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

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:
  $$\forall t_1, t_2 \in r, t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$
  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.
   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

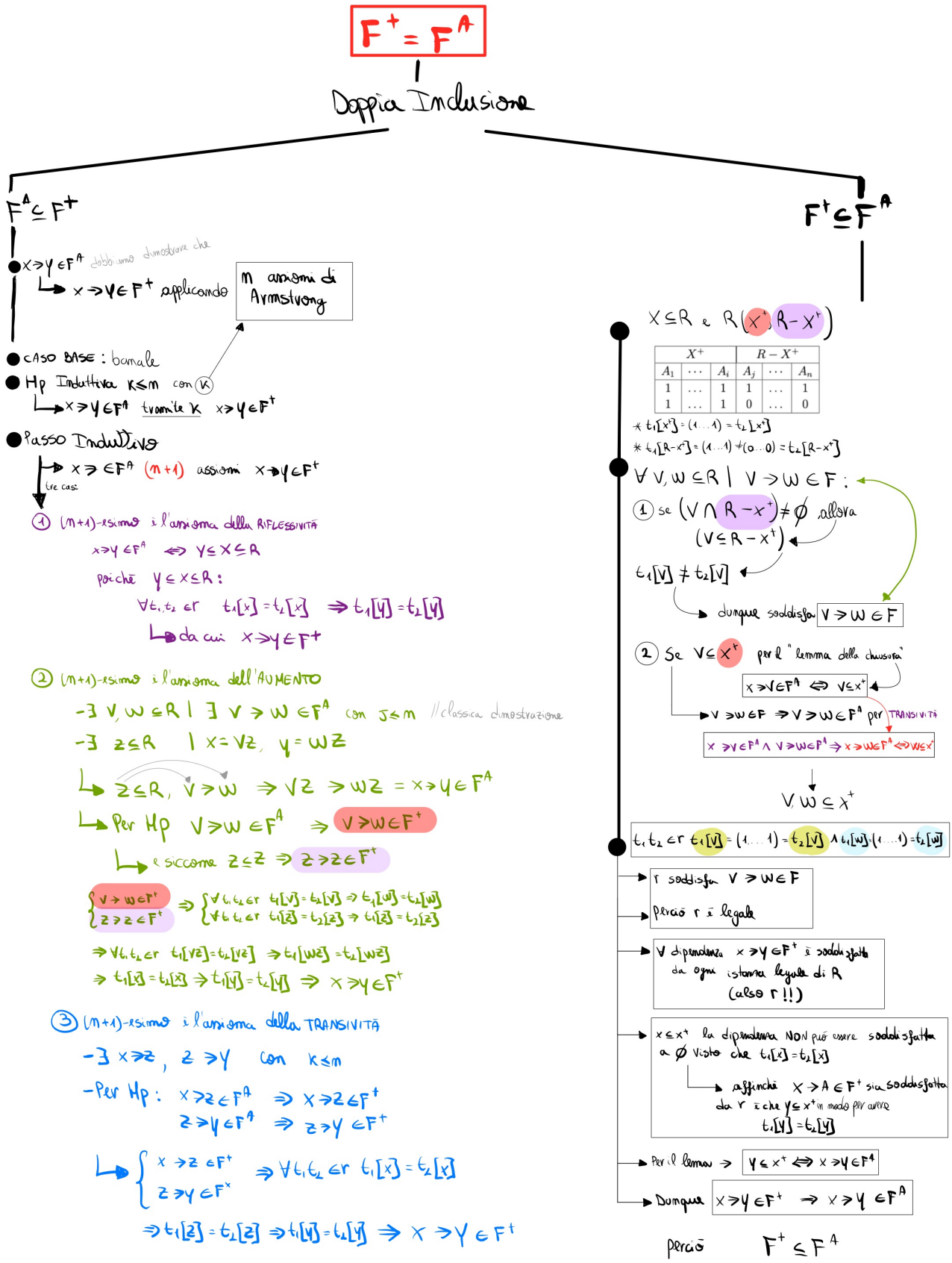
$$t_1, t_2 \in r, t_1[V] = (1, \ldots, 1) = t_2[V] \land t_1[W] = (1, \ldots, 1) = t_2[W]$$

    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 $|F^+| >> 2^{|R|}$.


