

```
delete from agents where city = 'New York';
```

3. 数据库设计

3.1 ER 模型

ER图分为实体, 属性, 关系三个核心部分. 实体是长方形体现, 而属性则是椭圆形, 关系为菱形.

ER图的实体 (entity) 即数据模型中的数据对象, 例如人, 学生, 音乐都可以作为一个数据对象, 用长方体来表示, 每个实体都有自己的实体成员 (entity member) 或者说实体对象 (entity instance) , 例如学生实体里包括张三, 李四等, 实体成员 (entity member) / 实体实例 (entity instance) 不需要出现在ER图中.

ER图的属性 (attribute) 即数据对象所具有的属性, 例如学生具有姓名, 学号, 年级等属性, 用椭圆形表示, 属性分为唯一属性 (unique attribute) 和非唯一属性, 唯一属性指的是唯一可用来标识该实体实例或者成员的属性, 用下划线表示, 一般来讲实体都至少有一个唯一属性.

ER图的关系 (relationship) 用来表现数据对象与数据对象之间的联系, 例如学生的实体和成绩表的实体之间有一定的联系, 每个学生都有自己的成绩表, 这就是一种关系, 关系用菱形来表示.

ER图中关联关系有三种:

1对1 (1:1): 1对1关系是指对于实体集A与实体集B, A中的每一个实体至多与B中一个实体有关系；反之, 在实体集B中的每个实体至多与实体集A中一个实体有关系.

1对多 (1:N): 1对多关系是指实体集A与实体集B中至少有N(N>0)个实体有关系；并且实体集B中每一个实体至多与实体集A中一个实体有关系.

多对多 (M:N): 多对多关系是指实体集A中的每一个实体与实体集B中至少有M(M>0)个实体有关系, 并且实体集B中的每一个实体与实体集A中的至少N (N>0) 个实体有关系.

3.2 规范化

函数依赖:

X -> Y, 常被读作 "X 函数决定 Y".

如果 Y 是 X 的子集, 则称 X -> Y 是一个平凡函数依赖.

如果 Y 不是 X 的子集, 则称 X -> Y 是一个非平凡函数依赖.

如果我们有另一个函数依赖 W -> Y 和 W ⊆ X, 则称 X -> Y 为部分函数依赖.

如果我们没有另一个函数依赖 W -> Y 和 W ⊆ X, 则称 X -> Y 为完全函数依赖.

函数依赖的闭包: F+ = { 根据 F 中已有函数依赖, 使用 Armstrong 公理可以推导得到的所有依赖 }

函数依赖的覆盖: 如果 G ⊆ F+, 则称 F 覆盖 G.

函数依赖等价: 如果 F 和 G 相互覆盖, 则称等价.

属性集的闭包: X_F+ = { A | X -> A 属于 F+ }

最小覆盖: M 覆盖 F, 没有冗余的函数依赖, 并且函数依赖左边没有多余的属性, 都是完全函数依赖.

无损分解: 指将一个关系模式分解成若干个关系模式后, 通过自然连接和投影等运算依然能够还原到原来的关系模式.

无损分解 ⇔ R1∩R2 → (R1-R2) 或 R1∩R2 → (R2-R1)

保持函数依赖 ⇔ (F1∪F2)+ = F+

最小覆盖求法:

1. 先将函数依赖右侧拆分为单项.
2. 一条一条依次去除冗余函数依赖, 直到不能去除.
3. 将函数依赖左侧一个一个地去除冗余, 直到不能去除.
4. 最后再将其合并.

Armstrong 公理:

基本规则: 自反规则, 传递规则, 增广规则.

扩充规则: 合并规则, 分解规则.

1. 自反律: 若 Y⊆X⊆U, 则 X→Y 为 F 所蕴含;
2. 增广律: 若 X→Y 为 F 所蕴含, 且 Z⊆U, 则 XZ→YZ 为 F 所蕴含;
3. 传递律: 若 X→Y, Y→Z 为 F 所蕴含, 则 X→Z 为 F 所蕴含.

根据上面三条推理规则, 又可推出下面三条推理规则:

4. 合并规则: 若 X→Y, X→Z, 则 X→YZ 为 F 所蕴含;
5. 伪传递规则: 若 X→Y, WY→Z, 则 XW→Z 为 F 所蕴含;
6. 分解规则: 若 X→Y, Z⊆Y, 则 X→Z 为 F 所蕴含.

3.3 基础概念

实体: 现实世界中客观存在并可以被区别的事物. 比如 "一个学生" , "一本书" , "一门课" 等等. 值得强调的是这里所说的 "事物" 不仅仅是看得见摸得着的 "东西" , 它也可以是虚拟的, 不如说 "老师与学校的关系" .

属性: 教科书上解释为: "实体所具有的某一特性" , 由此可见, 属性一开始是个逻辑概念, 比如说, "性别" 是 "人" 的一个属性. 在关系数据库中, 属性又是个物理概念, 属性可以看作是 "表的一列" .

元组: 表中的一行就是一个元组.

分量: 元组的某个属性值. 在一个关系数据库中, 它是一个操作原子, 即关系数据库在做任何操作的时候, 属性是 "不可分的" . 否则就不是关系数据库了.

