SAPIENZA
UNIVERSITÀ DI ROMA

Appunti di Basi Dati Modulo I

Colacel Alexandru Andrei

Disclaimer

Le fonti sono le "Hand Notes" del prof. Perelli tradotte in italiano, appunti presi dalle slides della prof. De Marsico ed eventuali e-mail.

Indice

1 Lemma della Chiusura **2**
1.1 Dimostrazione $\Rightarrow$ . . . 2
1.2 Dimostrazione $\Leftarrow$ . . . 2

2 Teorema $F^+ = F^A$ **3**
2.1 Dimostrazione $F^A \subseteq F^+$ . . . 3
2.2 Dimostrazione $F^+ \subseteq F^A$ . . . 5

3 Chiusura di X **6**
3.1 Teorema: L'algoritmo computa X_F^+ . . . 6

4 Lemma: Inclusione delle chiusure **8**
4.1 Dimostrazione . . . 8

5 Chiusura di X in G **9**
5.1 Dimostrazione . . . 9

6 Join senza perdita **11**
6.1 Dimostrazione $r \subseteq m_\rho(r)$ . . . 11
6.2 Dimostrazione $\pi_{R_i}(m_\rho(r)) = \pi_{R_i}(r) \forall R_i \in \rho$ . . . 11
6.3 Dimostrazione $m_\rho(m_\rho(r)) = m_\rho(r)$ . . . 11
6.4 Algoritmo controllo presenza join senza perdita . . . 12
6.4.1 Commenti sull'algoritmo . . . 12

7 Assiomi di Armstrong **14**

8 Decomposizione che preserva F **15**
8.1 Prima proprietà . . . 15
8.2 Seconda proprietà . . . 15
8.3 Terza proprietà . . . 15
8.4 Definizione G . . . 15

9 Dimostrazione ρ preserva F **16**

10 Hash **17**

11 B-tree **18**

12 Chiusura di un insieme di attributi **19**
12.1 Algoritmo . . . 19
12.2 Dimostrazione correttezza . . . 19

13 3NF (per dipendenza transitiva) **20**

14 Chiusura di F e primo lemma **21**

15 Isam **22**
15.1 Variante Isam con chiavi indice che hanno valore ultimo record . . . 22

16 Altre definizioni da sapere **23**
16.1 Chiave minimale . . . 23
16.2 Superchiave . . . 23

1 Lemma della Chiusura

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R. Si ha che:

$$X \to Y \in F^A \iff Y \subseteq X^+$$

1.1 Dimostrazione $\Rightarrow$

Dato $X \to Y \in F^A$, per la regola della decomposizione, otteniamo:

$$X \to A \in F^A, \quad \forall A \in Y$$

e quindi, per definizione di X^+, otteniamo che:

$$A \in X^+, \quad \forall A \in Y$$

che significa:

$$Y \subseteq X^+$$

1.2 Dimostrazione $\Leftarrow$

Dato:

$$Y \subseteq X^+$$

si ottiene che:

$$X \to A \in F^A \quad \forall A \in Y$$

che implica, per la regola dell'unione, che:

$$X \to Y \in F^A$$

2 Teorema $F^+ = F^A$

Dato uno schema R e un insieme F di dipendenze funzionali definite su R, si ha che:

$$F^+ = F^A$$

2.1 Dimostrazione $F^A \subseteq F^+$

Prendiamo $X \to Y \in F^A$, noi dobbiamo provare che $X \to Y \in F^+$ per induzione con n numero di applicazioni degli assiomi di Armstrong.

- **Caso base** (n = 0): se $X \to Y \in F^A$ senza aver applicato alcun assioma di Armstrong, allora l'unica possibilità è che:
$$X \to Y \in F \subseteq F^+$$
- **Ipotesi induttiva forte:** ogni dipendenza funzionale in F^A ottenuta da F applicando k $\leq$ n assiomi di Armstrong è anche in F^+:
$$X \to Y \in F^A \text{ tramite } k \leq n \text{ assiomi} \Rightarrow X \to Y \in F^+$$
- **Passo induttivo:** è necessario dimostrare che se $X \to Y \in F^A$ dopo aver applicato $n+1$ assiomi di Armstrong, allora $X \to Y \in F^+$.

