

Assignment Project Exam Help

Concurrency Control

<https://eduassistpro.github.io>

Imperial College London

Add WeChat edu_assist_pr

Transactions: ACID properties

ACID properties

Database management systems (DBMS) implements indivisible tasks called transactions

<https://eduassistpro.github.io>

```
BEGIN TRANSACTION
```

```
UPDATE branch
SET cash=cash-10000.00
WHERE sortcode=56
```

```
UPDATE branch
SET cash=cash+10000.00
WHERE sortcode=34
```

```
COMMIT TRANSACTION
```

Note that before the transaction, then it will be the same after the transaction.

SQL Conversion to Histories

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
4	'Goodge St'	8900.57
17	'Stand'	34005.00

Assignment Project Exam Help

BEGIN TRANSACTION T1

0.45,

<https://eduassistpro.github.io>


SET cash=cash+10000.00

WHERE sortcode=34

 w

COMMIT TRANSACTION T1

Add WeChat edu_assist_pr

history of transaction T_n

- 1 Begin transaction b_n (only given if necessary for discussion)
- 2 Various read operations on objects $r_n[o_j]$ and write operations $w_n[o_j]$
- 3 Either c_n for the commitment of the transaction, or a_n for the abort of the transaction

SQL Conversion to Histories

branch		
sortcode	bname	cash
56	'Wimbledon'	84340.45
64	'Goodge St'	18900.67
67	'Island'	34005.00

BEGIN TRANSACTION T2

0.67,

<https://eduassistpro.github.io>

SET cash=cash+2000.00

WHERE sortcode=67

w

COMMIT TRANSACTION T2

Add WeChat edu_assist_pr

history of transaction T_n

- 1 Begin transaction b_n (only given if necessary for discussion)
- 2 Various read operations on objects $r_n[o_j]$ and write operations $w_n[o_j]$
- 3 Either c_n for the commitment of the transaction, or a_n for the abort of the transaction

Concurrent Execution

Concurrent Execution of Transactions

- Interleaving of several transaction histories
- Order

<https://eduassistpro.github.io>

Add WeChat: edu_assist_pro

Some possible concurrent executions are

$H_1 = r_1$
 $H_2 = r_2[b_{34}], w_2[b_{34}], r_2[b_{67}], w_2[b_{67}], c_2$
 $H_x = r_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_2$
 $H_y = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], \quad 1 \quad 34 \quad 2 \quad 67 \quad 2 \quad 67 \quad 2, \quad c_1$
 $H_z = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, r_2[b_{67}], w_2[b_{67}], c_2$

Which concurrent executions should be allowed?

Assignment Project Exam Help

Concurrency control \rightarrow controlling interaction

serialisab

A concurr
some serial e

It as

<https://eduassistpro.github.io>

recoverability

No transaction commits depending on data that has been pro
transaction that has yet to commit

Add WeChat edu_assist_pr

Quiz 1: Serialisability and Recoverability (1)

$$H_x = r_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, w_2[b_{34}], r_2[b_{67}], w_2[b_{67}], c_2$$

Assignment Project Exam Help

A

Not Serialis

<https://eduassistpro.github.io>

B

Not Serialisable, Recoverable

Add WeChat edu_assist_pro

C

Serialisable, Not Recoverable

D

Serialisable, Recoverable

Quiz 2: Serialisability and Recoverability (2)

$$H_y = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], r_2[b_{67}], w_2[b_{67}], c_2, c_1$$

Assignment Project Exam Help

A

Not Serialis

<https://eduassistpro.github.io>

B

Not Serialisable, Recoverable

Add WeChat edu_assist_pro

C

Serialisable, Not Recoverable

D

Serialisable, Recoverable

Quiz 3: Serialisability and Recoverability (3)

$$H_z = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, r_2[b_{67}], w_2[b_{67}], c_2$$

Assignment Project Exam Help

A

Not Serialis

B

Not Serialisable, Recoverable

C

Serialisable, Not Recoverable

D

Serialisable, Recoverable

<https://eduassistpro.github.io>

Add WeChat edu_assist_pro

Anomaly 1: Lost update

BEGIN TRANSACTION T1

EXEC move_cash(56,34,10000.00)

