**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 | r₁(X); w₁(X); r₁(Y); w₁(Y); r₂(X); w₂(X) | Yes (serial) | T1 → T2 |
| S2 | r₁(X); w₁(X); r₁(Y); r₂(X); w₁(Y); w₂(X) | Yes (conflict-serializable) | T1 → T2 |
| S3 | r₁(X); w₁(X); r₁(Y); r₂(X); w₂(X); w₁(Y) | Yes (conflict-serializable) | T1 → T2 |
| S4 | r₁(X); w₁(X); r₂(X); r₁(Y); w₁(Y); w₂(X) | Yes (conflict-serializable) | T1 → T2 |
| S5 | r₁(X); w₁(X); r₂(X); r₁(Y); w₂(X); w₁(Y) | Yes (conflict-serializable) | T1 → T2 |
| S6 | r₁(X); w₁(X); r₂(X); w₂(X); r₁(Y); w₁(Y) | Yes (conflict-serializable) | T1 → T2 |
| S7 | r₁(X); r₂(X); w₁(X); r₁(Y); w₁(Y); w₂(X) | No | – |
| S8 | r₁(X); r₂(X); w₁(X); r₁(Y); w₂(X); w₁(Y) | No | – |
| S9 | r₁(X); r₂(X); w₁(X); w₂(X); r₁(Y); w₁(Y) | No | – |
| S10 | r₁(X); r₂(X); w₂(X); w₁(X); r₁(Y); w₁(Y) | No | – |
| S11 | r₂(X); r₁(X); w₁(X); r₁(Y); w₁(Y); w₂(X) | No | – |
| S12 | r₂(X); r₁(X); w₁(X); r₁(Y); w₂(X); w₁(Y) | No | – |
| S13 | r₂(X); r₁(X); w₁(X); w₂(X); r₁(Y); w₁(Y) | No | – |
| S14 | r₂(X); r₁(X); w₂(X); w₁(X); r₁(Y); w₁(Y) | No | – |
| S15 | r₂(X); w₂(X); r₁(X); w₁(X); r₁(Y); w₁(Y) | Yes (serial) | T2 → T1 |

**Problem 4:**

| **Schedule** | **Operations** | **Serializable?** | **Equivalent Serial Order** |
| --- | --- | --- | --- |
| **(a)** | r₁(X); r₃(X); w₁(X); r₂(X); w₃(X) | No | – |
| **(b)** | r₁(X); r₃(X); w₃(X); w₁(X); r₂(X) | No | – |
| **(c)** | r₃(X); r₂(X); w₃(X); r₁(X); w₁(X) | Yes | T2 → T3 → T1 |
| **(d)** | r₃(X); r₂(X); r₁(X); w₃(X); w₁(X) | No | – |