  È possibile ritrovarsi in uno dei seguenti tre casi:

  1. Se l'$(n+1)$-esimo assioma applicato è l'assioma di **riflessività**, allora l'unica possibilità è che:
  $$X \to Y \in F^A \Leftrightarrow Y \subseteq X \subseteq R$$
  Dunque, poiché, $Y \subseteq X \subseteq R$, per ogni istanza legale di R si ha che:
  $$\forall t_1, t_2 \in r_1, t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$
  da cui ne segue automaticamente che $X \to Y \in F^+$
  2. Se l'$(n+1)$-esimo assioma applicato è l'assioma di **aumento**, allora è obbligatoriamente necessario che:
     - $\exists V, W \subseteq R \,|\, \exists V \to W \in F_A$, ottenuta applicando $j \leq n$ assiomi di Armstrong
     - $\exists Z \subseteq R \,|\, X := VZ, Y := WZ$

     Affinché si abbia che:
     $$Z \subseteq R,\, V \to W \Rightarrow VZ \to WZ = X \to Y \in F^A$$
     Siccome per ipotesi induttiva si ha $V \to W \in F^A \Rightarrow V \to W \in F^+$ e siccome $Z \subseteq Z \Rightarrow Z \to Z \in F^+$, si vede facilmente che:
     $$\begin{cases} V \to W \in F^+ \\ Z \to Z \in F^+ \end{cases} \implies \begin{cases} \forall t_1, t_2 \in r, t_1[V] = t_2[V] \implies t_1[W] = t_2[W] \\ \forall t_1, t_2 \in r, t_1[Z] = t_2[Z] \implies t_1[Z] = t_2[Z] \end{cases} \implies$$
     $$\implies \forall t_1, t_2 \in r, t_1[VZ] = t_2[VZ] \implies t_1[WZ] = t_2[WZ] \implies$$
     $$\implies \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y] \implies X \to Y \in F^+$$

3. Se l'$(n+1)$-esimo assioma applicato è l'assioma di **transitività**, allora è obbligatoriamente necessario che $\exists X \to Z, Z \to Y \in F^A$, ottenute con $k \leq n$ assiomi di Armstrong, affinché si abbia che:

$$X \to Z \in F^A \vee Z \to Y \in F^A \Rightarrow X \to Y \in F^A$$

Siccome per ipotesi induttiva $X \to Z \in F^A \Rightarrow X \to Z \in F^+$ e $Z \to Y \in F^A \Rightarrow Z \to Y \in F^+$, si vede facilmente che:

$$\begin{cases} X \to Z \in F^+ \\ Z \to Y \in F^+ \end{cases} \implies$$

$$\implies \forall t_1, t_2 \in r, t_1[X] = t_2[X] \implies t_1[Z] = t_2[Z] \implies t_1[Y] = t_2[Y] \implies$$

$$\implies X \to Y \in F^+$$

2.2 Dimostrazione $F^+ \subseteq F^A$

- Sia $X \subseteq R$ e sia r istanza di $R(X^+, R - X^+)$ tale che

X^+			$R - X^+$		
A_1	$\cdots$	A_i	A_j	$\cdots$	A_n
1	...	1	1	...	1
1	...	1	0	...	0

  dunque tale che $\forall t_1, t_2 \in r$ si ha:

  * $t_1[X^+] = (1, \ldots, 1) = t_2[X^+]$
  * $t_1[R - X^+] = (1, \ldots, 1) \neq (0, \ldots, 0) = t_2[R - X^+]$

- Notiamo che $\forall V, W \subseteq R \mid V \to W \in F$ si ha che:

  * Se $V \cap R - X^+ \neq \emptyset$ (dunque anche se $V \subseteq R - X^+$) allora $t_1[V] \neq t_2[V]$, dunque r soddisfa $V \to W \in F$
  * Se invece $V \subseteq X^+$, per il lemma precedentemente visto si ha che

$$V \subseteq X^+ \iff X \to V \in F^A$$

    Siccome $V \to W \in F \implies V \to W \in F^A$, per transitività si ha che

$$X \to V \in F^A \land V \to W \in F^A \implies X \to W \in F^A \iff W \subseteq X^+$$