COMMIT TRANSACTION T1

BEGIN TRANSACTION T2

EXEC move_cash(34,67,2000.00)

COMMIT TRANSACTION T2

 $r_1[b_{56}]$ $w_1[b_{67}]$ $w_2[b_{67}]$ $r_1[b_{56}]$, cash=94340.45, $w_1[b_{56}]$, cash=84340. $r_2[b_{34}]$, cash=8900.67, $w_1[b_{34}]$, cash=18900.67 *lost update*, c_1 , $w_2[b_{34}]$, cash=6900.42 $r_2[b_{67}]$, cash=34005.00, $w_2[b_{67}]$, cash=36005.25, c_2

- serialisable

+ recoverable

Anomaly 2: Inconsistent analysis

BEGIN TRANSACTION T1

EXEC move_cash(5034,10000.00)

COMMIT TRANSACTION T1

BEGIN TRANSACTION T4

SELECT SUM(cash) FROM branch

COMMIT TRANSACTION T4

 $r_1[b_{56}]$, $r_4[b_{67}]$, c_1  $r_1[b_{56}]$, cash=94340.45, $w_1[b_{56}]$, cash=84

.45,

 $r_4[b_{34}]$, cash=8900.67, $r_4[b_{67}]$, cash=34005.00, $r_1[b_{34}]$, cash=8900.67, $w_1[b_{34}]$, cash=18900.67, c_1 , c_4

- serialisable

+ recoverable

Anomaly 3: Dirty Reads

BEGIN TRANSACTION T1

EXEC move_cash(5034,10000.00)

COMMIT TRANSACTION T1

BEGIN TRANSACTION T2

EXEC move_cash(3467,2000.00)

COMMIT TRANSACTION T2

 $r_1[b_{56}]$ <https://eduassistpro.github.io> $w_2[b_{67}]$, c_2  $r_1[b_{56}]$, cash=94340.45, $w_1[b_{56}]$, cash=847, c_1 $w_2[b_{34}]$, cash=6900.42, $r_1[b_{34}]$, cash=6900.67, $w_1[b_{34}]$, cash=16900.67, c_1 $r_2[b_{67}]$, cash=34005.00, $w_2[b_{67}]$, cash=36005.25, a_2 

serialisable



recoverable

Quiz 4: Anomalies (1)

Assignment Project Exam Help

$$H_x = r_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, w_2[b_{34}], r_2[b_{67}], w_2[b_{67}], c_2$$

Which ano

<https://eduassistpro.github.io>

A

None

Lost

C

Inconsistent Analysis

D

Dirt

[Add WeChat edu_assist_pro](#)

Quiz 5: Anomalies (2)

Assignment Project Exam Help

$$H_z = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, r_2[b_{67}], w_2[b_{67}], c_2$$

Which ano

A

None

Lost

C

Inconsistent Analysis

D

Dirt

Worksheet: Anomalies

Assignment Project Exam Help

rental_charge

 $H_1 = r_1[\text{transfer_}$ $02]$ $H_2 = r_2[$ <https://eduassistpro.github.io>

total_charge

 $H_3 = r_3[d_{1000}], r_3[d_{1001}], r_3[d_{1002}]$

Add WeChat edu_assist_pr

Account Table

Assignment Project Exam Help

account	
107	'current' 'Poulovassilis, A.'
119	'deposit' 'Poulovassilis, A.'
125	'current' 'Bailey, J.'

<https://eduassistpro.github.io>

Add WeChat edu_assist_pr

Anomaly 4: Dirty Writes

```

BEGIN TRANSACTION T5
UPDATE account
SET rate=5.5
WHERE type='deposit'
COMMIT TRANSACTION T5

```

```

BEGIN TRANSACTION T6
UPDATE account
SET rate=6.0
WHERE type='deposit'
COMMIT TRANSACTION T6

```



<https://eduassistpro.github.io>

$w_5[a_{119}]$, rate=5.5, c_5 w

Add WeChat edu_assist_pr

$w_6[a_{101}]$, rate=6.0, $w_5[a_{101}]$, rate=5.5, $w_5[a_{119}]$, rate=5.5,
 $w_6[a_{119}]$, rate=6.0, c_5 , c_6

