

Database - P23

Armstrong 公理系统 (Axiomatization) 是有效 (sound) 的且完备 (complete) 的。
对于定义在关系 R 的属性集合 U 上的函数依赖集合 F , 有闭包 F^+
有效性指: 由 F 根据 Armstrong 公理而蕴含的每个函数依赖均在闭包 F^+ 中
完备性指: 闭包 F^+ 中的每一个函数依赖, 皆可由 F 根据 Armstrong 公理导出

对于公理系统的有效性, 可由三条推理规则的正确性推导

对于公理系统的完备性证明, 可转换为对某逆否命题的证明

即如果函数依赖 $X \rightarrow Y$ 不可由 F 从公理系统导出, 则必然不为 F 所蕴含, 即不在 F^+ 中

(1) 对于属性集合 V, W , 如果有 $V \rightarrow W$ 且 $V \subseteq X_F^+$, 则有 $W \subseteq X_F^+$

由于 $V \subseteq X_F^+$, 则有 $X \rightarrow V$, 且 $V \rightarrow W$

则根据传递规则, 有 $X \rightarrow W$,

于是根据 X_F^+ 定义, 有 $W \subseteq X_F^+ \cap \underbrace{X_F^+}_{U \setminus X_F^+}$

(2) 构造由两个元组组成的关系 r , 有

则 r 为 $R(u)$ 的实例, 且 F 在 r 上成立

由于关系 R 的属性集合等于 U , 则 r 为 $R(u)$ 的实例

如果 F 在 r 上不成立, 则存在 F^+ 中的 FD: $V \rightarrow W$, 且 r 不满足 $V \rightarrow W$

由 r 的构成可知, $V \subseteq X_F^+$, 而 $W \not\subseteq X_F^+$

又 $W \subseteq U$, 则有 W 的子集 W' , 使得 $W' \subseteq U \setminus X_F^+$

又根据 $V \subseteq X_F^+$ 且 $V \rightarrow W$, 则有 $W \subseteq X_F^+$, 从而产生矛盾

于是关系 r 必定是 $R(u, F)$ 的一个实例

(3) 如果 $X \rightarrow Y$ 不可由 F 从公理系统导出, 其中 $X, Y \subseteq U$

则有 Y 不是 X_F^+ 的子集, 即存在 Y 的子集 Y' , 使得 $Y' \subseteq U \setminus X_F^+$

于是 $X \rightarrow Y$ 在 r 中必定不成立, 即关系 r 不满足 $X \rightarrow Y$

于是有 $X \rightarrow Y$ 必然不为 F 所蕴含, 即不在 F^+ 中

覆盖

(cover), 对于定义在关系 $R(u)$ 上的函数依赖集合 F 和 G

如果对于任意 $FD \in G$, 有 $F \vdash FD$, 则称 F 覆盖 G

等价地有, 如果有 $G^+ \subseteq F^+$, 则同样有 F 覆盖 G

等价

(equivalence), 对于定义在关系 $R(u)$ 上的函数依赖集合 F 和 G

如果有 $F^+ = G^+$, 则称 F 和 G 是等价的 (equivalent), 记为 $F \equiv G$

等价地有, $F^+ = G^+ \leftrightarrow F \subseteq G^+ \wedge G \subseteq F^+$

当 F 和 G 是等价的, 则有 F 覆盖 G 且 G 覆盖 F

Database - P24

2) - 2.3.1.2

非冗余覆盖 (non-redundant cover)

对于函数依赖 F , 如果不存在 F 的真子集 $F' \subset F$ 使得 $F' \equiv F$, 则称函数依赖是 非冗余的 (non-redundant)

使得 $F' \subsetneq F$, 则称函数依赖是 冗余的 (redundant)

否则如果这样的 F' 存在, 则称 F 是冗余的 (redundant)

对于函数依赖 F 和 G , 如果有 F 覆盖 G 且有 F 是非冗余的, 则称 F 是 G 的非冗余覆盖

且有 F 是非冗余的, 则称 F 是 G 的非冗余覆盖

对于非冗余的另一种方法是, 对于函数依赖 F

如果不存在 $FD: X \rightarrow Y \in F$, 使得 $F \setminus \{X \rightarrow Y\} \vdash X \rightarrow Y$

