Warren Spears

Computer Science 303: Database Management

6/5/2024

To identify functional and transitive dependencies in the given table, we need to understand how each field depends on the primary key and other fields. Functional dependency is a fundamental concept that describes the relationship between attributes (columns) in a table. Transitive dependency is an indirect relationship between non-key attributes in a database that occurs when the value of one attribute can be determined by the values of two or more other attributes.

Let us list out and analyze the given fields for the database:

- Employee ID
- Employee Last Name
- Employee First Name
- Street Address
- City
- State
- ZIP Code
- Department
- Manager ID
- Position
- Salary

Identifying the Primary Key

The primary key for this table is Employee ID because it uniquely identifies each record in the table. Each employee has a unique Employee ID.

Functional Dependencies

1. Employee ID → Employee Last Name, Employee First Name, Street Address, City, State, ZIP Code, Department, Manager ID, Position, Salary

- The Employee ID uniquely determines the employee's personal information, address, department, manager, position, and salary. This is the primary functional dependency as it uses the primary key to determine all other fields.

2. Street Address → City, State, ZIP Code

   - The street address uniquely determines the city, state, and ZIP code. Given a street address, you can determine the exact location in terms of city, state, and ZIP code.

3. Department → Manager ID, Position

   - The department typically determines the manager and the positions available within that department. For example, the Administration department is managed by Employee ID 1005 and has positions like Vice President.

4. Position → Salary

   - The position within the company determines the salary. For instance, the position of Vice President correlates with salaries of $95,000 or $75,000 depending on specific individuals.

Transitive Dependencies

Transitive dependencies occur when a non-prime attribute (an attribute not part of any candidate key) depends on another non-prime attribute via an intermediate attribute. In this case, several transitive dependencies can be identified:

1. Employee ID → Manager ID → Department

   - Employee ID determines the Manager ID, and the Manager ID determines the department. For instance, Employee ID 1010 has Manager ID 1005, which is linked to the Administration department.

2. Employee ID → Department → Position

   - Employee ID determines the department, and the department determines the position. For example, Employee ID 1010 is in the Administration department, which has the position of Vice President.

3. Street Address → City, State → ZIP Code

   - Street Address determines the City and State, which then determines the ZIP Code. For example, the address 312 Maple Drive is in Anytown, FL, which is within the ZIP Code 32829.

Analysis of Dependencies

1. Primary Key Dependencies:

- The primary key, Employee ID, ensures that each employee's data is unique and can be used to access all the related attributes.
- Every attribute in the table is directly functionally dependent on the Employee ID.

2. Non-Primary Key Functional Dependencies:

- Street Address, as a non-primary key attribute, functionally determines City, State, and ZIP Code. This means given a Street Address, the City, State, and ZIP Code can be uniquely identified.
- Department determines the Manager ID and Position, showing that within a specific department, there are certain roles and a specific manager overseeing the department.

3. **Transitive Dependencies:**

- There is a clear transitive dependency path from Employee ID through Manager ID to Department, indicating an indirect relationship through an intermediate attribute (Manager ID).
- Another transitive dependency is from Employee ID through Department to Position, demonstrating that knowing an employee's department helps in understanding their potential position within the company.
- Address-related transitive dependencies highlight the hierarchical structure of location data, where knowing a specific address reveals broader locational details like city and state, which then reveals the ZIP Code.

Implications of Dependencies

Redundancy and Anomalies:

- These dependencies suggest potential redundancy in the data. For example, if the department information changes, it has to be updated for all employees within that department to avoid inconsistencies.
- Update anomalies may occur if a manager's ID or department information changes, requiring multiple updates across different records.
- Deletion anomalies might arise if an employee who is a manager leaves, potentially orphaning the records of employees managed by them.

Normalization:

- To reduce redundancy and avoid anomalies, the table can be normalized. Normalization involves decomposing the table into smaller tables without losing data integrity.
- For instance, we could create separate tables for Employees, Departments, Positions, and Addresses to handle functional dependencies and reduce redundancy.

2. **Identify a Primary Key**

For the given data, the primary key is Employee ID. This field uniquely identifies each employee in the table. Each Employee ID is distinct, ensuring that no two records share the same primary key, which is essential for the unique identification of records in a relational database.

