

PROBLEMA ESECUZIONE CONCORRENTE:

Concetto di SCHEDULE del DBMS:

Sequenza di OPERAZIONI (lettura / Scrittura) eseguite su una RISORSA della Nostra Base di Dati da parte di Transazioni concorrenti.

S. $r_1(x)$ $r_2(z)$ $w_1(x)$ $w_2(z)$

Risorsa su cui lavora

Transazione

Schedule = Possibile Esecuzione Concorrente di Transazioni

Quali SCHEDULE evitano le Anomalie?

Andiamo ad Accettare gli Schedule che sono EQUIVALENTI ad uno SCHEDULE SERIALE

Quando le Operazioni di Ogni Transazione Compiono in Sequenza SENZA ESSERE INFRAMEZZATE da operazioni di altre TRANSAZIONI

$t_1: r_1(x)$ $w_1(x)$
 $t_2: r_2(z)$ $w_2(z)$

SCHEDULE SERIALE $S_1: t_1 t_2$
oppure $S_2: t_2 t_1$

Voglio Introdurre la CONCORRENZA mantenendo l'EQUIVALENZA

SCHEDULE è Serializzabile se è EQUIVALENTE ad una SCHEDULE SERIALE

EQUIVALENZA \Rightarrow Si Basa sugli Effetti prodotti Sulla BASE DI DATI, se hanno gli STESSI EFFETTI Sulla BASE DI DATI

HIPOTESI DI COMMIT PROIEZIONE \Rightarrow Posso Supporre che le ti abbiano un ESITO NOTO, quindi noi considereremo solo quelle che sono andate a buon fine

VIEW-EQUIVALENZA \Rightarrow Si BASA su 2 Relazioni fondamentali:

"LEGGE DA": dato uno Schedule S, si dice che un'operazione di lettura $r_i(x)$ che compare in S, legge da un'operazione di Scrittura $w_j(x)$ che compare in S se valgono le seguenti:

① $w_j(x)$ precede $r_i(x)$

② Se non esiste alcuna Operazione $w_k(x)$

Tra le 2

N.B! Sempre sulla stessa risorsa x

$S_1: r_1(x) \ r_2(x) \ w_1(x) \ w_2(x)$ Legge-Da (S_1) = \emptyset

$S_2: w_1(x) \ r_1(x) \ \underbrace{r_2(x)}_{\text{↑}} \ w_2(x)$ Legge-Da (S_2) = $\{r_2(x), w_1(x)\}$

SCRITTURE FINALI: dato SCHEDULE S si dice che un'operazione $w_i(x)$ è finale se è l'ultima operazione di scrittura della RISORSA x in S

$S_1: r_1(x) \quad r_2(x) \quad w_2(x) \quad w_3(x) \quad w_1(y)$

Scritture-finali (S_1) = $\{w_3(x), w_1(y)\}$

Per Ogni RISORSA Guardo Qual'è L'ULTIMA

2 Schedule Sono VIEW-EQUIVALENTE ($S_1 \approx S_2$) Se Possiedono le Stesse Relazioni "legge-da" e Scritture finali

S_1 È VIEW-SERIALIZZABILE se VIEW-EQUIVALENTE ad uno Schedule Seriele S_R

Posso Cambiare ORDINE delle TRANSAZIONI, ma non le operazioni all'Interno delle TRANSAZIONE

ESERCIZIO TIPO ESAME:

PERDITA DI AGGIORNAMENTO:

$T_1: r_1(x) \quad w_1(x)$

$T_2: r_2(x) \quad w_2(x)$

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

genera PROBLEMI

è VIEW-SERIALIZZABILE?

definisce:

$\text{LEGGE-DA} = \emptyset$

risorsa UNICA

$\text{SCRITTURE FINALI} = \{w_1(x)\}$

Per dare la RISPOSTA devo Generare Tutti i possibili Schedule Seriali e Trovare, se c'è, Quello con gli Stessi Schemi

④ GENERA SCHEDULE SERIALI

- $S_{12}: r_1(x) \quad w_1(x) \quad r_2(x) \quad w_2(x)$

- $S_{21}: r_2(x) \quad w_2(x) \quad r_1(x) \quad w_1(x)$

$\text{Legge-Da } (S_{12}) = \{r_2(x), w_1(x)\}$

$\text{Scritture Finali } (S_{12}) = \{w_2(x)\}$

$\} \neq \text{Quindi NON VIEW EQUIVALENTE}$

$$\text{legge-Da } (S_{21}) = \left\{ r_1(x), W_2(x) \right\}$$
$$\text{Scritture finali} = \left\{ W_1(x) \right\}$$

Anche Qui legge Da \neq

→ schedule Seriele \Rightarrow Uno Scheduler con PERDITA di Aggiorn.
non è VIEW SERIALIZZABILE (VSR)