则称函数依赖集 F 是非冗余的

极小函数依赖集 (irreducible function depending set), 指对于函数依赖集 F

有 (1) F 中任一函数依赖的右部仅含有一个属性

each right set of a function dependency of F

contains only one attribute

(2) F 中不存在函数依赖 $X \rightarrow A$, 使得 $F \equiv F \setminus \{X \rightarrow A\}$

reducing any function dependency will change the content of F

(3) F 中不存在函数依赖 $X \rightarrow A$,

X 有真子集 $Z \subset X$, 使得 $F \equiv F \setminus \{X \rightarrow A\} \cup \{Z \rightarrow A\}$

each left set of a function dependency of F is irreducible

reducing any attribute from left set will change the content of F

极小函数依赖集也称 最小依赖集 (canonical, minimal)

每一个函数依赖集 F , 均等价于一个极小函数依赖集 F_m , F_m 称为 F 的最小依赖集

构造性地证明有, 对于函数依赖集 F 进行“极小化处理”

(1) 对于 F 中的每一个函数依赖 $FD_i: X \rightarrow Y$,

如果 $Y = A_1, A_2 \dots A_k$, $k \geq 2$, 则用 $\{X \rightarrow A_j | j=1, \dots, k\}$ 代替 F 中的 $X \rightarrow Y$

(2) 对于 F 中的每一个函数依赖 $FD_i: X \rightarrow A$, 令 $G = F \setminus \{X \rightarrow A\}$

如果 $A \in X_g^+$, 则从 F 中去掉 $X \rightarrow A$

(3) 对于 F 中的每一个函数依赖 $FD_i: X \rightarrow A$, 如果有 $X = B_1, B_2 \dots B_m$, $m \geq 2$

则逐一检查 B_j , $j=1, \dots, m$, 如果 $A \in (X \setminus B_j)_F^+$, 则以 $(X \setminus B_j)$ 替换 X

在以上步骤完成后, 最终的 F 是最小依赖集, 且与 F 等价

注意: 由于 FD_i 和 B_j 的相对顺序, 最终形成的最小依赖集 F_m 不是唯一的

Database - P25

对于关系模式 $R(u, F)$, 其中 U 为关系 R 的属性集合, F 是 R 满足的函数依赖集合
令 U' 为属性集合 U 的子集, 即 $U' \subseteq U$
则有函数依赖集合 $\{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq U'\}$ 的一个覆盖 F'
称 F' 为 F 在属性集合 U' 上的投影

关系模式的分解, 指对于关系模式 $R(u, F)$, 记 P 为 $R(u, F)$ 的一个分解
有 $P = \{R_1(u_1, F_1), R_2(u_2, F_2), \dots, R_k(u_k, F_k)\}$
其中 $U_i \cup U_j = U$, 且对任意 $1 \leq i, j \leq k$, $U_i \neq U_j$
 F_i 为函数依赖集合 F 在属性集合 U_i 上的投影

令 r 为关系模式 $R(u, F)$ 的一个实例

定义 $\pi_{R_i}(r)$ 为 $\pi_{R_i}(r) = \{\pi_{U_i} t \mid t \in r\}$, 其中 $i = 1, 2, \dots, k$

定义 $m_p(r) = \bowtie_{i=1}^k \pi_{R_i}(r)$, 即 r 在分解 P 中各关系模式上投影的连接

于是有 (1) $r \subseteq m_p(r)$

(2) 令 $s = m_p(r)$, 则有 $\pi_{R_i}(s) = r_i$, 其中 $r_i = \pi_{R_i}(r)$

(3) $m_p(m_p(r)) = m_p(r)$

无损连接性 (lossless join), 指对于关系模式 $R(u, F)$ 的一个分解 $P = \{R_1(u_1, F_1), \dots, R_k(u_k, F_k)\}$
如果对于 R 的任意一个关系实例 r , 都有 $r = m_p(r)$
则称分解 P 具有无损连接性, 或称 P 为无损分解

保持函数依赖 (preserve functional dependency), 指对于一个分解 $P = \{R_1(u_1, F_1), \dots, R_k(u_k, F_k)\}$
如果有 $(\cup_{i=1}^k F_i)^+ = F^+$, 则称分解 P 保持函数依赖

希恩定理 (Heath's theorem), 指对于关系 R , R 有属性集合 U ,

如果关系 R 满足函数依赖 $X \rightarrow Y$, 另有属性集合 $U - XY = Z$

则关系 R 可以安全地分解 (split) 为两个更小的关系

即有 $\pi_{XY}(R) \bowtie \pi_{XZ}(R) = R$ 即有 $P = \{R_1(XY, F_{XY}), R_2(XZ, F_{XZ})\}$

可知分解 P 为无损连接分解 (lossless-join decomposition)