    Dunque, siccome $V, W \subseteq X^+$, in definitiva si ha che

$$t_1, t_2 \in r, t_1[V] = (1, \ldots, 1) = t_2[V] \land t_1[W] = (1, \ldots, 1) = t_2[W]$$

    e quindi r soddisfa ogni $V \to W \in F$

- Siccome in entrambi i casi r soddisfa ogni $V \to W \in F$, allora r è legale.
- A questo punto, una qualsiasi dipendenza $X \to Y \in F^+$ deve essere soddisfatta da qualsiasi istanza legale di R, inclusa r stessa
- Poiché $X \subseteq X^+$, ne segue che la dipendenza non può essere soddisfatta a vuoto poiché $t_1[X] = t_2[X]$. Dunque, l'unica possibilità affinché $X \to A \in F^+$ sia soddisfatta da r è che $Y \subseteq X^+$ in modo che si abbia $t_1[Y] = t_2[Y]$
- A questo punto, per il lemma si ha che $Y \subseteq X^+ \iff X \to Y \in F^A$
- Dunque, siccome $X \to Y \in F^+ \implies X \to Y \in F^A$, concludiamo che $F^A \subseteq F^+$

Nota

Poiché $F^+ = F^A$, per calcolare F^+ ci basta applicare gli assiomi di Armstrong sulle dipendenze in F in modo da trovare F^A.
Tuttavia, calcolare $F^+ = F^A$ richiede tempo esponenziale, quindi $O(2^{nk})$: considerando anche solo l'assioma di riflessività, siccome ogni possibile sottoinsieme di R genera una dipendenza e siccome i sottoinsiemi possibili di R sono $2^{|R|}$, allora ne segue che $|F^+| >> 2^{|R|}$.

3 Chiusura di X

`Input`:

- Relazione R
- Dipendenze Funzionali F
- Insieme $X \subseteq R$

`Output`: X^+

Algorithm 1 Closure Algorithm

```
1: Z ← X
2: S ← {A | ∃Y → V ∈ F, A ∈ V ∧ Y ⊆ Z}
3: while S ⊈ Z do
4:     Z ← Z ∪ S
5:     S ← {A | ∃Y → V ∈ F, A ∈ V ∧ Y ⊆ Z}
6: end while
7: return Z
```

Tale algoritmo viene eseguito in tempo polinomiale, ossia On^k

3.1 Teorema: L'algoritmo computa X_F^+

Il teorema calcola correttamente la chiusura di un insieme di attributi X rispetto ad un insieme F di dipendenze funzionali.

Dimostrazione:
Denotiamo:

$$Z_0, Z_1, \ldots, Z_i, \ldots$$

$$S_0, S_1, \ldots, S_i, \ldots$$

Indichiamo con Z_0 il valore iniiale di $Z(Z_0 = X)$ e con Z_i e S_i, $i \geq 1$ i valori di Z ed S dopo l'i-esima esecuzione del corpo del ciclo, infatti notiamo che $Z_i \subseteq Z_{i+1}$, per ogni i.

Sia j tale che $S_j \subseteq Z_j$ (cioè, Z_j è l'output di Z quando l'algoritmo termina); proveremo che:

$$A \in Z_j \Leftrightarrow A \in X^+$$

- Dimostriamo per induzione su i che $Z_f \subseteq X^+$
  - Caso base dell'induzione: i = 0.
    Alla 0-esima iterazione del while (ossia prima di esso) si ha $Z_0 = X$ e $X \subseteq X^+$
  - Ipotesi induttiva: Per ogni $i \in \mathbb{N}$ si ha che $Z_i \subseteq X^+$
  - **Passo induttivo** $(i > 0)$: Dato $A \in Z_{i+1} = Z_i \cup S_i$, si ha che $A \in Z_i \vee A \in S_i$.
    Dunque, si possono verificare due casi:
    * Se $A \in Z_i,$ allora per ipotesi $A \in Z_i \subseteq X^+$.
    * Se $A \in S_i,$ allora $\exists Y \to V \in F$ tale che $A \in V \subseteq R, Y \subseteq Z_i$.
      Siccome per ipotesi $Z_i \subseteq X^+$ e $Y \subseteq Z_i,$ allora $Y \subseteq Z_i \subseteq X^+ \iff X \to Y \in F^A$ e $Y \to V \in F \implies Y \to V \in F^A,$ quindi per transitività si ha che:

      $$X \to Y, Y \to V \in F^A \implies X \to V \in F^A \iff V \subseteq X^+$$

      Dunque,si ha che $A \in V \subseteq X^+$
    * Siccome in entrambi i casi $A \in Z_{i+1} \implies A \in X^+,$ allora concludiamo che $Z_{i+1} \subseteq X^+$.