3. **Explain Why Table is Not 3NF**

To understand why the given table is not in Third Normal Form (3NF), let's review the prerequisites for a table to be in 3NF:

1. First Normal Form (1NF): (Odugbesan, T. , 2017)

A table is said to be in 1NF if the following rules hold:

1. Columns must have single values
2. Columns must have unique names
3. Values of a given attribute must be of the same data type
4. No two records (or rows) can be identical

2. Second Normal Form (2NF): (Odugbesan, T. , 2017)

A table is said to be in 2NF if the following rules hold:

1. Table is in 1NF
2. There should be no partial dependencies of any column on the primary key

3. Third Normal Form (3NF): (Odugbesan, T. , 2017)

A table is said to be in the 3NF if the rules hold as follows:

1. The table should be in the 2NF
2. All non-primary fields are dependent on the primary field
3. Transitive dependencies are removed

Analysis of the Given Table

The table, with fields such as Employee ID, Employee Last Name, Employee First Name, Street Address, City, State, ZIP Code, Department, Manager ID, Position, and Salary, meets 1NF as it has a primary key (Employee ID), and all values are atomic. However, it does not meet the requirements of 2NF and 3NF due to partial and transitive dependencies.

Partial Dependencies

A partial dependency occurs when a non-key attribute is dependent on part of a composite primary key. Since the primary key here is a single attribute (Employee ID), there are no partial dependencies.

Transitive Dependencies

Transitive dependencies are present when a non-key attribute depends on another non-key attribute. Here are a few transitive dependencies identified in the table:

1. Employee ID → Street Address → City, State, ZIP Code
    - The street address determines the city, state, and ZIP code. Therefore, City, State, and ZIP Code are transitively dependent on Employee ID through Street Address.
2. Employee ID → Department → Manager ID, Position
    - The department determines the manager and the position within the department. Hence, Manager ID and Position are transitively dependent on Employee ID through the Department.
3. Employee ID → Position → Salary
    - The position determines the salary, making Salary transitively dependent on Employee ID through the Position.

Why the Table is Not in 3NF

The presence of these transitive dependencies means that the table is not in 3NF. For the table to be in 3NF, each non-key attribute must be directly dependent on the primary key, not on other non-key attributes. This table violates rules 2 and 3 in the 3NF definition.

Example Transitive Dependency:

Consider the following dependency:

- Employee ID → Department → Manager ID

Here, knowing the Employee ID helps to determine the Department, and knowing the Department helps to determine the Manager ID. This implies that Manager ID is indirectly dependent on Employee ID via the Department, indicating a transitive dependency.

Such dependencies can lead to anomalies:

- Update Anomaly: If a department's manager changes, multiple records need to be updated to reflect this change. If any record is missed, the data becomes inconsistent.

- Deletion Anomaly: If an employee leaves and their record is deleted, valuable information about the department or manager might be lost.
- Insertion Anomaly: Adding a new employee without complete information about the department or manager might be problematic.

To move the table to 3NF, we need to eliminate these transitive dependencies by creating separate tables for related attributes. This process will involve:

- Creating an Addresses table with attributes: Street Address, City, State, ZIP Code.
- Creating a Departments table with attributes: Department, Manager ID.
- Creating a Positions table with attributes: Position, Salary.

By doing this, each attribute will be dependent only on the primary key, thereby removing any transitive dependencies.

4. **Explain the Current Normalization Status**

Current Normalization Status: Second Normal Form (2NF)

The given table is in Second Normal Form (2NF) but not in Third Normal Form (3NF).

- First Normal Form (1NF): The table satisfies 1NF as all fields contain atomic values, and there are no repeating groups.
- Second Normal Form (2NF): The table satisfies 2NF as it has no partial dependencies on the primary key, which is a single field (Employee ID).

However, the table fails to meet the criteria for 3NF due to the presence of transitive dependencies, where non-key attributes depend on other non-key attributes rather than directly on the primary key.

To achieve 3NF, the table must be decomposed to eliminate these transitive dependencies, ensuring that all non-key attributes are directly dependent on the primary key and nothing else.

To create the tables in MySQL and ensure they are in 3NF, we need to decompose the original table into several smaller tables, each addressing specific dependencies and avoiding transitive dependencies.

Step-by-Step Table Creation

1. Employee Table: Contains employee-specific information.
2. Address Table: Contains address details.
3. Department Table: Contains department and manager information.
4. Position Table: Contains position and salary information.