– serialisable

+ recoverable

Serialisable Transaction Execution

■ Solve anomalies $\rightarrow H = \text{serial execution}$
 ■ Only interested in the committed projection

 $H_c =$
 $r_1[b_{56}]$
 $r_3[m_{10}]$
 $w_1[b_{56}], r_4[b_{56}],$
 $r_3[m_{1003}], r_3[m_{1004}], r_3[m_{1005}],$
 $r_1[b_{34}], a_3, w_1[b_{34}], c_1, r_4[b_{34}],$
 $r_2[b_{67}], w_2[b_{67}], c_2, r_4[b_{67}], c_4$


<https://eduassistpro.github.io>

Add WeChat edu_assist_pro

Possible Serial Equivalents

Assignment Project Exam Help

$H_{cp} = r_1[b_{56}] , r_2[b_{34}] , w_2[b_{34}] , w_1[b_{56}] , r_4[b_{56}] , r_1[b_{34}] , w_1[b_{34}] , c_1 , r_4[b_{34}] , r_2[b_{67}] , w_2[b_{67}]$



<https://eduassistpro.github.io>



- how to determine that histories are equivalent?
- how to check this during execution?

Add WeChat edu_assist_pro

Conflicts: Potential For Problems

conflict

A **conflict** occurs when there is an interaction between two transactions

- $r_x[o]$ and $w_y[o]$ are in H where $x \neq y$
- or
- $w_x[o]$ and $w_y[o]$ are in H where $x \neq y$

conflicts

$H_x = r_2[b_{34}] \quad 1 \quad 56 \quad 1 \quad 56 \quad 1 \quad 34 \quad 1 \quad 34 \quad 1 \quad 2 \quad 34 \quad 2 \quad 67 \quad w_2[b_{67}], c_2$

$H_y = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], \quad 2, c_1$

$H_z = r_2[b_{34}], u_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], \quad 7, c_2$

Conflicts

- $w_2[b_{34}] \rightarrow r_1[b_{34}]$ T1 reads from T2 in H_y, H_z
- $w_1[b_{34}] \rightarrow w_2[b_{34}]$ T2 writes over T1 in H_x
- $r_2[b_{34}] \rightarrow w_1[b_{34}]$ T1 writes after T2 reads in H_x

Quiz 6: Conflicts

Assignment Project Exam Help

$I_w = \{ r_2[a_{100}], w_2[a_{100}], r_2[a_{107}], r_1[a_{119}], w_1[a_{119}], r_1[a_{107}], w_1[a_{107}], c_1, w_2[a_{107}], c_2 \}$

Which of th

A

$r_2[a_{107}] \rightarrow r_1[a_{107}]$

C

$r_1[a_{107}] \rightarrow w_2[a_{107}]$

B

$r_2[a$

$w_1[a_{107}] \rightarrow w_2[a_{107}]$

Conflict Equivalence and Conflict Serialisable

Conflict Equivalence

Two histories H_1 and H_2 are conflict equivalent if:

- 1 Contain the same set of operations
- 2 Order

Conflict Serialisable

a history H is **conflict serialisable (CSR)** if $C(H) \equiv$ a serial history

Failure to be conflict serialisable

$H_x = r_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], 1 \quad 2 \quad 34 \quad 2 \quad 67 \quad 2 \quad 67], c_2$

Contains conflicts $r_2[b_{34}] \rightarrow w_1[b_{34}]$ and $w_1[b_{34}] \rightarrow w_2[b_{34}]$ and so is not conflict equivalence to H_1, H_2 nor H_2, H_1 , and hence is not conflict serialisable.

Testing for Conflict Equivalence

Assignment Project Exam Help

<https://eduassistpro.github.io>

1 H_{cp} and H_2, H_1, H_4 contain the same set of op

2 conflicting pairs are

$w_2[b_{34}] \rightarrow r_1[b_{34}], w_2[b_{67}] \rightarrow r_4[b_{67}],$

$w_1[b_{34}] \rightarrow r_4[b_{34}], w_1[b_{56}] \rightarrow r_4[b_{56}]$

3 $H_2, H_1, H_4 \equiv_{CE} H_{cp} \rightarrow H_{cp} \in CSR$

Serialisation Graph

Serialisation Graph

A **serialisation graph** $SG(H)$ contains a node for each transaction in H , and an edge $T_i \rightarrow T_j$ if there is some object o for which a conflict $rw_i[o] \rightarrow rw_j[o]$ exists in H . If $SG(H)$ is acyclic, then H is conflict serialisable.

