أسماء جمال عبد الحليم مبروك

Alexandria University
Faculty of Engineering
Comp. & Comm. Engineering
CC471: Database Systems
Spring 2021

جامعة الاسكندرية
كلية الهندسة
برنامج هندسة الحاسب والاتصالات
مادة نظم قواعد البيانات
ربيع ٢٠٢١

Sheet8
Transaction Processing & Concurrency Control

1) Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.
   a) $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X)$
   b) $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X)$
2) How many serial schedules exist for the three transactions in the Figure below? What are they? What is the total number of possible schedules?

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

3) Determine which of the following schedules are recoverable, which are cascadeless, and which are strict.
   a) S_1: $r_1(X); w_1(X); r_1(Y); w_1(Y); C_1; r_2(X); w_2(X); C_2;$
   b) S_3: $r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X); C_1; C_2;$
   c) S_4: $r_1(X); w_1(X); r_1(Y); w_1(Y); r_2(X); w_2(X); C_2; C_1;$
   d) S_7: $r_1(X); w_1(X); r_1(Y); r_2(X); w_1(Y); w_2(X); C_2; C_1;$
   e) S_8: $r_1(X); w_1(X); r_1(Y); r_2(X); w_2(X); w_1(Y); C_1; C_2;$
   f) S_9: $r_1(X); w_1(X); r_1(Y); r_2(X); w_2(X); w_1(Y); C_2; C_1;$
   g) S_{21}: $r_1(X); r_2(X); w_1(X); r_1(Y); w_1(Y); C_1; w_2(X); C_2;$
   h) S_{26}: $r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); C_2; w_1(Y); C_1;$
   i) S_{27} : $r_1(X); r_2(X); w_1(X); w_2(X); r_1(Y); w_1(Y); C_1; C_2;$
   j) S_{36} : $r_2(X); r_1(X); w_1(X); r_1(Y); w_1(Y); C_1; w_2(X); C_2;$
4) Prove that strict two-phase locking guarantees strict schedules.
5) No more questions are provided for concurrency control. It means that most exam questions on this part are review questions (اكتب مذكرات جغرافية عن - علل).

How to submit the homework assignments?

- Solve the sheet individually without looking up the solution on the Internet. The sheet is to practice; it is a learning tool not an exam.
- Assignments are to be **handwritten**.
- Papers are to be scanned (I like camscanner app). Put all images in a pdf file (camscanner does that for you)

Database

Sheet (8): Transaction Processing & Concurrency Control

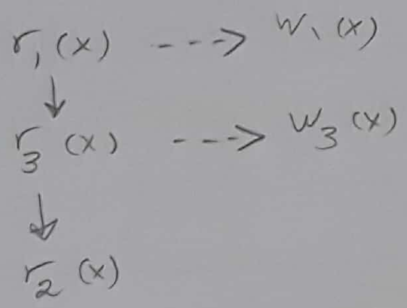
Name: Asmaa Gamal Abdel-Halem Mabrouk Nagy.

Course: Audiance course "DBMS"

Department: Communications & Electronics

$\boxed{1}$ using "Precedence graph" method, if the graph is acyclic $\xrightarrow{\because}$ the schedule is conflict Serializable.

for ⓐ:

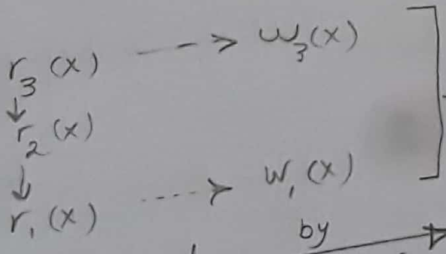


$\because$ acyclic $\longrightarrow$ $\therefore$ Conflict serializable schedule.

$\therefore$ equivalent serial schedule $\xrightarrow{\text{by}}$ topological sort

$\therefore$ $r_1(x)$; $w_1(x)$; $r_3(x)$; $w_3(x)$; $r_2(x)$

for ⓑ:



$\because$ acyclic $\longrightarrow$ $\therefore$ conflict

$\therefore$ equivalent $\xrightarrow[\text{topological Sort}]{\text{by}}$ $r_3(x)$; $r_2(x)$; $r_1(x)$; $w_3(x)$; $w_1(x)$

$\boxed{2}$ • # serial scheduales = # all possible transaction orderings * # all possible orderings for each T_1, T_2, T_3

• for each scheduale: # possible scheduales = $\begin{cases} 4! = 24 & \text{for } T_1 \\ 5! = 120 & \text{" } T_2 \\ 4! = 24 & \text{" } T_3 \end{cases}$

$\therefore$ # possible ordering of transactions = 6 = nPr = $\underset{\text{tables}}{3P3}$ = 6 (تباديل وتوافيق)

$\therefore$ # serial scheduals = 6 * 24 * 120 * 24 = 414,720 possible schedules

3 * Recoverable schedule: all transaction that T reads have already Committed.

* Cascadeless " : " " " T writes " " " !

* Strict " : " " " T read/write " " "

		Recoverable	Cascadeless	Strict
a	S_1	×	×	×
b	S_3	✓	×	×
c	S_4	×	×	×
d	S_7	✓	✓	✓
e	S_8	✓	✓	✓
f	S_9	×	×	×
g	S_{21}	×	×	×
h	S_{26}	×	×	×
i	S_{27}	✓	×	×
J	S_{36}	×	×	×

4 * to Prove : ∴ we need to show that strictness Property applies. and Stict (2PL) applies in → Growing Phase. shrinking phase.

Proof:

*assume: • atransaction T reads/writes by uncommitted transaction.

• T_1 modify before Committing,

T_2 read/write data item that T_1 modified

* According to Strict 2PL:

• T_1 → exclusive lock X (during growing Phase)

• T_2 → only Lock X (" Shrinking ")

* but : ∵ T_2 accesses X (modified by T_1)

∴ T_2 → lock X before T_1 release it

∴ Violate the strict 2pL Protocol

∴ false assumption

∴ Strict 2PL Protocol guarntees strict scheduales.

#