- Dimostriamo ora che $X^+ \subseteq Z_i$:
  - Sia $X \subseteq R$ e sia r istanza di $R(Z_i, R - Z_i)$.

Z_i			$R - Z_i$		
A_1	...	A_i	A_j	...	A_n
1	...	1	1	...	1
1	...	1	0	...	0

    dunque tale che per $t_1, t_2 \in r$ si ha:
    - $t_1[Z_i] = (1, \ldots, 1) = t_2[Z_i]$
    - $t_1[R - Z_i] = (1, \ldots, 1) \neq (0, \ldots, 0) = t_2[R - Z_i]$
  - Notiamo che $\forall V, W \subseteq R$ — $V \to W \in F$ si ha che:
    - Se $V \cap (R - Z_i) \neq \emptyset$ (quindi anche se $V \subseteq R - Z_i$) allora $t1[V] \neq t2[V]$, quindi r soddisfa $V \to W \in F$
    - Se invece $V \subseteq Z_i$, allora $W \subseteq S_f$, poiché per come viene calcolato S_f, si ha che:
      $V \to W \in F$, $V \subseteq Z_i$, $B \in W \subseteq R \Rightarrow B \in S_f \Rightarrow W \subseteq S_f$
      e quindi, siccome $S_f \subseteq Z_i$ è la condizione che termina l'algoritmo, allora $W \subseteq S_f \subseteq Z_i$
    - Siccome $V, W \subseteq Z_i$, in definitiva si ha che
      $t1, t2 \in r$, $t1[V] = (1, \ldots, 1) = t2[V]$ e $t1[W] = (1, \ldots, 1) = t2[W]$,
      e quindi r soddisfa $V \to W \in F$
  - Siccome in entrambi i casi r soddisfa ogni $V \to W \in F$, allora r è legale.
  - A questo punto, dato $A \in X^+$, si ha che $X \to A \in F^A = F^+$ deve essere soddisfatta da qualsiasi istanza legale di R, inclusa r stessa.
  - Poiché $X = Z_0 \subseteq Z_i$, ne segue che la dipendenza non può essere soddisfatta a vuoto poiché $t1[X] = t2[X]$. Dunque, l'unica possibilità affinché $X \to A \in F^+$ sia soddisfatta da r è $A \in Z_i$ in modo che si abbia $t1[A] = t2[A]$.
  - Dunque, siccome $A \in X^+ \Rightarrow A \in Z_i$, concludiamo che $X^+ \subseteq Z_i$.

4 Lemma: Inclusione delle chiusure

Dato uno schema R e due insiemi F e G di dipendenze funzionali su R, si ha che:

$$F \subseteq G^+ \Leftrightarrow F^+ \subseteq G^+$$

4.1 Dimostrazione

- Denotiamo come $G \xrightarrow{\text{A}} F$ la possibilità di ottenere F partendo da G applicando una determinata quantità di assiomi di Armstrong.
- Ricordando che G^A è l'insieme di tutte le dipendenze funzionali ottenibile applicando assiomi di Armstrong su G, allora:

$$G \xrightarrow{\text{A}} F \Leftrightarrow \forall X \to Y \in F, \text{ si ha } X \to Y \in G^A = G^+ \Leftrightarrow F \subseteq G^+$$

- Siccome $F \subseteq G \Leftrightarrow G \xrightarrow{\text{A}} F$, per definizione di $F^A = F^+$ si ha che:

$$F \subseteq G^+ \Rightarrow G \xrightarrow{\text{A}} F \xrightarrow{\text{A}} F^A = F^+ \Rightarrow F^+ \subseteq G^+$$