Demonst

Given H_{cp}

c_1 , $r_4[b_{34}]$ 2 67 2 67 2 4 67 4

Conflicts are $w_2[b_{34}] \rightarrow r_1[b_{34}]$, $w_2[b_{67}] \rightarrow r_4[b_{34}]$, $w_1[b_{56}] \rightarrow r_4[b_{56}]$

Then serialisation graph is



$SG(H_{cp})$ is acyclic, therefore H_{cp} is CSR

Worksheet: Serialisability

Assignment Project Exam Help

$$H_1 = r_1[o_1], w_1[o_1], w_1[o_2], w_1[o_3], c_1$$

$$H_2 = r_2$$

$$H_3 = r_3$$

<https://eduassistpro.github.io>

$$H = r_1[o_1], w_1[o_1], r_2[r_2], w_2[o_2], w_2[o_1], w_3[o_1], w_3[o_2], c_3, w_1[o_3], c_1$$

Add WeChat edu_assist_pro

Recoverability

- Serialisability necessary for isolation and consistency of committed transactions
- Recoverability necessary for isolation and consistency when there are also aborted transactions

Recoverability

A **recoverable** transaction history is one in which no transaction reads data written by an aborted transaction.

Execution avoiding cascading aborts

A history which **avoids cascading aborts** (A non-committed transaction does not read data written by a committed transaction.)

Strict execution

A **strict (ST)** history does not read from a non-committed transaction nor write over a non-committed transaction

$$ST \subset ACA \subset RC$$

Non-recoverable executions

BEGIN TRANSACTION T1

UPDATE branch

SET cash=cash-10000.00

WHERE surr_code=56

UPDATE branch

S

W

COMMIT

Assignment Project Exam Help

<https://eduassistpro.github.io> H_1 $=$ $r_1[b_{56}], a_1[b_{56}], a_1$ H_4 $=$ $r_4[b_{56}], a_4$  $H_c = r_1[b_{56}], \text{cash}=94340.45, w_1[b_{56}], \text{cash}=84340.45, r_4[b_{56}], \text{cash}=84340.45,$ $r_4[b_{34}], \text{cash}=8900.67, r_4[b_{67}], \text{cash}=34005.00, c_4, a_1$ $H_c \notin RC$

Cascading Aborts

BEGIN TRANSACTION T1

UPDATE branch

SET cash=cash-10000.00

WHERE surr_code=56

UPDATE branch

S

W

COMMIT

<https://eduassistpro.github.io>



$H_1 = r_1[b_{56}], w_1[b_{56}], a_1$

$H_2 = r_2[a_1], w_2[a_1]$



$H_c = r_1[b_{56}], \text{cash}=94340.45, w_1[b_{56}], \text{cash}=84340.45, r_4[b_{56}], \text{cash}=84340.45, r_4[b_{34}], \text{cash}=8900.67, r_4[b_{67}], \text{cash}=34005.00, a_1, a_4$

$H_c \in RC$
 $H_c \notin ACA$

Strict Execution

```

BEGIN TRANSACTION T5
UPDATE account
SET rate=5.5
WHERE type='deposit'
COMMIT TRANSACTION T5

```

```

BEGIN TRANSACTION T6
UPDATE account
SET rate=6.0
WHERE type='deposit'
COMMIT TRANSACTION T6

```

$H_5 = w_5[a_{101}], \text{rate}=5.5,$
 $H_6 = w_6[a_{101}], \text{rate}=6.0,$

$w_5[a_{119}], \text{rate}=5.5, a_5$
 $w_6[a_{119}]$

$H_c = w_6[a_{101}], \text{rate}=6.0, w_5[a_{101}], \text{rate}=5.5,$
 $w_5[a_{119}], \text{rate}=5.5, w_6[a_{119}], \text{rate}=6.0, a_5, c_6$

