Esami - Basi di dati

VR443470

giugno 2023

Indice

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.*

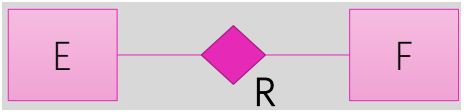
   Semantica. L'entità rappresenta una classe di oggetti che hanno proprietà comuni ed esistenza "autonoma" ai fini dell'applicazione di interesse. Il nome dato ad ogni entità è identificativo di quella determinata classe di oggetti e deve essere univoco all'interno dello schema.
   Sintassi grafica. Per 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 $E_1, E_2, ..., E_n$, un'istanza è composta da una ennupla del tipo:

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

Inoltre, esiste un'importante proprietà che afferma:

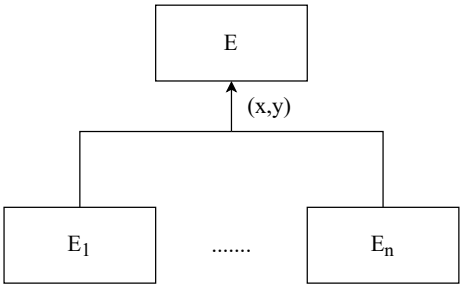
$$I(R) \subseteq I(E_1) \times I(E_2) \times ... \times I(E_n)$$

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 $E_1, ..., E_n$, chiamate figlie. Quindi, si dice che l'entità E (genitore) è la generalizzazione delle entità $E_1, ..., E_n$ (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 classificazioni 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:

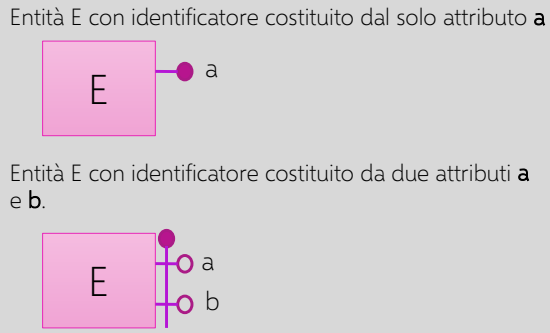


4. *Si illustri il concetto/costrutto di **identificatore** nel modello Entità-Relazione.*

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.

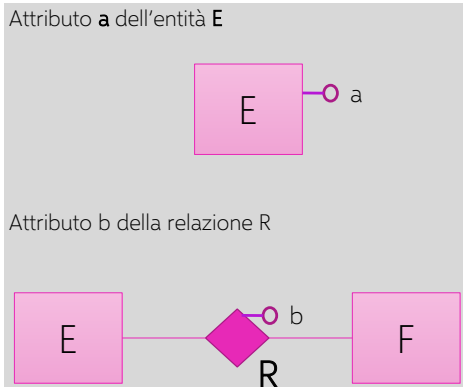
6. *Si illustri il concetto/costrutto di **attributo** nel modello Entità-Relazione.*

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:

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

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$^+$-tree

2.1.1 Esercizio 1

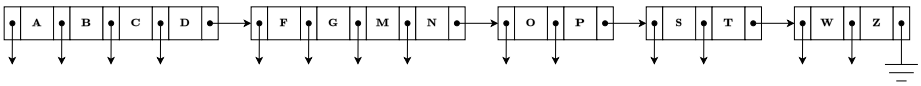
Costruire un B$^+$-tree di fan-out = 5 con i seguenti nodi foglia: (A, B, C, D), (F, G, M, N), (O, P), (S, T), (W, Z). I vincoli di riempimento sono:

- $2 \leq$ #chiavi ≤ 4
- $3 \leq$ #puntatori ≤ 5

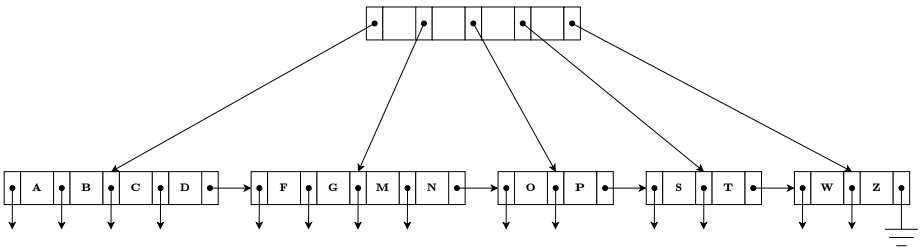
Dopodiché, inserire il valore chiave H nel B$^+$-tree ottenuto precedentemente. Infine, l'esercizio si conclude eseguendo una rimozione del valore chiave Z ottenuto precedentemente.

