San Jose State University

CMPE 180B: Database System

Spring 2025

Homework 3

Due 05/08/2025 @ 23:59

Name: Phuong Duy Lam, Nguyen

ID: 018229432

Problem 1:

In the EMP_PROJ table, because SSN only partially determines the employee name and PNUMBER determines the project's name and location, you end up with all three classic anomalies: you cannot add a new project unless at least one employee is assigned to it (insertion anomaly), deleting the last employee on a project also wipes out the project's own details (deletion anomaly), and changing a project's name or location forces you to update every row with that PNUMBER to keep data consistent (update anomaly). Likewise, EMP_DEPT has SSN→DNUMBER plus a transitive dependency DNUMBER→DNAME,DMGRSSN, so you cannot create a department without inserting an employee, removing the final employee in a department erases the department's information, and updating the department's name or manager requires touching every tuple for that DNUMBER. In both relations mixing entity-level data (projects or departments) with relationship-level data (which employee works where) violates 2NF/3NF and causes insertion, deletion, and update anomalies

Problem 2:

Assuming that each item carries its own discount rate, that Total_price refers to the price of a single item on the order, that Odate records the date the order was placed, and that Total_amount gives the overall dollar amount for the order, the natural join of Order and Order_Item produces a single relation with this schema:

O#, Odate, Cust#, Total_amount, I#, Qty_ordered, Total_price, Discount%

Every tuple in this joined table represents one line-item on an order, enriched with the order's header information. Because an order can contain many items (and the same item can appear in

different orders), you need both O# (order number) and I# (item number) together to uniquely identify a row—so the primary key is the composite (O#, I#).

Under these assumptions the following functional dependencies hold in the joined relation:

- O# → Odate, Cust#, Total_amount

  (the order header—its date, customer ID, and total cost—depends solely on the order number)

- I# → Discount%

  (each distinct item carries its own fixed discount percentage)

- (O#, I#) → Qty_ordered, Total_price

  (knowing the specific order-item tells you how many units were ordered and the per-unit price)

To satisfy BCNF, every non-trivial dependency must have a superkey on its left side. Here, O# alone is not a superkey (it doesn't identify which item in the order) yet it determines Odate, Cust#, and Total_amount; similarly I# alone is not a superkey (it doesn't identify which order) yet it determines Discount%. Because neither O# nor I# by itself is a superkey, those dependencies violate BCNF. Only the full composite (O#, I#) qualifies as a superkey, and its dependency covers only Qty_ordered and Total_price. Therefore the joined relation is not in BCNF.

Problem 3:

There are 4 operations in T1 and 2 in T2, so the total number of interleavings that preserve each transaction's internal order is

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

All 15 schedules listing:

Schedule ID	Operations	Serializable?	Equivalent Serial Order
S1	$r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; $r_2(X)$; $w_2(X)$	Yes (serial)	T1 → T2
S2	$r_1(X)$; $w_1(X)$; $r_1(Y)$; $r_2(X)$; $w_1(Y)$; $w_2(X)$	Yes (conflict-serializable)	T1 → T2
S3	$r_1(X)$; $w_1(X)$; $r_1(Y)$; $r_2(X)$; $w_2(X)$; $w_1(Y)$	Yes (conflict-serializable)	T1 → T2
S4	$r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_1(Y)$; $w_2(X)$	Yes (conflict-serializable)	T1 → T2
S5	$r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$	Yes (conflict-serializable)	T1 → T2
S6	$r_1(X)$; $w_1(X)$; $r_2(X)$; $w_2(X)$; $r_1(Y)$; $w_1(Y)$	Yes (conflict-serializable)	T1 → T2
S7	$r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; $w_2(X)$	No	–
S8	$r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$	No	–
S9	$r_1(X)$; $r_2(X)$; $w_1(X)$; $w_2(X)$; $r_1(Y)$; $w_1(Y)$	No	–
S10	$r_1(X)$; $r_2(X)$; $w_2(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$	No	–
S11	$r_2(X)$; $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; $w_2(X)$	No	–
S12	$r_2(X)$; $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$	No	–
S13	$r_2(X)$; $r_1(X)$; $w_1(X)$; $w_2(X)$; $r_1(Y)$; $w_1(Y)$	No	–
S14	$r_2(X)$; $r_1(X)$; $w_2(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$	No	–
S15	$r_2(X)$; $w_2(X)$; $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$	Yes (serial)	T2 → T1

Problem 4:

Schedule	Operations	Serializable?	Equivalent Serial Order
(a)	$r_1(X); r_3(X); w_1(X); r_2(X); w_3(X)$	No	–
(b)	$r_1(X); r_3(X); w_3(X); w_1(X); r_2(X)$	No	–
(c)	$r_3(X); r_2(X); w_3(X); r_1(X); w_1(X)$	Yes	T2 → T3 → T1
(d)	$r_3(X); r_2(X); r_1(X); w_3(X); w_1(X)$	No	–