資料庫系統
Class 4: Design of Relational DB

逢甲資工 許懷中

好的關聯式資料庫設計

- 消除重複的資料
- 保證資料之完整與正確

第一正規化 (First Normal Form)

- 如果一個 relational schema 其中所有的屬性都屬於 atomic domain，則稱此資料表符合 第一正規化 (First Normal Form, 1NF)
  - Atomic domain?
  - Non-atomic domain?
    - 一項屬性中包含多項子屬性，如姓名包含姓、名、綽號；電話號碼包含國碼、區碼、本地碼
    - 一項屬性是由可以進一步分割的字串所構成，如D0123456、+886424517250、台中市40724西屯區文華路100號資電館2F(資電201)

第一正規化 (cont.)

- 為什麼需要第一正規化？
  - Non-atomic 屬性會造成資訊的重複存儲
  - 應用程式需要額外的邏輯來處理以 Non-atomic 形式存儲的資訊

正規化 (Normalization)

- 正規化是建構 Data Model 的一種技術，目的是為了降低資料的重覆性，避免發生更新異常
- 第一正規化為正規化的一種形式
- 還有更多的正規化，幫助設計『好的』資料庫架構
- 若現有 relational schema 不符合好的形式
  - 將其解構為具有良好形式的 schema
  - 解構必須是 Lossless-Join-Decomposition

由於資料表合併造成的重複

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

並非所有合併都會造成重複

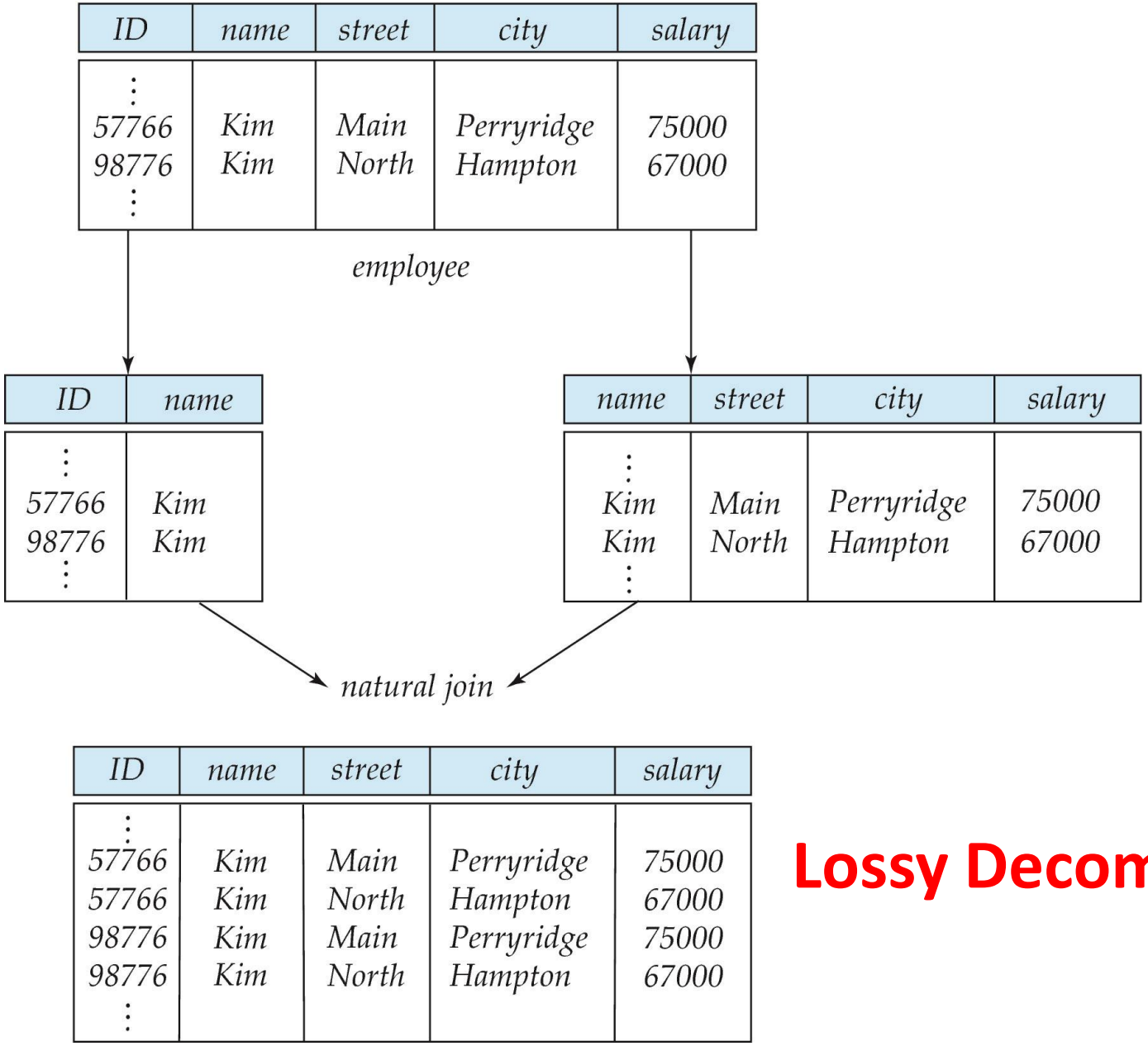
- 將
  - sec_class(sec_id, building, room_number) and
  - section(course_id, sec_id, semester, year)
- 合併為
  - section(course_id, sec_id, semester, year, building, room_number)

