

Esami - Basi di dati

VR443470

giugno 2023

# Indice

<b>1</b>	<b>Domande di teoria - Primo parziale</b>	<b>3</b>
<b>2</b>	<b>Esercizi terzo parziale</b>	<b>8</b>
2.1	B <sup>+</sup> -tree . . . . .	8
2.1.1	Esercizio 1 . . . . .	8
2.1.2	Esercizio 2 . . . . .	11
2.2	Verificare che uno schedule sia VSR (View-serializzabile) . . . . .	13
2.2.1	Esercizio 1 - Perdita di aggiornamento . . . . .	13
2.2.2	Esercizio 2 - Lettura inconsistente . . . . .	16
2.2.3	Sintesi dell'algoritmo . . . . .	19
2.3	Verificare che uno schedule sia CSR (Conflict-serializzabile) . . . . .	20
2.3.1	Esercizio 1 - Perdita di aggiornamento . . . . .	20
2.3.2	Esercizio 2 - Lettura inconsistente . . . . .	21
2.3.3	Sintesi dell'algoritmo . . . . .	21
2.4	Verificare che uno schedule sia NonSR, VSR e/o CSR . . . . .	22
2.4.1	Testo esercizio . . . . .	22
2.4.2	Schedule 1 . . . . .	22
2.4.3	Schedule 2 . . . . .	23
2.4.4	Schedule 3 . . . . .	24
2.4.5	Schedule 4 . . . . .	26
2.5	Ottimizzazione e stima di costo . . . . .	28
2.5.1	Esercizio 1 . . . . .	28
2.5.2	Esercizio 2 . . . . .	31
2.5.3	Esercizio 3 . . . . .	33
2.6	XML . . . . .	35
2.6.1	Esercizio 1 . . . . .	35
2.6.2	Esercizio 2 . . . . .	35
<b>3</b>	<b>Domande di teoria - Terzo parziale</b>	<b>35</b>
<b>4</b>	<b>Esami terzo parziale</b>	<b>41</b>
4.1	Terzo parziale - 06/2015 . . . . .	41
4.2	Terzo parziale - 07/06/2016 . . . . .	41
4.3	Terzo parziale - 21/04/2022 . . . . .	41
4.4	Terzo parziale - 22/04/2022 . . . . .	41
4.5	Terzo parziale - 10/06/2022 . . . . .	41

## 1 Domande di teoria - Primo parziale

Le domande più frequenti che si possono incontrare nel primo parziale di Basi di dati sono:

1. Si illustri il concetto/costrutto di **entità** nel modello Entità-Relazione.
2. Si illustri il concetto/costrutto di **relazione** nel modello Entità-Relazione.
3. Si illustri il concetto/costrutto di **generalizzazione** nel modello Entità-Relazione.
4. Si illustri il concetto/costrutto di **identificatore** nel modello Entità-Relazione.
5. Si illustri il concetto/costrutto di **superchiave** nel modello Entità-Relazione.
6. Si illustri il concetto/costrutto di **attributo multivalore** nel modello Entità-Relazione.

La risposta, per essere considerata perfetta, deve includere le seguenti caratteristiche:

- Semantica
- Sintassi grafica con esempio
- Istanza
- Eventuali proprietà

Qui di seguito vengono date le possibili risposte alle domande di teoria:

1. *Si illustri il concetto/costrutto di **entità** nel modello Entità-Relazione.*

**Semantica.** L'entità rappresenta una classe di oggetti che hanno proprietà comuni ed esistenza "autonoma" ai fini dell'applicazione di interesse. Il nome dato ad ogni entità è identificativo di quella determinata classe di oggetti e deve essere univoco all'interno dello schema.

**Sintassi grafica.** Per esempio, l'entità studenti rappresenta la classe di oggetti degli studenti di un'università e gli attributi possibili possono essere: matricola, nome, cognome, data di nascita, ecc. La sintassi grafica è la seguente:

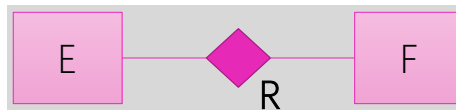


**Istanza.** L'istanza di un'entità è un oggetto della classe che lo rappresenta e non un unico valore. Per esempio, nell'entità studenti, lo studente Mario Rossi (in carne ed ossa) rappresenta un'istanza dell'entità.

2. *Si illustri il concetto/costrutto di **relazione** nel modello Entità-Relazione.*

**Semantica.** La relazione rappresenta legami logici tra una o più entità. Ogni relazione deve avere un nome univoco all'interno dello schema e non può avere identificatori. Esistono due tipi di relazioni: *ricorsive*, cioè in cui è coinvolta una sola entità, *n-arie*, in cui sono coinvolte  $n$  entità. Esse nascono solo quando le entità coinvolte contengono almeno una tupla.

**Sintassi grafica.** Un esempio di relazione è la “Residenza” tra le entità “Città” e “Impiegato”. La sua sintassi grafica è un rombo:



**Istanza.** Data una relazione  $R$  tra  $n$  entità  $E_1, E_2, \dots, E_n$ , un'istanza è composta da una ennupla del tipo:

$$(e_1, e_2, \dots, e_i) \text{ dove } e_i \in I(E_i), 1 \leq i \leq n$$

Inoltre, esiste un'importante proprietà che afferma:

$$I(R) \subseteq I(E_1) \times I(E_2) \times \dots \times I(E_n)$$

3. *Si illustri il concetto/costrutto di **generalizzazione** nel modello Entità-Relazione.*

**Semantica.** Le generalizzazioni rappresentano legami logici tra un'entità  $E$ , chiamata genitore, e più entità  $E_1, \dots, E_n$ , chiamate figlie. Quindi, si dice che l'entità  $E$  (genitore) è la generalizzazione delle entità  $E_1, \dots, E_n$  (figlie) e quest'ultime vengono chiamate specializzazioni. Inoltre, ogni occorrenza dell'entità figlia è anche un'occorrenza dell'entità padre, e ogni proprietà dell'entità padre è anche una proprietà dell'entità figlia.

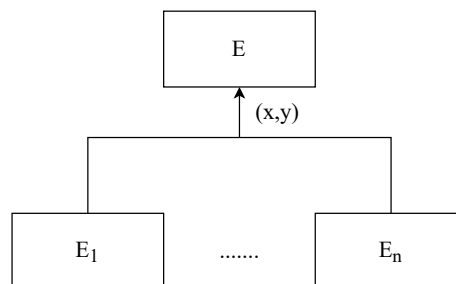
La classificazione sono coppie di valori che hanno diverso significato:

- (totale, esclusiva)
- (totale, sovrapposta)
- (parziale, esclusiva)
- (parziale, sovrapposta)

Con totale, il genitore ha ogni occorrenza posseduta da almeno un'entità figlia. In caso contrario è parziale.

Con esclusiva, il genitore ha ogni occorrenza che si ripete solamente in una delle entità figlie. In caso un'occorrenza del genitore sia di più entità figlie, si dice sovrapposta.

**Sintassi grafica.** Un esempio è la generalizzazione “Persona” con le specializzazioni “Uomo” e “Donna”. La sintassi grafica:



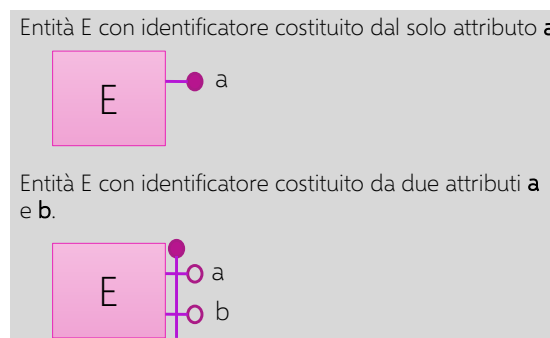
4. *Si illustri il concetto/costrutto di **identificatore** nel modello Entità-Relazione.*

**Semantica.** Gli identificatori descrivono i concetti (attributi/entità) dello schema che consentono di identificare in maniera univoca le occorrenze delle entità. Devono essere specificati per ogni entità e non possono apparire all'interno di relazioni. Un identificatore può essere:

- Interno, ovvero viene scelto un attributo dell'entità;
- Esterno, viene scelto un identificatore di un'altra identità;

È possibile utilizzare sia identificatori interni ed esterni insieme.

**Sintassi grafica.** Un esempio è l'entità "Studente" che possiede come identificatore la "Matricola" poiché unica. La sintassi grafica:



5. *Si illustri il concetto/costrutto di **superchiave** nel modello Entità-Relazione.*

**Semantica.** Una superchiave è un'insieme di attributi che non contiene tuple duplicate al suo interno. Una superchiave è una chiave prima se e solo se è una superchiave minimale. Invece, una chiave primaria è sempre superchiave (non viceversa!).

**Sintassi grafica.** Non esiste una sintassi grafica poiché è un concetto, ma un esempio:

**Tabella**

Matricola	Cognome	Nome	Data di nascita	Ufficio
2231	Rossi	Mario	22/08/1984	marketing
2232	Verdi	Paolo	11/03/1990	marketing
2233	Bianchi	Mario	07/05/1995	vendite
2234	Rossi	Giovanni	16/01/1978	personale

**superchiave**

Nessuna tupla si ripete, quindi "Matricola, Cognome, Nome" è una superchiave valida. Non è minimale poiché esiste "Matricola" che è chiave primaria e superchiave minimale.