- Viceversa, si ha che $F^+ \subseteq G^+ \Rightarrow F \subseteq F^+ \subseteq G^+$, quindi concludiamo che $F \subseteq G^+ \Leftrightarrow F^+ \subseteq G^+$

5 Chiusura di X in G

Dato uno schema R con decomposizione $\rho = R_1, ..., R_k$, dato un insieme F di dipendenze funzionali su R e posto:

$$G := \bigcap_{i=0}^{k} \pi_{R_i}(F)$$

preso $X \subseteq R$, il seguente algoritmo calcola X_G^+ tramite F:

Algorithm 2 Calcolo di X_G^+ tramite F

```
1:  procedure CalculateX_G^+(R: schema, F: set of dependencies, X: set of attributes)
2:      Z ← X
3:      S ← ∅
4:      for i ← 1 to k do
5:          S ← S ∪ ((Z ∩ R_i)_F^+ ∩ R_i)
6:      end for
7:      while S ⊈ Z do
8:          Z ← Z ∪ S
9:          for i ← 1 to k do
10:             S ← S ∪ ((Z ∩ R_i)_F^+ ∩ R_i)
11:         end for
12:     end while
13:     X_G^+ ← Z
14:     return X_G^+
15: end procedure
```

5.1 Dimostrazione

- Siano $Z_0, Z_1, \ldots, Z_i, \ldots$ e $S_0, S_1, \ldots, S_i, \ldots$ gli insiemi calcolati ad ogni iterazione del ciclo while dell'algoritmo
- Osserviamo che $Z_i \subseteq Z_{i+1}$ per ogni $i \in \mathbb{N}$, dunque $Z_0, Z_1, \ldots, Z_i, \ldots$ è una sequenza monotona limitata da R, implicando che esiste un $f \in \mathbb{N}$ tale che $Z_f = Z_{f+1}$
- Siccome ciò può accadere solo se $S_f \subseteq Z_f$, ossia quando l'algoritmo termina, si ha che Z_f è l'output dell'algoritmo
- Dimostriamo per induzione che $Z_f \subseteq X_G^+$:
  - **Caso base ($i = 0$):**
    Alla 0-esima iterazione del `while`, prima di esso, si ha $Z_0 = X \subseteq X_G^+$.
  - **Ipotesi induttiva:**
    Per ogni $i \in \mathbb{N}$ si ha che $Z_i \subseteq X_G^+$.
  - **Passo induttivo ($i > 0$):**
    Dato $A \in Z_{i+1} = Z_i \cup S_i$, abbiamo che $A \in Z_i$ oppure $A \in S_i$. Quindi possiamo considerare due casi:
    - Se $A \in Z_i$, allora per ipotesi induttiva abbiamo che $A \in Z_i \subseteq X_G^+$.
    - Se $A \in S_i$, allora per la definizione stessa di S_i, $\exists\ j \leq k$ tale che $A \in ((Z_i \cap R_j)_F^+ \cap R_j)$.

      A questo punto, abbiamo che:

      $$A \in ((Z_i \cap R_j)_F^+ \cap R_j) \Leftrightarrow A \in (Z_i \cap R_j)_F^+ \wedge A \in R_j$$

      da cui otteniamo che:

      $$A \in (Z_i \cap R_j)_F^+ \Leftrightarrow (Z_i \cap R_j) \rightarrow A \in F^A = F^+.$$

      Dunque, siccome $(Z_i \cap R_j) \subseteq R_j$ e $A \in R_j$, allora si ha che:

$(Z_i \cap R_j) \to A \in \pi_{R_j}(F) = \{X \to Y \in F^+ \mid XY \in R_j\}$
Dai quali deduciamo che:

$(Z_i \cap R_j) \to A \in \pi_{R_j}(F) \subseteq G \subseteq G^+ = G^A$.