F⁺ = F^A
Doppia Inclusione
F^A ⊆ F⁺
X→Y ∈ F^A dobbiamo dimostrare che
X→Y ∈ F⁺ applicando
m assiomi di Armstrong
CASO BASE: banale
Hp Induttiva k≤m con (k)
X→Y ∈ F^A tramite k X→Y ∈ F⁺
Passo Induttivo
X→ ∈ F^A (n+1) assiomi X→Y ∈ F⁺
tre casi:
(n+1)-esimo è l'assioma della RIFLESSIVITÀ
X→Y ∈ F^A ⇔ Y ⊆ X ⊆ R
poiché Y ∈ X ⊆ R:
∀t1,t2 ∈ r t1[X] = t2[X] ⇒ t1[Y] = t2[Y]
da cui X→Y ∈ F⁺
(n+1)-esimo è l'assioma dell'AUMENTO
-∃ V,W ⊆ R | ∃ V→W ∈ F^A con J≤m //classica dimostrazione
-∃ Z ⊆ R | X = VZ, Y = WZ
Z ⊆ R, V→W ⇒ VZ→WZ = X→Y ∈ F^A
Per Hp V→W ∈ F^A ⇒ V→W ∈ F⁺
e siccome Z⊆Z ⇒ Z→Z ∈ F⁺
V→W ∈ F⁺, Z→Z ∈ F⁺ ⇒ ∀t1,t2 ∈ r t1[V]=t2[V] ⇒ t1[W]=t2[W], ∀t1,t2 ∈ r t1[Z]=t2[Z] ⇒ t1[Z]=t2[Z]
⇒ ∀t1,t2 ∈ r t1[VZ]=t2[VZ] ⇒ t1[WZ]=t2[WZ]
⇒ t1[X]=t2[X] ⇒ t1[Y]=t2[Y] ⇒ X→Y ∈ F⁺
(n+1)-esimo è l'assioma della TRANSITIVITÀ
-∃ X→Z, Z→Y con k≤m
-Per Hp: X→Z ∈ F^A ⇒ X→Z ∈ F⁺
Z→Y ∈ F^A ⇒ Z→Y ∈ F⁺
X→Z ∈ F⁺, Z→Y ∈ F^A ⇒ ∀t1,t2 ∈ r t1[X] = t2[X]
⇒ t1[Z] = t2[Z] ⇒ t1[Y] = t2[Y] ⇒ X→Y ∈ F⁺
F⁺ ⊆ F^A
X ⊆ R e R(X⁺, R−X⁺)
X⁺ | R − X⁺
A1 ... Ai | Aj ... An
1 ... 1 | 1 ... 1
1 ... 1 | 0 ... 0
* t1[X⁺] = (1...1) = t2[X⁺]
* t1[R−X⁺] = (1...1) ≠ (0...0) = t2[R−X⁺]
∀ V,W ⊆ R | V→W ∈ F:
1 se (V ∩ R−X⁺) ≠ ∅ allora (V ⊆ R−X⁺)
t1[V] ≠ t2[V]
dunque soddisfa V→W ∈ F
2 Se V ⊆ X⁺ per il "lemma della chiusura"
X→V ∈ F^A ⇔ V ⊆ X⁺
V→W ∈ F ⇒ V→W ∈ F^A per TRANSITIVITÀ
X→V ∈ F^A ∧ V→W ∈ F^A ⇒ X→W ∈ F^A ⇔ W ⊆ X⁺
V,W ⊆ X⁺
t1,t2 ∈ r t1[V] = (1...1) = t2[V] ∧ t1[W] = (1...1) = t2[W]
r soddisfa V→W ∈ F
perciò r è legale
∀ dipendenza X→Y ∈ F⁺ è soddisfatta da ogni istanza legale di R (anche r!!)
X ⊆ X⁺ la dipendenza NON può essere soddisfatta a ∅ visto che t1[X] = t2[X]
affinché X→A ∈ F⁺ sia soddisfatta da r è che Y ⊆ X⁺ in modo per avere t1[Y] = t2[Y]
Per il lemma → Y ⊆ X⁺ ⇔ X→Y ∈ F^A
Dunque X→Y ∈ F⁺ ⇒ X→Y ∈ F^A
perciò F⁺ ⊆ F^A

3 Chiusura di X

3.1 Chiusura di un insieme di attributi

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R.