解構資料表

- 將有重複資訊的資料表解構成較小的資料表以消除重複
- 如何知道 *inst_dept*資料表應該解構為 *instructor* 以及 *department* 資料表？
  - 觀察 *inst_dept* 資料表，發現假如存在 (*dept_name*, *building*, *budget*) 這樣的關聯式綱要 (relational schema)，那麼 **dept_name** 將是這張該 schema 的**候選鍵**
  - 這就是所謂的 **functional dependency**
    - *dept_name* → *building*, *budget*

解構資料表 (cont.)

- 並非所有的解構都是『好的』
  - 考慮將 employee(ID, name, street, city, salary) 解構為
    - employee1 (ID, name)
    - employee2 (name, street, city, salary)
- 發生什麼事情？為什麼說這樣不好？



Lossy Decomposition

Lossless-Join Decomposition

將 *r1*(*A*, *B*, *C*) 解構為 *r2*(*A*, *B*) 、*r3*(*B*, *C*)

A	*B*	*C*
α	1	A
β	2	B

r

A	*B*
α	*1*
β	*2*

$\Pi_{A,B}(r)$

B	*C*
1	A
2	B

$\Pi_{B,C}(r)$

Π_A (r) $\bowtie$ Π_B (r)

Natural Join

A	*B*	*C*
α	1	A
β	2	B

功能相依 (Functional Dependency)

- 一個資料表中的某些屬性，可以決定某些屬性之值的唯一性
  - Department(*dept_name*, *building*, *budget*)
  - *dept_name* → *building*, *budget*
  - *building* 與 *budget* 屬性，相依於 *dept_name*
- 功能相依是鍵 (key) 的普遍化形式
  - 功能表中所有的屬性，都功能相依於該資料表的超鍵 (Super key)

功能相依 (cont.)

- 令 R 是一個 relational schema，α、β 是 R 之中屬性的集合： $\alpha \subseteq R$、$\beta \subseteq R$
- 在 R 之中 β 相依於 α ($\alpha \rightarrow \beta$)，若且唯若對於關聯式資料庫 $r(R)$中任意兩個 tuple t_1, t_2，當 t_1, t_2 屬性 α 的值相同則屬性 β 的值也相同
  - $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$
  - Example: r(A, B)
    - $A \rightarrow B$ 成立？
    - $B \rightarrow A$ 成立？

A	B
1	4
1	5
3	7

功能相依 (cont.)

- 如果 K 是 R 的超鍵： $K \rightarrow R$
- K 是 R 的候選鍵 若且唯若
  - $K \rightarrow R$
  - $\exists$ no α where $\alpha \subset K$ that $\alpha \rightarrow R$
- Example
  - *inst_dept* (*ID, name, salary, dept_name, building, budget*)
  - *dept_name* $\rightarrow$ *building*?
  - *ID* $\rightarrow$ *building*?
  - *dept_name* $\rightarrow$ *salary*?

功能相依 (cont.)

- 如何運用功能相依
  - 檢查資料表是否符合功能相依的限制
  - F 是一個功能相依集，如果 F 在 relation *r* 中都成立，則 *r* 滿足 F (*r* satisfies F)
  - F 在一個 relational schema R 中都成立，代表 R 中所有的 relation *r* 都滿足 F
- 注意相反條件
  - 某一個 *relation* 的部分 instances 可能滿足某一個功能相依，既使就整個 relation 而言，並不成立
  - 如 name → ID

功能相依 (cont.)

- 下列功能相依被稱為 **trivial**
  - ID, name $\rightarrow$ ID
  - name $\rightarrow$ name
- , $\alpha \rightarrow \beta$ 是 trivial 若 $\beta \subseteq \alpha$

功能相依 (cont.)

- 若 $A \rightarrow B$ 與 $B \rightarrow C$ 則 $A \rightarrow C$ *成立*
- 功能相依閉包 (Closure of set of functional dependencies)
- F 為一個功能相依集，則由所有可以由 F 所產生的功能相依之集合，稱做功能相依閉包 (Closure of set of functional dependencies)
- The *closure* of *F* 寫作 $\mathbf{F^+}$.
- F^+ 是 *F* 的超集 *(Superset)*.