Inoltre, siccome $(Z_i \cap R_j) \subseteq Z_i$ e, per ipotesi induttiva, $Z_i \subseteq X_G^+$, otteniamo che:

$(Z_i \cap R_j) \subseteq Zi \subseteq X_G^+$

implicando quindi che $X \to (Z_i \cap R_j) \in G^A$.
Infine, per transitività otteniamo che:

$$X \to (Z_i \cap R_j),\ (Z_i \cap R_j) \to A \in G^A \Rightarrow X \to A \in G^A \Rightarrow A \in X_G^+$$

- Dunque, siccome in entrambi i casi si ha $A \in Z_f \Rightarrow A \in X_G^+$, possiamo concludere che $Z_f \subseteq X_G^+$.

- Dimostriamo per induzione che $X_G^+ \subseteq Z_f$[1]:
  - Osserviamo che $X \subseteq Y \Rightarrow X_F^+ \subseteq Y_F^+$
  - Adesso sippiamo che $X = Z_0 \subseteq Z_f$, noi otteniamo che $X_G^+ \subseteq (Z_f)_G^+$. Noi possiamo dimostrare che $Z_f = (Z_f)_G^+$ che prova questo stato. Ovviamente $Z_f \subseteq (Z_f)_G^+$
  - Dobbiamo ora dimostrare l'Inclusione. Consideriamo $S_1 = A|Y \to V \in G, Y \subseteq Z_f, A \in V$ e otteniamo per l'esecuizione del primo passo della chiususa dell'algoritmo di Z_f su G. Noi abbiamo dimostrare che $S_1 \subseteq Z_f$. Poniamo $A \in S_1$. Questo significa che esiste $Y \to V \in G$ tale che $Y \subseteq Z_f$ e $A \in V$.
  - Adesso per definizione di F, esiste un indice j che $Y, V \subseteq R_j$ che significa $Y \subseteq Z_f \cap R_j$ e $A \in R_j$ e implica che $A \in (Z_j \cap R_j)_F^+ \cap R_j$.
  - Detto questo $A \in S_f \subseteq Z_f$

Abbiamo dimostrato che $Z_f = X_G^+$

[1] Dalle slides del Prof "Hand Notes"

6 Join senza perdita

Teorema Sia R uno schema con decomposizione $\rho = R_1, \ldots, R_k$ e sia F un insieme di dipendenze funzionali su R. La decomposizione ρ presenta un join senza perdita se per ogni istanza legale r di R si ha che:

$$r = m_\rho(r) := \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_k}(r)$$

Sia R uno schema con decomposizione $\rho = R_1, \ldots, R_k$ e sia F un insieme di dipendenze funzionali su R. Posto $m_\rho(r) := \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_k}(r)$, per ogni istanza legale r di R si ha che:

1. $r \subseteq m_\rho(r)$
2. $\pi_{R_i}(m_\rho(r)) = \pi_{R_i}(r)$, per ogni $R_i \in \rho$
3. $m_\rho(m_\rho(r)) = m_\rho(r)$

6.1 Dimostrazione $r \subseteq m_\rho(r)$

Data una qualsiasi tupla $t \in r$, si ha che:

$$t \in r \Rightarrow t \in \{t[R_1]\} \bowtie \ldots \bowtie \{t[R_k]\} \subseteq \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_k}(r) = m_\rho(r)$$

dunque $r \subseteq m_\rho(r)$.

6.2 Dimostrazione $\pi_{R_i}(m_\rho(r)) = \pi_{R_i}(r) \forall R_i \in \rho$

Poiché $r \subseteq m_\rho(r)$, allora effettuando una proiezione con $R_i \in \rho$ su entrambe, ne segue che $\pi_{R_i}(r) \subseteq \pi_{R_i}(m_\rho(r))$.

Inoltre, per definizione di proiezione, si ha che:

$$t \in \pi_{R_i}(m_\rho(r)) \Rightarrow \exists t' \in m_\rho(r) \text{ tale che } t_{R_i} = t'[R_i]$$

Infine, per definizione stessa di $m_\rho(r)$, si ha che:

$$t' \in m_\rho(r) \Rightarrow \exists t_1, \ldots, t_k \in r \text{ tale che } \forall R_j \in \rho, t_j[R_j] = t'[R_j]$$

In particolare, quindi, otteniamo che:

$$t_{R_i} \in \pi_{R_1}(m_\rho(r)) \Rightarrow t_{R_i} = t'[R_i] = t_i[R_i] \in \pi_{R_i}(r) \Rightarrow \pi_{R_1}(m_\rho(r)) \subseteq \pi_{R_i}(r)$$

