San Jose State University

CMPE 138: Database Systems

Homework 3

Please submit your solution to Canvas for this assignment, as a PDF document. Follow the homework policy document for instructions on how to submit this homework. References to figures, sections, tables and queries are from your textbook.

Problem 1 **(20 points)**

What update anomalies occur in the EMP_PROJ and EMP_DEPT relations of Figure 14.3 and 14.4?

Solution:

In EMP_PROJ, the partial dependencies {SSN}->{ENAME} and {PNUMBER}->{PNAME, PLOCATION} can cause anomalies. For example, if a PROJECT temporarily has no EMPLOYEEs working on it, its information (PNAME, PNUMBER, PLOCATION) will not be represented in the database when the last EMPLOYEE working on it is removed (deletion anomaly). A new PROJECT cannot be added unless at least one EMPLOYEE is assigned to work on it (insertion anomaly). Inserting a new tuple relating an existing EMPLOYEE to an existing PROJECT requires checking both partial dependencies; for example, if a different value is entered for PLOCATION than those values in other tuples with the same value for PNUMBER, we get an update anomaly. Similar comments apply to EMPLOYEE information. The reason is that EMP_PROJ represents the relationship between EMPLOYEEs and PROJECTs, and at the same time represents information concerning EMPLOYEE and PROJECT entities.

In EMP_DEPT, the transitive dependency {SSN}->{DNUMBER}->{DNAME, DMGRSSN} can cause anomalies. For example, if a DEPARTMENT temporarily has no EMPLOYEEs working for it, its information (DNAME, DNUMBER, DMGRSSN) will not be represented in the database when the last EMPLOYEE working on it is removed (deletion anomaly). A new DEPARTMENT cannot be added unless at least one EMPLOYEE is assigned to work on it (insertion anomaly). Inserting a new tuple relating a new EMPLOYEE to an existing DEPARTMENT requires checking the transitive dependencies; for example, if a different value is entered for DMGRSSN than those values in other tuples with the same value for DNUMBER, we get an update anomaly. The reason is that EMP_DEPT represents the relationship between EMPLOYEEs and DEPARTMENTs, and at the same time represents information concerning EMPLOYEE and DEPARTMENT entities.

Problem 2 **(25 points)**

Consider the following relations for an order-processing application database at ABC, Inc.

ORDER (O#, Odate, Cust#, Total_amount)

ORDER-ITEM (O#, I#, Qty_ordered, Total_price, Discount%)

Assume that each item has a different discount. The Total_price refers to one item, Odate is the date on which the order was placed, and the Total_amount is the amount of the order. If we apply a natural join on the relations Order-Item and Order in this database, what does the resulting relation schema look like? What will be its key? Show the FDs in this resulting relation. Is it in 2NF? Is it in 3NF? Why or why not? (State any assumptions you make.)

Solution:

Given relations

Order(O#, Odate, Cust#, Total_amt)

Order_Item(O#, I#, Qty_ordered, Total_price, Discount%),

the schema of Order * Order_Item looks like

T_1(O#,I#,Odate, Cust#, Total_amount, Qty_ordered, Total_price, Discount%)

and its key is O#,I#.

It has functional dependencies

O#I# . Qty_ordered

O#I# .Total_price

O#I# .Discount%

O# . Odate

O# .Cust#

O# .Total_amount

It is not in 2NF, as attributes Odate, Cut#, and Total_amount are only partially dependent on the primary key, O#I#

Nor is it in 3NF, as a 2NF is a requirement for 3NF.

Problem 3 **(25 points)**

List all possible schedules for transactions T 1 and T 2, and determine which are conflict serializable (correct) and which are not.

T 1	T 2
read_item(X);	read_item(X);
X := X - N	X := X + M;
write_item(X);	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

The transactions can be written as follows using shorthand notation:

T 1 : r 1 (X); w 1 (X); r 1 (Y); w 1 (Y);

T 2 : r 2 (X); w 2 (X);

An example of a possible schedule is: r 1 (X); r 2 (X); w 1 (X); r 1 (Y); w 1 (Y); w 2 (X);

This schedule is not serializable because w1(X) happens after r2(X).

Solution

In this case, m =2 and n1 = 4 and n2 = 2, so the number of possible schedules is:

(4+2)! / (4! * 2!) = 6*5*4*3*2*1/ 4*3*2*1*2*1 = 15.