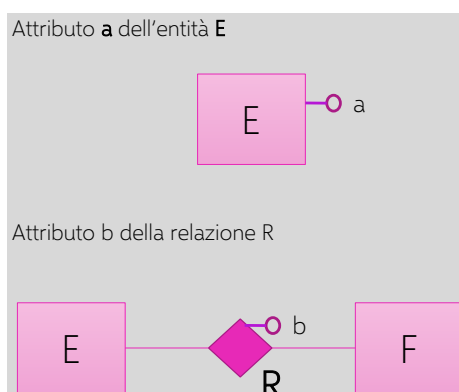
6. *Si illustri il concetto/costrutto di **attributo** nel modello Entità-Relazione.*

**Semantica.** Gli attributi descrivono le proprietà elementari di entità o relazioni che sono di interesse ai fini dell'applicazione. Ogni attributo ha un suo dominio e quindi può essere visto come una funzione che ha come dominio le istanze dell'entità/relazione e come codominio l'insieme dei valori ammissibili:

$$f_a : I(E) \rightarrow D$$

Dove  $a$  è un attributo dell'entità  $E$ ,  $I(E)$  è l'insieme delle istanze di  $E$  e  $D$  è l'insieme dei valori ammissibili.

**Sintassi grafica.** Un esempio di attributo è “Cognome”, “Stipendio” ed “Età” dell'entità “Impiegato”. La sintassi grafica è la seguente:



**Istanza.** L'istanza si ottiene tramite una funzione che data un'istanza dell'entità  $E$  (o relazione  $R$ ), restituisce l'attributo  $a$ :

$$\text{valore di } a \text{ su } e = f_a(e)$$

## 2 Esercizi terzo parziale

### 2.1 B<sup>+</sup>-tree

#### 2.1.1 Esercizio 1

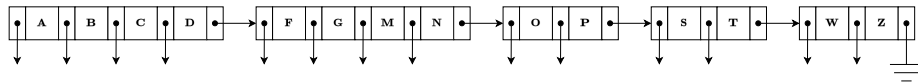
Costruire un B<sup>+</sup>-tree di fan-out = 5 con i seguenti nodi foglia: (A, B, C, D), (F, G, M, N), (O, P), (S, T), (W, Z). I vincoli di riempimento sono:

- $2 \leq \# \text{chiavi} \leq 4$
- $3 \leq \# \text{puntatori} \leq 5$

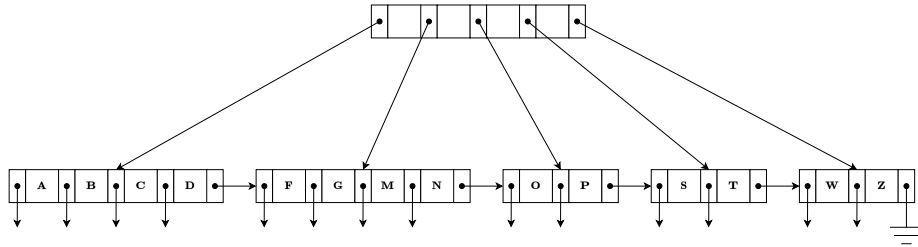
Dopodiché, inserire il valore chiave H nel B<sup>+</sup>-tree ottenuto precedentemente. Infine, l'esercizio si conclude eseguendo una rimozione del valore chiave Z ottenuto precedentemente.

#### *Soluzione*

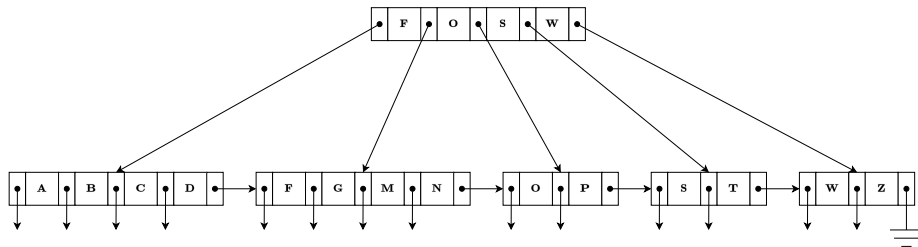
Il primo passo è costruire i vari livelli dei nodi foglia:



Adesso è necessario costruire tutti i puntatori richiesti. Fan-out è uguale a 5 quindi viene costruito un nodo intermedio con 5 puntatori e si collegano tutti i nodi:

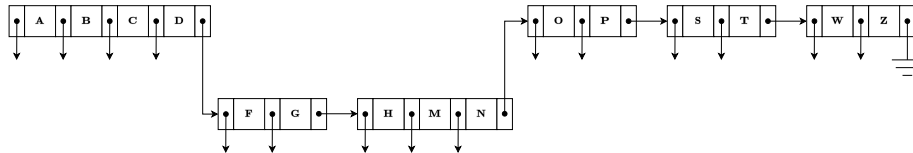


Adesso si aggiungono le lettere che devono essere raggiunte dopo aver visitato ogni nodo:

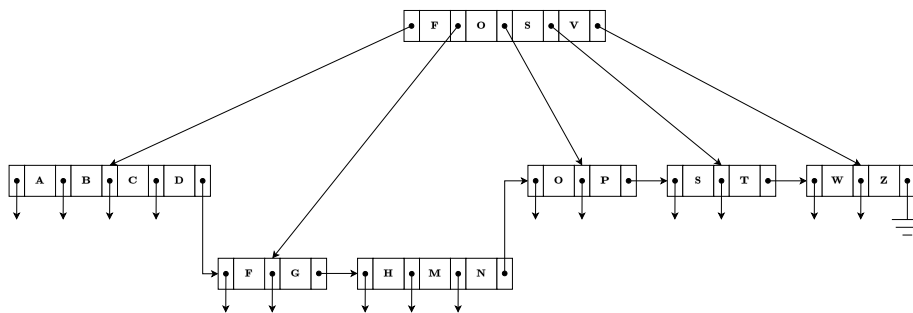




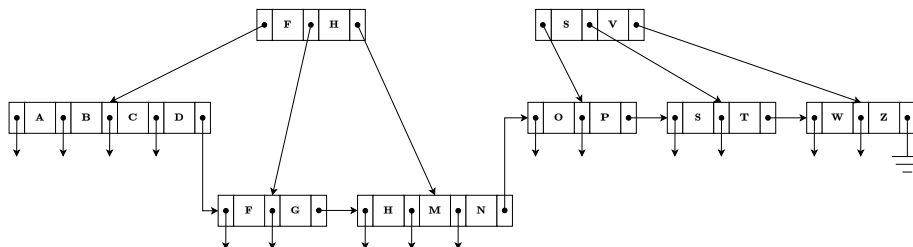
Per inserire il valore chiave è necessario avere a disposizione una posizione libera. Tuttavia, questo non è possibile, per cui viene applicato uno split. Viene divisa la radice contenente  $(F, G, M, N)$  così da inserire la chiave H tra la G e la M:



A questo punto è necessario riadattare il nodo radice che attualmente punta ad un nodo errato (attenzione c'è un errore, il nodo V in realtà è il nodo W):



Per farlo, è necessario eseguire una divisione anche nel nodo radiceaggiustando i valori delle chiavi (attenzione c'è un errore, il nodo V in realtà è il nodo W):



```

graph TD
    Root["O •"] --> FH["F • H •"]
    Root --> SV["S • V •"]
    FH --> ABCD["• A • B • C • D •"]
    FH --> FGMN["• F • G • H • M • N •"]
    SV --> OP["• O • P •"]
    SV --> ST["• S • T •"]
    SV --> WZ["• W • Z •"]
    ABCD --> A1[ ]
    ABCD --> B1[ ]
    ABCD --> C1[ ]
    ABCD --> D1[ ]
    FGMN --> F2[ ]
    FGMN --> G2[ ]
    FGMN --> H2[ ]
    FGMN --> M2[ ]
    FGMN --> N2[ ]
    OP --> O3[ ]
    OP --> P3[ ]
    ST --> S4[ ]
    ST --> T4[ ]
    WZ --> W5[ ]
    WZ --> Z5[ ]
    style A1 fill:none,stroke:none
    style B1 fill:none,stroke:none
    style C1 fill:none,stroke:none
    style D1 fill:none,stroke:none
    style F2 fill:none,stroke:none
    style G2 fill:none,stroke:none
    style H2 fill:none,stroke:none
    style M2 fill:none,stroke:none
    style N2 fill:none,stroke:none
    style O3 fill:none,stroke:none
    style P3 fill:none,stroke:none
    style S4 fill:none,stroke:none
    style T4 fill:none,stroke:none
    style W5 fill:none,stroke:none
    style Z5 fill:none,stroke:none
  