6.3 Dimostrazione $m_\rho(m_\rho(r)) = m_\rho(r)$

Siccome $\pi_{R_i}(r) = \pi_{R_1}(m_\rho(r))$, allora si ha che:

$$m_\rho(m_\rho(r)) = \pi_{R_1}(m_\rho(r)) \bowtie \ldots \bowtie \pi_{R_k}(m_\rho(r)) = \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_k}(r) = m_\rho(r)$$

6.4 Algoritmo controllo presenza join senza perdita

Dato uno schema $R = A_1, \ldots, A_n$ con decomposizione $\rho = R_1, \ldots, R_k$ e un insieme F di dipendenze funzionali su R, presa l'istanza legale r di R dove per ogni $i \in [1, k]$ e per ogni $j \in [1, n]$ si ha:

Un sistema di equazioni con l'ambiente `cases`:

$$r_{i,j} \begin{cases} \text{"}a\text{"} & se A_j \in R \\ \text{"}b_i\text{"} & se A_j \notin R \end{cases}$$

il seguente algoritmo determina se ρ presenta un join senza perdita.

Algorithm 3 Verifica se ρ ha un join senza perdita

```
1:  function HasLosslessJoin(R, F, ρ)
2:      unchanged ← False
3:      while not unchanged do
4:          unchanged ← True
5:          for X → Y ∈ F do
6:              for t1 in r do
7:                  for t2 in r do
8:                      if t1[X] = t2[X] and t1[Y] ≠ t2[Y] then
9:                          unchanged ← False
10:                         for A_j ∈ Y do
11:                             if t1[A_j] = "a" then
12:                                 t2[A_j] ← t1[A_j]
13:                             else
14:                                 t1[A_j] ← t2[A_j]
15:                             end if
16:                         end for
17:                     end if
18:                 end for
19:             end for
20:         end for
21:     end while
22:     for t ∈ r do
23:         if t = ("a", ..., "a") then
24:             return True
25:         end if
26:     end for
27:     return False
28: end function
```

6.4.1 Commenti sull'algoritmo

- L'algoritmo modifica l'istanza di partenza r in modo che tutte le dipendenze di F vengano soddisfatte:
- Ogni volta che l'algoritmo trova due tuple aventi lo stesso valore nel determinante ma valori differenti, quest'ultimo viene modificato in modo che essi siano uguali.
- Nel fare ciò, il simbolo "a" viene considerato prioritario, ovvero "a" non può mai diventare "b", mentre "b" può diventare "a".
- Se due tuple hanno lo stesso valore nel determinante ma valori differenti nel determinato, e solo una delle due tuple ha un valore "a" nel determinato, il valore "b" dell'altra tupla viene cambiato in "a".
- Se due tuple hanno lo stesso valore nel determinante ma valori differenti nel determinato, ma nessuna delle due tuple ha un valore "a" nel determinato, il valore "b" di una delle due tuple viene cambiato in modo che esse abbiano lo stesso valore "b".
- Due valori vengono considerati uguali se sono entrambi "a" (indipendentemente dal pedice che hanno) o se entrambi hanno una "b" con lo stesso identico pedice.

- L'algoritmo termina quando tutte le coppie di tuple soddisfano le dipendenze di F.
- Infine, r diventa un'istanza legale di R.
- Una volta terminato l'algoritmo, se esiste almeno una tupla avente tutti valori "a" al suo interno, allora ρ presenta un join senza perdita, altrimenti no.

7 Assiomi di Armstrong

8 Decomposizione che preserva F

8.1 Prima proprietà

8.2 Seconda proprietà

8.3 Terza proprietà

8.4 Definizione G

9 Dimostrazione ρ preserva F

10 Hash

11 B-tree

12 Chiusura di un insieme di attributi

12.1 Algoritmo

12.2 Dimostrazione correttezza

13 3NF (per dipendenza transitiva)

14 Chiusura di F e primo lemma

15 Isam

15.1 Variante Isam con chiavi indice che hanno valore ultimo record

16 Altre definizioni da sapere

16.1 Chiave minimale

16.2 Superchiave