Dato $X \subseteq R$, definiamo come chiusura di X rispetto a F, indicata con X_F^+ (o solo X^+ se F è l'unico insieme di dipendenze su R), il seguente insieme:

$$X_F^+ = \{A \in R \mid X \to A \in F^A\}$$

dove $A \in R$ implica che A sia un singolo attributo di R.
`Input`:

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

`Output`: X^+

Algorithm 1 Algoritmo per la chiusura di un insieme di attributi

1: $Z \leftarrow X$
2: $S \leftarrow \{A \mid \exists Y \to V \in F, A \in V \wedge Y \subseteq Z\}$
3: **while** $S \not\subseteq Z$ **do**
4: $\quad Z \leftarrow Z \cup S$
5: $\quad S \leftarrow \{A \mid \exists Y \to V \in F, A \in V \wedge Y \subseteq Z\}$
6: **end while**
7: **return** Z

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

3.2 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 correttezza dell'algoritmo:

Denotiamo:

$$Z_0, Z_1, \ldots, Z_i, \ldots$$
$$S_0, S_1, \ldots, S_i, \ldots$$

Indichiamo con Z_0 il valore iniziale 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** 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

  Proviamo se legale:
  * Prendiamo una dipendenza funzionale $V \to W \in F$
    - Se $t_1[V] \neq t_2[V]$ allora r soddisfa $V \to W$
    - Se $t_1[V] = t_2[V]$ allora è verificato che $V \subseteq Z_i$. Otteniamo così che $W \subseteq S_i$, come ciascun attributo $B \in W$ è aggiunto a S_i ciò significa che $V \to W \in F$
    - Adesso se Z_i terminale noi otteniamo $W \subseteq S_i \subseteq Z_i$ e quindi che $t_1[W] = t_2[W]$ perciò r soddisfa $V \to W$
  * Essendo r un'istanza legale, noi abbiamo che r soddisfa $X \to A$. Adesso visto che abbiamo $X \subseteq Z_0 \subseteq Z_i$ otteniamo che $t_1[X] = t_2[X]$ e quindi che $t_1[A] = t_2[A] \Rightarrow A \in Z_i$

CHIUSURA DI X

ALGORITMO | $X \subseteq R$

```
Z = X
S = { A | ∃Y → V ∈ F , A ∈ V ∧ Y ⊆ Z }
While S ≠ Z do
    Z = Z ∪ S
    S = { A | ∃Y → V ∈ F , A ∈ V ∧ Y ⊆ Z }
end while
return Z
```

$X^+ = Z_f$

Denotiamo: $Z_0, Z_1, \dots, Z_i, \dots$; $S_0, S_1, \dots, S_i, \dots$ → **i-esima iterazione del while**

Sia j tale che $S_j \subseteq Z_j$: (output alla terminazione)

Proveremo che:

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

$Z_i \subseteq X^+$

- CASO BASE $i = 0 \Rightarrow Z_0 = X$ e $X \subseteq X^+$
- IPOTESI INDUTTIVA $\Rightarrow \forall i \in \mathbb{N}$ si ha che $Z_i \subseteq X^+$
- Passo Induttivo $i > 0 \Rightarrow A \in Z_{i+1} = Z_i \cup S_i \Rightarrow A \in Z_i \vee A \in S_i$

  Due casi:

  - if $A \in Z_i$ allora $A \in Z_i \subseteq X^+$
  - if $A \in S_i$ allora $\exists\, Y \to V \in F$ tale che $A \in V \subseteq R$, $Y \subseteq Z_i$

    Per Hp $Z_i \subseteq X^+$ e $Y \subseteq Z_i$ allora:

    X Lemma

    $$\left\{ \begin{array}{l} Y \subseteq Z_i \subseteq X^+ \iff X \to Y \in F^A \\ \text{e} \\ Y \to V \in F \Rightarrow Y \to V \in F^A \end{array} \right.$$

    X transitività

    $$X \to Y,\ Y \to V \in F^A \Rightarrow X \to V \in F^A \overset{\text{X Lemma}}{\iff} V \subseteq X^+$$

    $A \in V \subseteq X^+$

- Visto che $A \in Z_{i+1} \Rightarrow A \in X^+$ in entrambi i casi allora

$$Z_{i+1} \subseteq X^+$$

$X^+ \subseteq Z_i$

- Sia $x \subseteq R$ e r istanza di $R(Z_i, R - Z_i)$

Z_i			$R - Z_i$		
A_1	...	A_i	A_{i+1}	...	A_m
1	...	1	1	...	1
1	...	1	0	...	0

Verifichiamo se legale:

Prendiamo $V \to W \in F$

- se $t_1[V] \neq t_2[V]$ allora r soddisfa $V \to W$
- Se $t_1[V] = t_2[V]$ allora $V \subseteq Z_i$. Otteniamo $W \subseteq S_i$
  - Ciascun attributo $B \in W$ è aggiunto a $S_i \Rightarrow V \to W \in F$
  - se Z_i terminale $W \subseteq S_i \subseteq Z_i$ e quindi $t_1[W] = t_2[W]$