功能相依之內範圍關聯集合

- 給定F 後，透過反覆運用這些規則，我們可以找到所有的 F^+。這個系列的規則被稱為**阿姆斯壯定理(Armstrong's Axioms)**：
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(反身規則)**
  - if $\alpha \rightarrow \beta$, then $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(擴充規則)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(遞移規則)**
- 阿姆斯壯定理是
  - **合理的(sound)**
  - **完整的(complete)**

例子

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$
    - 由於$A \rightarrow B$ 和$B \rightarrow H$ 符合，我們使用於遞移規則
  - $AG \rightarrow I$
    - 由於$A \rightarrow C$ 和$CG \rightarrow I$，偽遞移規則意味著$AG \rightarrow I$ 符合
  - $CG \rightarrow HI$
    - 由於$CG \rightarrow H$ 和$CG \rightarrow I$，聯集規則意味著$CG \rightarrow HI$

計算 F^+ 的過程

計算功能相依 F 之內範圍關聯集合:

$F^+ = F$
repeat
    for each functional dependency f in F^+
        apply reflexivity and augmentation rules on f
        add the resulting functional dependencies to F^+
    for each pair of functional dependencies f_1 and f_2 in F^+
        if f_1 and f_2 can be combined using transitivity
            then add the resulting functional dependency to F^+
until F^+ does not change any further

功能相依之內範圍關聯集合

- 其他規則:
  - 如果$\alpha \rightarrow \beta$符合和$\alpha \rightarrow \gamma$符合，那麼$\alpha \rightarrow \beta\gamma$就符合 (**聯集規則**)
  - 如果$\alpha \rightarrow \beta\gamma$ 符合，則$\alpha \rightarrow \beta$符合和$\alpha \rightarrow \gamma$也符合　(**分解規則**)
  - 如果$\alpha \rightarrow \beta$符合和 $\gamma\ \beta \rightarrow \delta$符合，則$\alpha\ \gamma \rightarrow \delta$也符合 (**偽遞移規則**)

屬性的內範圍集合

- 假設 a 是屬性集合。我們稱所有處於功能相依F 集合下，由a 功能確定屬性的集合為F 之下的 a **內範圍集合(*closure*)**，並以 a^+ 表示
- 計算 a^+ 的演算法如下：

result := a;
while (changes to *result*) **do**
  for each $\beta \rightarrow \gamma$ **in** F **do**
    begin
      if $\beta \subseteq$ *result* **then** *result* := *result* $\cup \gamma$
    end

屬性內範圍的用途

- 超鍵的測試:
  - 測試α 是否為超鍵,我們計算α^+，並檢查α^+ 是否包含了所有R 內的屬性
- 功能相依的測試:
  - 藉由檢查$\beta \subseteq \alpha^+$ 是否成立，來檢查一個功能相依$\alpha \rightarrow \beta$ 是否符合（換句話說，是在F^+ 中）
- 計算 F^+ 的替代方法
  - 在每個 $\gamma \subseteq R$ ，我們都發現了γ^+ 內範圍集合，至於每個 $S \subseteq \gamma^+$, ，我們產出了$\gamma \rightarrow S$功能相依

簡化集合(canonical cover)

- 功能相依集合可能有重複的相依被引用到其他的集合中

- F 的簡化集合就是最小的功能相依集合，沒有重複的相依或重複的部分相依

外來屬性 (Extraneous Attributes)

- 考慮功能相依 F 集合和 F 中的 $\alpha \rightarrow \beta$ 功能相依
  - 如果 $A \in \alpha$ 和 F 邏輯上暗示 $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$，那$A$ 屬性在α 中則為外來的
  - 如果$A \in \beta$ 和 $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ 功能相依集合邏輯上蘊涵F，那A 屬性在β 中則為外來的
- Example: Given $F = \{A \rightarrow C, AB \rightarrow C\}$
  - 假設我們在F 中有$AB \rightarrow C$ 和$A \rightarrow C$ 功能相依。那麼，B 在$AB \rightarrow C$ 中是外來的
- Example: Given $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - 假設我們在F 中有$AB \rightarrow CD$ 和$A \rightarrow C$ 功能相依。那麼C 在$AB \rightarrow CD$ 的右邊亦為外來的

外來屬性的測試

- 考慮功能相依F 集合和F 中的$\alpha \rightarrow \beta$功能相依
- 如果是$A \in \alpha$ ，為了檢查A 是否為外來的，應考慮
  1. compute $\gamma = (\{\alpha\} - A)^+$
  2. 檢查$\gamma \rightarrow \beta$ 可否由F 推斷。若 γ^+ 包含了所有β內的屬性 ，那麼A 在α 是外來的
- 如果是$A \in \beta$ ，為了檢查A 是否為外來的，應考慮
  1. $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  2. 檢查$\alpha \rightarrow A$ 可否由F' 推斷。為此，需根據F' 來計算α^+（ α 的內範圍集合 ）;若α^+ 包含A，那麼A 在β 是外來的