```

```

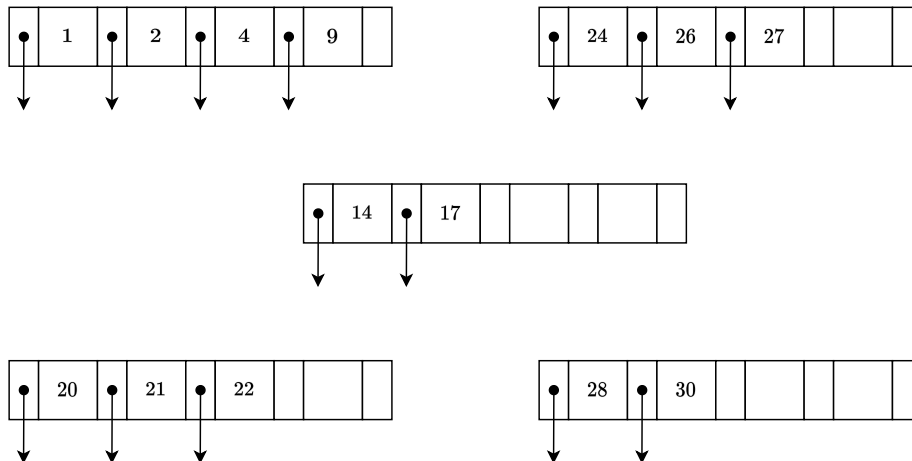
graph TD
    Root["FISH"] --> A["A"]
    Root --> B["B"]
    Root --> O1["O"]
    Root --> S1["S"]
    A --> B1["B"]
    A --> C["C"]
    A --> D["D"]
    B1 --> F["F"]
    B1 --> G["G"]
    C --> H["H"]
    C --> M["M"]
    D --> N["N"]
    D --> O2["O"]
    O1 --> P["P"]
    O1 --> S2["S"]
    P --> S3["S"]
    P --> T["T"]
    P --> W["W"]
    S2 --> T1["T"]
    S2 --> W1["W"]
  
```

### 2.1.2 Esercizio 2

Data la seguente lista di possibili valori chiave:

$L = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30)$

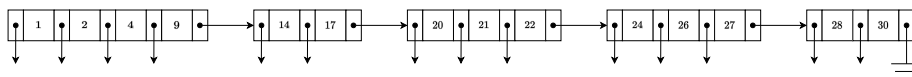
Costruire un B+-tree ( $fan-out = 5$ ) che contenga i seguenti nodi foglia:



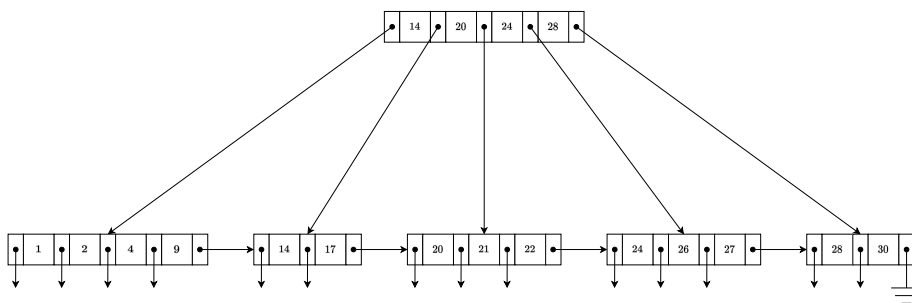
Mostrare l'albero dopo l'inserimento del valore chiave 5 e partendo da questo risultato, mostrare l'albero dopo la cancellazione del valore chiave 28.

### Soluzione

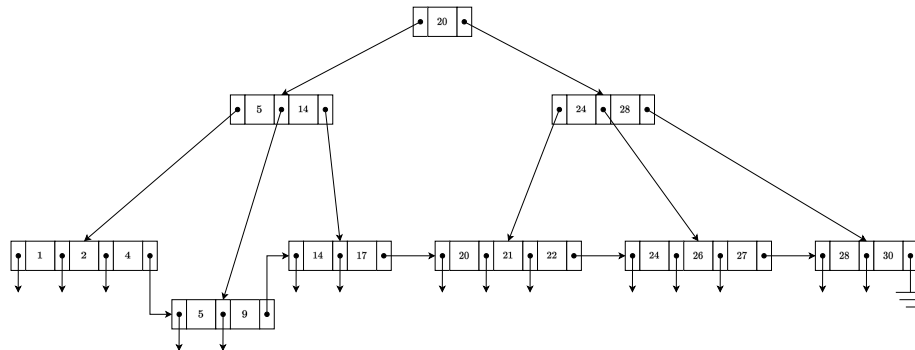
Si costruisce la lista:



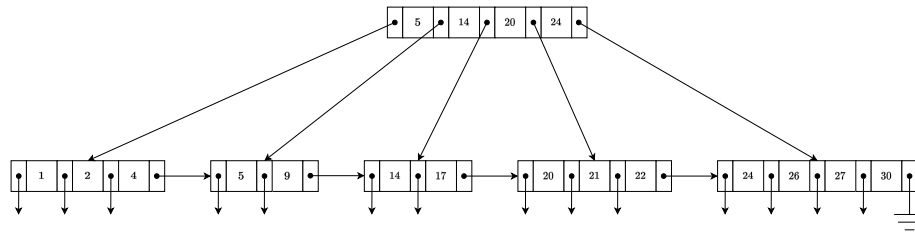
E poi l'albero:



Si inserisce il valore 5:



E infine si rimuove il valore 28:



## 2.2 Verificare che uno schedule sia VSR (View-serializzabile)

### 2.2.1 Esercizio 1 - Perdita di aggiornamento

Date due transazioni  $T_1$  e  $T_2$  di seguito descritte:

$$\begin{array}{lcl} T_1 & : & r_1(x) \ w_1(x) \\ T_2 & : & r_2(x) \ w_2(x) \end{array}$$

Lo schedule che rappresenta l'anomalia è il seguente:

$$S_{PA} = r_1(x) \ r_2(x) \ w_2(x) \ w_1(x)$$

Per verificare che uno schedule sia VSR o meno, è necessario caratterizzare  $S_{PA}$  calcolando l'insieme delle relazioni LeggeDa e l'insieme delle ScrittureFinali.

Quindi, per l'insieme LeggeDa viene cercato per ogni operazione di lettura, una precedente scrittura sulla stessa risorsa fatta da un'altra transazione. In questo caso, l'insieme è vuoto poiché nessuna risorsa scrive prima di una lettura.

Invece, per l'insieme ScrittureFinali, per ogni risorsa indicata nello schedule si specifica l'ultima scrittura eseguita. In questo caso, l'unica risorsa è  $x$  e l'ultima scrittura è  $w_1(x)$ .

Quindi, gli insiemi sono composti da:

$$\begin{array}{lcl} \text{LeggeDa}(S_{PA}) & = & \emptyset \\ \text{ScrittureFinali}(S_{PA}) & = & \{w_1(x)\} \end{array}$$

Adesso si generano tutti i possibili schedule seriali che eseguono le due transazioni. Essi si ottengono generando le possibili permutazioni dell'insieme di transazioni che partecipano allo schedule. In questo caso, date solo due transazioni  $T_1$  e  $T_2$ , le possibili combinazioni sono:

$$\begin{array}{lcl} S_1 & = & T_1 \ T_2 = r_1(x) \ w_1(x) \ r_2(x) \ w_2(x) \\ S_2 & = & T_2 \ T_1 = r_2(x) \ w_2(x) \ r_1(x) \ w_1(x) \end{array}$$

Adesso, si verifica che almeno uno dei due schedule seriali è view-equivalente a  $S_{PA}$ .

Verifica partendo dallo schedule  $S_1$ :

1. Creazione dell'insieme  $\text{LeggeDa}(S_1)$ . Data la sequenza:

$$S_1 = r_1(x) \ w_1(x) \ r_2(x) \ w_2(x)$$

L'unica scrittura che precede una lettura è  $w_1(x)$ . Quindi, l'insieme è composto dalla scrittura che avviene prima della lettura e da quest'ultima:

$$\text{LeggeDa}(S_1) = \{(r_2(x), w_1(x))\}$$

2. Creazione dell'insieme  $\text{ScrittureFinali}(S_1)$ . Data la sequenza:

$$S_1 = r_1(x) \ w_1(x) \ r_2(x) \ w_2(x)$$

L'unica risorsa  $x$  ha come ultima scrittura  $w_2(x)$ , quindi l'insieme è composto da:

$$\text{ScrittureFinali}(S_1) = \{w_2(x)\}$$

3. Si esegue il confronto degli insiemi ottenuti da  $S_1$  e dagli insiemi ottenuti da  $S_{PA}$ :

$$\begin{array}{ll} \text{LeggeDa}(S_{PA}) & = \emptyset \\ \text{LeggeDa}(S_1) & = \{(r_2(x), w_1(x))\} \\ \text{ScrittureFinali}(S_{PA}) & = \{w_1(x)\} \\ \text{ScrittureFinali}(S_1) & = \{w_2(x)\} \end{array}$$

Come è evidente, nessuno dei due insiemi è equivalente:

$$\begin{array}{ll} \text{LeggeDa}(S_{PA}) & \not\equiv \text{LeggeDa}(S_1) \\ \text{ScrittureFinali}(S_{PA}) & \not\equiv \text{ScrittureFinali}(S_1) \end{array}$$

Quindi, è possibile concludere che  $S_{PA}$  non è view-equivalente a  $S_1$ .