  ↳ r soddisfa $V \to W$
- Essendo r istanza legale abbiamo che $X \to A$.
- Avendo $X \subseteq Z_0 \subseteq Z_i \Rightarrow t_1[X] = t_2[X]$ e quindi $t_1[A] = t_2[A] \Rightarrow A \in Z_i$

4 Lemma: Inclusione delle chiusure

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

$$F \subseteq G^+ \Rightarrow F^+ \subseteq G^+$$

4.1 Dimostrazione

- Denotiamo come $F \xrightarrow{\text{A}} F'$ la possibilità di ottenere F' partendo da F applicando una determinata quantità di assiomi di Armstrong.
- Ricordando che F^A è l'insieme di tutte le dipendenze funzionali ottenibile applicando assiomi di Armstrong su F.
- Otteniamo che $F \xrightarrow{\text{A}} F' \Leftrightarrow F' \subseteq F^+$ e in particolare $F \rightarrow F^+$
- Adesso abbiamo ottenuto che $F \subseteq G^+ \Leftrightarrow G \xrightarrow{\text{A}} F \xrightarrow{\text{A}} F^+$
- Quindi $G \xrightarrow{\text{A}} F^+$ il quale implica che $F^+ \subseteq G^+$

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:

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

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" che **NON** sono oggetto di orale

6 Join senza perdita

Una decomposizione ha un join senza perdita se per ogni istanza legale r di R si ha che r è ottenibile facendo il join naturale tra le proiezioni delle decomposizioni in ρ di r.

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`:

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

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

Denotiamo con F^A l'insieme di dipendenze funzionali definito nel modo seguente:

- Se $f \in F$, allora $f \in F^A$.
- Se $Y \subseteq X \subseteq R$, allora $X \to Y \in F^A$ (**assioma della riflessività**).
- Se $X \to Y \in F^A$ e $Z \subseteq R$, allora $XZ \to YZ \in F^A$ (**assioma dell'aumento**).
- Se $X \to Y \in F^A$ e $Y \to Z \in F^A$, allora $X \to Z \in F^A$ (**assioma della transitività**).

Introduciamo altre tre regole conseguenza degli assiomi che consentono di derivare da dipendenze funzionali F^A altre dipendenze funzionali in F^A:

- Se $X \to Y \in F^A$ e $X \to Z \in F^A$, allora $X \to YZ \in F^A$ (**regola dell'unione**).
- Se $X \to Y \in F^A$ e $Z \subseteq Y$, allora $X \to Z \in F^A$ (**regola della decomposizione**).
- Se $X \to Y \in F^A$ e $WY \to Z \in F^A$, allora $WX \to Z \in F^A$ (**regola della pseudotransitività**).

- **Regola dell'unione**:
  - Siano $X \to Y, X \to Z \in F^A$.
  - Per assioma di aumento, si ha che:
    $$X \subseteq R, X \to Y \in F^A \implies XX \to XY = X \to XY \in F^A$$
  - Analogamente, si ha che:
    $$Y \subseteq R, X \to Z \in F^A \implies XY \to ZY = XY \to YZ \in F^A$$
  - Infine, per assioma di transitività si ha che:
    $$X \to XY \in F^A \land XY \to YZ \in F^A \implies X \to YZ \in F^A$$
- **Regola della decomposizione:**
  - Sia $Z \subseteq Y \subseteq R$ e sia $X \to Y \in F^A$.
  - Per assioma di riflessività, si ha che:
    $$Z \subseteq Y \subseteq R \implies Y \to Z \in F^A$$
  - Infine, per assioma di transitività si ha che:
    $$X \to Y \in F^A \land Y \to Z \in F^A \implies X \to Z \in F^A$$
- **Regola della pseudo-transitività:**
  - Sia $X \to Y, YW \to Z \in F^A$.
  - Per assioma di aumento, si ha che:
    $$W \subseteq R, X \to Y \in F^A \implies XW \to YW \in F^A$$
  - Infine, per assioma di transitività si ha che:
    $$XW \to YW \in F^A \land YW \to Z \in F^A \implies XW \to Z \in F^A$$

8 Definizioni utili

8.0.1 Cos'è una dipendenza funzionale?

Una dipendenza funzionale su una relazione R è una coppia ordinata di sottoinsiemi di R e viene denotata con $X \to Y$.

8.0.2 Cos'è la chiusura di un insieme di dipendenze funzionali?

La chiusura di un insieme di dipendenze funzionali è l'insieme di dipendenze funzionali soddisfatte da ogni istanza legale di R, essa viene denotata con F^+.