键: 表中可以唯一确定一个元组的某个属性 (或者属性组), 如果这样的键有不止一个, 那么大家都叫候选键, 我们从候选键中挑一个出来做主要使用的键, 它就叫主键.

全键: 如果一个键包含了所有的属性, 这个键就是全键.

主属性: 一个属性只要在任何一个候选键中出现过, 这个属性就是主属性.

非主属性: 与上面相反, 没有在任何候选键中出现过, 这个属性就是非主属性.

外键: 一个属性 (或属性组) , 它不是键, 但是它别的表的键, 它就是外键.

3.4 范式

第一范式 (1NF): 属性不可分. 只要是关系数据库就是第一范式.

第二范式 (2NF): 符合 1NF, 并且, 非主属性完全依赖于键. 对于 X -> A, A 是一个不在 X 中的单非主属性, 则 X 不能真包含于任何键中.

第三范式 (3NF): 符合 2NF, 并且, 消除传递依赖. 对于 X -> A, A 是一个不在 X 中的单非主属性, 则 X 必须是一个超键.

BC范式 (BCNF): 符合 3NF, 并且, 主属性不依赖于主属性. 对于 X -> A, A 是一个不在 X 中的单属性, 则 X 必须是一个超键.

3NF 分解:

1. 先求出最小覆盖集 Fc
2. 对于 Fc 里面的所有函数依赖 a->b, 均转化为 Ri=ab
3. 对于所有的模式 Ri
    1. 如果包含候选键, 进行第 4
    2. 如果都不包含候选键, 将任意一个候选键添加到模式 Ri 里面
4. 如果一个模式被另一个模式包含, 则去掉此被包含的模式.

BCNF 分解:

1. 求出候选键
2. 观察函数依赖集, 如果左边不是超键 (候选键), 则不满足条件
3. 用不满足条件的函数依赖 (A->B) 进行分解, 这样分解之后就满足了
    1. R1=AB (这样就满足了)
    2. R2=(R-R1)∪A
    3. F2={…} 去掉 B 的所有函数依赖, 尽可能写全
4. 对 F2 进行步骤 1 的计算。
5. 重复直到所有的满足条件

4. 视图

4.1 数据完整性约束

- NOT NULL, DEFAULT, CHECK
- Primary Key & Unique
- Foreign Key

4.2 视图语法

```
create view v1(cno, stud_count) as
    select cno, count(*) from Study group by cno;
```

4.3 安全性

授权: GRANT ... TO ...

权限回收: REVOKE ... FROM ...

5. 事务

5.1 事务定义

数据库事务 (transaction) 是访问并可能操作各种数据项的一个数据库操作序列, 这些操作要么全部执行,要么全部不执行, 是一个不可分割的工作单位. 事务由事务开始与事务结束之间执行的全部数据库操作组成.

5.2 事务的 ACID 特性

Atomicity (原子性): 一个事务 (transaction) 中的所有操作, 要么全部完成, 要么全部不完成, 不会结束在中间某个环节. 事务在执行过程中发生错误, 会被恢复 (Rollback) 到事务开始前的状态, 就像这个事务从来没有执行过一样.

Consistency (一致性): 在事务开始之前和事务结束以后, 数据库的完整性没有被破坏. 这表示写入的资料必须完全符合所有的预设规则, 这包含资料的精确度, 串联性以及后续数据库可以自发性地完成预定的工作.

Isolation (隔离性): 数据库允许多个并发事务同时对其数据进行读写和修改的能力, 隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致. 事务隔离分为不同级别, 包括读未提交 (Read uncommitted), 读提交 (read committed) , 可重复读 (repeatable read) 和串行化 (Serializable).

Durability (持久性): 事务处理结束后, 对数据的修改就是永久的, 即便系统故障也不会丢失

5.3 封锁

排它锁又称为写锁. 若事务T对数据对象A加上X锁, 则只允许T读取和修改A, 其它任何事务都不能再对A加任何类型的锁, 直到T释放A上的锁. 这就保证了其它事务在T释放A上的锁之前不能再读取和修改A.

共享锁又称为读锁. 若事务T对数据对象A加上S锁, 则事务T可以读A但不能修改A, 其它事务只能再对A加S锁, 而不能加X锁, 直到T释放A上的S锁. 这就保证了其它事务可以读A, 但在T释放A上的S锁之前不能对A做任何修改.

5.4 冲突可串行化调度

1. 冲突操作指的是不同事务对于同一数据的读写操作与写写操作.
2. 有些冲突操作是可以交换次序的, 有些冲突操作不能交换次序.
3. 不能交换位置的次序为：(1) 不同事务的冲突操作. (2) 同一事务的两个操作.
4. 一个调度 Sc 在保证冲突操作 (即 3 中的两种情况) 次序不变的情况下, 通过交换两个事务不冲突操作的次序得到另一个调度Sc', 如果Sc'是串行的, 则称调度Sc为冲突可串行化的调度.

只需要关注每一个操作对象, 将 RW 和 WW 的情况用箭头将对应事务连接起来, 最后看看有没有环, 无环就是冲突可串行化调度.