

# Database Technology

## Topic 10: Concurrency Control

Olaf Hartig

[olaf.hartig@liu.se](mailto:olaf.hartig@liu.se)

# Concurrency Control

## Basic Concepts



- Remember that *schedules* may contain operations from multiple transactions
- How many operations from each transaction can be in one such schedule?
  1. only one
  2. at most 64 if we assume a 64-bit computer architecture
  3. all of them
  4. all of them but they must all come directly after one another (i.e., without operations from other transactions in between)



# QUIZ

- Which of the following types of schedules is guaranteed to produce a state of the database that is correct?  
(assuming all transactions in such schedules have the consistency preservation property)
  1. serial schedules
  2. schedules with operations from only one transaction
  3. all serializable schedules
  4. all of the above



# Concurrency Control

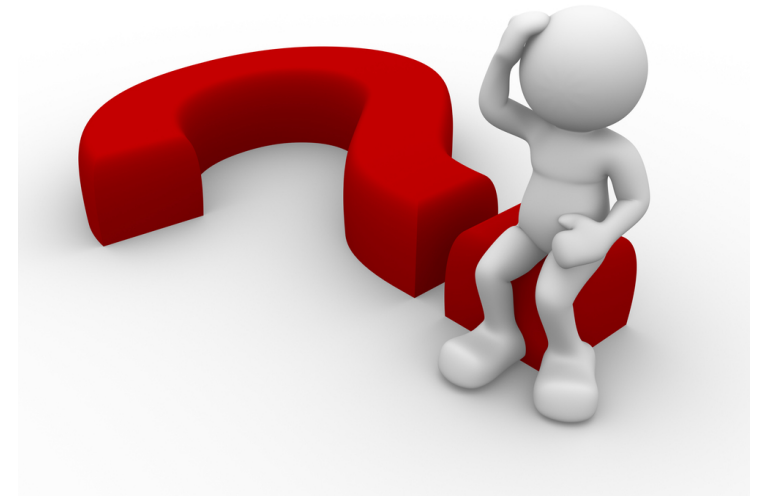
## Conflict Equivalence

# QUIZ

- Consider the following schedule

$S_1$ :  $b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- How many pairs of conflicting operations are in this schedule?
  1. *only one*
  2. *three*
  3. *four*
  4. *six*

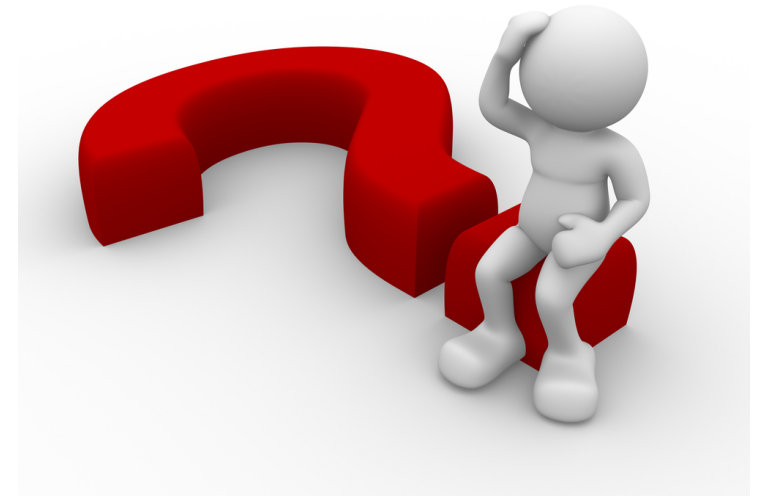


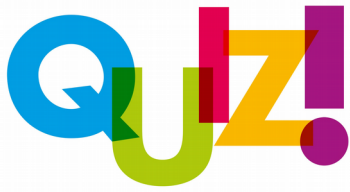
# QUIZ

- Consider the following schedule

$S_1: b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- How many pairs of conflicting operations are in this schedule?
  - ~~only one~~
  - ~~three~~
  - four
  - ~~six~~





- Consider the following two schedules

$S_1$ :  $b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

$S_2$ :  $b_2, r_2(Y), b_1, r_1(X), w_1(X), b_3, w_2(Y), e_2, r_3(X), r_1(Y), e_3, w_1(Y), e_1$