8.0.3 Quali sono le caratteristiche di una chiave per una relazione?

Un sottoinsieme K di R è chiave per R se è un insieme minimale che determina R. Ovvero se $K \to R$ appartiene a F^+ e non esiste un K' sottoinsieme di K tale che $K' \to R$ appartiene a F^+.

8.0.4 Cos'è un attributo primo?

Un attributo A dello schema R è primo se e solo se fa parte di almeno una chiave di R. In caso contrario A è detto non-primo.

8.0.5 Cos'è un superchiave?

Un sottoinsieme X di R è una superchiave se contiene una chiave di R.

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R. Definiamo il sottoinsieme di attributi $K \subseteq R$ come superchiave di R se:

- $K \to R \in F^+$
- $\exists K' \subseteq K \mid K' \to R \in F^+ \wedge \nexists K'' \subset K' \mid K'' \to R \in F^+$ (ossia contiene una chiave)

8.0.6 Chiave Minimale (chiave primaria)

Una chiave minimale, anche nota come chiave primaria, è un insieme di attributi (colonne) all'interno di una tabella (relazione) che può essere utilizzato per identificare univocamente ogni riga all'interno di quella tabella.

- **Unicità**: Ogni valore nell'insieme di attributi deve essere unico all'interno della tabella. Non possono esistere due righe con lo stesso valore per l'insieme di attributi che costituisce la chiave minimale.
- **Irreducibilità**: Nessun sottoinsieme dell'insieme di attributi può conservare la proprietà di unicità. Questo significa che se si rimuove un qualsiasi attributo dall'insieme di attributi della chiave, la capacità di identificare univocamente le righe viene persa.

8.0.7 Cos'è una copertura minimale di un insieme di dipendenze funzionali?

Una copertura minimale di un insieme di dipendenze F è un insieme di dipendenze nel quale:

- Ogni dipendenza funzionale ha la parte sinistra che è un singleton
- Ogni dipendenza funzionale ha la parte sinistra non ridondante
- Non sono presenti dipendenze funzionali ridondanti

8.0.8 Cosa sono le dipendenze parziali e le dipendenze transitive?

Una dipendenza $X \to A$ in F^+ si dice parziale se A non è primo e X è contenuto propriamente in una chiave. Una dipendenza $X \to A$ in F^+ si dice transitiva se A non è primo ed X non è contenuto in nessuna chiave K in R e $K - X \neq \emptyset$.

8.0.9 Terza Forma Normale (3NF)

Sia R uno schema e sia F un insieme di dipendenze funzionali definite su R. Lo schema R viene detto in terza forma normale (3NF) se:

$$\forall X \to A \in F^+,\ A \in R - X,\ \exists K \subseteq R \text{ chiave} \,|\, K \subseteq X \vee A \in K$$

In altre parole, uno schema viene detto in terza forma normale se per ogni dipendenza funzionale non banale $X \to A \in F^+$, il determinante X è superchiave o il determinato A è primo.
Se uno schema è in 3NF, la quantità di **anomalie** e di **ridondanze dei dati** è estremamente ridotta.

8.0.10 Quando è che una relazione è in 3NF? (Dimostra l'equivalenza tra le due affermazioni)

- Una relazione R è in 3NF se per ogni dipendenza funzionale $X \to A$ in F^+ si ha che A è primo oppure X è una superchiave.
- Una relazione R è in 3NF se in F^+ non ci sono né dipendenze **transitive** né dipendenze **parziali**.

8.0.11 F^A

Definiamo come F^A l'insieme di tutte le dipendenze funzionali ottenibili assioma di aumento partendo da F applicando i seguenti assiomi di Armstrong.

8.0.12 Decomposizione che preserva F e Spiegazione

Una decomposizione ρ preserva un insieme di dipendenze funzionali F se e solo se $F \equiv G$ dove (definizione di G):

$$G := \bigcup_{i=1}^{k} \{X \to Y \mid X \to Y \in F^+ \wedge XY \subseteq R_i\}$$

Spiegazione: Proiettare ogni dipendenza funzionale in F sul sottoschema R_i vuol dire prendere in considerazione tutte e sole le dipendenze derivabili da F (tramite assiomi di Armstrong, quindi F^+) che hanno tutti gli attributi in R_i.