Verifica partendo dallo schedule  $S_2$ :

1. Creazione dell'insieme  $\text{LeggeDa}(S_2)$ . Data la sequenza:

$$S_2 = r_2(x) \ w_2(x) \ r_1(x) \ w_1(x)$$

L'unica scrittura che precede una lettura è  $w_2(x)$ . Quindi, l'insieme è composto dalla scrittura che avviene prima della lettura e da quest'ultima:

$$\text{LeggeDa}(S_2) = \{(r_1(x), w_2(x))\}$$

2. Creazione dell'insieme  $\text{ScrittureFinali}(S_2)$ . Data la sequenza:

$$S_2 = r_2(x) \ w_2(x) \ r_1(x) \ w_1(x)$$

L'unica risorsa  $x$  ha come ultima scrittura  $w_2(x)$ , quindi l'insieme è composto da:

$$\text{ScrittureFinali}(S_2) = \{w_1(x)\}$$

3. Si esegue il confronto degli insiemi ottenuti da  $S_1$  e dagli insiemi ottenuti da  $S_{PA}$ :

$$\begin{aligned} \text{LeggeDa}(S_{PA}) &= \emptyset \\ \text{LeggeDa}(S_1) &= \{(r_1(x), w_2(x))\} \\ \text{ScrittureFinali}(S_{PA}) &= \{w_1(x)\} \\ \text{ScrittureFinali}(S_1) &= \{w_1(x)\} \end{aligned}$$

Come è evidente, soltanto uno dei due insiemi è equivalente:

$$\begin{aligned} \text{LeggeDa}(S_{PA}) &\not\equiv \text{LeggeDa}(S_1) \\ \text{ScrittureFinali}(S_{PA}) &\equiv \text{ScrittureFinali}(S_1) \end{aligned}$$

Quindi, è possibile concludere che  $S_{PA}$  non è view-equivalente a  $S_1$  poiché entrambi gli insiemi non sono equivalenti.

L'esercizio si conclude qua. Nessuna combinazione è view-equivalente allo schedule di partenza  $S_{PA}$ . Quindi, si conclude affermando che  $S_{PA}$  non è VSR.

### 2.2.2 Esercizio 2 - Lettura inconsistente

Date due transazioni  $T_1$  e  $T_2$  di seguito descritte:

$$\begin{array}{lcl} T_1 & : & r_1(x) \ r'_1(x) \\ T_2 & : & r_2(x) \ w_2(x) \end{array}$$

Lo schedule che rappresenta l'anomalia è il seguente:

$$S_{LI} = r_1(x) \ r_2(x) \ w_2(x) \ r'_1(x)$$

Per verificare che uno schedule sia VSR o meno, è necessario caratterizzare  $S_{LI}$  calcolando l'insieme delle relazioni LeggeDa e l'insieme delle ScrittureFinali.

Quindi, per l'insieme LeggeDa viene cercato per ogni operazione di lettura, una precedente scrittura sulla stessa risorsa fatta da un'altra transazione. In questo caso, l'insieme è composto da  $w_2(x)$  perché precede  $r'_1(x)$ .

Invece, per l'insieme ScrittureFinali, per ogni risorsa indicata nello schedule si specifica l'ultima scrittura eseguita. In questo caso, l'unica risorsa è  $x$  e l'ultima scrittura è  $w_2(x)$ .

Quindi, gli insiemi sono composti da:

$$\begin{array}{lcl} \text{LeggeDa}(S_{LI}) & = & \{(r'_1(x), w_2(x))\} \\ \text{ScrittureFinali}(S_{LI}) & = & \{w_2(x)\} \end{array}$$

Adesso si generano tutti i possibili schedule seriali che eseguono le due transazioni. Essi si ottengono generando le possibili permutazioni dell'insieme di transazioni che partecipano allo schedule. In questo caso, date solo due transazioni  $T_1$  e  $T_2$ , le possibili combinazioni sono:

$$\begin{array}{lcl} S_1 & = & T_1 \ T_2 = r_1(x) \ r'_1(x) \ r_2(x) \ w_2(x) \\ S_2 & = & T_2 \ T_1 = r_2(x) \ w_2(x) \ r_1(x) \ r'_1(x) \end{array}$$

Adesso, si verifica che almeno uno dei due schedule seriali è view-equivalente a  $S_{LI}$ .



Verifica partendo dallo schedule  $S_1$ :

1. Creazione dell'insieme  $\text{LeggeDa}(S_1)$ . Data la sequenza:

$$S_1 = r_1(x) \ r'_1(x) \ r_2(x) \ w_2(x)$$

L'unica scrittura che precede una lettura è  $w_1(x)$ . Quindi, l'insieme è composto dalla scrittura che avviene prima della lettura e da quest'ultima:

$$\text{LeggeDa}(S_1) = \emptyset$$

2. Creazione dell'insieme  $\text{ScrittureFinali}(S_1)$ . Data la sequenza:

$$S_1 = r_1(x) \ r'_1(x) \ r_2(x) \ w_2(x)$$

L'unica risorsa  $x$  ha come ultima scrittura  $w_2(x)$ , quindi l'insieme è composto da:

$$\text{ScrittureFinali}(S_1) = \{w_2(x)\}$$

3. Si esegue il confronto degli insiemi ottenuti da  $S_1$  e dagli insiemi ottenuti da  $S_{LI}$ :

$$\begin{array}{ll} \text{LeggeDa}(S_{LI}) & = \{(r'_1(x), w_2(x))\} \\ \text{LeggeDa}(S_1) & = \emptyset \\ \text{ScrittureFinali}(S_{LI}) & = \{w_2(x)\} \\ \text{ScrittureFinali}(S_1) & = \{w_2(x)\} \end{array}$$

Come è evidente, soltanto uno dei due insiemi è equivalente:

$$\begin{array}{ll} \text{LeggeDa}(S_{LI}) & \not\equiv \text{LeggeDa}(S_1) \\ \text{ScrittureFinali}(S_{LI}) & \equiv \text{ScrittureFinali}(S_1) \end{array}$$

Quindi, è possibile concludere che  $S_{LI}$  non è view-equivalente a  $S_1$ .

Verifica partendo dallo schedule  $S_2$ :

1. Creazione dell'insieme  $\text{LeggeDa}(S_2)$ . Data la sequenza:

$$S_2 = r_2(x) \ w_2(x) \ r_1(x) \ r'_1(x)$$

L'unica scrittura che precede due letture è  $w_2(x)$ . Quindi, l'insieme è composto dalla scrittura che avviene con due letture:

$$\text{LeggeDa}(S_2) = \{(r'_1(x), w_2(x)), (r_1(x), w_2(x))\}$$

2. Creazione dell'insieme  $\text{ScrittureFinali}(S_2)$ . Data la sequenza:

$$S_2 = r_2(x) \ w_2(x) \ r_1(x) \ r'_1(x)$$

L'unica risorsa  $x$  ha come ultima scrittura  $w_2(x)$ , quindi l'insieme è composto da:

$$\text{ScrittureFinali}(S_2) = \{w_2(x)\}$$

3. Si esegue il confronto degli insiemi ottenuti da  $S_1$  e dagli insiemi ottenuti da  $S_{PA}$ :

$$\begin{aligned} \text{LeggeDa}(S_{LI}) &= \{(r'_1(x), w_2(x))\} \\ \text{LeggeDa}(S_2) &= \{(r'_1(x), w_2(x)), (r_1(x), w_2(x))\} \\ \text{ScrittureFinali}(S_{LI}) &= \{w_2(x)\} \\ \text{ScrittureFinali}(S_2) &= \{w_2(x)\} \end{aligned}$$

Come è evidente, soltanto uno dei due insiemi è equivalente:

$$\begin{aligned} \text{LeggeDa}(S_{PA}) &\not\equiv \text{LeggeDa}(S_1) \\ \text{ScrittureFinali}(S_{PA}) &\equiv \text{ScrittureFinali}(S_1) \end{aligned}$$

Quindi, è possibile concludere che  $S_{LI}$  non è view-equivalente a  $S_2$  poiché entrambi gli insiemi non sono equivalenti.

L'esercizio si conclude qua. Nessuna combinazione è view-equivalente allo schedule di partenza  $S_{LI}$ . Quindi, si conclude affermando che  $S_{LI}$  non è VSR.

### 2.2.3 Sintesi dell'algoritmo

In sintesi l'algoritmo per capire se uno schedule è VSR:

1. Si tiene bene in considerazione lo schedule che rappresenta l'anomalia, ovvero quello che viene dato;
2. Si compongono i due insiemi:
  - (a) Creazione insieme LeggeDa cercando per ogni operazione di lettura ( $r_i$  (risorsa)) una precedente operazione di scrittura sulla stessa risorsa fatta da un'altra transazione. Nel caso in cui si trovi, si aggiunge all'insieme la scrittura incriminata e la relativa lettura;
  - (b) Creazione insieme ScrittureFinali cercando per ogni risorsa indicata nello schedule l'ultima scrittura eseguita.
3. Date le varie transazioni, si creano tutti i possibili schedule creando così una lista;
4. Si verifica che almeno uno schedule della lista sia view-equivalente allo schedule dato al punto 1. Per farlo si esegue questo piccolo algoritmo:
  - (a) Creazione dell'insieme LeggeDa (vedi punto 2.a);
  - (b) Creazione dell'insieme ScrittureFinali (vedi punto 2.b);
  - (c) Confronto degli insiemi creati precedente con quelli creati per lo schedule dato al punto 1. Se non sono uguali tutti uguali, allora lo schedule creato tramite combinazione non è equivalente allo schedule dato al punto 1. Altrimenti, è possibile affermare di aver trovato una combinazione view-equivalente.
5. Al termine della creazione degli insiemi e dei vari confronti, se esiste almeno una combinazione che è view-equivalente allo schedule del punto 1, allora è possibile affermare che lo schedule di partenza è VSR.

