**COP 4722 – Survey of Database Systems**

**Assignment 2**

Instructor: Nagarajan Prabakar

Student: Carlos Barrios

Florida International University

Spring 2019

========================================================  
  
 Name: Carlos Barrios  
  
 Panther-ID: 6048821  
  
 Course: COP 4722  
  
 Assignment#: 2  
  
 Due: Wed, Feb 20, 2019  
  
 I hereby certify that this work is my own and none of  
 it is the work of any other person.  
  
 Signature: CBarrios  
=========================================================

The purpose of this assignment is to familiarize with **serializability**. Show all intermediate steps for each answer.

1. Consider the three transactions Ta, Tb, and Tc, and the schedules S1, S2, S3 and S4 given below. Which of the schedules is (conflict) serializable? The subscript for each database operation in a schedule denotes the transaction number for that operation. For each schedule, show all conflicts, draw the precedence graph, determine and write down if it is serializable or not, and the equivalent serial schedules if exist.

Ta: ra(x); wa(x);

Tb: rb(x);

Tc: rc(x); wc(x);

**S1: ra(x); rc(x); wa(x); rb(x); wc(x);**

**S2: ra(x); rc(x); wc(x); wa(x); rb(x);**

**S3: rc(x); rb(x); wc(x); ra(x); wa(x);**

**S4: rc(x); rb(x); ra(x); wc(x); wa(x);**

1. Consider the three transactions T1, T2, and T3, and the schedules S5 and S6 given below. Show all conflicts and draw the serializability (precedence) graphs for S5 and S6, and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).

T1: r1(p); r1(r); w1(p);

T2: r2(r); r2(q); w2(r); w2(q);

T3: r3(p); r3(q); w3(q);

**S5: r1(p); r2(r); r1(r); r3(p); r3(q); w1(p); w3(q); r2(q); w2(r); w2(q);**

**S6: r1(p); r2(r); r3(p); r1(r); r2(q); r3(q); w1(p); w2(r); w3(q); w2(q);**

1. **ANSWERS**

Legend

WAR (Write after Read)

RAW (Read after Write)

WAW (Write after Write)

**S1: ra(x); rc(x); wa(x); rb(x); wc(x);**

|  |  |  |
| --- | --- | --- |
| Ta | Tb | Tc |
| 1 read(X); |  |  |
| 2 |  | read(X); |
| 3 write(X); |  |  |
| 4 | read(X); |  |
| 5 |  | write(X); |

Conflict Precedence Graph:

**Since the precedence graph has the following two cycles:**

**Ta 🡪 Tc 🡪 Ta**

**Ta 🡪 Tb 🡪 Tc 🡪 Ta**

**This schedule is not conflict serializable.**

**So, there is no\_conflict\_equivalent\_serial\_schedule for this schedule.**

**S2: ra(x); rc(x); wc(x); wa(x); rb(x);**

|  |  |  |
| --- | --- | --- |
| Ta | Tb | Tc |
| 1 read(X); |  |  |
| 2 |  | read(X); |
| 3 |  | write(X); |
| 4 write(X); |  |  |
| 5 | read(X); |  |

Conflict Precedence Graph:

**Since the precedence graph has the following cycle:**

**Ta 🡪 Tc 🡪 Ta**

**This schedule is not conflict serializable.**

**So, there is no\_conflict\_equivalent\_serial\_schedule for this schedule.**

**S3: rc(x); rb(x); wc(x); ra(x); wa(x);**

|  |  |  |
| --- | --- | --- |
| Ta | Tb | Tc |
| 1 |  | read(X); |
| 2 | read(X); |  |
| 3 |  | write(X); |
| 4 read(X); |  |  |
| 5 write(X); |  |  |

Conflict Precedence Graph:

**Since the precedence graph has no cycle:**

**This schedule is conflict serializable.**

**So, the conflict\_equivalent\_serial\_schedule for this schedule is:**

**Tb 🡪 Tc 🡪 Ta**

|  |  |  |
| --- | --- | --- |
| Ta | Tb | Tc |
| 1 | read(X); |  |
| 2 |  | read(X); |
| 3 |  | write(X); |
| 4 read(X); |  |  |
| 5 write(X); |  |  |

**S4: rc(x); rb(x); ra(x); wc(x); wa(x);**

|  |  |  |
| --- | --- | --- |
| Ta | Tb | Tc |
| 1 |  | read(X); |
| 2 | read(X); |  |
| 3 read(X); |  |  |
| 4 |  | write(X); |
| 5 write(X); |  |  |

Conflict Precedence Graph:

**Since the precedence graph has the following cycle:**

**Ta 🡪 Tc 🡪 Ta**

**This schedule is not conflict serializable.**

**So, there is no\_conflict\_equivalent\_serial\_schedule for this schedule.**

1. **ANSWERS**

Legend

WAR (Write after Read)

RAW (Read after Write)

WAW (Write after Write)

**S5: r1(p); r2(r); r1(r); r3(p); r3(q); w1(p); w3(q); r2(q); w2(r); w2(q);**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| 1 read(P); |  |  |
| 2 | read(R); |  |
| 3 read(R); |  |  |
| 4 |  | read(P); |
| 5 |  | read(Q); |
| 6 write(P); |  |  |
| 7 |  | write(Q); |
| 8 | read(Q); |  |
| 9 | write(R); |  |
| 10 | write(Q); |  |

Conflict Precedence Graph:

**Since the precedence graph has no cycle:**

**This schedule is conflict serializable.**

**So, the conflict\_equivalent\_serial\_schedule for this schedule is:**

**T3 🡪 T1 🡪 T2**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| 1 |  | read(P); |
| 2 |  | read(Q); |
| 3 |  | write(Q); |
| 4 read(P); |  |  |
| 5 read(R); |  |  |
| 6 write(P); |  |  |
| 7 | read(R); |  |
| 8 | read(Q); |  |
| 9 | write(R); |  |
| 10 | write(Q); |  |

**S6: r1(p); r2(r); r3(p); r1(r); r2(q); r3(q); w1(p); w2(r); w3(q); w2(q);**

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| 1 read(P); |  |  |
| 2 | read(R); |  |
| 3 |  | read(P); |
| 4 read(R); |  |  |
| 5 | read(Q); |  |
| 6 |  | read(Q); |
| 7 write(P); |  |  |
| 8 | write(R); |  |
| 9 |  | write(Q); |
| 10 | write(Q); |  |

Conflict Precedence Graph:

**Since the precedence graph has the following two cycles:**

**T2 🡪 T3 🡪 T2**

**T2 🡪 T3 🡪 T1🡪 T2**

**This schedule is not conflict serializable.**

**So, there is no\_conflict\_equivalent\_serial\_schedule for this schedule.**