即保证在连接回去时, 不损失任何数据
ensuring no data is lost when two parts are joined back

于是可知关系中的函数依赖提供了一种简单的方法

用于构建关系 R 的无损连接分解 P

Database - P26

Heath 定理

在数据库中，可以从一个关系 R 中提取 $X \rightarrow Y$ ，并存储为一个新的关系 $\pi_{XY}(R)$

(1) $\pi_{XY}(R)$ 中的 X 值没有重复 (no value repetition)

(2) 可以高效地根据 X 的值检索 Y

effective lookup table for Y keyed by X

(3) 在更新 Y 的值时，仅需在 $\pi_{XY}(R)$ 中进行修改

only one place to update Y corresponding to each X

注意， $\pi_{XY}(R)$ 中的 X 还应作为 R 的剩余部分 $\pi_{XZ}(R)$ 中的外键

包含依赖 (inclusion dependency)，指对于两个关系 R 和 S，分别有属性集合 $\{r_1, \dots, r_m, c_1, \dots, c_k\}$ 和 $\{s_1, \dots, s_n, c_1, \dots, c_k\}$

其中 r_1, \dots, r_m 为 R 的特有属性， s_1, \dots, s_n 为 S 的特有属性， c_1, \dots, c_k 为公共属性

如果有属性集合 $\{a_1, \dots, a_i\} \subseteq \{c_1, \dots, c_k\}$ 使得 $\pi_{a_1, \dots, a_i}(R) \subseteq \pi_{a_1, \dots, a_i}(S)$

则称关系 R 包含依赖于 S

于是可知，包含依赖为形式化的外键 (formalism for foreign key)

保证了关系型数据库中的参照完整性 (referential integrity)

与函数依赖相比：

函数依赖表示了一个关系内部的约束 (express constraint over one relation)

是生成等价的依赖 (equality-generating dependency)

包含依赖表示关系间的约束 (express constraint between relation schemas)

是生成元组的依赖 (tuple-generating dependency)

注意 使用 Heath 定理分解关系时，实际上是否保证参照完整性的

需要引入包含依赖，即外键，加以约束

规范化

(normalization)，对于一个属于低一级的范式 (normal form) 的关系模式 (relation schema)

通过模式分解 (schema decomposition) 转换为若干高一级的范式

用于消除数据冗余 (reduce data redundancy) 和提高数据完整性 (improve data integrity)

规范化主要针对解决 3 个问题：

更新异常 (update anomaly)，指由于数据冗余，当更新数据库中的数据时 inconsistency，系统需要付出巨大代价来维护数据库的完整性，否则会面临逻辑上不一致 (logical inconsistency)

插入异常 (insertion anomaly)，指对于本应可以独立存在的数据

由于其他部分暂缺而无法插入数据库的情形

删除异常 (deletion anomaly)，指对于本应可以独立存在的数据

由于删除其他部分而导致这些数据从数据库中丢失的情形

Database - P27

范式

(normal form), 指关系型数据库中关系所满足的一系列要求

通常称满足不同程度要求的~~关系~~为不同范式

现在也把范式的概念理解为符合某-范式要求的关系集合

如对于关系R, 记Rⁿ范式为R满足第n范式

注意这个表示方法类似于算法复杂度中渐近记号的使用

由于通常高-级的范式是低-级范式的基础上满足额外~~条件~~得到的

所以也有 0范式 ⊂ 1范式 ⊂ 2范式 ⊂ ... (UNF ⊂ INF ⊂ 2NF ⊂ ...)

未规范化形式 (unnormalized form, UNF), 又称 0范式 (non first normal form)

UNF 的关系仅需满足最基本的要求:

1. 具有主键 (primary key, 主码), ~~主键唯一且非空~~

即有一个属性集合, 其值可以唯一地指向关系中的一个元组

同时也蕴含了关系中不存在重复的元组 (no duplicate tuple)

2. 属性名~~均为~~唯一的, 即没有重复的属性

no repeating groups in individual table

第一范式 (first normal form, 1NF), 比UNF高-级的范式

在UNF的基础上增加要求:

关系中的每一个~~属性~~的值域只能包含原子性的值,

即每一个分量必须是不可分的数据项

the domain of each attribute contains only atomic (indivisible) values

and the value of each attribute contains only a single value from domain

注意: 有时也将UNF的条件合并入1NF, 并认为1NF是要求最低的范式

第二范式 (second normal form, 2NF), 比1NF高-级的范式

在1NF的基础上增加要求:

每一个非主属性完全依赖于~~关系的~~每一个候选~~键~~键

every non-prime attribute is dependent on whole of every candidate key

或者说 非主属性不依赖于某一个候选键的真子集

not have any non-prime attribute that is dependent on

any proper ~~subset~~ subset of any candidate key of the relation

其中非主属性指不属于任何候选键的属性

not a part of any candidate key of the relation

注意: 如果关系不满足2NF, 则会产生插入异常, 删除异常, 修改复杂的问题

Database - P28

第三范式

第三范式

第三范式 (third normal form, 3NF), 比 2NF 高一级的范式

在 2NF 的基础上增加要求:

不存在候选码 X , 属性集合 Y 和非主属性集合 Z

使得 $X \rightarrow Y$, $Y \rightarrow X$, $Y \rightarrow Z$, 即 $X \xrightarrow{\text{传递}} Z$

即每一个非主属性均不传递依赖于候选码

every non-prime attribute is non-transitively dependent on every key

基本函数依赖 (elementary function dependency), 对于函数依赖集合 G , 则 $f: X \rightarrow A \in G$

如果 $f: X \rightarrow A$ 是非平凡的且是完全依赖, 则称 $f: X \rightarrow A$ 是 G 的基本函数依赖

对于关系 $R^{(u, f)}$, 如果函数依赖 $f: X \rightarrow A$ 是 F 的基本函数依赖

则称 $f: X \rightarrow A$ 是关系 R 的基本函数依赖

基本键 (elementary key), 指对于关系 $R^{(u, f)}$, X 是关系 R 的候选键.

如果有属性 A , 使得 $f: X \rightarrow A$ 是 R 的基本函数依赖, 则称 X 是关系 R 的基本键

基本键范式 (elementary key normal form, EKNF), 比 3NF 高一级的范式

在 3NF 的基础上增加要求:

所有的基本函数依赖 或者 左部是完整的候选键, 或者 右部是基本键的子集

for every full non-trivial FD: $X \rightarrow Y$, either X is a key, or Y is (part of) elementary key

通常应对关系中存在多个不同的复合主键且 交集非空

3.5 NF

比 3NF 高一级

BC 范式 (Boyce-Codd normal form, BCNF), 比 EKNF 高一级的范式, 通常忽略 EKNF 而认为

在 EKNF 的基础上增加要求:

对于每一个非平凡的 FD: $X \rightarrow Y$, X 是 R 的 superkey, 即 X 包含候选键

如果关系 R 满足 BCNF, 则关系 R 满足:

所有的非主属性对每一个键都是完全函数依赖

所有的主属性对每一个不包含它的键, 也是完全函数依赖

没有任何属性完全函数依赖于非码的任何一组属性

BCNF 排除了任何属性对码的传递依赖与部分依赖, 即所有在函数依赖上的冗余

多值依赖 (multivalued dependency, MVD), 对于关系 $R^{(u)}$, 有属性集合 X, Y, Z , 且有 $Z = u - X - Y$

对于某个 X 值, 一组 Y 值和一组 Z 值, 有 V_Y 与 V_Z 分立, 或者说 $V_Y \times V_Z$ 对于某个 X 值全部存在

记为 $X \rightarrow \rightarrow Y$, 如果 $Z = \emptyset$, 则称 $X \rightarrow \rightarrow Y$ 为平凡的多值依赖

Database - P29

平凡多值依赖(trivial multivalued dependency), 指对于关系 $R(U)$, 且有 $X \rightarrow\!\!\rightarrow Y$
如果有 $Y \subseteq X$, 或者 $X \cup Y = U$, 即 $Z = U - X - Y = \emptyset$, 则称 $X \rightarrow\!\!\rightarrow Y$ 是平凡的多值依赖

对于关系 $R(U)$, 有属性集合 X, Y, Z .

对称性: 如果有 $Z = U - X - Y$, 且有 $X \rightarrow Y$, 则有 $X \rightarrow Z$

传递性: 如果有 $X \rightarrow Y$ 且 $Y \rightarrow Z$, 则有 $X \rightarrow Z$

函数依赖可视为多值依赖的特殊情形, 即如果有 $X \rightarrow Y$, 则有 $X \rightarrow Z$. 反之不一定成立

即对于每一个 X 的值 x , 都有一个确定的 Y 值 y 与之对应

集合运算: 如果有 $X \rightarrow Y$ 且 $X \rightarrow Z$, 则有

并集: $X \rightarrow Y \cup Z$, 交集: $X \rightarrow Y \cap Z$, 差集 $X \rightarrow Y - Z$ 且 $X \rightarrow Z - Y$

多值依赖与函数依赖的基本区别:

有效性范围 对于关系 $R(U)$ 上的属性集合 $XY \subseteq W \subseteq U$,

如果 $X \rightarrow Y$ 在 U 上成立, 则 $X \rightarrow Y$ 在 W 上一定成立, 但反之不一定成立

但是对于 $X \rightarrow Y$ 在 U 上成立, 则 $X \rightarrow Y$ 在任何属性集 W 上也成立, 反之亦然

嵌入多值依赖(Embedded MVD, EMVD), 对于属性集合 $XY \subseteq W \subseteq U$

如果 $X \rightarrow Y$ 在 U 上不成立但在 W 上成立, 则称为 R 上的嵌入多值依赖, 记为 $(X \rightarrow Y)_W$

子集有效性 如果 $X \rightarrow Y$ 在 $R(U)$ 上成立, 则对于任意 $Y' \subseteq Y$, 有 $X \rightarrow Y'$ 成立

但是如果 $X \rightarrow Y$ 在 $R(U)$ 上成立, 则无法断言 $X \rightarrow Y'$ 对所有 $Y' \subseteq Y$ 成立

第四范式(fourth normal form, 4NF), 比 BCNF 高一级的范式

在 3NF 基础上增加要求:

对于每一个非平凡的多值依赖 $X \rightarrow\!\!\rightarrow Y$ 且 $Y \not\subseteq X$, X 都是 super key, 即包含候选键

于是 4NF 不允许存在非平凡且非函数依赖的多值依赖

由于对于每个 $X \rightarrow Y$, X 都包含候选键, 于是有 $X \rightarrow Y$

可以用投影分解的方法消去非平凡且非函数依赖的多值依赖

连接依赖(join dependency, JD), 对于关系模式 $R(U, F)$, 如果有 U 的划分 R_1, R_2, \dots, R_n

则有 R 的分解 $P = \{R_1, R_2, \dots, R_n\}$. 如果对于每一个 R 的实例 r

都满足 $m_P(r) = \prod_{i=1}^n \pi_{R_i}(r) = r$,

则称 R 满足连接依赖 $*(R_1, R_2, \dots, R_n)$

如果某个 $R_i = R$, 则称 $*(R_1, R_2, \dots, R_n)$ 为平凡的连接依赖

特别地, 多值依赖实际上是二元(2-ary)的连接依赖

即 $X \rightarrow\!\!\rightarrow Y$ 在 R 上成立 当且仅当 R 满足 $*(XUY, XU(U-Y))$

Database - P30

→ 3NF → 4NF → ...

ETNF

(essential tuple normal form), 比 4NF 高一级的范式
在 4NF 的基础上增加要求
对于任意的连接依赖 $\star(R_1, R_2, \dots, R_n)$, 其中存在 R_i 是关系 $R(u)$ 的 superkey

第五范式

(fifth normal form), 比 ETNF 高一级的范式, 通常忽略 ETNF 而认为是比 4NF 高一级
在 ETNF 的基础上增加要求
对于关系模式 $R(u)$ 的任意连接依赖, 均由 R 的候选键蕴含
 $\text{join dependency } \star(R_1, R_2, \dots, R_n) \text{ implied by the } R \text{'s candidate key}$
iff each of R_1, R_2, \dots, R_n is R 's superkey
第五范式有时也称为投影连接范式 (project-join normal form, PJNF)

DKNF

(domain-key normal form), 比 5NF 高一级的范式
在 5NF 的基础上增加要求
关系模式中仅包含域约束 (domain constraint) 和键约束 (key constraint)
domain constraint specifies the permissible value for given attribute
key constraint specifies the attributes that uniquely identify a tuple

第六范式

(sixth normal form, 6NF), 在 3NF 上增加要求

关系模式中不存在非平凡的连接依赖 (nontrivial join dependency)

atomic column no repeating groups
primary key (no duplicate tuples)

	1NF	2NF	3NF	4NF	ETNF	5NF	DKNF	6NF
atomic column	✓							
no repeating groups	✓							
(on candidate key) no partial dependency	X	✓						
(on candidate key) no transitive dependency	X	✓						
every nontrivial FD involves either a superkey or contained by elementary key	X	✓						
no redundancy from any FD	X	✓						
every nontrivial MVD has superkey	X	✓						
a component of every explicit JD is superkey			X	✓				
every explicit JD implied by candidate key			X	✓				
Contain only domain constraint and key constraint				X	✓			
no nontrivial join dependency				X	✓			

every component of every explicit JD is superkey

every explicit JD implied by candidate key

Contain only domain constraint and key constraint

no nontrivial join dependency