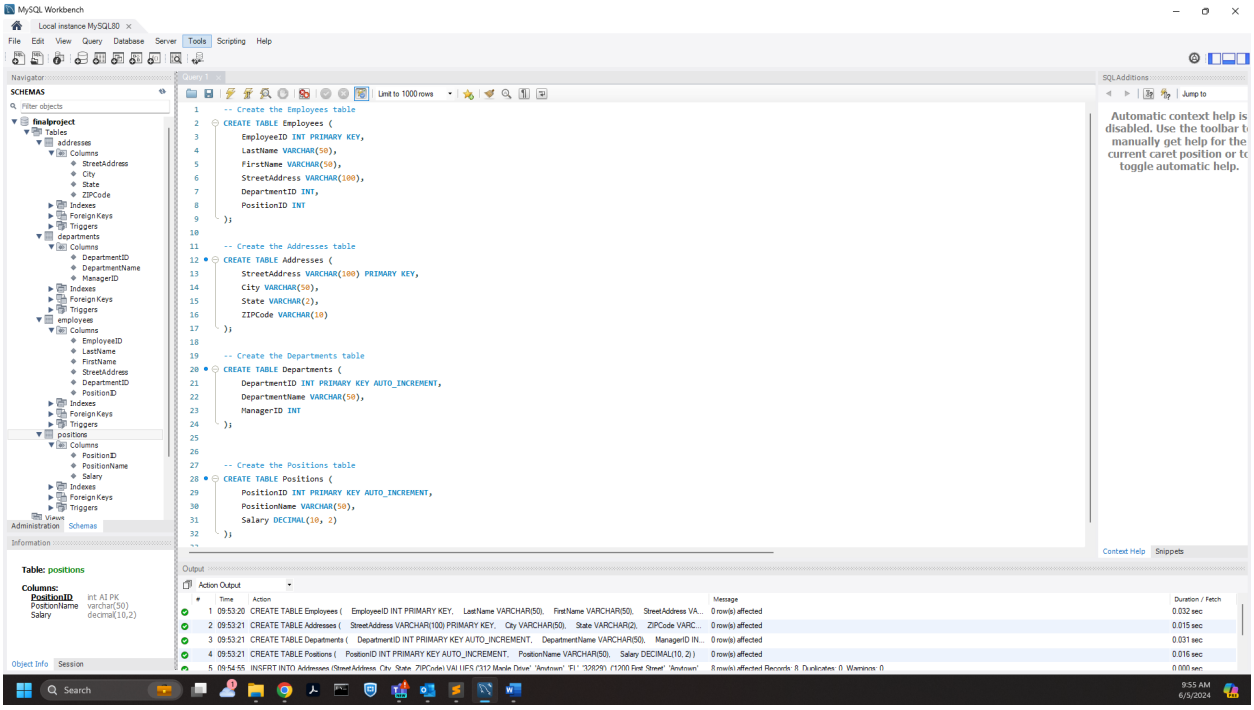
SQL Statements to Create Tables

```
-- Create the Employees table
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    LastName VARCHAR(50),
    FirstName VARCHAR(50),
    StreetAddress VARCHAR(100),
    DepartmentID INT,
    PositionID INT
);

-- Create the Addresses table
CREATE TABLE Addresses (
    StreetAddress VARCHAR(100) PRIMARY KEY,
    City VARCHAR(50),
    State VARCHAR(2),
    ZIPCode VARCHAR(10)
);

-- Create the Departments table
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    DepartmentName VARCHAR(50),
    ManagerID INT
);
```

```
-- Create the Positions table
CREATE TABLE Positions (
    PositionID INT PRIMARY KEY AUTO_INCREMENT,
    PositionName VARCHAR(50),
    Salary DECIMAL(10, 2)
);
```



```
-- Insert data into Addresses table
INSERT INTO Addresses (StreetAddress, City, State, ZIPCode) VALUES
('312 Maple Drive', 'Anytown', 'FL', '32829'),
('1200 First Street', 'Anytown', 'FL', '32829'),
('4989 Fleur de Lane', 'Sometown', 'FL', '32829'),
('12 Arcadia Avenue', 'Anytown', 'FL', '32829'),
('687 Gulf View Street', 'Sometown', 'FL', '32830'),
('1209 Pine Tree Lane', 'Sometown', 'FL', '32831'),
('5435 Main Street', 'Anytown', 'FL', '32831'),
('3 Post Drive', 'Sometown', 'FL', '32831');

-- Insert data into Departments table
INSERT INTO Departments (DepartmentName, ManagerID) VALUES
('Board of Directors', 1000),
('Administration', 1005),
('Information Technology', 1015),
('Sales', 1021);



-- Insert data into Positions table
INSERT INTO Positions (PositionName, Salary) VALUES
('President', 100000.00),
('Vice President', 95000.00),
('Vice President', 75000.00),
('Programmer II', 70000.00),
('Programmer I', 45000.00),
```