$H_c \in ACA$
 $H_c \notin ST$

Quiz 7: Recoverability

Assignment Project Exam Help

$$H_z = r_2[b_{34}], w_2[b_{34}], r_1[b_{56}], w_1[b_{56}], r_1[b_{34}], w_1[b_{34}], c_1, r_2[b_{67}], w_2[b_{67}], c_2$$

Which des

A

Non-recoverable

C

Avoids Cascading Aborts

Reco

D

Stric

Worksheet: Recoverability

Assignment Project Exam Help

$$H_w = r_2[o_1], r_2[o_2], w_2[o_2], r_1[o_2], w_2[o_1], r_2[o_3], c_2, c_1$$

$$H_x = r_2[$$

<https://eduassistpro.github.io>

$$H_y = r_2[\begin{matrix} 1 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 3 & 2 \end{matrix}]$$

$$H_z = r_2[o_1], w_1[o_1], r_2[o_1], w_2[o_2], r_2[o_3], c_1$$

Add WeChat edu_assist_pro

Maintaining Serialisability and Recoverability

■ two-phase locking (2PL)

- conflict based
- uses **locks** to prevent problems
- c

■ time

- a
-
- may only read or write objects with earlier timestamp
- abort when object has new timestamp
- common technique

■ optimistic concurrency control

- do nothing until commit
- at commit, inspect history for problems
- good if few conflicts

Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu_assist_pro

The 2PL Protocol

Assignment Project Exam Help

1 read locks $rl[o], \dots, r[o], \dots, ru[o]$

2 write l

3 Two p

i gr

ii shrinking phase

4 refuse $rl_i[o]$ if $wl_j[o]$ already held

refuse $wl_i[o]$ if $rl_j[o]$ or $wl_j[o]$ already held

5 $rl_i[o]$ or $wl_i[o]$ refused \rightarrow delay T_i

no.
locks
in H_i ▲



Quiz 8: Two Phase Locking (2PL)

Which history is not valid in 2PL?

Assignment Project Exam Help

A
 $rl_1[a_{107}]$,

<https://eduassistpro.github.io>

B
 $wl_1[a_{107}]$, $wl_1[a_{100}]$, $r_1[a_{107}]$, $w_1[a_{107}]$, $r_1[a_{100}]$, $w_1[a_{100}]$, $wu_1[a_{100}]$, $wu_1[a_{107}]$

Add WeChat edu_assist_pro

C
 $wl_1[a_{107}]$, $r_1[a_{107}]$, $w_1[a_{107}]$, $wu_1[a_{107}]$, $wl_1[100]$, $r_1[100]$, $wl_1[100]$, $wu_1[a_{100}]$

D
 $wl_1[a_{107}]$, $r_1[a_{107}]$, $w_1[a_{107}]$, $wl_1[a_{100}]$, $r_1[a_{100}]$, $wu_1[a_{107}]$, $w_1[a_{100}]$, $wu_1[a_{100}]$

Anomaly 1: Lost update

BEGIN TRANSACTION T1

EXEC move_cash(56,34,10000.00)

COMMIT TRANSACTION T1

BEGIN TRANSACTION T2

EXEC move_cash(34,67,2000.00)

COMMIT TRANSACTION T2

 $r_1[b_{56}]$  $w_1[b_{67}]$ $w_2[b_{67}]$  $r_1[b_{56}]$, cash=94340.45, $w_1[b_{56}]$, cash=84340. $r_2[b_{34}]$, cash=8900.67, $w_1[b_{34}]$, cash=18900.67 *lost update*, c_1 , $w_2[b_{34}]$, cash=6900.42 $r_2[b_{67}]$, cash=34005.00, $w_2[b_{67}]$, cash=36005.25, c_2

- serialisable

+ recoverable

Lost Update Anomaly with 2PL

BEGIN TRANSACTION T1
 EXEC move_cash(56 34, 10000 00)
 COMMIT TRANSACTION T1

BEGIN TRANSACTION T2
 EXEC move_cash(34 67 2000 00)
 COMMIT TRANSACTION T2



b_1 , $wl_1[b_{34}]$, $r_1[b_{34}]$, $w_1[b_{34}]$, c_1 , $wu_1[b_{56}]$, $wu_1[b_{34}]$