簡化集合

- 一個F的F_c **簡化集合(canonical cover)** 是
  - 一個邏輯上意味著所有Fc內相依的相依集合
  - 而F_c於邏輯上意味著所有F中的相依
  - F_c中的功能相依不包含外來屬性
  - 每個F_c中的功能相依左側都是唯一的

計算簡化集合

- $R = (A, B, C)$
  $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$
- 結合 $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$
  - 集合現在為 $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A 在 $AB \rightarrow C$ 中是外來的
  - 因為F 邏輯上蘊涵 (F - {AB → C}) ∪ {B → C}，所以A在AB → C 中是外來的
    - 因為B → C 已經在我們功能相依的集合中了
- C 在$A \rightarrow BC$ 中是外來的
  - 因為 $A \rightarrow BC$ 邏輯上蘊涵$A \rightarrow B$ 和$B \rightarrow C$
- 簡化集合為：　　$A \rightarrow B, B \rightarrow C$

無損結合分解 (lossless-join decomposition)

- 對於 $R = (R_1, R_2)$, 我們需要在架構R中所有可能的關聯 r

$$r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$$

- 如果至少有一個以下的功能相依在F^+ 中，$R1$ 和$R2$ 會組成一個R 無損分解：
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

相依保存

- 存在一功能相依$\alpha \rightarrow \beta$與關聯綱要 R
- 若$\alpha \cup \beta \subseteq R$，則 $\alpha \rightarrow \beta$被**保存** (preserved) 於R ⇨ 相依保存 (Dependency Preservation)
- 相依保存，可以降低檢查資料庫 constraints (如功能相依) 時的代價
- 在一張關聯綱要中就可以檢查所有相關的功能相依

解構 與 相依保存

- F 對 R_i 的限制(restriction) 為 F_i 集合，是F^+中所有只包含 R_i 屬性的功能相依
  - 如果以下條件成立，則該分解為相依性保存

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$

  - 反之，則要對每次的更新做功能相依檢查，會有較高成本

相依性保存測試

- 該測試用於下列程序，確認在F 內 $\alpha \rightarrow \beta$的相依性是否保存在從R 中分解出來的$R_1, R_2, ..., R_n$
  - $result = \alpha$
    while (changes to *result*) do
      for each R_i in the decomposition
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
  - 若result 在β 中包含了所有屬性，那功能相依$\alpha \rightarrow \beta$就得以保留
- 若且唯若 所有的相依關聯在F 會被保留，分解才算是是相依性保存的分解

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C\}$
  Key = $\{A\}$
- R 並非 BCNF
- 分解為 $R_1 = (A, B)$, $R_2 = (B, C)$
  - R_1 和 R_2 為 BCNF
  - 無損的分解
  - 相依保存

Boyce-Codd 正規化 (BCNF)

- 對一個具有功能相依集 F 的關聯式綱要(relation schema) R，若所有屬於 F^+ 的功能相依 $\alpha \rightarrow \beta$ ($\alpha \subseteq R$, $\beta \subseteq R$) 符合下列條件之一，則 R 符合 BCNF
  - $\alpha \rightarrow \beta$ 是 trivial (i.e., $\beta \subseteq \alpha$)
  - α 是 R 的超鍵
- Example
  - *inst_dept* (*ID, name, salary, dept_name, building, budget*) 不符合 BCNF
  - 因為 *dept_name*→ *building, budget* 是屬於 *inst_dept* 的功能相依而 *dept_name* 並非 *inst_dept* 的 超鍵

解構關聯綱要以符合 BCNF

- 對於一個關聯綱要 R ，假如一個功能相依 $\alpha \rightarrow \beta$ 使得 R 無法符合 BCNF
- 將 R 解構為
  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$
- Example
  - α = *dept_name*、β = *building, budget*
  - $(\alpha \cup \beta)$ = (*dept_name, building, budget*)
  - $(R - (\beta - \alpha))$ = (*ID, name, salary, dept_name*)

BCNF 分解演算法

result := {R };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
        and $\alpha \cap \beta = \varnothing$;
      result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
      end
    else *done* := **true;**

該演算法產生的分解不僅在BCNF 內，同時也是個無損分解

BCNF 分解演算法實例

- *class* (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
- F = {*course_id*→ *title*, *dept_name*, *credits*
  building, *room_number*→*capacity*
  course_id, *sec_id*, *semester*, *year*
  →*building*, *room_number*, *time_slot_id*
- 候選鍵為 {*course_id*, *sec_id*, *semester*, *year*}.

