Dalla relazione

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

**Analisi delle dipendenze funzionali**

Facendo un’analisi delle dipendenze funzionali abbiamo:

1. (Ogni prodotto ha il suo prezzo PT. A prodotti uguali corrispondono PT uguali)
2. (A tipi uguali corrispondono componenti uguali)
3. (Per ogni componente di un prodotto abbiamo associati Q, PC, Fornitore)

**Analisi delle anomalie**

Analizzare le possibili anomalie, date le seguenti operazioni sulla tabella:

1. *Inserimento di un nuovo prodotto*

Se voglio inserire un nuovo prodotto, devo ripetere tutte le informazioni ad esso annesse (come PT) per ogni componente di tale prodotto. In altre parole, vengono introdotti dati ridondanti ad ogni inserimento.

1. *Cancellazione di un prodotto*

Se voglio cancellare un prodotto allora devo prima eliminare tutte le tuple relative al prodotto (una per componente).

1. *Aggiunta di una componente a un prodotto*

Implica l’aggiunta di una nuova tupla che introduce ripetizione (vado a ripetere ad esempio PT ad aggiunta).

1. *Modifica del prezzo di un prodotto*

Volendo modificare il prezzo della libreria ad esempio (PT), devo modificare tutte le tuple associate a questo prodotto che sono esattamente tante quante sono le componenti di libreria.

Posso trovare una decomposizione che mi evita queste anomalie?

**Ricerca di una decomposizione che eviti anomalie**

Sfruttando le dipendenze funzionali possiamo trovare una decomposizione che non presenti anomalie. Le dipendenze funzionali trovate prima sono:

Da queste dipendenze funzionali possiamo costruirci la nostra decomposizione .

1. diventa
2. diventa
3. diventa

La decomposizione .

**Verifichiamo se la decomposizione è una loss-less join**

Usiamo l’algoritmo di verifica:

1. Costruisco la matrice

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  | Prodotto | Componente | Tipo | Quantità | PC | Fornitore | PT |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |

1. Riempiamo la matrice:

Per la relazione :

1. Prodotto è presente in R1, allora metto
2. Componente non è presente in R1, allora metto
3. Tipo non è presente in R1, allora metto
4. Quantità non è presente in R1, allora metto
5. PC non è presente in R1, allora metto
6. Fornitore non è presente in R1, allora metto
7. PT è presente in R1, allora metto

Per la relazione R2 e R3 continuiamo con lo stesso algoritmo.

1. Per ogni dipendenza funzionale:
2. Per la prima dipendenza funzionale :
   1. Esistono due righe in Prodotto con valore uguale? Si .
   2. Facciamo in modo che in PT le due righe corrispondono cambiando da a .

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  | Prodotto | Componente | Tipo | Quantità | PC | Fornitore | PT |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |

1. Per la seconda dipendenza funzionale :
   1. Esistono due righe in tipo con valore uguale? Si .
   2. Facciamo in modo che in componente le due righe corrispondono cambiando da a .

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  | Prodotto | Componente | Tipo | Quantità | PC | Fornitore | PT |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |

1. Per la terza dipendenza funzionale :
   1. Esistono due righe in prodotto e tipo che hanno valore uguale? No
   2. Non cambio nessun valore nella matrice
2. Esiste una riga della tabella con valori ? Si l’ultima.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  | Prodotto | Componente | Tipo | Quantità | PC | Fornitore | PT |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |

Per l’algoritmo di verifica la decomposizione è una loss-less join.