Date: November 15, 2023

Group 4 Topic # 26 Payroll Management DBMS		
Names	Student #	Signature
Hamza Malik	501112545	Malik
Omer Zulfiqar	501101201	omerz
Amanat Sodhi	501108395	AS

Assignment 8 Goal:

See the diagram examples posted for A7 (4 marks)

Lab 9) Assignment 8: Normalization 3NF/ BCNF by Algorithm: (4 marks). For A8 to change to 3NF you need to add additional FDs to be able to breakdown at least one table by Bernstein's Algorithm and show the details for making that table 3NF. Then verify all tables whether or not they are in 3NF or BCNF as shown in class. You can explain why each table is BCNF by showing FDs. You need to use BCNF algorithm (by adding FDs if needed) and showing the details for at least one table to BCNF. Note that during assignments 8 you use algorithm for 3NF or BCNF by both Bernstein's Algorithm for 3NF and BCNF algorithms each of them at least for one table. It is recommended to start creating Java/web based UI

FOLLOWING FDs for the Code:

For the EMPLOYEE table:
- EMPLOYEE_ID → NAME
- EMPLOYEE_ID → DESIGNATION_ID

For the SALARY table:
- SALARY_ID → EMPLOYEE_ID
- SALARY_ID → AMOUNT

For the PAYMENT table:
- PAYMENT_ID → EMPLOYEE_ID
- PAYMENT_ID → AMOUNT
- PAYMENT_ID → DATE_RECEIVED

For the TAX table:
- TAX_ID → EMPLOYEE_ID
- TAX_ID → TAX_AMOUNT

For the DEDUCTION table:
- DEDUCTION_ID → EMPLOYEE_ID
- DEDUCTION_ID → DEDUCTION_AMOUNT

- DEDUCTION_ID → REASON

USING THE BernStien's Algorithm:

As Employee and Salary tables (FD):

- EMPLOYEE_ID → NAME
- EMPLOYEE_ID → DESIGNATION_ID

For the SALARY table:

- EMPLOYEE_ID → AMOUNT

- For the EMPLOYEE table, EMPLOYEE_ID is the candidate key.
- For the SALARY table, EMPLOYEE_ID uniquely determines AMOUNT.

DECOMPOSITION TABLE:

```
CREATE TABLE EMPLOYEE (
   EMPLOYEE_ID VARCHAR2(100) NOT NULL PRIMARY KEY,
   NAME VARCHAR2(100) NOT NULL,
   DESIGNATION_ID VARCHAR2(100) REFERENCES DESIGNATION(DESIGNATION_ID)
);

CREATE TABLE SALARY (
   SALARY_ID VARCHAR2(100) NOT NULL PRIMARY KEY,
   EMPLOYEE_ID VARCHAR2(100) REFERENCES EMPLOYEE(EMPLOYEE_ID),
   AMOUNT NUMBER(10,2) NOT NULL CHECK (AMOUNT >= 0)
);
```

TO check whether if it's 3NF or BCNF:

- For 3NF, every non-prime attribute must be non-transitively dependent on every key.
- For BCNF, for every FD X → Y, X must be a superkey.

FIRST TABLE (DESIGNATION TABLE):

- FD: DESIGNATION_ID → TITLE
- DESIGNATION_ID is the primary key.
- No non-prime attributes exist (all attributes are part of a key).

- Conclusion: The DESIGNATION table is in BCNF (and therefore also in 3NF) because the only FD has a superkey on the left-hand side.

SECOND TABLE (EMPLOYEE TABLE):

- FDs: EMPLOYEE_ID → NAME, EMPLOYEE_ID → DESIGNATION_ID
- EMPLOYEE_ID is the primary key.
- Both NAME and DESIGNATION_ID are non-prime attributes and are fully functionally dependent on the primary key.
- There are no transitive dependencies.
- Conclusion: The EMPLOYEE table is in BCNF (and therefore also in 3NF) because every FD has a superkey (EMPLOYEE_ID) on the left-hand side.

THIRD TABLE (SALARY TABLE):

- FD: EMPLOYEE_ID → AMOUNT
- Initially, this might seem incorrect because SALARY_ID is the primary key, not EMPLOYEE_ID. However, if EMPLOYEE_ID is unique (which is suggested by the UNIQUE constraint in the provided SQL), then it can be considered an alternate key.

PAYMENT, TAX, and DEDUCTION Tables:

- AYMENT_ID → EMPLOYEE_ID, AMOUNT, DATE_RECEIVED
- TAX_ID → EMPLOYEE_ID, TAX_AMOUNT
- DEDUCTION_ID → EMPLOYEE_ID, DEDUCTION_AMOUNT, REASON

All three tables are in BCNF (and therefore also in 3NF) because, for all FDs, the left-hand side is a superkey.

You need to use BCNF algorithm (by adding FDs if needed) and showing the details for at least one table to BCNF:

DEPARTMENT_ID → AMOUNT

```
CREATE TABLE SALARY (
   SALARY_ID VARCHAR2(100) NOT NULL PRIMARY KEY,
   EMPLOYEE_ID VARCHAR2(100) UNIQUE REFERENCES EMPLOYEE(EMPLOYEE_ID),
   DEPARTMENT_ID VARCHAR2(100),
   AMOUNT NUMBER(10,2) NOT NULL CHECK (AMOUNT >= 0)
);
```

And the functional dependencies are:

SALARY_ID → EMPLOYEE_ID, DEPARTMENT_ID, AMOUNT
EMPLOYEE_ID → DEPARTMENT_ID
DEPARTMENT_ID → AMOUNT

Converting this to BCNF:

Decompose the Table:

decompose the SALARY table into two tables to remove the violation:

- SALARY_BANDS table: This table will store the relationship between DEPARTMENT_ID and AMOUNT.

```
CREATE TABLE SALARY_BANDS (
   DEPARTMENT_ID VARCHAR2(100) NOT NULL PRIMARY KEY,
   AMOUNT NUMBER(10,2) NOT NULL CHECK (AMOUNT >= 0)
);
```

Which connects to employee salary:

```
CREATE TABLE EMPLOYEE_SALARY (
   SALARY_ID VARCHAR2(100) NOT NULL PRIMARY KEY,
   EMPLOYEE_ID VARCHAR2(100) UNIQUE REFERENCES EMPLOYEE(EMPLOYEE_ID),
   DEPARTMENT_ID VARCHAR2(100) REFERENCES SALARY_BANDS(DEPARTMENT_ID)
);
```

After this decomposition:

- SALARY_BANDS satisfies BCNF because there's only one FD, and DEPARTMENT_ID is a superkey.
- EMPLOYEE_SALARY satisfies BCNF because SALARY_ID is a superkey and there are no non-key attributes.

The new schema for the salary table is now in BCNF.