## 2.3 Verificare che uno schedule sia CSR (Conflict-serializzabile)

### 2.3.1 Esercizio 1 - Perdita di aggiornamento

Date due transizioni  $T_1$  e  $T_2$ :

$$\begin{array}{lcl} T_1 & : & r_1(x) \ w_1(x) \\ T_2 & : & r_2(x) \ w_2(x) \end{array}$$

Lo schedule che rappresenta l'anomalia è il seguente:

$$S_{PA} = r_1(x) \ r_2(x) \ w_2(x) \ w_1(x)$$

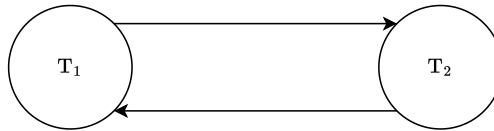
Per verificare CSR è necessario caratterizzare  $S_{PA}$  calcolando l'insieme dei conflitti. Si ricorda che due azioni sono in conflitto se operano sullo stesso oggetto e se almeno una di esse è in scrittura (quindi le combinazioni:  $rw, wr, ww$ ). Quindi, si calcola l'insieme dei conflitti di  $S_{PA}$ :

1.  $r_1(x) \ r_2(x) \ w_2(x) \ w_1(x)$
2.  $r_1(x) \ r_2(x) \ w_2(x) \ w_1(x)$
3.  $r_1(x) \ r_2(x) \ w_2(x) \ w_1(x)$

L'insieme è quindi così costituito:

$$\text{Conflitti}(S_{PA}) = \{(r_1(x), w_2(x)), (r_2(x), w_1(x)), (w_2(x), w_1(x))\}$$

Si costruisce il grafo nel seguente modo. Si rappresentano tanti nodi quanti sono le transazioni e ogni arco (orientato) viene tracciato da  $t_i$  a  $t_j$  se vengono rispettate due condizioni: se c'è almeno un conflitto fra un'azione  $a_i$  e un'azione  $a_j$  tale che  $a_i$  precede  $a_j$ . Quindi:



Se il grafo è aciclico allora  $S_{PA}$  è CSR. In questo caso, il grafo non è aciclico ma ciclico, per cui  $S_{PA}$  non è CSR.

### 2.3.2 Esercizio 2 - Lettura inconsistente

Date due transizioni  $T_1$  e  $T_2$ :

$$\begin{aligned} T_1 &: r_1(x) \ r'_1(x) \\ T_2 &: r_2(x) \ w_2(x) \end{aligned}$$

Lo schedule che rappresenta l'anomalia è il seguente:

$$S_{LI} = r_1(x) \ r_2(x) \ w_2(x) \ r'_1(x)$$

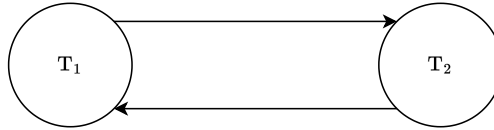
Per verificare CSR è necessario caratterizzare  $S_{LI}$  calcolando l'insieme dei conflitti. Si ricorda che due azioni sono in conflitto se operano sullo stesso oggetto e se almeno una di esse è in scrittura (quindi le combinazioni:  $rw, wr, ww$ ). Quindi, si calcola l'insieme dei conflitti di  $S_{LI}$ :

1.  $r_1(x) \ r_2(x) \ w_2(x) \ r'_1(x)$
2.  $r_1(x) \ r_2(x) \ w_2(x) \ r'_1(x)$

L'insieme è quindi così costituito:

$$\text{Conflitti}(S_{LI}) = \{(r_1(x), w_2(x)), (w_2(x), r'_1(x))\}$$

Si costruisce il grafo nel seguente modo. Si rappresentano tanti nodi quanti sono le transazioni e ogni arco (orientato) viene tracciato da  $t_i$  a  $t_j$  se vengono rispettate due condizioni: se c'è almeno un conflitto fra un'azione  $a_i$  e un'azione  $a_j$  tale che  $a_i$  precede  $a_j$ . Quindi:



Se il grafo è aciclico allora  $S_{LI}$  è CSR. In questo caso, il grafo non è aciclico ma ciclico, per cui  $S_{LI}$  non è CSR.

### 2.3.3 Sintesi dell'algoritmo

In sintesi l'algoritmo per capire se uno schedule è CSR:

1. Si calcola l'insieme dei conflitti. Un conflitto si manifesta quando due azioni **differenti** operano sullo stesso oggetto e quando almeno una di esse è in scrittura. Quindi, le combinazioni che possono esserci sono:  $rw, wr, ww$ ;
2. Si costruisce il grafo dall'insieme dei conflitti. I nodi rappresentano le transizioni e gli archi si disegnano solo se due azioni non riguardano la stessa transizione.

## 2.4 Verificare che uno schedule sia NonSR, VSR e/o CSR

### 2.4.1 Testo esercizio

Classificare i seguenti schedule (come: NonSR, VSR, CSR); nel caso uno schedule sia VSR oppure CSR, indicare tutti gli schedule seriali a esso equivalenti.

1.  $S_1 = r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$
2.  $S_2 = r_1(x) w_1(x) w_3(x) r_2(y) r_3(y) w_3(y) w_1(y) r_2(x)$
3.  $S_3 = r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
4.  $S_4 = r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$

### 2.4.2 Schedule 1

Dato il seguente schedule:

$$S_1 = r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$$

Le transizioni sono:

$$\begin{aligned} T_1 &: r_1(x) w_1(x) r_1(y) w_1(y) \\ T_2 &: r_2(z) r_2(x) w_2(x) w_2(z) \end{aligned}$$

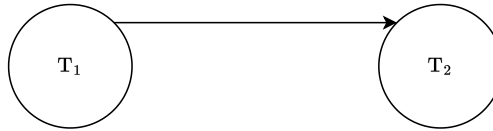
Si verifica se è CSR. Quindi, si crea l'insieme dei conflitti:

1.  $r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$
2.  $r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$
3.  $r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$

Quindi l'insieme è:

$$\text{Conflitti}(S_1) = \{(r_1(x) w_2(x)), (w_1(x) r_2(x)), (w_1(x) w_2(x))\}$$

Si costruisce il grafo:



Il ciclo è aciclico quindi è  $S_1$  è CSR. Dato che  $\text{CSR} \subset \text{VSR}$ , allora  $S_1$  è anche VSR.

### 2.4.3 Schedule 2

Dato il seguente schedule:

$$S_2 = r_1(x) \ w_1(x) \ w_3(x) \ r_2(y) \ r_3(y) \ w_3(y) \ w_1(y) \ r_2(x)$$

Le transazioni sono:

$$\begin{array}{lll} T_1 & : & r_1(x) \ w_1(x) \ w_1(y) \\ T_2 & : & r_2(y) \ r_2(x) \\ T_3 & : & w_3(x) \ r_3(y) \ w_3(y) \end{array}$$

Si verifica se è VSR. Si inizia analizzando l'insieme  $S_2$ :

$$\begin{array}{ll} \text{LeggeDa}(S_2) & = \{ (r_2(x), w_3(x)) \} \\ \text{ScrittureFinali}(S_2) & = \{ w_3(x), w_1(y) \} \end{array}$$

Dato che è impossibile provare tutte le combinazioni (3 transazioni e quindi  $3! = 6$ ), si fanno alcune considerazioni. Per esempio, dato che nelle LeggeDa si deve mantenere l'ordine  $(r_2(x), w_3(x))$ , e sapendo che  $r_2$  appartiene a  $T_2$  e  $w_3$  a  $T_3$ , si può concludere che  $T_3$  deve per forza precedere  $T_2$ . Quindi, le combinazioni si riducono a:

- $T_1 \ T_3 \ T_2$
- $T_3 \ T_1 \ T_2$
- $T_3 \ T_2 \ T_1$

Tuttavia, se  $T_3$  anticipa  $T_2$ , tutte le combinazioni avranno come insieme LeggeDa almeno i due valori:

$$\text{LeggeDa}(S_2) = \{ (r_2(x), w_3(x)), (r_2(y), w_3(y)) \}$$

Quindi, è possibile concludere che nessuna combinazione ha un insieme LeggeDa equivalente al LeggeDa di  $S_2$ . È possibile concludere che  $S_2$  non è VSR.

