Databases-Week03

-Tao Nijia, He Yiyang, Liang Ruyi

BASIC TASKS

Task 1.

a. Candidate key: A candidate key is a minimal set of attributes within a table that can uniquely identify each record in that table. There can be one or more candidate keys in a table.

b. Composite key: A composite key is a key that consists of two or more attributes that together uniquely identify a record in a table.

c. Foreign key: A foreign key is an attribute or a set of attributes in one table that refers to the primary key of another table. It is used to establish a relationship between two tables and maintain referential integrity.

d. Functional dependency: In a database, a functional dependency is a relationship between two sets of attributes in a table, where the value of one set of attributes (the dependent attributes) is determined by the value of another set of attributes (the determinant

attributes). It is denoted as A -> B, where A is the determinant and B is the dependent.

Task 2.

In the relational model, the three integrity rules/constraints are:

1. Entity Integrity: This rule states that the primary key of a table must have a unique value and cannot be null. Each row in the table must be uniquely identifiable by the primary key.
2. Referential Integrity: This rule ensures that the values of foreign keys in a table must match the values of the primary keys in the referenced table. If a foreign key value exists in a table, it must refer to an existing row in the referenced table.
3. Domain Integrity: This rule specifies that the values entered into a column must be of the correct data type and must fall within a specified range or set of values. It ensures the validity and consistency of the data stored in the database.

Task 3

Table1:

1. The line filmNo: 008 is missing the value of genre.
2. There is no directorNo value for "Snakes on a Plane" stored in the Table Directors: 753.

Table2:

Violation of referential integrity: The value of the Price column of the Aziz row is ABC, which does not meet the numerical requirements, which may lead to data inconsistency and affect the referential integrity (assuming that the correct data is a numerical value).

Table3:

The song A Kind of Magic is missing the primary key Artist value

MEDIUM TASKS

Task 4

a. The new project lacks a primary key called EmployeeNo
b. Deleting Prc10 will lose the information of employees

c. Moving J Kirk from department L004 to Department L009 results in inconsistent department information in multiple related records

d. 1NF

Analyze the original table structure

The original ProjectEmployee table contains a variety of information about projects and employees, For example, ProjectCode, ProjectTitle, ProjectManager, ProjectBudget, EmployeeNo, EmployeeName, DepartmentNo, DepartmentName, and HourlyRate. Where there may be multiple employees involved for each project, there are issues of data redundancy and duplicate groups.

Steps to convert to 1NF

Split the duplicate sets of data so that each tuple (row) represents an employee's information on a project. In this way, each cell in the table contains only one atomic value, meeting the 1NF requirement. For example, if there is a project PRC10 in the original table with three employees involved, there will be three rows of data in the table 1NF that correspond to the information of those three employees in the

PRC10 project.

e. 2NF

Project: Includes ProjectCode (primary key), ProjectTitle, ProjectManager, and ProjectBudget. This relationship mainly stores basic information about the project.

Employee relationship: Contains EmployeeNo (primary key), EmployeeName, DepartmentNo, DepartmentName, and HourlyRate. This relationship mainly stores basic information about employees.

f. 3NF

Employee relationship is further decomposed into two relationships:

EmployeeBasic: Contains EmployeeNo (primary key), EmployeeName, and DepartmentNo. This relationship stores the employee's basic information and department number.

Department: Includes DepartmentNo (primary key) and DepartmentName. This relationship stores department information separately.

Through such decomposition, the transfer function

dependence is eliminated and the requirement of 3NF is satisfied.

Task 5

a. Anomaly analysis

1. Insert exception: When inserting a new order, if there is No customer information (such as ac.no. Is empty), it may not be inserted correctly.

2. Deletion exception: Deletion of an order may result in loss of customer information.

3. Modification exception: To modify the customer address, it needs to be modified in multiple order records, which is prone to error

b. 1NF

Split the item information in each order so that each item has its own separate row and each cell in the table contains only one atomic value. In this way, the data in the table meets the requirements of 1NF. For example, if an order contains three items, there will be three rows of data in the 1NF table for each item.

b. 2NF

Decompose the table into three new relationships:

1. Order relation (Order) : Contains Order No. (primary key), Date, Acc.No., Customer, Address. This relationship mainly stores the basic information of the order and the customer information.
2. Item relation: Contains Item (primary key), Qty., and Price. This relationship mainly stores basic information about the product.
3. OrderItem relation: contains Order No. (foreign key, refer to the Order No.), Item (foreign key, refer to the Item of the commodity relation). This relationship is used to establish many-to-many connections between orders and goods.

d. 3NF

The order relationship is further decomposed into two relationships:

1. OrderBasic information relationship (OrderBasic) : contains Order No. (primary key), Date, Acc.No. This relationship stores the basic information of the order and the customer account number.
2. Customer relationship (Customer) : Contains AC.no. (primary key), Customer, and Address. This relationship stores customer information separately.

