# 数据库系统原理

陈岭

浙江大学计算机学院

# 10 BCNF、3NF和4NF

- BCNF
- 3NF
- 多值依赖
- 4NF
- 数据库设计过程

# Boyce-Codd范式

- 具有函数依赖集合$F$ 的关系模式$R$ 属于BCNF的条件是，当且仅当对$F^+$ 中所有函数依赖$\alpha \rightarrow \beta$（其中，$\alpha \subseteq R$ 且$\beta \subseteq R$），下列至少有一项成立
  - $\alpha \rightarrow \beta$ 是平凡的函数依赖（即$\beta \subseteq \alpha$）
  - $\alpha$ 是$R$ 的超码(即，$R \subseteq \alpha^+$ ，$\alpha \rightarrow R$）

# Boyce-Codd范式

- 例，$R = (A, \quad B, \quad C)$

  $F = \{A \rightarrow B$

  $B \rightarrow C\}$

  键= $\{A\}$

- $R$ 不属于BCNF（$\because F^+ = \{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C, \quad \cdots\}$，因为$B \rightarrow C$，$B$ 不是超码）

- 分解成$R_1 = (A, \quad B)$，$R_2 = (B, \quad C)$
  - $R_1$ 与$R_2$ 属于BCNF（$A$ 是$R_1$的超码，$B$ 是$R_2$的超码）
  - 无损连接分解（$R_1 \cap R_2 = B$，并且$B$ 是$R_2$的超码）
  - 保持依赖（$F_1 = \{A \rightarrow B\}$ 在$R_1$上保持依赖，$F_2 = \{B \rightarrow C\}$ 在$R_2$上保持依赖）

# 检查是否为BCNF

- 为检查非平凡依赖 $\alpha \rightarrow \beta$ 是否违反BCNF的要求
  - 计算$\alpha^+$
  - 检验$\alpha^+$ 是否包含$R$ 的所有属性，即，是否为$R$ 的超码
- 简化的测试：为检查具有函数依赖集合$F$ 的关系模式$R$ 是否属于BCNF，只需检查$F$ 中的函数依赖是否违反BCNF即可，而不需检查$F^+$ 中的所有函数依赖
  - 可以证明如果$F$ 中没有违反BCNF的依赖，则$F^+$ 中也没有违反BCNF的依赖

> ∵ $F^+$是由Armstrong的3个公理从$F$ 推出的，而任何公理都不会使FD左边变小(拆分)，故如果F中没有违反BCNF的FD(即左边是superkey)，则$F^+$中也不会

# 检查是否为BCNF

- 但是，当检查$R$ 的分解后的关系时仅用$F$是错误的
- 例，考虑 $R$ ($A$，$B$，$C$，$D$)，具有$F$ = {$A \rightarrow B$，$B \rightarrow C$}
  - 分解$R$ 到$R_1$($A$，$B$) 与$R_2$($A$，$C$，$D$)
  - $F$ 中的函数依赖都不是只包含($A$，$C$，$D$)中的属性， 因此我们可能错误地认为$R_2$ 满足BCNF
  - 事实上， $F^+$ 中的依赖$A \rightarrow C$ 显示$R_2$ 不属于BCNF

可在$F$下判别$R$ 是否违反BCNF，但须在$F^+$下判别$R$ 的分解式是否违反BCNF

# BCNF分解算法

```
result := {R};
done := false;
compute F+;
while (not done) do
  if (result 中存在模式Ri 不属于BCNF)
    then begin
          令α → β 是Ri 上的一个非平凡函数依赖
            使得α → Ri 不属于F+，且α ∩ β = ∅;
          result := (result - Ri) ∪ (Ri - β) ∪ (α, β);
          end
    else done := true;
```

将$R_i$分解为二个子模式：$R_{i1}=(\alpha, \beta)$和$R_{i2}=(R_i - \beta)$，$\alpha$是$R_{i1}$和$R_{i2}$的共同属性

$R_{i1}$　　$R_{i2}$

- 注意：每个$R_i$都属于BCNF，且分解是无损连接的

# BCNF分解示例

- *class* (*course_id, title, dept_name, credits, sec_id, semester, year, building, room_number, capacity, time_slot_id*)
- 函数依赖：
  - *course_id* → *title, dept_name, credits*
  - *building, room_number* → *capacity*
  - *course_id, sec_id, semester, year* → *building, room_number, time_slot_id*
- 候选码：{*course_id, sec_id, semester, year*}
- BCNF分解：
  - *course_id* → *title, dept_name, credits*，但是*course_id* 不是超码

# BCNF分解示例

- 我们将*class* 分解为：
  - *course*(*course_id, title, dept_name, credits*)
  - *class-1* (*course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id* )
- *course*是BCNF
- *building, room_number* → *capacity* 在*class-1*上存在依赖，但是{*building, room_number*}不是*class-1*的超码
- 我们将*class-1* 分解为：
  - *classroom* (*building, room_number, capacity*)
  - *section* (*course_id, sec_id, semester, year, building, room_number, time_slot_id*)
- *classroom* 和*section* 是BCNF

# BCNF与保持依赖

- BCNF分解不总是保持依赖的
- 例，$R = (J, K, L)$

  $F = \{JK \rightarrow L$

  $L \rightarrow K\}$

J---student，K---course，L---teacher（一门课有多个教师，一个教师上一门课，一个学生选多门课，一门课有多个学生选）

- 两个候选码：JK 和JL
- $R$ 不属于BCNF（$\because L \rightarrow K$，$L$ 不是超码）
- $R$ 的任何分解都不满足保持依赖：$JK \rightarrow L$
  - 例，$R_1 = (L, K)$，$R_2 = (J, L)$ $\subseteq$ BCNF，但不保持依赖

# BCNF与保持依赖

- 因此，我们并不总能满足这三个设计目标：
  - 无损连接
  - BCNF
  - 保持依赖

# 3NF：动机

- 存在这样的情况
  - BCNF不保持依赖
  - 但是，有效检查更新是否违反FD是重要的
- 解决方法：定义一种较弱的范式，称为第三范式（3NF）
  - 允许出现一些冗余（从而会带来一些问题）
  - 但FD可以在单个关系中检查，不必计算连接
  - 总是存在到3NF的无损连接分解，且是保持依赖的