- Are these two schedules conflict equivalent?
  - yes
  - no
  - that's a trick question because only operations can be conflict equivalent
  - sorry, I don't know





# Concurrency Control

## Serializability

# QUIZ

- Consider the following schedule

$S_1$ :  $b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- Is this schedule serializable?
  1. yes
  2. no

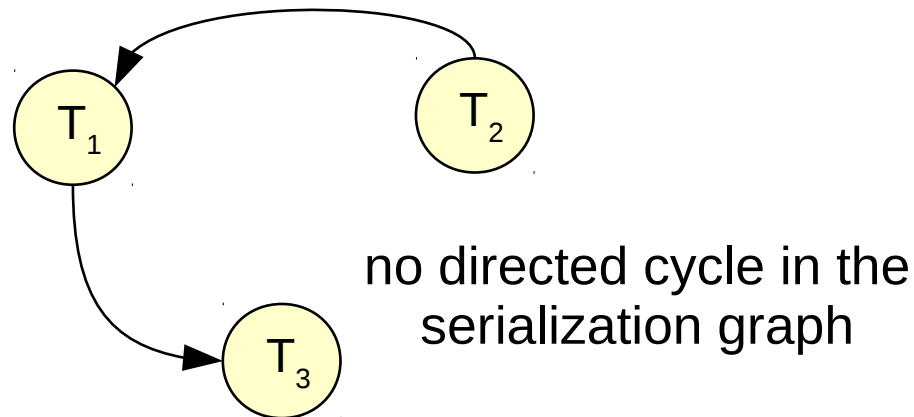


# QUIZ

- Consider the following schedule

$S_1: b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- Is this schedule serializable?
  - yes
  - ~~no~~

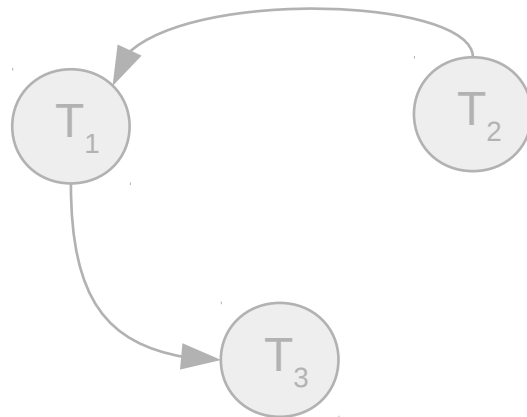


# EXERCISE

- Consider the following schedule

$S_1$ :  $b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- Write down a serial schedule that is conflict equivalent with  $S_1$



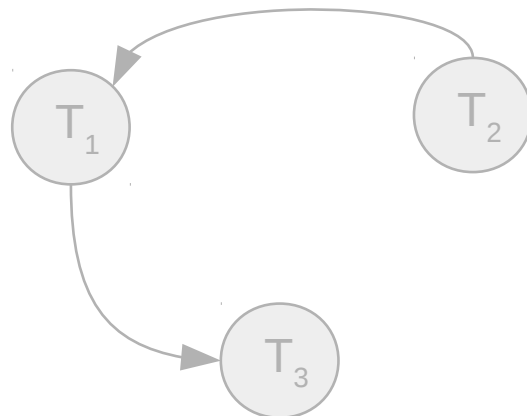
# EXERCISE

- Consider the following schedule

$S_1$ :  $b_1, r_1(X), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_1(Y), r_3(X), e_3, w_1(Y), e_1$

- Write down a serial schedule that is conflict equivalent with  $S_1$

$S_3$ :  $b_2, r_2(Y), w_2(Y), e_2, b_1, r_1(X), w_1(X), r_1(Y), w_1(Y), e_1, b_3, r_3(X), e_3$



# EXERCISE

- Consider the following schedule (which is a different one now!)  
 $S_4: b_1, r_1(Y), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_3(X), e_3, w_1(Y), e_1$
- Is this schedule serializable?
  1. yes
  2. no