Below are the 15 possible schedules, and the type of each schedule:

S 1 : r 1 (X); w 1 (X); r 1 (Y); w 1 (Y); r 2 (X); w 2 (X); serial (and hence also serializable)

S 2 : r 1 (X); w 1 (X); r 1 (Y); r 2 (X); w 1 (Y); w 2 (X); (conflict) serializable

S 3 : r 1 (X); w 1 (X); r 1 (Y); r 2 (X); w 2 (X); w 1 (Y); (conflict) serializable

S 4 : r 1 (X); w 1 (X); r 2 (X); r 1 (Y); w 1 (Y); w 2 (X); (conflict) serializable

S 5 : r 1 (X); w 1 (X); r 2 (X); r 1 (Y); w 2 (X); w 1 (Y); (conflict) serializable

S 6 : r 1 (X); w 1 (X); r 2 (X); w 2 (X); r 1 (Y); w 1 (Y); (conflict) serializable

S 7 : r 1 (X); r 2 (X); w 1 (X); r 1 (Y); w 1 (Y); w 2 (X); not (conflict) serializable

S 8 : r 1 (X); r 2 (X); w 1 (X); r 1 (Y); w 2 (X); w 1 (Y); not (conflict) serializable

S 9 : r 1 (X); r 2 (X); w 1 (X); w 2 (X); r 1 (Y); w 1 (Y); not (conflict) serializable

S 10 : r 1 (X); r 2 (X); w 2 (X); w 1 (X); r 1 (Y); w 1 (Y); not (conflict) serializable

S 11 : r 2 (X); r 1 (X); w 1 (X); r 1 (Y); w 1 (Y); w 2 (X); not (conflict) serializable

S 12 : r 2 (X); r 1 (X); w 1 (X); r 1 (Y); w 2 (X); w 1 (Y); not (conflict) serializable

S 13 : r 2 (X); r 1 (X); w 1 (X); w 2 (X); r 1 (Y); w 1 (Y); not (conflict) serializable

S 14 : r 2 (X); r 1 (X); w 2 (X); w 1 (X); r 1 (Y); w 1 (Y); not (conflict) serializable

S 15 : r 2 (X); w 2 (X); r 1 (X); w 1 (X); r 1 (Y); w 1 (Y); serial (and hence also serializable)

Problem 4 **(20 points)**

Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

(a) r1 (X); r3 (X); w1(X); r2(X); w3(X)

(b) r1 (X); r3 (X); w3(X); w1(X); r2(X)

(c) r3 (X); r2 (X); w3(X); r1(X); w1(X)

(d) r3 (X); r2 (X); r1(X); w3(X); w1(X)

Solution

Let there be three transactions T1, T2, and T3. They are executed concurrently and produce a schedule S. S is serializable if it can be reproduced as at least one serial schedule

(a) This schedule is not serializable because T1 reads X (r1(X)) before T3 but T3 reads X

(r3(X)) before T1 writes X (w1(X)), where X is a common data item. The operation r2(X) of T2 does not affect the schedule at all so its position in the schedule is irrelevant. In a serial schedule T1, T2, and T3, the operation w1(X) comes after r3(X), which does not happen in the question.

(b) This schedule is not serializable because T1 reads X (r1(X)) before T3 but T3 writes X

(w3(X)) before T1 writes X (w1(X)). The operation r2(X) of T2 does not affect the schedule at all so its position in the schedule is irrelevant. In a serial schedule T1, T3, and T2, r3(X) and w3(X) must come after w1(X), which does not happen in the question.

(c) This schedule is **serializable** because all conflicting operations of T3 happens before all conflicting operation of T1. T2 has only one operation, which is a read on X (r2(X)), which does not conflict with any other operation. Thus this serializable schedule is equivalent to r2(X); r3(X); w3(X); r1(X); w1(X) (e.g., T2 □□T3 □□T1) serial schedule.

(d) This is not a serializable schedule because T3 reads X (r3(X)) before T1 reads X (r1(X)) but r1(X) happens before T3 writes X (w3(X)). In a serial schedule T3, T2, and T1, r1(X) will happen after w3(X), which does not happen in the question.

10 points will be awarded for following the homework guidelines document.