Soluzione

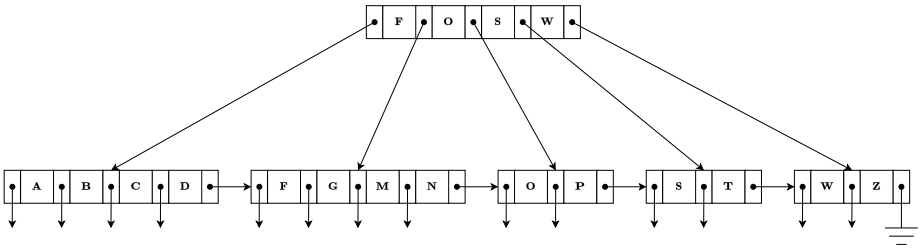
Il primo passo è costruire i vari livelli dei nodi foglia:



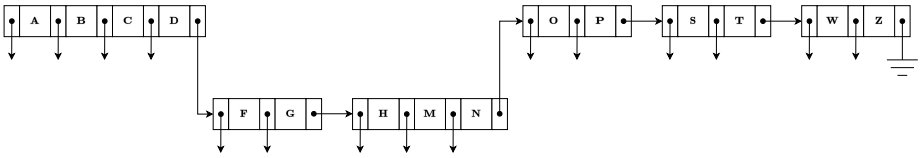
Adesso è necessario costruire tutti i puntatori richiesti. Fan-out è uguale a 5 quindi viene costruito un nodo intermedio con 5 puntatori e si collegano tutti i nodi:



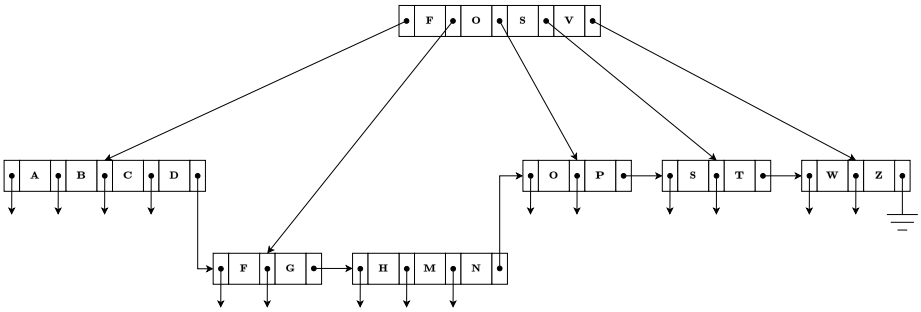
Adesso si aggiungono le lettere che devono essere raggiunte dopo aver visitato ogni nodo:



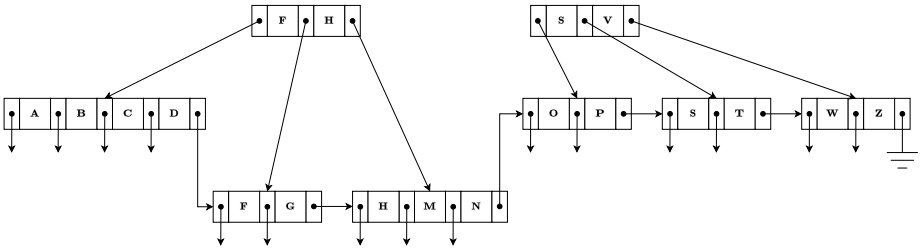
Per inserire il valore chiave è necessario avere a disposizione una posizione libera. Tuttavia, questo non è possibile, per cui viene applicato uno split. Viene divisa la radice contenente (F, G, M, N) così da inserire la chiave H tra la G e la M:



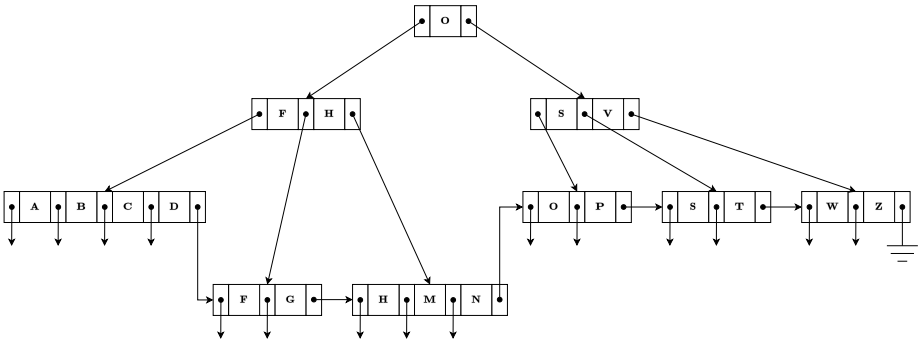
A questo punto è necessario riadattare il nodo radice che attualmente punta ad un nodo errato (attenzione c'è un errore, il nodo V in realtà è il nodo W):



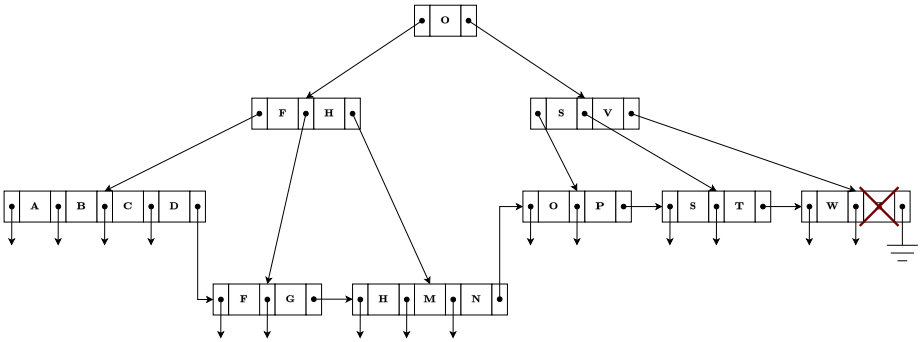
Per farlo, è necessario eseguire una divisione anche nel nodo radice aggiustando i valori delle chiavi (attenzione c'è un errore, il nodo V in realtà è il nodo W):



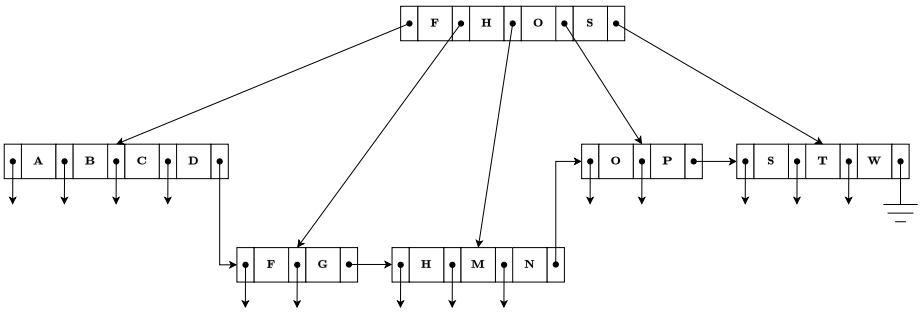
E infine, collegare i due nodi divisi con un nodo di congiunzione. Inoltre, quest'ultimo viene riempito con un valore chiave (attenzione c'è un errore, il nodo V in realtà è il nodo W):



La rimozione della chiave Z comporta che l'ultimo nodo abbia come chiave solo il valore W. Questo comporta un'irregolarità poiché il numero minimo di ogni chiave in ogni nodo deve essere minimo di due e massimo di quattro. Per cui è necessario effettuare un merge:



Eliminazione della chiave Z.



Merge degli ultimi due nodi.