('Programmer I', 60000.00),

('Sales Representative', 50000.00),

('Sales Representative', 35000.00);

-- Insert data into Employees table

INSERT INTO Employees (EmployeeID, LastName, FirstName, StreetAddress, DepartmentID, PositionID) VALUES

(1005, 'Doe', 'Janet', '312 Maple Drive', 1, 1),

(1010, 'Eyre', 'Jane', '1200 First Street', 2, 2),

(1011, 'Bronte', 'Charlotte', '4989 Fleur de Lane', 2, 3),

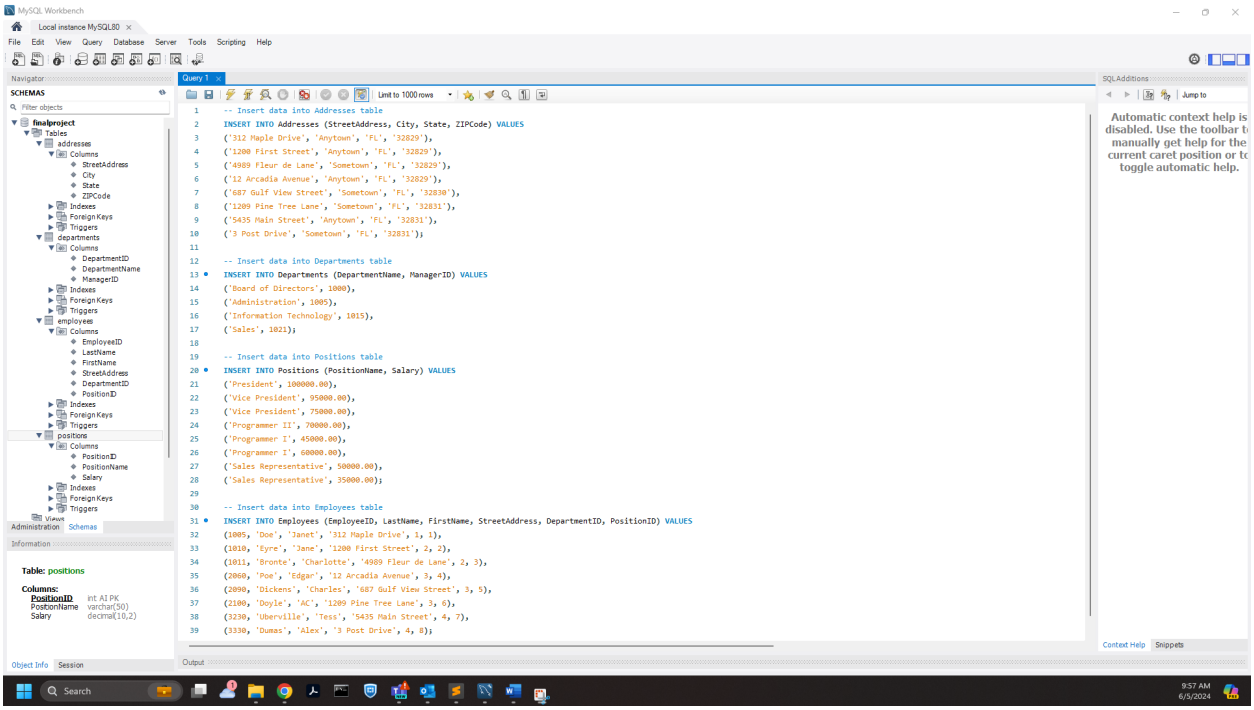
(2060, 'Poe', 'Edgar', '12 Arcadia Avenue', 3, 4),

(2090, 'Dickens', 'Charles', '687 Gulf View Street', 3, 5),

(2100, 'Doyle', 'AC', '1209 Pine Tree Lane', 3, 6),

(3230, 'Uberville', 'Tess', '5435 Main Street', 4, 7),

(3330, 'Dumas', 'Alex', '3 Post Drive', 4, 8);



References:

Odugbesan, T. (2017, December 30). What is Normal Form in DBMS? - Types & Examples. Study.com. https://study.com/academy/lesson/what-is-normal-form-in-dbms-types-examples.html