8.1 Organizzazione con file Heap

- È il modello di organizzazione dei file primaria più basilare
- I nuovi record vengono inseriti alla fine del file
- Non vi è relazione tra gli attributi dei record e la loro locazione fisica
- L'unica opzione per il recupero dei record è la ricerca lineare(ossia controllando sequenzialmente ogni record)
- Ad ogni record è associata una chiave di ricerca che lo identifica
- Per un file con N blocchi, il tempo medio impiegato per trovare un record in base alla sua chiave univoca di ricerca è $\lceil \frac{N}{2} \rceil$
- Cercare record in base a chiavi di ricerca non univoche richiede la lettura dell'intero file, in modo da selezionare il record giusto

8.2 Organizzazione con file Sequenziali

- I record vengono archiviati in ordine crescente (o discendente) in base al valore della loro chiave di ricerca
- Essendo i record ordinati in base alla loro chiave di ricerca, viene utilizzata la ricerca binaria, rendendo il numero atteso di accessi ai blocchi necessari per recuperare un record pari a $\lceil \log_2(N) \rceil$ RBA, poiché i blocchi su cui si accede non sono sequenziali per natura stessa dell'algoritmo di ricerca binaria.
- Può essere utilizzata anche la ricerca lineare, rendendo il numero atteso di accessi ai blocchi necessari per recuperare un record pari a $\lceil \frac{N}{2} \rceil$ SBA, risultando tuttavia meno efficiente.
- Aggiornare i file sequenziali è più laborioso rispetto all'aggiornamento di un file heap, poiché richiede il riordinamento delle chiavi. Per tale motivo, spesso vengono fatti più aggiornamenti simultaneamente

8.3 Organizzazione con file hash

- Viene utilizzato un algoritmo di hashing per effettuare una conversione da chiave all'indirizzo fisico dove il record è archiviato
- Risulta più efficiente quando viene utilizzata una chiave primaria
- Poiché viene utilizzata una funzione di hash, non è garantito che tutte le chiavi vengano mappate a valori hash diversi, per via delle possibili collisioni, dunque vengono utilizzati dei "recipienti", detti bucket, che contengono tutti i record il cui hash generato coincide.
- Viene mantenuta salvata in memoria principale una bucket directory, ossia un insieme di record dove ogni entrata corrisponde ad un puntatore al primo blocco di un bucket associato. Ogni entrata è indicizzata da un valore assumibile dalla funzione di hash.
- L'algoritmo di hashing deve distribuire i record il più possibile in modo uniforme, spesso realizzata tramite l'operatore modulo (ad esempio address(key_i) = key_i *mod* M)

Se si ha una quantità di record mappati allo stesso bucket maggiore della capienza stabilita per il bucket stesso, allora il record viene considerato in overflow. La tecnica più utilizzata per la gestione degli overflow è la tecnica di indirizzamento aperto, dove gli overflow vengono archiviati nel primo slot disponibile. L'efficienza dell'algoritmo di hash utilizzato è misurata in base al numero atteso di RBA e di SBA:[2]

- Recuperare un record non in overflow richiede un singolo RBA per raggiungere l'indirizzo del primo blocco del bucket, ottenuto dall'algoritmo di hash, per poi (potenzialmente) eseguire uno o più SBA, poiché i record nello stesso bucket sono archiviati in modo sequenziale
- Recuperare un record in overflow richiede accessi ai blocchi aggiuntivi in base alla percentuale di record in overflow, dipendente dalla tecnica di hashing utilizzata, e in base alla tecnica adottata per gestirli

[2] **RBA (Relative Block Address)**: L'indirizzo relativo del blocco fa riferimento alla posizione di un blocco di dati all'interno di un dispositivo di archiviazione. Questo indirizzo è relativo al blocco di partenza o ad un punto di riferimento, spesso indicato come "blocco base". In altre parole, l'RBA ti dice quanto è distante il blocco di dati dal blocco base. Quando si accede ai dati, è necessario calcolare l'RBA per determinare la posizione esatta del blocco che si desidera recuperare.
SBA (Sequential Block Address): L'indirizzo sequenziale del blocco fa riferimento all'indirizzo del blocco successivo in sequenza rispetto a quello corrente. Questo è particolarmente rilevante quando si accede a dati archiviati in modo sequenziale su un dispositivo di archiviazione, come ad esempio quando i record sono memorizzati uno dopo l'altro in un blocco. L'SBA è utilizzato per spostarsi in modo efficiente attraverso i dati archiviati in sequenza.