2.2 Verificare che uno schedule sia VSR (View-serializzabilità)

2.2.1 Esercizio 1 - Perdita di aggiornamento

Date due transazioni T_1 e T_2 di seguito descritte:

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

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 prime 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:

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

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:

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

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 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 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}:

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

   Come è evidente, nessuno dei due insiemi è equivalente:

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

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

Verifica partendo dallo schedule S_2:

1. Creazione dell'insieme 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 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}:

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

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

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

   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:

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

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 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 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}:

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

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

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

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

Verifica partendo dallo schedule S_2:

1. Creazione dell'insieme 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 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}:

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

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

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

   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

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:

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

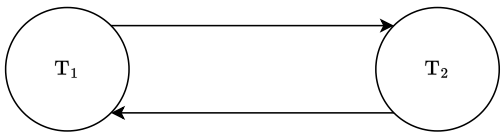
Per verificare CSR è necessario caratterizzare S_{PA} calcolando l'insieme dei conflitti. Si ricorda che due azioni sono in conflitto se operano sullo stesso oggetto e se almeno una di esse è in scrittura (quindi le combinazioni: rw, wr, ww). Quindi, si calcola l'insieme dei conflitti di S_{PA}:

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:

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

Lo schedule che rappresenta l'anomalia è il seguente:

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

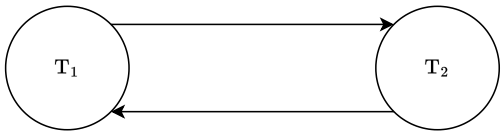
Per verificare CSR è necessario caratterizzare S_{LI} calcolando l'insieme dei conflitti. Si ricorda che due azioni sono in conflitto se operano sullo stesso oggetto e se almeno una di esse è in scrittura (quindi le combinazioni: rw, wr, ww). Quindi, si calcola l'insieme dei conflitti di S_{LI}:

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 costruire 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:

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

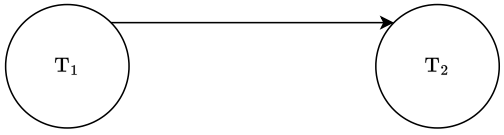
Si verifica se è CSR. Quindi, si crea l'insieme dei conflitti:

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 CSR $\subset$ 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:

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

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

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

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:

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

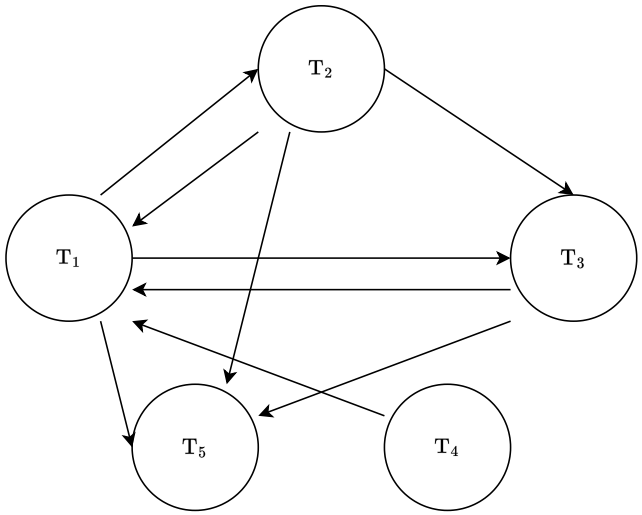
Si verifica se CSR. Quindi, si cerca l'insieme di conflitti:

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

L'insieme dei conflitti è quindi formato da:

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

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 5^a 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:

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

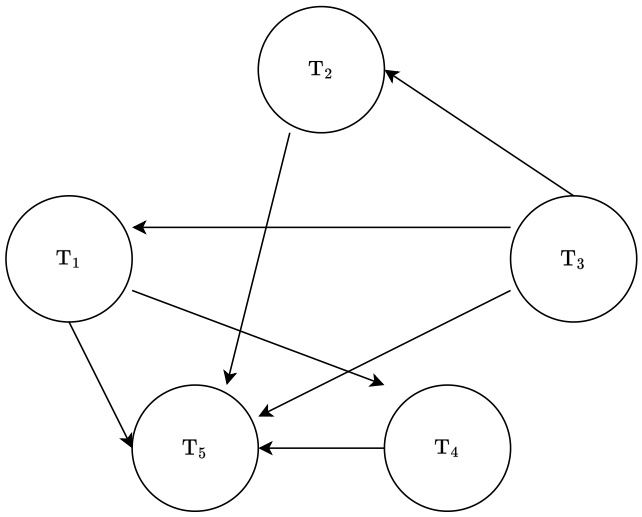
Si verifica CSR prima di tutto. Quindi, si inizia costruendo l'insieme dei conflitti:

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

Quindi, l'insieme dei conflitti è:

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

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 una 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