Task 6

a. Patient(PatientNo,Surname,FirstName)

Admission(PatientNo,Admitted,Discharged,Ward)

Doctor(DoctorNo,Surname,FirstName,Ward)

Ward(Ward,WardName,DoctorNo-InCharge)

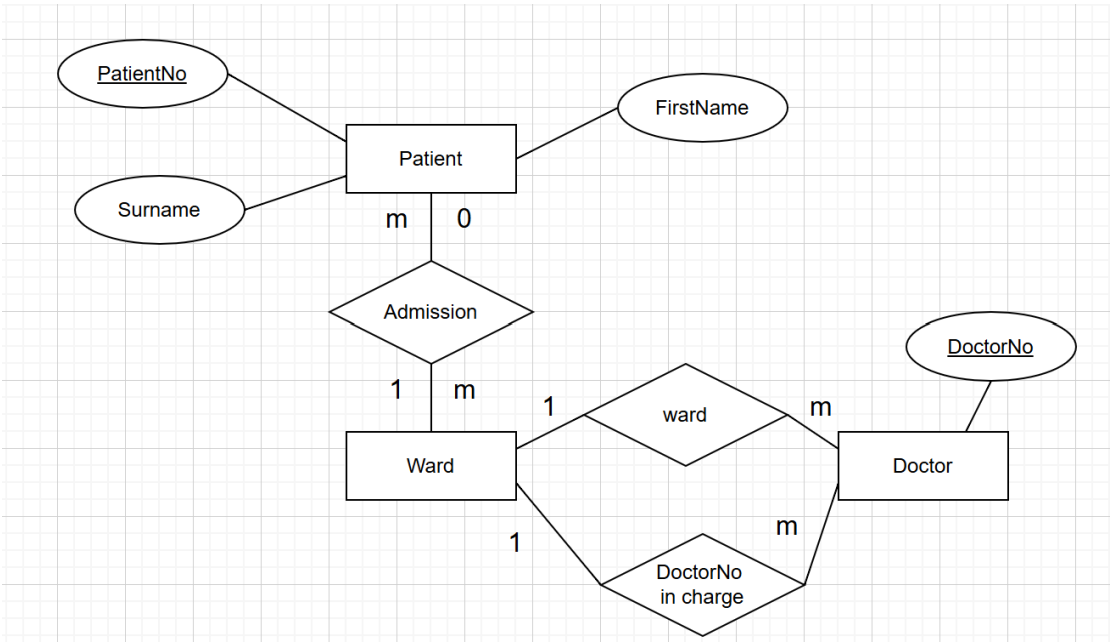
b. The relationships between tables are as follows:

The Patient table and the Admission table are associated through PatientNo.

Admission tables and Ward tables are associated by Ward.

The Doctor table and Ward table are associated by Ward.

c.



ADVANCED TASKS

Task 7.

1.

Table 1

- Order No → Account No, Customer, Address, Date (Order No determines the customer and their details)
- Order No, Item → Quantity, Item Price (The combination of Order No and Item determines the quantity and item price)

Table 2:

- Student No → Name, Course, Course Duration (A student number determines the student's details)
- Student No, Module No → Module Name, Lecturer (The combination of student number and module number determines the module name and lecturer)

2

Table 1

1NF

Table 1 is already in 1NF as all attributes contain atomic values, and there are no repeating groups.

2NF

- **Orders Table**:
    - Order No → Account No, Customer, Address, Date

- **Order Details Table**:
    - Order No, Item → Quantity, Item Price

3NF:

In this case, there are no transitive dependencies, as all non-key attributes directly depend on the primary key, so the table is already in 3NF after the 2NF step.

Table 2

1NF:

The table is already in 1NF since all the data values are atomic and there are no repeating groups.

2NF:

The primary key here is Student No, Module No. The non-key attributes Name, Course, Course Duration depend only on Student No, not Module No. So, we split the table to achieve 2NF:

- **Students Table**:
    - Student No → Name, Course, Course Duration
- **Student Modules Table**:
    - Student No, Module No → Module Name, Lecturer

3NF (Third Normal Form):

To ensure 3NF, we must remove any transitive dependencies. In this case, there are no transitive dependencies, so the tables are

already in 3NF after the 2NF step.

8

a. Branch Code → Branch Name, Supervisor ID, Supervisor Name

- Car Plate No → Car Type
- Bill No → Bill Date, Penalty, Final Bill Amount
- Supervisor ID → Supervisor Name

b.

