Db3

Definitions:

a. Candidate Key

A candidate key is an attribute or a set of attributes that can uniquely identify a tuple (row) in a database table. A candidate key must be unique (no two tuples can have the same value for the candidate key) and minimal (the key cannot contain any unnecessary attributes; if an attribute is removed from the key, it would no longer be a candidate key). Among all the candidate keys, one is chosen to be the primary key.

b. Composite Key

A composite key is a type of candidate key that consists of two or more attributes that together uniquely identify a record in a database table. Each individual attribute in a composite key may not be unique on its own, but when combined, they form a unique identifier for each record.

c. Foreign Key

A foreign key is a column or a combination of columns in one table that refers to the primary key or a unique key in another table. The purpose of the foreign key is to ensure referential integrity between the tables, meaning that only valid entries (as defined by the primary or unique key in the referenced table) can be inserted into the foreign key column(s).

d. Functional Dependency

In the context of database design, a functional dependency is a relationship between two sets of attributes in a relation (table) such that the value of one set of attributes (called the determinant) determines the value of another set of attributes (called the dependent). It is denoted as X → Y, which means that if two tuples agree on the values of X, then they must also agree on the values of Y.

Integrity Rules/Constraints:

1. Entity Integrity Constraint

This rule states that every table must have a primary key, and the primary key cannot contain null values. This ensures that each row in the table can be uniquely identified without ambiguity.

2. Referential Integrity Constraint

This rule governs the relationships between tables through foreign keys. It stipulates that a foreign key in one table must either match a primary key value in another table or be null. This constraint prevents orphaned records and maintains the consistency of data across related tables.

3. Domain Integrity Constraint

This rule ensures that the data in a specific column adheres to a particular set of values or constraints. For example, a column might be constrained to accept only certain types of data (e.g., integers, dates, etc.), or the values might be restricted to a predefined list. This helps to maintain the accuracy and consistency of the data stored in the database.

These rules and constraints are fundamental to ensuring that a relational database remains accurate, consistent, and reliable.

3...Table 1: Film and Director

Film Table:

Primary Key: filmNo

Foreign Key: directorNo (references Director table)

Director Table:

Primary Key: directorNo

Violations:

The film with filmNo 111 has a directorNo of 753, which does not exist in the Director table. This violates referential integrity because there is no corresponding director with directorNo 753.

Table 2: Supplier

Supplier Table:

Primary Key: Supplier

Composite Key: (PartNo, Quantity, Price)

Violations:

There are no apparent violations in this table based on the given data. Each supplier has unique part numbers with associated quantities and prices.

Table 3: Song

Song Table:

Primary Key: Song

Composite Key: (Artist, Position, Month, Year)

Violations:

There are no apparent violations in this table based on the given data. Each song has a unique combination of artist, position, month, and year.

In summary, the only violation identified is in Table 1: the film with filmNo 111 has a directorNo that does not exist in the Director table.

4Anomalies Resulting from Operations

a. Insertion Management wishes to create and insert a new project into the table: ProjectCode= PrC30, ProjectTitle= Skills Matrix, ProjectManager = M. Uhura, ProjectBudget = 20000;

Anomaly: Insertion Anomaly

The insertion of a new project requires all attributes for every employee working on it. If no employees have been assigned yet, this information cannot be inserted.

b. Deletion Project ‘PrC10’ ended abruptly and was deleted from the table.

Anomaly: Deletion Anomaly

Deleting the entire row for Project 'PrC10' will also delete all associated employee data, even if they continue to work on other projects.

c. Modification Due to her outstanding performance ‘J Kirk’ was moved from department ‘L004’ to department ‘L009’

Anomaly: Update Anomaly

Updating J Kirk's department requires updating multiple rows where she appears, which can lead to inconsistencies if not done correctly.

Deriving Normal Forms

To address these anomalies, we need to decompose the table into normal forms.

d. Derive 1NF (First Normal Form)

A relation is in 1NF if it has atomic values in each column. The current table already satisfies 1NF because there are no repeating groups or composite columns.

e. Derive 2NF (Second Normal Form)

A relation is in 2NF if it is in 1NF and every non-key attribute is fully dependent on the primary key. To achieve 2NF, we need to eliminate partial dependencies.

Decomposition:

Projects Table:

ProjCode

ProjTitle

ProjManager

ProjBudget

Employees Table:

EmpNo

EmpName

DeptNo

DeptName

HrlyRate

Assignments Table:

ProjCode

EmpNo

This decomposition ensures that each non-key attribute depends only on the primary key.

f. Derive 3NF (Third Normal Form)

A relation is in 3NF if it is in 2NF and there are no transitive dependencies among non-key attributes. In our case, after achieving 2NF, we need to ensure that there are no further dependencies between non-key attributes.

Further Decomposition:

Departments Table:

DeptNo

DeptName

Now, the tables are in 3NF:

Projects Table:

ProjCode

ProjTitle

ProjManager

ProjBudget

Employees Table:

EmpNo

EmpName

DeptNo

HrlyRate

Assignments Table:

ProjCode

EmpNo

Departments Table:

DeptNo

DeptName

These decompositions help in eliminating the anomalies identified earlier by ensuring proper normalization.

5

### 1. First Normal Form (1NF)

- Eliminate repeating groups.

- Ensure that each column value is atomic (indivisible).