LETTURA INCONSIESTENTE:

$$T_1 : r_i(x) \dots r_i'(x)$$

$$\bar{T}_2 : r_2(x) \quad w_2(x)$$

$$S_{LI} : r_i(x) \quad r_2(x) \quad w_2(x) \quad r_i'(x)$$

$$\text{Legge-Da } (S_{LI}) = \{r_i'(x), w_2(x)\}$$

$$\text{Scritture finali} = \{w_2(x)\}$$

Schedule Seriali:

$$S_{1,2} : r_i(x) \quad r_i'(x) \quad r_2(x) \quad w_2(x)$$

$$S_{2,1} : r_2(x) \quad w_2(x) \quad r_i(x) \quad r_i'(x)$$

$$\text{Legge-da } (S_{1,2}) = \emptyset$$

$$\text{Scritture finali} = \{w_2(x)\}$$

$$S_{LI} \not\approx_{v} S_{1,2}$$

$$\text{Legge-da } (S_{2,1}) = \{r_i(x), w_2(x), r_i'(x), w_2(x)\}$$

$$\text{Scritture finali} = \{w_2(x)\}$$

$$S_{2,1} \not\approx_{v} S_{LI}$$

S_{LI} non è VSR

ESEMPI PIÙ COMPLICATI: (da Esame)

S: $r_1(x) w_1(x) r_2(z) r_1(y) w_1(y) r_2(x) w_2(x) w_2(z)$

Legge-Da = $\{r_2(x), w_1(x)\}$

Scritture finali = $\{w_2(z), w_2(x), w_1(y)\}$


Partice del fondo e cercare la prima
occorrenza

$T_1: r_1(x) w_1(x) r_1(y) w_1(y)$
 $T_2: r_2(z) r_2(x) w_2(x) w_2(z)$

} date dall'Esercizio

$S_{12}:$

Legge-Da = $\{r_2(x), w_1(x)\}$

Scritture finali = $\{w_2(z), w_2(x), w_1(y)\}$

Questo è VSR essendo VIEW EQUIVALENTE

TEST VIEW-EQUIVALENZA ha Complessità Lineare, il problema è il TEST di VSR che ha Complessità ESPOENZIALE

Genero Tutti i Possibili
Schedule Seriali

VSR ha 2 problemi:

- COMPLESSITÀ
- IPOTESI DI COMMIT PROIEZIONE

RISOLVIAMO PROBLEMA DELLA COMPLESSITÀ:

CONFLICT-EQUIVALENZA \Rightarrow Si basa sul CONFLITTO, dato una Schedule S, una coppia di OPERAZIONI (a_i, a_j) rappresentano un Conflitto se:

- ① $i \neq j$ (operazioni di Trans. \neq)
- ② Operano Sulla Stessa RISORSA
- ③ Almeno una è un'operazione di Scrittura
- ④ a_i compare in S prima di a_j

2 Schedule sono C.E. se hanno lo stesso Insieme di Conflitti

S è CSR se è CONFLICT EQUIVALENTE ad una Schedule Seriele S_s

ed è migliore perché il TEST ha Complessità Lineare per Scorrere S e popolare il Grafo dei CONFLITTI dove i nodi sono le Transazioni e gli archi i Conflitti



Solo se ho (a_i, a_j) in S

Se Grafo è ACICLICO allora è CONFLICT Serializzabile

ESEMPIO COSTRUZIONE GRAFO:

$S : r_1(x) \quad w_1(x) \quad r_2(z) \quad r_1(y) \quad w_1(y) \quad r_2(x) \quad w_2(x) \quad w_2(z)$

$$\text{Conflict}(S) = \left\{ \underbrace{(r_1(x), w_2(x))}_{\text{Co ho già}}, \underbrace{(w_1(x), r_2(x))}_{\text{Co ho già}}, \underbrace{(w_1(x), w_2(x))}_{\text{Co ho già}} \right\}$$

per ciascuna RISORSA

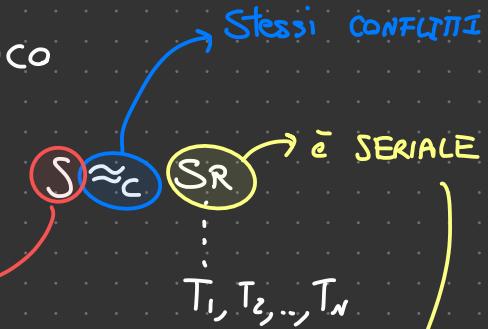


Grafo è ACICLICO, quindi Schedule S è CSR

DIMOSTRAZIONE:

$S \in CSR \Leftrightarrow$ grafo è ACICCLICO

① (\Rightarrow) $S \in CSR \Rightarrow \exists S_R .$



archi Possono Essere Solo del TIPO

(a_i, a_j) con $i < j$ essendo le Transazioni ordinate



Quindi un GRAFO ACICCLICO

② (\Leftarrow) se S ha grafo aciclico $\Rightarrow S \in CSR$



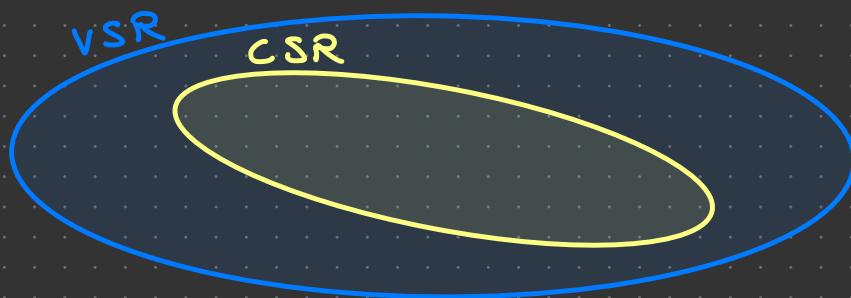
Posso definire un ORDINAMENTO TOPOLOGICO tra i NODI (Seguire gli Archi e ordinare le Transazioni)

Posso quindi ENUMERARE i NODI e avrò Archi: (i, j) con $i < j$ per l'ordinamento fatto

Se Seguo l'
ORDINAMENTO TOPOLOGICO
e definisco uno
Scheduler S che
Esegue le Transazioni
NELL'ORDINE stabilito
dell'ORDINAMENTO
Topologico

È UNO SCHEDULE SERIALE CON STESSO GRAFO DEI CONFLITTI ESSENDO PARTITO DA QUELLO.

RELAZIONE TRA CSR e VSR:



CSR è una Condizione SUFFICIENTE ma NON necessaria per VSR

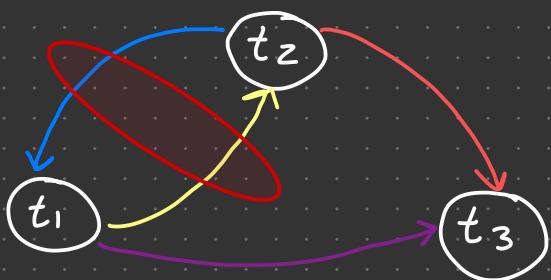
S: $r_1(x)$ $w_2(x)$ $w_1(x)$ $w_3(x)$

$$\left. \begin{array}{l} \text{Legge-Da}(S) = \emptyset \\ \text{Scritture finali} = \{w_3(x)\} \end{array} \right\} \begin{array}{l} T_1: r_1(x) w_1(x) \\ T_2: w_2(x) \\ T_3: w_3(x) \end{array}$$

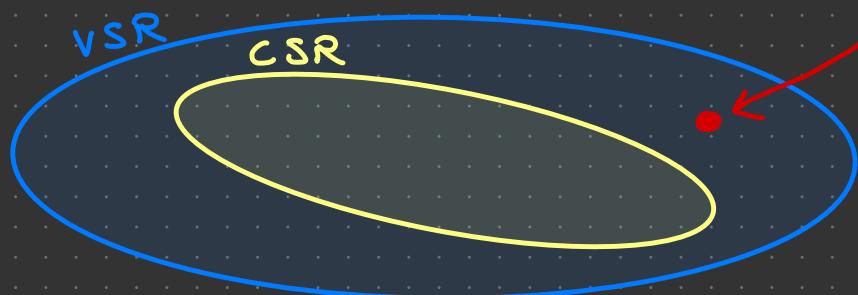
$S_{123}: T_1 T_2 T_3$

$$\left. \begin{array}{l} \text{Legge-Da}(S_{123}) = \emptyset \\ \text{Scritture finali}(S_{123}) = \{w_3(x)\} \end{array} \right\} \text{Quindi } S_{123} \text{ è VSR}$$

$$\text{Conflitti}(S) = \left\{ (r_1(x), w_2(x)), (r_1(x), w_3(x)), (w_2(x), w_3(x)), (w_2(x), w_1(x)), (w_1(x), w_3(x)) \right\}$$



S NON È CSR, allora S sta qui



CSR \Rightarrow VSR:

$S_1 \approx_c S_2$

Perché Siano \approx_v devono Avere:

■ Stesso Insieme Scritture finali

Se i 2 Insiemi di Scritture finali fossero diverse, allora vuol dire che \exists due Scritture fatte sulla stessa risorsa ciascuna da una Transazione che vengono messe in ordine opposto

■ siccome coppie di Scrittura Sulle STESSA risorse Sono un Conflitto, se Cambio l'ordine Cambio il Grafo

■ Legge-Da Uguali, altrimenti ho Tolto un Conflitto, se (aggiunge) non sono più CONFLICT EQUIVALENTI avendo aggiunto un CONFLITTO

↓
Modificando legge-d- Modifico insieme dei CONFLITTI

PERDITA AGGIORNAMENTO:

$$S_{PA} = r_1(x) r_2(x) W_2(x) W_1(x)$$

$$\text{CONFLITTO } (S_{PA}) = \left\{ \underline{(r_1(x), W_2(x))}, \underline{(r_2(x), W_1(x))}, \underline{(W_2(x), W_1(x))} \right\}$$



non è CSR

LETTURA INCONSISTENTE:

$$\text{Consistency (SLI)} = \left\{ \left(r_1(x), w_2(x) \right), \left(w_2(x), r'_2(x) \right) \right\}$$