8.4 ISAM (Indexed Sequential Access Method)

- Il file principale contenente i record è diviso in partizioni, ognuna corrispondente ad un blocco, ognuna rappresentata da un'entrata <Chiave di ricerca primo record, Puntatore indirizzo fisico primo record>, le quali vengono archiviate in un file indice.
- Il file principale e il file indice sono entrambi ordinati in base alle chiavi di ricerca presenti nelle loro entrate
- Il file indice può essere di tipo denso, dove esiste un'entrata per ogni possibile valore della chiave di ricerca, o di tipo sparso, dove esiste solo l'entrata del primo record di ogni partizione, implicando quindi che ogni entrata faccia riferimento ad un gruppo di record

Dato un numero di blocchi del file principale pari a NBFP ed un numero di blocchi del file indice pari a NBFI, l'efficienza delle ricerche risulta essere:

- Ricerca lineare sul file principale: $\mathrm{NB}_F P$ SBA
- Ricerca binaria sul file principale: $\lceil \log_2(NB_{FP}) \rceil$ RBA
- Ricerca binaria tramite file indice: $\lceil \log_2(NB_{FI}) \rceil + 1$ RBA, dove l'ultimo RBA è dovuto all'accesso al blocco dell'entrata trovata

Effettuando la ricerca binaria sul file indice, se k è la chiave del record che si sta cercando, a e b sono gli estremi dell'intervallo di chiavi del file indice che si sta considerando attualmente e m è il valore medio di tale intervallo, allora:

- Se k < m allora l'algoritmo controllerà l'insieme di chiavi compreso tra [a, m - 1], poiché la chiave k sicuramente non si trova all'interno del blocco m
- Se k = m allora l'algoritmo selezionerà tale blocco
- Se k > m allora l'algoritmo controllerà l'insieme di chiavi compreso tra [m, b], poiché la chiave k potrebbe essere all'interno del blocco m

8.4.1 Variante Isam con chiavi indice che hanno valore ultimo record

TO-DO

8.5 B-Tree

L'organizzazione del file tramite B-Tree è una **generalizzazione** del file ISAM. In un file B-Tree non si accede ai blocchi del file principale, come nell'ISAM, attraverso un file indice, bensì attraverso una **gerarchia** di indici.

L'**indice** a livello più alto è costituito da un unico blocco ed è quindi allocato in **memoria principale**. Durante la ricerca di un record con un dato valore per la chiave si parte dall'indice di livello più alto e, man mano che si scende, si restringe la porzione di blocchi del file principale in cui deve trovarsi il record, fino a che nell'ultimo livello tale porzione è ristretta ad un solo blocco. Per evitare ulteriori passaggi per l'inserimento, la cancellazione e la modifica, i blocchi degli indici e del file principale devono essere sempre pieni almeno per metà.
Il B-Tree è utile per file **grossi** dove non conviene caricare tutti i file non necessari alla query.

Ogni blocco del file indice è costituito da una **coppia** (v, b) dove v è il valore della **chiave del primo** record della porzione del file principale accessibile attraverso il **puntatore** b, il quale punta ad un blocco del file indice a livello immediatamente inferiore o, se è l'indice più basso della gerarchia, ad un blocco del file principale

8.5.1 Ricerca

Per ricercare un record con un valore v per la chiave si procede nella seguente maniera: Si parte dall'indice a livello più alto, il quale è costituito da un solo blocco. Ad ogni passo si esamina un solo blocco. Se esso è del file principale, allora il record ricercato dovrà essere in tale blocco, altrimenti si cerca un valore che ricopra v e si segue il puntatore associato. Ad ogni passo si scende di livello quindi si effettuano al massimo h+1 passi, dove h è l'altezza dell'albero.
Il costo della ricerca è di $O(h - 1 + 1)$ (-1 perchè la radice già presente nella memoria principale).

8.5.2 Inserimento

Il costo dell'inserimento dipende dal fatto che ci sia o meno spazio sufficiente all'interno del blocco.

- Se c'è spazio, il record viene ricercato con un costo di $h + 1$ e viene modificato il blocco con un costo di 1, quindi il costo totale è di $h + 2$ accessi.
- Se non c'è spazio, il record viene **ricercato** con un costo di $h + 1$. Viene poi **chiesto** al sistema un nuovo blocco e vengono ripartiti i record del blocco pieno in modo da tenere entrambi i blocchi **pieni almeno per metà**. Lo stesso discorso va effettuato al blocco indice di livello 1 se esso è pieno, e al livello 2 se anche esso è pieno, e così via. Se **tutti** i blocchi sono pieni, l'**altezza dell'albero aumenterà di uno** poiché tutti i blocchi indice, fino alla radice, verranno **sdoppiati**. Verrà quindi richiesto un ulteriore blocco che farà da **nuova radice**.

