**Dire se il seguente schedule è in CSR, se è VSR e se rispetta la 2PL**

*Uno schedule è in CSR se il grafo dei conflitti è aciclico*. Per costruirci il grafo dei conflitti, classifichiamo le operazioni delle varie transazioni per gli oggetti in gioco .

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

I tipi di conflitti possibili sono:

* **Conflitti RW**, T2 scrive un oggetto precedentemente letto da T1
* **Conflitti WR**, T1 legge un oggetto precedentemente scritto da T2 che non è ancora conclusa
* **Conflitti WW**, T2 scrive un oggetto precedentemente scritto da T1 (per cui sovrascrive il cambiamento di T1)

Vediamo quindi possibili schemi partendo dalla tabella.

* Conflitti su
  + In c’è un conflitto RW tra la transazione e
  + In c’è un conflitto WW tra la transazione e
  + In c’è un conflitto RW tra la transazione e
* Conflitti su
  + In c’è un conflitto RW tra la transazione e
* Conflitti su
  + In c’è un conflitto RW tra la transazione e
  + In c’è un conflitto RW tra la transazione e
  + In c’è un conflitto RW tra la transazione e
  + In c’è un conflitto WW tra la transazione e
* Conflitti su
  + In c’è un conflitto WR tra la transazione e
  + In c’è un conflitto WR tra la transazione e

Rappresentiamo il grafo dei conflitti:

Immagine che contiene cerchio, luna, sfera

Descrizione generata automaticamente

Il grafo dei conflitti è aciclico quindi lo schedule è in CSR. *Uno schedule è in VSR se è in CSR*. Quindi lo schedule è anche in VSR.

Verifichiamo se lo schedule rispetta la 2PL. *Uno schedule rispetta la 2PL se non ci sono deadlock*.

Vediamo cosa succede ogni singolo step ricordando che:

Se una transazione chiede un allora questo viene concesso se:

* La risorsa è completamente libera (sia in che )
* La risorsa è occupata in lettura dalla stessa transazione, cioè si ha un *lock escalation*

Se una transazione chiede un allora questo viene concesso se:

* La risorsa è completamente libera
* Se non è bloccata in scrittura
* Se è già presente una seconda transazione che ha un blocco in lettura (no in scrittura)

1. : Viene concesso un su per la transazione

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Viene concesso un su per la transazione

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Non viene concesso un su per la transazione perché la transizione ha già un su questa risorsa. Tutte le altre operazioni di vengono messe in attesa:

**attende**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Non viene concesso un su per la transazione perché la transizione ha già un su questa risorsa. Tutte le altre operazioni di vengono messe in attesa:

**attende**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Non viene concesso un su per la transazione perché la transizione ha già un su questa risorsa. Tutte le altre operazioni di vengono messe in attesa:

**attende** (che attende )

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Viene concesso un su per la transazione .

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Viene concesso un *condiviso* su per la transazione .

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Notiamo che è l’ultima operazione per la transazione viene rilasciata la memoria e la nuova tabella sarà:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. : Viene concesso un su per la transazione perché la transizione ha già un su questa risorsa e nessun’altra transazione sta leggendo , Viene fatta la cosiddetta *lock escalation*.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Notiamo che è l’ultima operazione per la transazione viene rilasciata la memoria e la nuova tabella sarà:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

A questo punto lo scheduler verifica se le transazioni che erano state messe in attesa possono essere svegliate ed eseguite. Le transazioni messe in attesa sono:

* **che attende per**
* **che attende per**
* **che attende per**

1. La risorsa è stata liberata da eventuali lock quindi può essere risvegliata

Al risveglio tutte le operazioni della possono essere eseguite. La nuova tabella sarà:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

è anche l’ultima operazione della transazione che terminando libererà la memoria. La nuova tabella dei lock è:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. La risorsa è stata liberata da eventuali lock quindi può essere risvegliata

Al risveglio tutte le operazioni della possono essere eseguite. La nuova tabella sarà:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

è anche l’ultima operazione della transazione che terminando libererà la memoria. La nuova tabella dei lock è:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

11. La risorsa è stata liberata da eventuali lock quindi può essere risvegliata

Al risveglio tutte le operazioni della possono essere eseguite. La nuova tabella sarà:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

è anche l’ultima operazione della transazione che terminando libererà la memoria. La nuova tabella dei lock è:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Lo schedule non presenta deadlock. D'altronde, *ogni schedule CSR non può avere deadlock*.