BCNF 分解演算法實例

- *course_id* 並非超鍵，所以 class 需要解構
  course(*course_id*, *title*, *dept_name*, *credits*)
  class-1(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
  course 符合 BCNF
- *building*, *room_number*→*capacity* 存在於 *class-1*
  但 {*building*, *room_number*} 並非超鍵，所以
  class-1 需要解構
  classroom (*building*, *room_number*, *capacity*)
  section (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
  classroom 和*section* 符合 BCNF

BCNF 與 相依保存

- R = (A, B, C), F = (C→B, AB→C)
- 假如 C → B 使得 R 無法符合 BCNF
  - 進行解構 R1 = (A, C), R2 = (B, C)
  - AB→C 必須要 R1 ⋈ R2 才能表達
  - **違背相依保存**
- BCNF 與 相依保存不一定總是能同時達成

第三正規化 (Third Normal Form)

- 對一個關聯式綱要(relation schema) R，若所有屬於 F^+ 的功能相依 $\alpha \rightarrow \beta$ ($\alpha \subseteq R$, $\beta \subseteq R$) 符合下列條件之一，則 R 符合 3NF
  - $\alpha \rightarrow \beta$ 是 trivial (i.e., $\beta \subseteq \alpha$)
  - α 是 R 的超鍵
  - 任一 β - α 中的成員 A，是某個候選鍵的成員
- R = (A, B, C), F = (C→B, AB→C)
  - 假如 C → B 使得 R 無法符合 BCNF，因為 B ⊆ AB，所以 R 符合 3NF
- 3NF 是 BCNF 的超集，同時最低程度地在 BCNF 上做出妥協，以保證相依保存

範例1 (用 3NF 取代 BCNF)

Person	Shop Type	Nearest Shop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

{Person, ShopType} → {NearestShop}

{Nearestshop} → {ShopType}

範例2 (符合 3NF 但是不符合 BCNF)

Court	Start Time	End Time	Rate Type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	14:00	16:30	PREMIUM-A

SAVER: for Court 1 bookings made by members
STANDARD: for Court 1 bookings made by non-members
PREMIUM-A: for Court 2 bookings made by members
PREMIUM-B: for Court 2 bookings made by non-members

範例2 (cont.)

- 候選鍵
  - {Court, Start Time}, {Court, End Time}, {Rate Type, Start Time}, {Rate Type, End Time}
- 不符合 BCNF:
  - Rate Type → Court
- 符合 3NF: Court 是上述某一候選鍵的成員

範例2 (cont.)

Rate Type	Court	Member Flag
SAVER	1	Yes
STANDARD	1	No
PREMIUM-A	2	Yes
PREMIUM-B	2	No

Member Flag	Court	Start Time	End Time
Yes	1	09:30	10:30
Yes	1	11:00	13:30
No	1	14:00	15:30
No	2	10:00	11:30
No	2	11:30	13:30
Yes	2	14:00	16:30

為什麼需要正規化？

- 希望所有的關聯綱要都是『**好的**』
- 利用正規化，將不好的關聯綱要解構
  - 解構後的關聯綱要都是**好的** (符合 BCNF)
  - 解構不能造成資訊流失(Lossless-join Decomposition)
  - 解構後的關聯綱要能達成相依保存 (Dependency Preservation)
- 由於這三者不見得總能全被滿足，所以我們只能被迫在BCNF 和3NF 相依保存之間做出選擇
  - 缺乏相依保存
  - 使用3NF造成的重複性

BCNF 夠好嗎？

- BCNF 比 3NF 還要更嚴謹，但是這樣就夠了嘛？
- 考慮一張資料表
  - *inst_info*(ID, child_name, phone)
  - 每位老師可以登錄多個電話號碼，同時可能有多個小孩

ID	Child_name	phone
99999	David	04-2451-1234
99999	Alan	04-2451-5678
99999	David	04-2451-5678
99999	Alan	04-2451-1234

BCNF 夠好嗎？(cont.)

- 是 *inst_info* 事實上是符合 BCNF 的
  - 所有的功能相依都是 trivial 的
- *inst_info* 資料表看起來有些冗餘
  - 要為教師 99999 新增一支電話，得插入兩行資料
    - (99999, David, 04-2451-8888)
    - (99999, Alan, 04-2451-8888)
  - *inst_info* 應該要解構成為兩張資料表
    - *inst_child*(ID, child_name)
    - *inst*_phone(ID, phone)
- 這代表我們需要其他的關聯式 => 4NF

多值依存 (Multivalued Dependency)

- 假設一個關於學校教授的關聯綱要
  - *inst*(*ID, dept_name, name, address*)
- 假設一個教授可能屬於多間系所，同時可能有多個住址，但是每個教授只會有一個獨一無二的 ID
  - ID → name
- 一個符合 BCNF 的解構如下
  - *inst*(*ID*, *name*)
  - *inst_info*(*ID*, *dept_name*, *address*)

多值依存 (cont.)

ID	Dept_name	Name	Address
X001	心靈感應系	X教授	超能學院
X001	心靈感應系	X教授	小三的家
X001	心理系	X教授	超能學院
X001	心理系	X教授	小三的家

解構

ID	Dept_name	Address
X001	心靈感應系	超能學院
X001	心靈感應系	小三的家
X001	心理系	超能學院
X001	心理系	小三的家

ID	Name
X001	X教授

符合 BCNF? 有沒有冗餘？

多值依存 (cont.)

