课程名称: 数据库系统

第12讲:(第8章)

关系模式设计优化

单 位：重庆大学计算机学院

1.函数依赖和键（码）

- 给定 R(A, B, C).
  - A→ABC 意味着A 是一个键（码）.
- 通常,X是R的属性集,
  - X → R 意味着 X 是一个(超)码.
- 超码⊇候选码⊇主码

2.函数依赖集的闭包F^+

- **函数依赖集的闭包？由F逻辑蕴含的所有函数依赖的集合。**
- 计算函数依赖集的闭包：

$F^+ = F$

repeat

for each F^+ 中的函数依赖 f

在 f 上应用自反律和增补律

将结果加入到 F^+ 中

for each F^+ 中的一对函数依赖 f_1 和 f_2

if f_1 和 f_2 可以使用传递律结合起来

将结果加入到 F^+ 中

until F^+ 不再发生变化

F+例子

- F = {A → B, B → C, C D → E }
- Step 1: F中的每一个函数依赖, 使用自反律
  - 得到:CD → C; CD → D
  - 加到F上:
    F = {A → B, B → C, C D → E; CD → C; CD → D }
- Step 2: F中的每一个函数依赖, 使用增广律
  - A → B 得到: A → AB; AB → B; AC → BC; AD → BD; ABC →BC; ABD → BD; ACD →BCD
  - B → C 得到: AB → AC; BC → C; BD → CD; ABC → AC; ABD → ACD, etc.
- Step 3: 使用传递律
- 重复1~3步骤…

可以看出计算F+代价太高.

3.属性集闭包

- 函数依赖集闭包的大小是（属性的）指数级的
- 很多时候，我们仅仅是想判断一个函数依赖*X→Y是否在F的闭包中. 一个有效的方式是:*
  - 计算属性*X的闭包 (记为X^+) :*
    - *X的闭包 就是由X在F上蕴含的所有属性的集合。*
    - 计算属性的闭包仅仅需要一个线性的时间算法就够了.
  - F = {A → B, B → C, C D → E }

    A →E成立吗?

将判断A →E是否在F闭包里面的问题转换为E是否在A的属性集闭包里

属性集闭包的计算

- **Input F (a set of FDs), and X (a set of attributes)**
- **Output: Result=X^+ (under F)**

Method:

– Step 1: Result :=X;

– Step 2: Take Y → Z in F, and Y is in Result, do:

Result := Result ∪ Z

– Repeat step 2 until Result cannot be changed and then output Result.

属性集闭包例子

- F = {A → B, B → C, C D → E }

  A →E是否成立?

  - 也就是, 判断 A → E 是否在 F^+中?

    等价于, E 是否在A^+中?

- Step 1: Result = A
- Step 2: 考虑A → B, Result = AB

  考虑B → C, Result = ABC

  考虑CD → E, CD 不在ABC, 不添加
- Step 3: A^+ = {ABC}

  E 不在A^+中,所以 A → E 不在F^+中

属性集闭包例子

- F = {A →B, AC →D, AB →C}?
- 计算A^+。

- Answer: A^+ = ABCD

属性集闭包例子

- $R = (A, B, C, G, H, I)$
  $F = \{A \rightarrow B; A \rightarrow C; CG \rightarrow H; CG \rightarrow I; B \rightarrow H\}$
  $(AG)^+ = ?$
- Answer: ABCGHI
- *AG 是候选键吗?*

– 这个问题包括两部分:

1. AG 是一个超键吗?
   - $AG \rightarrow R? == Is\ (AG)^+ \supseteq R$
2. AG的子集是否是一个超键?
   - $A \rightarrow R? == Is\ (A)^+ \supseteq R$
   - $G \rightarrow R? == Is\ (G)^+ \supseteq R$

属性集闭包的作用

- 属性集闭包的作用:
  1. *检测超键*:

  – 判断X是否是一个超键？只需要计算 X^+, 检查 X^+ 是否包括*R的所有属性.*

  2. *检测函数依赖*

  – 判断$X \rightarrow Y$ 是否成立 (或者说, 是否在F^+中), *只需要判断*$Y \subseteq X^+$.