# EXERCISE

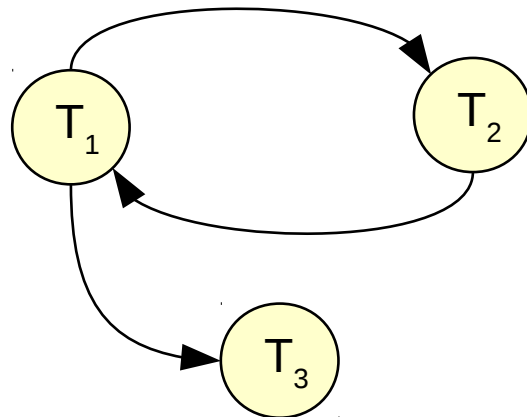
- Consider the following schedule

$S_4: b_1, r_1(Y), b_2, r_2(Y), w_1(X), b_3, w_2(Y), e_2, r_3(X), e_3, w_1(Y), e_1$

- Is this schedule serializable?

1. ~~yes~~

2. no, because its serialization graph contains a cycle



# Concurrency Control

## Locking



# Quiz

- Consider the following situation:
  - transaction TA1 holds exclusive lock on data item D1
  - transaction TA2 holds shared lock on data item D2
  - transaction TA2 is currently waiting for shared lock on D1
- Now, TA1 wants to read data item D2
- Which lock does TA1 need, and will it get this lock immediately?
  1. shared lock; can get it immediately
  2. shared lock; will have to wait for it
  3. exclusive lock; can get it immediately
  4. exclusive lock; will have to wait for it

# Quiz

- Consider the following transaction (with lock operations):  
exclLock(X), r<sub>1</sub>(X), w<sub>1</sub>(X), unlock(X), exclLock(Y), r<sub>1</sub>(Y), w<sub>1</sub>(Y), unlock(Y)
- Is this transaction valid in terms of the locks that it needs to hold for the operations that it aims to do?
  1. yes
  2. no, because it needs to obtain shared locks as well
  3. no, because it needs to obtain all locks in the beginning
  4. no, because of both of the aforementioned reasons

# Quiz

- Consider the following transaction (with lock operations):  
exclLock(X), r<sub>1</sub>(X), w<sub>1</sub>(X), unlock(X), exclLock(Y), r<sub>1</sub>(Y), w<sub>1</sub>(Y), unlock(Y)
- Does this TA follow the two-phase locking (2PL) protocol?
  1. yes
  2. no

# Quiz

- Consider the following transaction (with lock operations):  
exclLock(X), r<sub>1</sub>(X), w<sub>1</sub>(X), **unlock(X)**, **exclLock(Y)**, r<sub>1</sub>(Y), w<sub>1</sub>(Y), unlock(Y)
- Does this TA follow the two-phase locking (2PL) protocol?
  1. ~~yes~~
  2. no

# Quiz

- Consider the following transaction (with lock operations):  
exclLock(X),  $r_1(X)$ ,  $w_1(X)$ , unlock(X), exclLock(Y),  $r_1(Y)$ ,  $w_1(Y)$ , unlock(Y)
- Does this TA follow the two-phase locking (2PL) protocol?
  1. ~~yes~~
  2. no
- Modify this TA so that it follows 2PL



# Exercise

- Consider the following transaction (with lock operations):

exclLock(X),  $r_1(X)$ ,  $w_1(X)$ , unlock(X), exclLock(Y),  $r_1(Y)$ ,  $w_1(Y)$ , unlock(Y)

- Modify this TA so that it follows 2PL

- Option 1:

exclLock(X),  $r_1(X)$ ,  $w_1(X)$ , exclLock(Y), unlock(X),  $r_1(Y)$ ,  $w_1(Y)$ , unlock(Y)

- Option 2:

exclLock(X), exclLock(Y),  $r_1(X)$ ,  $w_1(X)$ , unlock(X),  $r_1(Y)$ ,  $w_1(Y)$ , unlock(Y)

- Option 3:

exclLock(X),  $r_1(X)$ , exclLock(Y),  $w_1(X)$ ,  $r_1(Y)$ , unlock(X),  $w_1(Y)$ , unlock(Y)

- etc.



[www.liu.se](http://www.liu.se)