- 令 R 為一關聯綱要，其中$\alpha \subseteq R, \beta \subseteq R$；則多值依存 (MVD) 寫作 $\alpha \rightarrow\rightarrow \beta$ 成立當
- 對 $r(R)$ 中任意一對 tuples t_1, t_2, 其 $t_1[\alpha] = t_2[\alpha]$, $r(R)$ 中存在 tuples t_3, t_4 使得:

$$
\begin{aligned}
&t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \\
&t_3[\beta] = t_1[\beta] \\
&t_3[R - \beta] = t_2[R - \beta] \\
&t_4[\beta] = t_2[\beta] \\
&t_4[R - \beta] = t_1[R - \beta]
\end{aligned}
$$

多值依存 (cont.)

	α	β	$R - \alpha - \beta$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

範例

- 令R 為一關聯綱要，而其屬性可以被分為三個非空子集 Y, Z, W
- $Y \rightarrow\rightarrow Z$ (Y **multidetermines** Z) 若且唯若對所有 $r(R)$

  若 $< y_1, z_1, w_1 > \in r$ and $< y_1, z_2, w_2 > \in r$

  則

  $< y_1, z_1, w_2 > \in r$ and $< y_1, z_2, w_1 > \in r$
- 由於 Z 與 W 在此推導中的作用相同，*因此* 若 $Y \rightarrow\rightarrow W$ 則 $Y \rightarrow\rightarrow Z$ 也同時成立

範例 (cont.)

- 回到前面的例子
  - $ID \rightarrow\!\!\!\rightarrow dept\_name$

<X001, 心靈感應系, 超能學院> $\in r$, <X001, 心理系, 小三的家> $\in r$
則
<X001, 心靈感應系, 小三的家> $\in r$, <X001, 心理系, 超能學院> $\in r$

  - $ID \rightarrow\!\!\!\rightarrow address$

<X001, 心靈感應系, 超能學院> $\in r$, <X001, 心理系, 小三的家> $\in r$
則
<X001, 心理系, 超能學院> $\in r$, <X001, 心靈感應系, 小三的家> $\in r$

範例 (cont.)

- 上面這個正規定義表示了，在 Y (ID) 上的**特定值**，分別與 Z ($address$) 以及 W ($dept\_name$)兩屬性上一**組**值之間的關聯性，雖然 Z ($address$) 以及 W ($dept\_name$) 在概念上是獨立的
- 若 $Y \rightarrow Z$ 則 $Y \rightarrow\!\!\!\rightarrow Z$，任何功能相依皆為多值相依

多值相依Closure

- 多值相依集 D 的 **closure** D^+ 包含了所有可以從 D 中推導出來的功能與多值相依
  - 詳細推導請見課本的 Appendix C

第四正規化
(The Fourth Normal Form, 4NF)

- 任意關聯綱要 R 以及相關之功能依存與多值依存之集合 D 符合 **4NF**，當 D^+ 中任意依存 $\alpha \rightarrow\rightarrow \beta$ $(\alpha \subseteq R, \beta \subseteq R)$ *符合下列條件之一*
  - $\alpha \rightarrow\rightarrow \beta$ 是 trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α 是 R 的 超鍵 (superkey)
- 當一個關聯符合 4NF 則其也符合 BCNF

依據 4NF 進行解構

- 對於 R 的子集 R_i 其 MVD 集合 D_i 可以從與 R 相關的 MVD 集合 D 中獲得， D_i 包含
  - D^+ 所有只含有 R_i 中屬性的功能相依
  - $\alpha \rightarrow\rightarrow (\beta \cap R_i)$
    - (對於任意 $\alpha \subseteq R_i$ 且 $\alpha \rightarrow\rightarrow \beta$ 屬於 D^+)

依據 4NF 進行解構 (演算法)

result: = {*R*};
done := false;
compute D^+;
Let D_i denote the restriction of D^+ to R_i

while (**not** *done*)
if (there is a schema $\boldsymbol{R}_i$ in *result* that is not in 4NF) **then**
begin

let $\alpha \rightarrow\!\!\!\rightarrow \beta$ be a nontrivial multivalued dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in D_i, and $\alpha \cap \beta = \phi$;
result := (*result* - R_i) $\cup$ (R_i - β) $\cup$ (α, β);
end
else *done*:= true;

Note: each R_i is in 4NF, and decomposition is lossless-join

依據 4NF 進行解構 (範例)

R =(A, B, C, G, H, I)
D ={ $A \rightarrow\rightarrow B$, $B \rightarrow\rightarrow HI$, $CG \rightarrow\rightarrow H$ }
由於 A 不是 R 的超鍵，所以 R 不符合4NF