$[b_{34}]$, $wl_2[b_{67}]$, $r_2[b_{67}]$, $w_2[b_{67}]$, c_2 , $wu_2[b_{34}]$, $wu_2[b_{67}]$

Add WeChat edu_assist_pr

b_1 , $wl_1[b_{56}]$, $r_1[b_{56}]$, $w_1[b_{56}]$, $wl_1[b_{34}]$, $r_1[b_{34}]$, b_2 , $wl_2[b_{34}]$, $r_2[b_{34}]$, $w_1[b_{34}]$, c_1 , $wu_1[b_{56}]$, $wu_1[b_{34}]$, $w_2[b_{34}]$, $wl_2[b_{67}]$, $r_2[b_{67}]$, $w_2[b_{67}]$, c_2 , $wu_2[b_{34}]$, $wu_2[b_{67}]$

Lost Update history not permitted by 2PL, since $wl_2[b_{34}]$ not granted

Lost Update Anomaly with 2PL

BEGIN TRANSACTION T1
 EXEC Move_cash(5634,10000,00)
 COMMIT TRANSACTION T1

BEGIN TRANSACTION T2
 EXEC Move_cash(3467,2000,00)
 COMMIT TRANSACTION T2



$b_1, wl_1[b_{34}], r_1[b_{34}], w_1[b_{34}], c_1, wu_1[b_{56}], wu_1[b_{34}]$

$r_2[b_{67}], w_2[b_{67}], c_2, wu_2[b_{34}], wu_2[b_{67}]$

Add WeChat edu_assist_pr

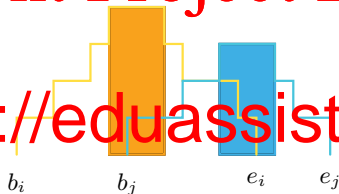
$b_1, wl_1[b_{56}], r_1[b_{56}], w_1[b_{56}], wl_1[b_{34}], r_1[b_{34}], b_2, w_1[b_{34}], c_1, wu_1[b_{56}], wu_1[b_{34}], wl_2[b_{34}], r_2[b_{34}], w_2[b_{34}], wl_2[b_{67}], r_2[b_{67}], w_2[b_{67}], c_2, wu_2[b_{34}], wu_2[b_{67}]$

2PL causes T2 to be delayed

Why does 2PL Work?

Assignment Project Exam Help

<https://eduassistpro.github.io>



- two-phase rule — maximum lock period
- can re-time history so all operations take place during ma
- CSR since *all* conflicts prevented during maximum lock period

When to lock: Aggressive Scheduler

Assignment Project Exam Help

 $rl_1[b_{34}] \quad wl_1[b_{34}] \quad wl_1[b_{34}]$
<https://eduassistpro.github.io>

- delay taking locks as long as possible
- maximises concurrency
- might suffer delays later on

Add WeChat edu_assist_pr

When to lock: Conservative Scheduler

Assignment Project Exam Help

 $wl_2[b_{34}] \quad wl_2[b_{34}]$

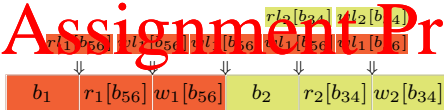
<https://eduassistpro.github.io>

- take locks as soon as possible
- removes risks of delays later on
- might refuse to start

Add WeChat edu_assist_pr

Deadlock Detection: WFG with No Cycle = No Deadlock

Assignment Project Exam Help



<https://eduassistpro.github.io>

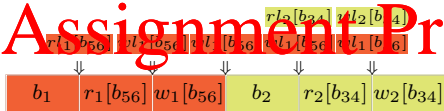


Add WeChat edu_assist_pr

- waits-for graph (WFG)
- describes which transactions waits for others

Deadlock Detection: WFG with No Cycle = No Deadlock

Assignment Project Exam Help



<https://eduassistpro.github.io>



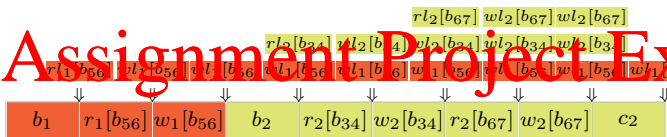
Add WeChat edu_assist_pr

H_1 attempts $r_1[b_{34}]$, but is refused since H_2 has a write-lock, and so is put on WFG