8.5.3 Cancellazione

Anche per la cancellazione, come per l'inserimento, il costo dipende da quanto siano pieni i blocchi:

- Se dopo la cancellazione il blocco rimane pieno almeno per metà, allora il record viene ricercato con un costo di $h + 1$ e viene modificato il blocco con un costo di 1. In questo caso, il costo totale è di $h + 2$. *(Modifica + Ricerca = 1 + 1 + altezza).*
- Se dopo la cancellazione il blocco non soddisfa la condizione di essere **pieno almeno per metà**, si verifica se c'è spazio nei blocchi adiacenti per inserire il contenuto di tale blocco. Lo stesso principio viene applicato ai blocchi indice se essi non rimangono pieni almeno per metà dopo la cancellazione del blocco. Nel caso peggiore, le modifiche riguardano tutti i blocchi fino alla radice, e l'altezza dell'albero può essere diminuita di uno.

8.5.4 Modifica

- Se la modifica non coinvolge una chiave, il record viene ricercato con un costo di $h + 1$ e il blocco viene modificato con un costo di 1.
- Se la modifica coinvolge una chiave, il costo totale è dato dalla somma degli accessi necessari per la cancellazione del record con chiave v e quelli necessari per l'inserimento del record con il nuovo valore della chiave.

8.5.5 Altezza di un B-Tree

Dato un B-Tree avente N chiavi, posto m il numero massimo di figli che un nodo può avere e $d := \frac{m}{2}$ il numero minimo di figli che un nodo può avere, l'altezza h del B-Tree è compresa tra:

$$\lceil \log_m(N+1) \rceil \leq h \leq \left\lfloor \log_d\left(\frac{N+1}{2}\right) \right\rfloor + 1$$

Dunque, l'altezza massima viene raggiunta quando l'albero possiede il minimo numero di nodi, mentre l'altezza minima quando possiede il massimo numero di nodi.

8.5.6 Performance di un B-Tree

Data l'altezza h di un B-Tree, si ha che:

- L'operazione `get(k)` impiega massimo h accessi al disco
- L'operazione `put(k)` impiega massimo $3h + 1$ accessi al disco
- L'operazione `remove(k)` impiega massimo $3h$ accessi al disco

8.6 B$^+$-Tree

Un B+-Tree è un B-Tree avente le seguenti proprietà aggiuntive:

- Tutti i valori chiave presenti nei nodi non foglia sono ripetuti nei nodi foglia, in modo che ogni possibile chiave dell'albero sia presente nei nodi foglia stessi.
- I nodi di livello più alto contengono sottoinsiemi delle chiavi presenti nei nodi foglia.
- Ogni nodo foglia ha un puntatore aggiuntivo, il quale punta a un nodo foglia adiacente.

Per via di tali proprietà, un B+-Tree risulta essere più efficiente rispetto a un normale B-Tree, poiché la sua altezza è generalmente inferiore.

9 Formulario

9.1 Hash

$$\text{NRpBk} = \left\lceil \frac{NR}{NBr} \right\rceil$$

$$\text{RpB} = \left\lfloor \frac{BS - PS}{RS} \right\rfloor$$

$$\text{BBk} = \left\lceil \frac{NRpBk}{RpB} \right\rceil$$

$$\text{PpB} = \left\lfloor \frac{BS}{RS} \right\rfloor$$

$$\text{BBD} = \left\lceil \frac{NBk}{PpB} \right\rceil$$

$$\text{NMA} = \left\lceil \frac{BBk}{2} \right\rceil$$

$$Tot = NMA \cdot BBk + BBD$$

9.2 Isam

$$\text{RpB} = \left\lfloor \frac{BS}{RS} \right\rfloor$$

$$\text{BFP} = \left\lceil \frac{NR}{RpB} \right\rceil$$

$$\text{EpB} = \left\lfloor \frac{BS}{KS + PS} \right\rfloor$$

$$\text{BFI} = \left\lceil \frac{BFP}{EpB} \right\rceil$$

$$\text{NA} = \lceil log_2(BFI) \rceil + 1$$

9.3 B-Tree

Se al **massimo** allora:

$$\text{RpB} = \left\lfloor \frac{BS}{RS} \right\rfloor$$

Se al **minimo** allora:

$$\text{RpB} = \left\lfloor \frac{BS \cdot 0.5}{RS} \right\rfloor$$

$$\text{masismo numero figli per nodo } m = \text{RI} = \left\lceil \frac{BS - PS}{KS + PS} \right\rceil$$

$$\text{BFP} = \left\lceil \frac{NR}{RpB} \right\rceil$$

$$L_0 \dots L_k = \left\lceil \frac{BFP}{RI} \right\rceil$$