- Decomposition
  - a) R1 = (A, B) (R1 符合 4NF)
  - b) R2 = (A, C, G, H, I) (R2 不符合 4NF, 進一步解構)
  - c) R3 = (C, G, H) (R3 符合 4NF)
  - d) R4 = (A, C, G, I) (R4 不符合 4NF, 進一步解構)
    - $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$，根據遞移律 $A \rightarrow\rightarrow HI$，而對於 R4 而言 $A \rightarrow\rightarrow I$
  - e) R5 = (A, I) (R5 符合 4NF)
  - f) R6 = (A, C, G) (R6 符合 4NF)

4NF 解構

ID	Dept_name	Name	Address
X001	心靈感應系	X教授	超能學院
X001	心靈感應系	X教授	小三的家
X001	心理系	X教授	超能學院
X001	心理系	X教授	小三的家

解構

ID	Dept_name
X001	心靈感應系
X001	心理系

ID	Address
X001	超能學院
X001	小三的家

ID	Name
X001	X教授

符合 4NF

設計資料庫

- 一個關聯綱要 R 來自
  - 相對應的 E-R Model
  - 一個包含所有需要屬性的大 table ，然後藉由 normalization 將其解構為較小的 table
  - 由其他 ad-hoc 手段設計而來的關聯，需要測試其是否符合正規化，並加以適當的轉換

ER-Model 與 正規化

- 由一個仔細設計的 ER-Model ，並且依照規則轉換而來的 Relational Model，不需要正規化
- 然而，有時會有非鍵值屬性，對其他屬性產生功能相依，如：
  - 在 *employee* entity 中包含了 *department_name* 與 *building* 屬性，因此存 d*epartment_name*→ *building*
- 有時候非鍵值屬性的功能相依會來自 relationship 中的 屬性
  - 將所有 relationship 轉換為 binary 可以避免這種狀況

正規化與效能的衝突

- 為了效能需求，有時候會希望放置非正規化的資料表
- 例如，想要顯示課程的先修課程名稱，但若所有 schema 都符合正規化，則需要 join *course* 與 *prereq* 資料表
- 解決方案 1: 使用非正規化資料表，在 *prereq* 資料表中登載 course 資料表的其他屬性
  - 提升查詢速度，但是
  - 需要額外的儲存空間，當資料表更新時，需要額外的執行時間
  - 需要額外針對可能發生的錯誤進行檢查
- 解決方案 2: 使用 View
  - 無須額外撰寫錯誤檢查碼

其他與正規化無關的設計議題

- 應避免的不良資料庫設計範例
  - 將類別標籤以屬性方式儲存，如將 *earnings*(*company_id, year, amount*) 寫成 *earnings*(company_id, *earnings_2014, earnings_2015, earnings_2016 …*)
  - 這類設計叫做 **crosstab**
  - 在資料分析工具以及試算表中常見，但是不應該出現在資料庫中

時序資料

- **時序資料 (Temporal data) 表示資料的有效期間**
- **快照 (snapshot) 表示資料在某個時間點的狀態值)**
- 如何另令 ER model 可以表達時序資料？
  - attributes, e.g., 在某個時間點上，雇員的住址
  - entities, e.g., 在某段時間內有效的學籍資料
  - relationships, e.g., 學生在某教授指導下的期間
- 目前沒有共通標準