#### 2.4.4 Schedule 3

Dato il seguente schedule:

$$S_3 = r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$$

Le transazioni sono:

$$\begin{aligned} T_1 &: r_1(x) w_1(x) w_1(y) w_1(z) \\ T_2 &: r_2(x) w_2(x) \\ T_3 &: r_3(x) w_3(y) w_3(x) \\ T_4 &: r_4(z) \\ T_5 &: w_5(x) w_5(y) r_5(z) \end{aligned}$$

Si verifica se CSR. Quindi, si cerca l'insieme di conflitti:

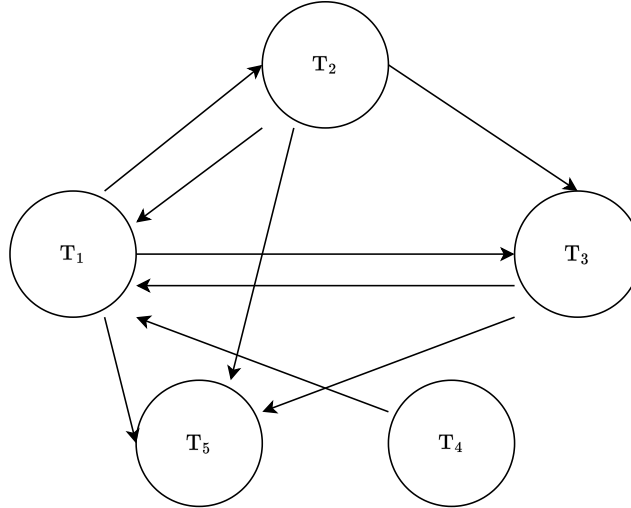
1.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
2.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
3.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
4.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
5.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
6.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
7.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
8.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
9.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
10.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
11.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
12.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
13.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
14.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
15.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
16.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
17.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
18.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
19.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$
20.  $r_1(x) r_2(x) w_2(x) r_3(x) r_4(z) w_1(x) w_3(y) w_3(x) w_1(y) w_5(x) w_1(z) w_5(y) r_5(z)$



L'insieme dei conflitti è quindi formato da:

$$\begin{aligned} \text{Conflitti}(S_3) = \{ & (r_1(x) \ w_2(x)), (r_1(x) \ w_3(x)), (r_1(x) \ w_5(x)), \\ & (r_2(x) \ w_1(x)), (r_2(x) \ w_3(x)), (r_2(x) \ w_5(x)), \\ & (w_2(x) \ r_3(x)), (w_2(x) \ w_1(x)), (w_2(x) \ w_3(x)), (w_2(x) \ w_5(x)), \\ & (r_3(x) \ w_1(x)), (r_3(x) \ w_5(x)), \\ & (r_4(z) \ w_1(z)), \\ & (w_1(x) \ w_3(x)), (w_1(x) \ w_5(x)), \\ & (w_3(y) \ w_1(y)), (w_3(y) \ w_5(y)), \\ & (w_3(x) \ w_5(x)), \\ & (w_1(y) \ w_5(y)), \\ & (w_1(z) \ r_5(z)) \} \end{aligned}$$

Adesso si crea il grafo dei conflitti. Le transazioni sono 5, quindi ci saranno 5 nodi:



Il grafo non è aciclico, quindi  $S_3$  non è CSR, quindi si verifica se può essere VSR. Si inizia calcolando l'insieme LeggeDa e ScrittureFinali:

$$\begin{aligned} \text{LeggeDa}(S_3) &= \{(r_3(x) \ w_2(x)), (r_5(z) \ w_1(z))\} \\ \text{ScrittureFinali}(S_3) &= \{w_5(x), w_1(z), w_5(y)\} \end{aligned}$$

Adesso si cerca almeno una combinazione tra le 5 transizioni, tale per cui una di esse sia view-equivalente a  $S_3$ . Con una considerazione rapida è possibile già ottenere il risultato finale. Guardando l'insieme delle ScrittureFinali, è possibile vedere che la 5ª transizione dovrebbe avere tra la scrittura sulla risorsa  $x$  e tra la scrittura sulla risorsa  $y$ , la scrittura sulla risorsa  $z$ . Questo non è possibile poiché quest'ultima appartiene alla transazione 1. Per cui, è già possibile affermare che nessuna combinazione tra le 5 transazioni, può dare una combinazione view-equivalente a  $S_3$ . Si conclude che  $S_3$  non è VSR e quindi è NonSR.

#### 2.4.5 Schedule 4

Dato il seguente schedule:

$$S_4 = r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$$

Le transazioni sono:

$$\begin{aligned} T_1 & : r_1(x) w_1(y) w_1(t) \\ T_2 & : r_2(z) w_2(z) \\ T_3 & : r_3(y) r_3(z) \\ T_4 & : w_4(x) r_4(t) \\ T_5 & : w_5(x) w_5(z) r_5(t) \end{aligned}$$

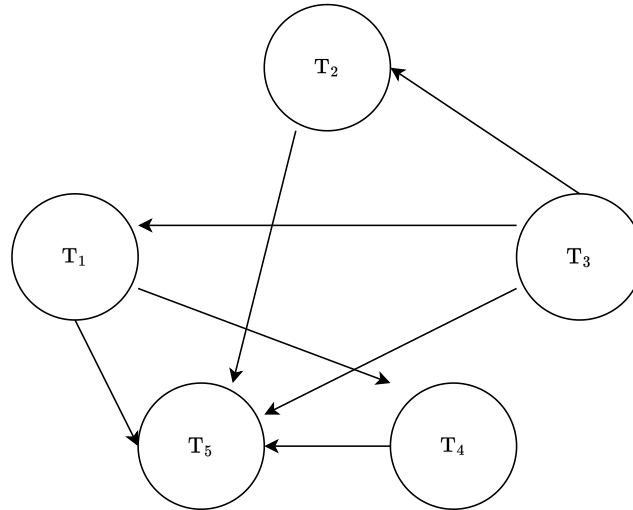
Si verifica CSR prima di tutto. Quindi, si inizia costruendo l'insieme dei conflitti:

1.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
2.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
3.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
4.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
5.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
6.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
7.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
8.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
9.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$
10.  $r_1(x) r_3(y) w_1(y) w_4(x) w_1(t) w_5(x) r_2(z) r_3(z) w_2(z) w_5(z) r_4(t) r_5(t)$

Quindi, l'insieme dei conflitti è:

$$\begin{aligned} \text{Conflitti}(S_4) = \{ & (r_1(x) w_4(x)), (r_1(x) w_5(x)), \\ & (r_3(y) w_1(y)), \\ & (w_4(x) w_5(x)), \\ & (w_1(t) r_4(t)), (w_1(t) r_5(t)) \\ & (r_2(z) w_5(z)), \\ & (r_3(z) w_2(z)), (r_3(z) w_5(z)), \\ & (w_2(z) w_5(z)) \} \end{aligned}$$

Adesso si costruisce il grafo. I nodi sono 5 poiché il numero di transazioni è 5:



Il grafo risulta aciclico, quindi è possibile affermare con certezza che  $S_4$  è CSR e per definizione anche VSR ( $CSR \subset VSR$ ). Per concludere l'esercizio, è necessario calcolare gli schedule seriali equivalenti. Per capire quali combinazioni sono accettate, si fanno alcune considerazioni guardando il grafo:

- Il nodo  $T_5$  non ha archi orientati in uscita, ma solo in entrata. Quindi, è possibile affermare che la transazione  $T_5$  deve essere l'ultima.
- Il nodo  $T_3$  non ha archi orientati in entrata, ma solo in uscita. Quindi, è possibile affermare che la transazione  $T_3$  deve essere la prima.
- I nodi  $T_2$  e  $T_1$  sono gli unici che vengono raggiunti da  $T_3$ , ovvero dal nodo d'inizio. Quindi, è possibile affermare che le uniche combinazioni potranno essere:
  1.  $T_3, T_1, T_2, T_4, T_5$
  2.  $T_3, T_1, T_4, T_2, T_5$
  3.  $T_3, T_2, T_1, T_4, T_5$
  4.  $T_3, T_2, T_4, T_1, T_5$
- Dal nodo  $T_1$  è possibile andare al nodo  $T_4$  senza problemi ma non viceversa poiché altrimenti si creerebbe un ciclo. Quindi le uniche combinazioni ammesse sono:
  1.  $T_3, T_1, T_2, T_4, T_5$
  2.  $T_3, T_1, T_4, T_2, T_5$
  3.  $T_3, T_2, T_1, T_4, T_5$
  4.  $T_3, T_2, T_4, T_1, T_5$

## 2.5 Ottimizzazione e stima di costo

### 2.5.1 Esercizio 1

Si consideri il seguente schema relazionale contenente le ricette di una catena di ristoranti:

INGREDIENTE (Codice, Nome, Calorie);  
COMPOSIZIONE (Ricetta, Ingrediente, Quantità);  
RICETTA (CodiceRicetta, Nome, Regione, TempoPreparazione).

N.B.: la quantità della tabella COMPOSIZIONE è espressa in grammi.

*Vincoli di integrità referenziale:*

COMPOSIZIONE.Ricetta → RICETTA  
COMPOSIZIONE.Ingrediente → INGREDIENTE

Data la seguente interrogazione SQL che consente di: *trovare gli ingredienti usati in ricette della Regione Veneto, riportando, il codice della ricetta e il nome e le calorie dell'ingrediente.*

```
1 SELECT R.CodiceRicetta, I.Nome, I.Calorie
2 FROM RICETTA R JOIN COMPOSIZIONE C
3 ON R.CodiceRicetta = C.Ricetta
4 JOIN INGREDIENTE I
5 ON C.Ingrediente = I.Codice
6 WHERE R.Regione = 'Veneto'
```

Calcolare il costo dell'interrogazione in termini di numero di accessi a memoria secondaria sotto le seguenti ipotesi:

- La selezione su ricetta richiede una scansione sequenziale della tabella RICETTA;
- L'ordine di esecuzione del join è:

$(RICETTA \bowtie COMPOSIZIONE) \bowtie INGREDIENTE$

- Le operazioni di join vengono eseguite con la tecnica Nested Loop join con una pagina di buffer disponibile per ogni tabella;
- Il risultato intermedio del primo join viene interamente memorizzato nel buffer;
- $NP(INGREDIENTE) = 40$ ,  $NP(COMPOSIZIONE) = 200$ ,  $NP(RICETTA) = 12$ ;
- $NR(INGREDIENTE) = 1200$ ,  $NR(COMPOSIZIONE) = 13000$ ,  $NR(RICETTA) = 260$ ;
- $VAL(Regione, RICETTA) = 20$ ,  $VAL(Ricetta, COMPOSIZIONE) = 260$ .