- **waits-for graph (WFG)**
- describes which transactions waits for others

Deadlock Detection: WFG with No Cycle = No Deadlock

Assignment Project Exam Help



<https://eduassistpro.github.io>

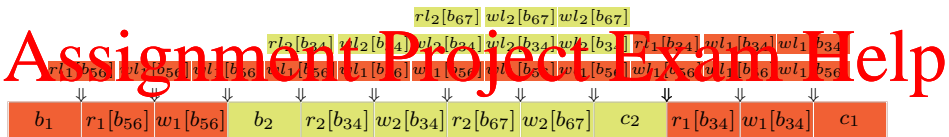


Add WeChat edu_assist_pr

H_2 can proceed to complete its execution, after which it will have released all its locks

- **waits-for graph (WFG)**
- describes which transactions waits for others

Deadlock Detection: WFG with No Cycle = No Deadlock



<https://eduassistpro.github.io>

 H_1

Add WeChat edu_assist_pr

- **waits-for graph (WFG)**
- describes which transactions waits for others

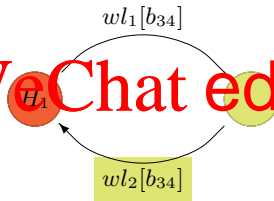
Deadlock Detection: WFG with Cycle = Deadlock

Assignment Project Exam Help

$rl_1[b_{34}]$ $rl_1[b_{34}]$ $rl_1[b_{34}]$ $rl_1[b_{34}]$
 $rl_2[b_{34}]$ $rl_2[b_{34}]$
 $rl_1[b_{56}]$ $wl_1[b_{56}]$ $wl_1[b_{56}]$ $wl_1[b_{56}]$ $wl_1[b_{56}]$ $wl_1[b_{56}]$

<https://eduassistpro.github.io>

Add WhatsApp edu_assist_pro



Cycle in WFG means DB in a deadlock state, must abort either H_1 or H_2

Quiz 9: Resolving Deadlocks in 2PL

$$H_1 = r_1[p_1], r_1[p_2], r_1[p_3], r_1[p_4], r_1[p_5], r_1[p_6]$$

$$H_2 = r_2[p_5], w_2[p_5], r_2[p_1], w_2[p_1]$$

$$H_3 = r_3[p_6], w_3[p_6], r_3[p_2], w_3[p_2]$$

$$H_4 = r_4[$$

Suppose the

$$H_d = r_1[$$

$$r_3[p_6], w_3[p_6], r_3[p_2], r_4[p_4]$$

Which transaction should be aborted?

A

 H_1

B

 H_2

C

 H_3

D

 H_4

Worksheet: Deadlocks

Assignment Project Exam Help

$$H_1 = w$$

$$H_2 = r_2$$

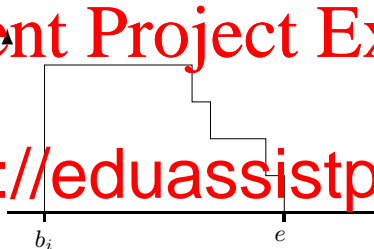
$$H_3 = r_3[o_4], w_3[o_4], r_3[o_3], w_3[o_3]$$

Add WeChat edu_assist_pr

Conservative Locking

Assignment Project Exam Help

<https://eduassistpro.github.io>



Conservative Locking

- prevents deadlock
- when to release locks problem
- not recoverable

Strict Locking



<https://eduassistpro.github.io>

Strict Lock

- prevents write locks being released before transaction
- recoverable (with cascading aborts) but allows deadlocks

Add WeChat edu_assist_pro

Strong Strict Locking

- no locks released before end \rightarrow recoverable
- allows deadlocks
- no problem determining when to release locks
- suitable for distributed transactions (using atomic commit)

Transaction Isolation Levels

Assignment Project Exam Help

- Do we always need ACID properties?

BEGIN TRANSACTION T3

co

<https://eduassistpro.github.io>

- Some transactions only need 'approximate' results

e.g. Management overview

e.g. Estimates

- May execute these transactions at a 'lower' level of conc

SQL allows you to vary the level of concurrency control

Add WeChat edu_assist_pr