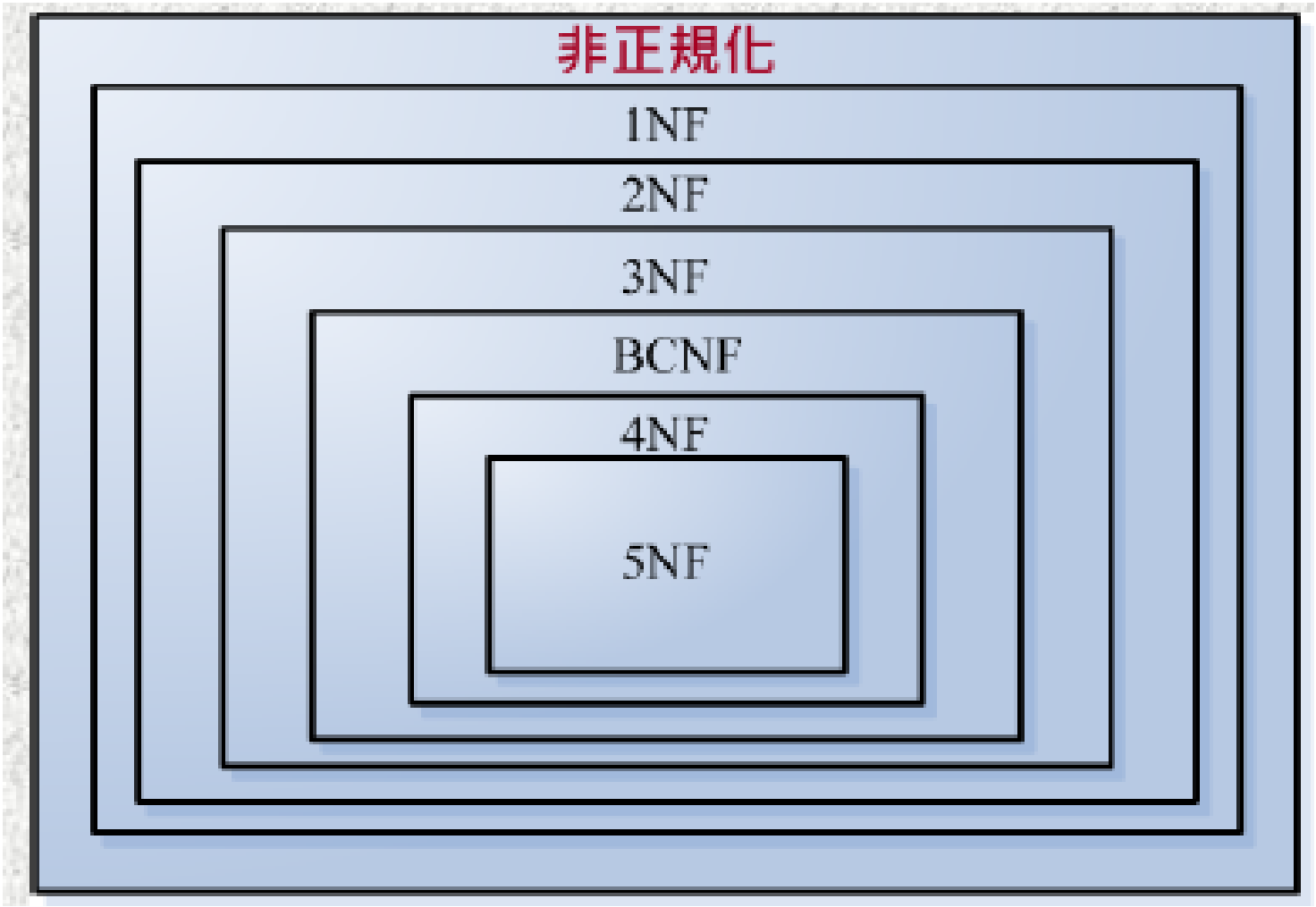
時序資料(cont.)

- 時序令 *ID* → *street, city* 不成立，因為地址有可能隨時間的流逝而改變
- 一個時序功能相依 X → Y 必須在一個關聯綱要的所有快照中都成立才算成立
- 實務上，設計者可能會在資料表中加上起始時間與結束時間的屬性
  - E.g., *course*(*course_id, course_title*) 被改成
    course(*course_id, course_title, start, end*)
  - Constraint: 任意兩個 Tuple 其存在時程不得重疊
    - 難以實作
- 外來鍵也會加上參考的時間點
  - E.g., 學生課表應該參照本學期的課程資訊

Summary

- 正規化是一種在設計資料庫時避免資料表中出現冗餘資訊的系統化方法
- 好的資料庫設計希望能遵守
  - Boyce-Codd 正規化 (BCNF)
  - 資訊無損失的解構 (Lossless Join Decomposition)
  - 相依保存 (Dependency Preservation)
- 在無法嚴格遵守上述規範的狀況下，使用妥協的正規化，如 3NF
- 在功能相依無法完整描述資料表的狀況下，使用多值相依與 4NF

補充範例



補充範例

學號	姓名	性別	課程代碼	課程名稱	學分	必選修	成績	老師ID	老師姓名
S001	幫人	男	C001 C101	資料庫 古蹟巡禮	3 2	必 選	85 99	T001 T101	HJ 古人
S002	欲享	男	C001 C102 C002	資料庫 咖啡實務 密碼學	3 2 3	必 選 選	77 60 75	T001 T102 T002	HJ 優良教師 優良教師

補充範例 (1NF)

學號	姓名	性別	課程代碼	課程名稱	學分	必選修	成績	老師ID	老師姓名
S001	幫人	男	C001	資料庫	3	必	85	T001	HJ
S001	幫人	男	C101	古蹟巡禮	2	選	99	T101	古人
S002	欲享	男	C001	資料庫	3	必	77	T001	HJ
S002	欲享	男	C102	咖啡實務	2	選	60	T102	優良教師
S002	欲享	男	C002	密碼學	3	選	75	T002	優良教師

補充範例 (BCNF)

學號	課程代碼	成績
S001	C001	85
S001	C101	99
S002	C001	77
S002	C102	60
S002	C002	75

學號	姓名	性別
S001	幫人	男
S002	欲享	男

老師ID	老師姓名
T001	HJ
T101	古人
T102	優良教師
T002	優良教師

課程代碼	課程名稱	學分	必選修	老師ID
C001	資料庫	3	必	T001
C101	古蹟巡禮	2	選	T101
C001	資料庫	3	必	T001
C102	咖啡實務	2	選	T102
C002	密碼學	3	選	T002