1NF

The data already meets the 1NF criteria, as each field contains atomic values and there are no repeating groups.

2NF

1. **Branch Table**:
    - Branch Code → Branch Name, Supervisor ID, Supervisor Name
2. **Car Table**:
    - Car Plate No → Car Type
3. **Bill Table**:
    - Bill No → Bill Date, Penalty, Final Bill Amount
4. **Branch-Car-Bill Table**:

- Branch Code, Car Plate No, Bill No (to associate branches, cars, and bills)

3NF:

1. **Branch Table**:
   - Branch Code → Branch Name, Supervisor ID
2. **Supervisor Table**:
   - Supervisor ID → Supervisor Name

Task 8

a.

Branch Code → Branch Name, Supervisor ID, Supervisor Name-

Car Plate No → Car Type` (Car plate number determines the car type)

Bill No → Bill Date, Penalty, Final Bill Amount

Supervisor ID → Supervisor Name

b. Normalization Process: 1NF, 2NF, and 3NF

1NF

The data already meets the 1NF criteria, as each field contains atomic values and there are no repeating groups.

2NF:

1. Branch Table:

-Branch Code → Branch Name, Supervisor ID, Supervisor Name

2. Car Table

Car Plate No → Car Type`

3. Bill Table:

-Bill No → Bill Date, Penalty, Final Bill Amount

4. Branch-Car-Bill Table:

-Branch Code, Car Plate No, Bill No

3NF:

1. Branch Table:

Branch Code → Branch Name, Supervisor ID

2. Supervisor Table:

Supervisor ID → Supervisor Name

9. CHALLENGE YOURSLEF!

I. Logical Model Design

- Entities:
    - Boat: Attributes include boat_id and equipment_type.
    - Engineer: Attributes are engineer_id and name.
    - Task: Attributes are task_id and task_type.
    - ServiceRecord: Attributes are service_record_id, service_time, man_hours, date, engineer_id (foreign key referencing Engineer), boat_id (foreign key referencing Boat), and task_id (foreign key referencing Task).
- Relationships:
    - Boat and Task: Many-to-many relationship.
    - Engineer and ServiceRecord: One-to-many relationship.
    - ServiceRecord and Task: One-to-many relationship.
    - ServiceRecord and Boat: One-to-many relationship.

II. Physical Model Design

- Table Structures:
    - boats: boat_id (primary key), equipment_type.
    - engineers: engineer_id (primary key), name.
    - tasks: task_id (primary key), task_type.

- service_records: service_record_id (primary key), engineer_id (foreign key), boat_id (foreign key), task_id (foreign key), service_time, man_hours, date.

IV. Functional Dependencies

- In service_records table, service_record_id determines all other attributes, i.e., service_record_id → engineer_id, service_record_id → boat_id, service_record_id → task_id, service_record_id → service_time, service_record_id → man_hours, service_record_id → date.
- In engineers table, engineer_id determines name, i.e., engineer_id → name.
- In tasks table, task_id determines task_type, i.e., task_id → task_type.
- In boats table, boat_id determines equipment_type, i.e., boat_id → equipment_type.

V. Populating Test Information

- boats table:

    - boat_id: 1, equipment_type: Fishing boat equipment.
    - boat_id: 2, equipment_type: Yacht equipment.
- engineers table:
    - engineer_id: 1, name: John.
    - engineer_id: 2, name: Mike.
- tasks table:
    - task_id: 1, task_type: Service.
    - task_id: 2, task_type: Software upgrade.
    - task_id: 3, task_type: Repair.
    - task_id: 4, task_type: Safety inspection.
    - task_id: 5, task_type: Customer specific request.
- service_records table:
    - service_record_id: 1, engineer_id: 1, boat_id: 1, task_id: 1, service_time: October 10, 2024, 8 am, man_hours: 4, date: October 10, 2024.
    - service_record_id: 2, engineer_id: 2, boat_id: 2, task_id: 2, service_time: October 11, 2024, 9 am, man_hours: 3, date: October 11, 2024.

VI. Normalization

- First Normal Form (1NF): Each attribute is atomic and indivisible. The design already meets 1NF.

- Second Normal Form (2NF): Eliminate partial functional dependencies of non-key attributes on candidate keys. In service_records table, service_record_id is the candidate key and all other attributes fully depend on it. In engineers table, engineer_id is the candidate key and name fully depends on it. In tasks table, task_id is the candidate key and task_type fully depends on it. In boats table, boat_id is the candidate key and equipment_type fully depends on it. So, it meets 2NF.
- Third Normal Form (3NF): Eliminate transitive functional dependencies. There are no transitive functional dependencies in the design, so it meets 3NF.