#### Table Structure:

| Order No. | Acc. No. | Customer | Address | Date | Item | Qty. | Price | Total Cost |

|-----------|----------|----------------|------------------------|--------|----------------|------|-------|------------|

| 7823 | 178 | Daisy's Café | 27 Bay Drive, Cove | 16-Jul | Bakewell Tart | 20 | 0.15 | £12.35 |

| 7823 | 178 | Daisy's Café | 27 Bay Drive, Cove | 16-Jul | Danish Pastry | 13 | 0.20 | £12.35 |

| 7823 | 178 | Daisy's Café | 27 Bay Drive, Cove | 16-Jul | Apple Pie | 45 | 0.15 | £12.35 |

| 4633 | 526 | Smiths | 12 Dee View, Aberdeen | 16-Jul | Butteries | 120 | 0.20 | £24.00 |

| 2276 | 167 | Sally's Snacks | 3 High Street, Banchory| 17-Jul | Apple Pie | 130 | 0.15 | £56.50 |

| 2276 | 167 | Sally's Snacks | 3 High Street, Banchory| 17-Jul | Cherry Pie | 100 | 0.18 | £56.50 |

| 2276 | 167 | Sally's Snacks | 3 High Street, Banchory| 17-Jul | Steak Pie | 30 | 0.50 | £56.50 |

| 2276 | 167 | Sally's Snacks | 3 High Street, Banchory| 17-Jul | Meringue Pie | 20 | 0.20 | £56.50 |

| 1788 | 32 | Tasty Bite | 17 Wood Place, Insch | 18-Jul | Apple Pie | 15 | 0.15 | £7.50 |

| 1788 | 32 | Tasty Bite | 17 Wood Place, Insch | 18-Jul | Danish Pastry | 50 | 0.20 | £7.50 |

### Anomalies Identified:

- \*\*Insertion Anomaly\*\*: If you want to add a new customer but they have not placed an order yet, you cannot insert their information because there is no order number associated with them.

- \*\*Update Anomaly\*\*: If you need to update the address of a customer, you would have to update it for every row where that customer appears.

- \*\*Deletion Anomaly\*\*: If you delete an order, you also lose the customer information associated with that order.

### 2. Second Normal Form (2NF)

- Eliminate partial dependencies by creating separate tables for entities that depend on only part of the primary key.

#### Tables:

1. \*\*Customers\*\*

- Acc. No.

- Customer

- Address

2. \*\*Orders\*\*

- Order No.

- Acc. No.

- Date

3. \*\*Order\_Items\*\*

- Order No.

- Item

- Qty

- Price

- Total Cost

### 3. Third Normal Form (3NF)

- Eliminate transitive dependencies by creating separate tables for entities that depend on other non-key attributes.

#### Tables:

1. \*\*Customers\*\*

- Acc. No.

- Customer

- Address

2. \*\*Orders\*\*

- Order No.

- Acc. No.

- Date

3. \*\*Items\*\*

- Item

- Price

4. \*\*Order\_Items\*\*

- Order No.

- Item

- Qty

- Total Cost

By following these steps, we can ensure that the data is properly normalized and free from anomalies.

6

Sure, let's address each part of the question step by step.

### a. Produce the schema for each table

#### Patient Table Schema:

- \*\*PatientNo (Primary Key)\*\*

- Surname

- FirstName

#### Admission Table Schema:

- \*\*PatientNo (Foreign Key referencing Patient.PatientNo)\*\*

- Admitted

- Discharged

- Ward (Foreign Key referencing Ward.Ward)

#### Doctor Table Schema:

- \*\*DoctorNo (Primary Key)\*\*

- Surname

- FirstName

- Ward (Foreign Key referencing Ward.WardName)

#### Ward Table Schema:

- \*\*Ward (Primary Key)\*\*

- WardName

- DoctorNoInCharge (Foreign Key referencing Doctor.DoctorNo)

### b. Identify the relations between the tables and specify the links between the foreign keys and their corresponding primary keys

1. \*\*Patient to Admission:\*\*

- The `PatientNo` in the `Admission` table is a foreign key that references the `PatientNo` in the `Patient` table.

2. \*\*Admission to Ward:\*\*

- The `Ward` in the `Admission` table is a foreign key that references the `Ward` in the `Ward` table.

3. \*\*Doctor to Ward:\*\*

- The `Ward` in the `Doctor` table is a foreign key that references the `Ward` in the `Ward` table.

4. \*\*Ward to Doctor:\*\*

- The `DoctorNoInCharge` in the `Ward` table is a foreign key that references the `DoctorNo` in the `Doctor` table.

### c. Generate the corresponding ER diagram

An Entity-Relationship (ER) diagram can be described as follows:

1. \*\*Entities:\*\*

- Patient

- Admission

- Doctor

- Ward

2. \*\*Relationships:\*\*

- A patient has many admissions (`Patient` to `Admission`)

- An admission belongs to one ward (`Admission` to `Ward`)

- A doctor is associated with one ward (`Doctor` to `Ward`)

- A ward is managed by one doctor (`Ward` to `Doctor`)

The ER diagram would show these entities connected by lines indicating the relationships, with arrows pointing from the "many" side to the "one" side where applicable. Each entity would have its attributes listed within it, and the primary and foreign keys would be indicated accordingly.