Come cambia il costo se è disponibile un indice B+-tree sull'attributo Codice della tabella INGREDIENTE che ha profondità 2.

Il *nested loop join* è un algoritmo di join che unisce due set usando due cicli nidificati. Quindi, i cicli sono:

1. Viene selezionata la regione Veneta (`WHERE R.Regione = 'Veneto'`) e successivamente eseguito il primo Join:

`seleziona_veneto (RICETTA) ⋈ COMPOSIZIONE`

Quindi il costo risulta essere una lettura della tabella esterna `RICETTA` poiché deve essere applicato una condizione `WHERE` e infine viene eseguita una moltiplicazione:

$$\text{Costo} = \text{NP}(\text{RICETTA}) + \text{NR}(\text{RICETTA con selezione Regione='Veneto'}) \times \text{NP}(\text{COMPOSIZIONE})$$

2. Il secondo Join non prevede condizioni `WHERE` e il costo non tiene in considerazione una lettura da una tabella esterna poiché la tabella interessata è già in buffer:

$$\text{Costo} = \text{NR}(\text{COMPOSIZIONE per le ricette con selezione Regione='Veneto'}) \times \text{NP}(\text{INGREDIENTE})$$

I dati dell'esercizio sui costi sono i seguenti:

Tabella	NP	NR	VAL(Regione)	VAL(Ricetta)
Ricetta	12	260	20	<b>X</b>
Ingrediente	40	1200	<b>X</b>	<b>X</b>
Composizione	200	13000	<b>X</b>	260

I dati presi sono stati acquisiti dai requisiti elencati, niente di difficile. Adesso si cerca di ottenere un valore tangibile, quindi si sostituiscono i valori:

$$\text{Costo} = 12 + \text{NR}(\text{RICETTA}) \div \text{VAL}(\text{Regione}, \text{RICETTA}) \times 200$$

La selezione della regione Veneto sulla tabella `RICETTA` è eseguita con la condizione:

$$\text{NR}(\text{RICETTA}) \div \text{VAL}(\text{Regione}, \text{RICETTA})$$

Andando a sostituire nuovamente i dati:

$$\text{Costo} = 12 + 260 \div 20 \times 200 = 2'612$$

Il costo del secondo Join è possibile calcolarlo allo stesso modo.

$$\text{Costo} = \text{NR}(\text{COMPOSIZIONE}) \div \text{VAL}(\text{Ricetta}, \text{COMPOSIZIONE}) \times 13 \times 40$$

- $\text{NR}(\text{COMPOSIZIONE}) \div \text{VAL}(\text{Ricetta}, \text{COMPOSIZIONE})$ , rappresenta il calcolo da eseguire poiché nella tabella **COMPOSIZIONE** la **Ricetta** è una chiave esterna;
- 13, il valore ottenuto dal costo precedente, ovvero rappresenta il numero di regioni del veneto sulla tabella **RICETTA**;
- 40, rappresenta  $\text{NP}(\text{INGREDIENTE})$ , come da formula.

Quindi, il costo totale del secondo Join:

$$\text{Costo} = 13'000 \div 260 \times 13 \times 40 = 26'000$$

E infine, il costo totale delle operazioni:

$$\text{Costo totale} = 2'612 + 26'000 = 28'612$$

Qua di seguito, viene riportato il costo totale nel caso in cui sia disponibile un indice B+-tree sull'attributo Codice della tabella **INGREDIENTE** che ha profondità 2. Il costo sul primo Join non viene influenzato poiché non riguarda la tabella **INGREDIENTE** e quindi rimane uguale a 2'612. Invece, il secondo Join viene influenzato poiché al posto di utilizzare  $\text{NP}(\text{INGREDIENTE})$ , si utilizza la profondità dell'albero, ovvero:

$$\text{Costo secondo Join} = (13'000 \div 260) \times 13 \times (2 + 1) = 650 \times 3 = 1'950$$

E quindi il costo totale:

$$\text{Costo totale} = 2'612 + 1'950 = 4'562$$

### 2.5.2 Esercizio 2

Si consideri il seguente schema relazionale contenete le ricette di una catena di ristoranti:

RICETTA (Nome, Descrizione, Regione) ;  
IN\_RICETTA (Ricetta, Ingrediente, Quantità) ;  
INGREDIENTE (Nome, Descrizione) .

N.B.: la quantità della tabella COMPOSIZIONE è espressa in grammi.

*Vincoli di integrità referenziale:*

IN\_RICETTA.Ricetta  $\rightarrow$  RICETTA  
IN\_RICETTA.Ingrediente  $\rightarrow$  INGREDIENTE

Data la seguente interrogazione SQL:

```
1 SELECT IR.Ingrediente, IR.Quantita, IR.Ricetta, R.Regione
2 FROM IN_RICETTA IR, RICETTA R
3 WHERE IR.Ricetta = R.Nome AND
4       R.Regione = 'Veneto' AND
5       IR.Quantita = 4
```

Calcolare il costo dell'interrogazione in termini di numero di accessi a memoria secondaria sotto le seguenti ipotesi:

- La selezione sulla tabella IN\_RICETTA eseguita attraverso una scansione della tabella medesima, il risultato della selezione viene salvato in memoria secondaria in 180 pagine e riutilizzato per il join;
- La selezione delle ricette viene eseguita attraverso una scansione della tabella RICETTA, il risultato della selezione viene mantenuto nel buffer;
- L'ordine di esecuzione del join è:

RICETTA  $\bowtie$  IN\_RICETTA

E le operazioni di join vengono eseguite con la tecnica "Nested Loop Join" con una pagina di buffer disponibile per ogni tabella;

- NP (INGREDIENTE) = 30, NP (IN\_RICETTA) = 780, NP (RICETTA) = 150;
- NR (INGREDIENTE) = 270, NR (IN\_RICETTA) = 19'800, NR (RICETTA) = 900;
- VAL (Regione, RICETTA) = 18, VAL (Ricetta, IN\_RICETTA) = 90,  
VAL (Quantita, IN\_RICETTA) = 50.

Come cambia il costo se è disponibile un indice B+-tree sull'attributo Ricetta della tabella IN\_RICETTA che ha profondità 2.

I dati dell'esercizio sui costi sono i seguenti:

Tabella	NP	NR	VAL(Regione)	VAL(Ricetta)	VAL(Quantita)
Ricetta	150	900	18	<b>X</b>	<b>X</b>
In_ricetta	780	19800	<b>X</b>	90	50
Ingrediente	30	270	<b>X</b>	<b>X</b>	<b>X</b>

Come viene esplicitato dall'esercizio, la selezione sulla tabella IN\_RICETTA viene eseguita:

1. Facendo una scansione della tabella medesima:

$$NP(IN\_RICETTA) = 780$$

2. Il risultato della selezione viene salvato in memoria secondaria in 180 pagine, quindi:

$$NP(IN\_RICETTA) = 780 \implies 180$$

3. Viene utilizzato il valore per il Join. Quindi, dato che non vi sono letture di tabelle poiché il buffer contiene già i valori d'interesse, il costo è:

$$\begin{aligned}
 \text{Costo} &= NR(RICETTA) \div VAL(Regione, RICETTA) \times NP(IN\_RICETTA) \\
 &= 900 \div 18 \times 180 \\
 &= 50 \times 180 \\
 &= 9'000
 \end{aligned}$$

Ricordando che si utilizzano 180 pagine e non 780 (punto 2).

Quindi, il costo totale è dato, rispettivamente, dalla somma delle pagine di IN\_RICETTA, dalle pagine scritte nella memoria secondaria che si riferiscono a IN\_RICETTA, dal Join eseguito qua sopra e infine dalla scansione della tabella RICETTA:

$$\text{Costo totale} = 780 + 180 + 9'000 + 150 = 10'110$$

Per quanto riguarda il caso in cui fosse presente un indice B+-tree, è necessario modificare il Join. Quindi diventa la formula:

$$\begin{aligned}
 \text{Costo Join} &= NR(RICETTA \text{ con selezione}) \div VAL(Regione, RICETTA) \times \\
 &\quad (\text{Prof. Indice} + NR(IN\_RICETTA \text{ con selezione}) \div VAL(q.ta, IN\_RICETTA)) \\
 &= 900 \div 18 \times (19'800 \div 50) \div 90 \\
 &= 220
 \end{aligned}$$

Il totale quindi diventa:

$$\text{Costo totale} = 780 + 180 + 150 + 220 = 1'330$$



### 2.5.3 Esercizio 3

Si consideri il seguente schema relazionale:

COMUNE (CodISTAT, Nome, Abitanti, Superficie, Prov, TipoTerritorio)  
ADIACENTE (Comune1, Comune2)  
PROVINCIA (Codice, NomeProv, SuperficieProv, Regione)

Con il dato TipoTerritorio: {montagna, mare, pianura, collina}.