  – 因此, 我们计算X^+, 然后检测这个属性集闭包是否包括 Y.

  – 简单有用的方法

  3. *计算F的函数依赖集闭包*

计算 F^+

- F={ A → B, B → C}. 计算F+ (属性包括A, B, C).

Step 1: 构建二维表, 行和列列出所有可能的属性组合

	A	B	C	AB	AC	BC	ABC
A							
B							
C							
AB							
AC							
BC							
ABC							

Step 2: 计算所有的属性组合的属性集闭包

Attribute closure
A^+=?
B^+=?
C^+=?
AB^+=?
AC^+=?
BC^+=?
ABC^+=?

Step 3: 将结果填写到二维表中

计算 F^+

- F={ A → B, B → C}. 计算F^+ (属性包括A, B, C).

- 例如：A^+.
  Step 1: Result = A
  Step 2: 考虑A → B, Result = A ∪ B = AB
  考虑B → C, Result = AB ∪ C = ABC
  Step 3: A^+ = {ABC}

Computing F^+

- F={ A → B, B → C}. 计算F^+ (包括属性A, B, C).

Step 1: 构建一个空的二维表, 行和列列出所有可能的属性组合

	A	B	C	AB	AC	BC	ABC
A	√	√	√	√	√	√	√
B							
C							
:							

Step 2: 计算所有的属性组合的属性集闭包

Attribute closure
A^+=ABC
B^+=?
C^+=?
AB^+=?
AC^+=?
BC^+=?
ABC^+=?

Step 3:将结果填写到二维表中.

由于**A+=ABC.** 填写标的时候，考虑第一列，**A** 是 **A+**的一部分吗**?** 是, 勾选**. B** 是 **A+** 的一部分吗**?** 是, 勾选**...**

计算F+

- F={ A → B, B → C}. Compute F+ (包括属性A, B, C).

	A	B	C	AB	AC	BC	ABC
A	√	√	√	√	√	√	√
B		√	√			√	
C			√				
AB	√	√	√	√	√	√	√
AC	√	√	√	√	√	√	√
BC		√	√			√	
ABC	√	√	√	√	√	√	√

Attribute closure
$A^+=ABC$
$B^+=BC$
$C^+=C$
$AB^+=ABC$
$AC^+=ABC$
$BC^+=BC$
$ABC^+=ABC$

- 每一个√ 表示FD (行) → (列) 在 F+中.
- 每一个√ (列) 在(行)+中

计算F+

- F={ A → B, B → C}. Compute F+ (包括属性A, B, C).

A→BC

	A	B	C	AB	AC	BC	ABC
A	√	√	√	√	√	√	√
B		√	√			√	
C			√				
AB	√	√	√	√	√	√	√
AC	√	√	√	√	√	√	√
BC		√	√			√	
ABC	√	√	√	√	√	√	√

Attribute closure
A^+=ABC
B^+=BC
C^+=C
AB^+=ABC
AC^+=ABC
BC^+=BC
ABC^+=ABC

- 每一个√ 表示FD (行) → (列) 在 F^+中.
- 每一个√ (列) 在(行)+中

例子

- F={ A → BC, B →C }. 判断 C →AB 是否在 F^+?

- Answer: 不在.
  Reason 1) C^+=C, 不包括 AB.
  Reason 2) 反例，不存在 C → AB.

A	B	C
1	1	2
2	1	2

例子

- R(A, B, C, D, E),
- F = {A → B, C → D}
- 候选键?
- ACE.
- 怎么计算?
- Intuitively,
  - A is not determined by any other attributes (like E), and A has to be in a candidate key (because a candidate key has to determine all the attributes).
  - Now if A is in a candidate key, B cannot be in the same candidate key, since we can drop B from the candidate without losing the property of being a “key”.
  - So B cannot be in a candidate key
  - Same reasoning apply to others attributes.