CH5 Relational Database Design Theory Part2关系数据库设计理论2

一、Relation Decomposition关系分解

1. Lossless Join无损连接

将关系R分解之后得到的关系仍然可以恢复成R，称该分解有一个*无损连接*

- 自然连接是可结合和可交换的
- R中的任意一条元组t，一定满足$t \in \pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \pi_{S_k}(R)$
- $\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \pi_{S_k}(R) = R$

```
def structTable(R, ds):
    # 构造一个分解数 * 属性数的表
    # 每一行为当前投影所拥有的属性
    # 当前投影没有的属性使用行下标表示
    res = np.zeros_like((ds.shape[0], R.shape[1]))
    for i in len(ds):
        for e in ds[i]:
            res[i][e] = e
        for oe in other element:
            res[i][oe] = os_i
    return res
def IsLosslessJoin(R, ds, fd):
    table = structTable(R, ds)      # 根据分解构造R的关系
    for f in fd:
        table.apply(f)              # 使用函数依赖改变表中的下标
    for row in table.rows:
        if(every element in row is unsubscripted):  # 存在一行，其中所有元素都没有下标
            return True
    return False
```

2. Dependency Preservation保持依赖

无法将关系分解为同时具有*无损连接*和*保持依赖*的关系

```
def DependencyPreservation(R, dR, fds):
    res = None
    for r in dR:
        res = res Union fd_projection(R, r, fds)
    return fds is_subset(res)
```

保持依赖性：

任意原有关系的函数依赖在至少一个分解中成立。

二、Third Norm Form

1. 1NF & 2NF

第一范式：原子性（存储数据不可再分）

第二范式：唯一性（不存在非主键中元素部分依赖联合主键中的部分字段）

e.g.

$AB \to CD$，AB为主键，但是也存在$B \to C$，则该关系不为第二范式。

唯一性，即每一行数据具有唯一可区分的特性。

第二范式要求非主键部分必须完全依赖主键。

2. Definition

主属性：为某个键中的一员

第三范式：独立性（消除传递依赖）

非主键值不依赖于另一个非主键值。

第三范式要求数据的每一列都与主键直接相关，而不是简介相关。

一个关系满足第三范式*当*：每个非平凡的函数依赖，要么左边是超键，要么右边只存在主属性。

$\underline{a} \to b, b \to c$：出现传递依赖

$\underline{ac} \to b, b \to \underline{c}$：不属于传递依赖，因为$ac \to c$为平凡依赖。

当且仅当：（$X \to Y$）是关系中的任意函数依赖

> $X \to Y$为平凡依赖
>
> X为超键
>
> $Y - X$为主属性

3. Abstract

key, whole key, nothing but the key

4. Synthesis Algorithm for 3NF

将关系R分解为具有以下性质的关系：

> 分解结果满足第三范式
>
> 分解可以无损连接
>
> 保持依赖

```
def DeposFor3NF(R, F):
    G = MinimalBasis(R, F)
    Rs = None
    for fd(X, A) in G:
        Rs.insert(R(X, A))
    if NotExistSuperkey(Rs):
        Rs.insert(any key)
    return Rs
```

三、Multivalued Dependencies

To be continued.....