La tabella ADIACENTE rappresenta la relazione di adiacenza tra comuni: poiché la relazione è simmetrica per rappresentare l'adiacenza tra i comuni A e B si memorizza sia la tupla (A,B) sia la tupla (B,A).

Vincoli d'integrità:

ADIACENTE.Comune1 → COMUNE  
ADIACENTE.Comune2 → COMUNE  
COMUNE.Prov → PROVINCIA

Dato lo schema dell'esercizio, si consideri la seguente interrogazione:

```
1 SELECT Nome, Abitanti, Superficie, Comune2
2 FROM COMUNE JOIN ADIACENTE ON CodISTAT = Comune1
3 WHERE TipoTerritorio <> 'pianura'
```

Indicare una stima del costo dell'interrogazione in termini di numero di accessi a memoria secondaria sapendo che: (i) la selezione dei comuni viene eseguita attraverso una scansione della tabella COMUNE e il risultato viene mantenuto nel buffer:

- L'ordine di esecuzione del join è:

COMUNE ⋈ ADIACENTE

E viene applicata la tecnica “Nested Loop Join” e con indice B+-tree di profondità 3 sull'attributo Comune1 della tabella ADIACENTE;

- I dati sono:

NP (COMUNE) = 15;  
NR (COMUNE) = 1'900;  
NP (ADIACENTE) = 155;  
NR (ADIACENTE) = 38'000;  
VAL (TipoTerritorio, COMUNE) = 4;  
VAL (Comune1, ADIACENTE) = 1'900;

I dati messi a disposizione sono i seguenti:

Tabella	NP	NR	VAL(TipoTerritorio)	VAL(Comune1)
COMUNE	15	1'900	4	<b>X</b>
ADIACENTE	155	38'000	<b>X</b>	1'900
PROVINCIA	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>

Viene richiesto il costo per:

- Eseguire una scansione della tabella **COMUNE** poiché viene eseguita una selezione dei comuni. Questo risultato è semplice poiché corrisponde a:

$$\text{Costo} = \text{scrittura} + \text{lettura} = \text{NP}(\text{COMUNE}) + 0 = 15$$

- Eseguire il Join con la tecnica “Nested Loop Join” senza albero B+-tree. Quindi, viene calcolato nel seguente modo il costo:

$$\text{Costo} = 0 + \text{NR}(\text{COMUNE}) \div \text{VAL}(\text{TipoTerritorio}, \text{COMUNE}) \times \text{NP}(\text{ADIACENTE})$$

Il valore zero indica che nel buffer non viene eseguita nessuna scrittura poiché è già stata fatta in precedenza (punto precedente). Quindi, andando a sostituire i valori:

$$\text{Costo} = 0 + 1'900 \div 4 \times 155 = 475 \times 155 = 73'625$$

- Eseguire il Join con la tecnica “Nested Loop Join” con albero B+-tree di profondità 3 sull'attributo Comune1. Quindi, viene calcolato nel seguente modo il costo:

$$\begin{aligned}
 \text{Costo} &= 0 + \text{NR}(\text{COMUNE}) \div \text{VAL}(\text{TipoTerritorio}, \text{COMUNE}) \times \\
 &\quad (\text{Profondità Indice} + \text{NR}(\text{ADIACENTE}) \div \text{VAL}(\text{Comune1}, \text{ADIACENTE})) \\
 &= 0 + 1'900 \div 4 \times (3 + 38'000 \div 1'900) \\
 &= 475 \times 23 \\
 &= 10'925
 \end{aligned}$$

## 2.6 XML

### 2.6.1 Esercizio 1

Dato il seguente frammento XML:

```
1 <banca>
2   <conto>
3     <num_conto>101</num_conto>
4     <agenzia>Citta</agenzia>
5     <saldo>555</saldo>
6     <cliente>
7       <nome>Mario</nome>
8       <cognome>Rossi</cognome>
9       <via>Garibaldi</via>
10      <citta>Verona</citta>
11    </cliente>
12  </conto>
13  <conto>
14    <num_conto>1010</num_conto>
15    <agenzia>Desenzano</agenzia>
16    <saldo>5550</saldo>
17    <cliente>
18      <nome>Mario</nome>
19      <cognome>Rossi</cognome>
20      <via>Garibaldi</via>
21      <citta>Verona</citta>
22    </cliente>
23  </conto>
24  <conto>
25    <num_conto>301</num_conto>
26    <agenzia>Negrar</agenzia>
27    <saldo>1020</saldo>
28    <cliente>
29      <nome>Paolo</nome>
30      <cognome>Verdi</cognome>
31      <via>Roma</via>
32      <citta>Verona</citta>
33    </cliente>
34  </conto>
35 </banca>
```

- Generare il corrisponde XSD, supponendo che ogni conto abbia uno o più clienti intestatari.
- Modificare la struttura dell'XML per ridurre la ridondanza e rappresentare una sola volta gli elementi ripetuti.

### 2.6.2 Esercizio 2

## 3 Domande di teoria - Terzo parziale

Domande di teoria tratte dalla terza prova intermedia dell'esame 06/2015.

1. (*3 punti*) Illustrare l'architettura di un DBMS descrivendo in particolare il modulo di gestione dei buffer; si indichi inoltre, per ogni modulo dell'architettura, quali sono le proprietà delle transazioni che contribuisce a garantire.

2. (*2 punti*) Si presenti in dettaglio la definizione di Conflict-Serializzabilità (CSR).
3. (*2 punti*) Lo studente illustri la struttura di accesso ai dati denominata indice primario denso: caratteristiche della struttura, ricerca, inserimento e cancellazione di entry dall'indice.

Domande di teoria tratte dalla terza prova intermedia dell'esame 07/06/2016.

1. (*3 punti*) Illustrare l'architettura di un DBMS descrivendo in particolare il modulo di gestione dei guasti (o gestore dell'affidabilità); si indichi inoltre, per ogni modulo dell'architettura, quali sono le proprietà delle transazioni che contribuisce a garantire.
2. (*3 punti*) Si presenti in dettaglio la politica di concessione dei lock applicata dal gestore dell'esecuzione concorrente secondo la tecnica detta "Locking a due fasi".
3. (*2 punti*) Lo studente illustri la struttura di accesso ai dati denominata indice primario sparso, si descrivano in particolare i seguenti punti: (i) le caratteristiche della struttura di accesso, (ii) l'algoritmo di ricerca di una tupla con chiave K usando l'indice.
4. (*2 punti*) Lo studente illustri le differenze tra i costrutti **elemento** e **attributo** del linguaggio XML, mostrando un esempio di documento XML dove vengono utilizzati entrambi.

Domande di teoria tratte dalla terza prova intermedia dell'esame 21/04/2022.

1. (*3 punti*) Illustrare l'architettura di un DBMS descrivendo in particolare il modulo di gestione dei buffer; si indichi inoltre quali moduli garantiscono le proprietà di persistenza e consistenza delle transazioni.
2. (*2 punti*) Si presenti in dettaglio la definizione di conflict-equivalenza tra due schedule.
3. (*2 punti*) Lo studente illustri la struttura di accesso ai dati denominata struttura ad accesso calcolato (hashing), si descrivano in particolare i seguenti punti: (i) le caratteristiche della struttura di accesso, (ii) l'algoritmo di ricerca di una tupla con chiave K usando l'indice.
4. (*2 punti*) Lo studente illustri l'algoritmo di ripresa a caldo.

Domande di teoria tratte dalla terza prova intermedia dell'esame 22/04/2022.

1. (*3 punti*) Illustrare le proprietà delle transazioni; si indichi inoltre quali moduli di un DBMS garantiscono ciascuna di tali proprietà.
2. (*2 punti*) Si presenti in dettaglio la definizione di view-equivalenza tra due schedule.
3. (*2 punti*) Lo studente illustri la struttura di accesso ai dati denominata B+-tree, si descrivano in particolare i seguenti punti: (i) le caratteristiche della struttura di accesso, (ii) l'algoritmo di ricerca di una tupla con chiave K usando l'indice.
4. (*2 punti*) Illustrare il meccanismo di 2PL stretto.

Domande di teoria tratte dalla terza prova intermedia dell'esame 10/06/2022.

1. (*2 punti*) Illustrare le proprietà delle transazioni ed indicare da quali moduli del DBMS vengono garantite.
2. (*3 punti*) Si presenti la definizione di View-serializzabilità e la relazione tra l'insieme degli schedule VSR e l'insieme degli schedule CSR. Presentare la dimostrazione formale di tale relazione.
3. (*2 punti*) Lo studente illustri la struttura di accesso ai dati denominata indice secondario, si descrivano in particolare i seguenti punti: (i) le caratteristiche della struttura di accesso, (ii) l'algoritmo di ricerca di una tupla con chiave K usando l'indice.
4. (*2 punti*) Lo studente illustri l'algoritmo di ripresa a caldo.



## 4 Esami terzo parziale

4.1 Terzo parziale - 06/2015

4.2 Terzo parziale - 07/06/2016

4.3 Terzo parziale - 21/04/2022

4.4 Terzo parziale - 22/04/2022

4.5 Terzo parziale - 10/06/2022