Logicql Design MNHS

Data Management Course
UM6P College of Computing

Professor: Karima Echihabi   **Program:** Computer Engineering
Session: Fall 2025

Team Information

Team Name	Sekel
Member 1	Anass Mahdad
Member 2	Youssef Ouazzani
Member 3	Rania Moujahed
Member 4	Abdelmoughit Labrigui
Member 5	Mohamed Idriss Nana
Member 6	Haitam Laghmam
Repository Link	https://github.com/Data-Project-Hierarchy/

1 Introduction

Making the logical database from the ER diagram is like turning a rough sketch into something you can actually use. The ER diagram shows the main things and how they connect, and the logical design figures out exactly how to put it all into tables with keys so it works in a real database. If you mess this part up, the database won't work properly later.

2 Requirements and Methodology

Logical Design

Note:

- **Primary Keys** are underlined.

- **Foreign Keys** are in bold.

Relation [Have] (N:M)

Patient (IID: INTEGER, CIN: STRING, name: STRING, sex: CHAR(1), birth: DATE, blood_group: STRING, phone: STRING)

Contact_Location (CLID: INTEGER, city: STRING, province: STRING, street: STRING, number: STRING, postal_code: STRING, phone: STRING)

Have (**CLID**: INTEGER, **IID**: INTEGER)

Foreign Key (FK)

- Key **CLID** represents Contact_Location(CLID)

- Key **IID** represents Patient(IID)

Composite Key   Have (**CLID, IID**): Relationship of type N:M between Patient and Contact_Location.

Relation [Covers] (N:M)

Patient (IID: INTEGER, CIN: STRING, name: STRING, sex: CHAR(1), birth: DATE, blood_group: STRING, phone: STRING)

Insurance (InsID: INTEGER, type: STRING)

Covers (**InsID**: INTEGER, **IID**: INTEGER)
   Note: Since this is an N:M relationship, a new relation (Covers) is created. It inherits the primary keys from both related entities, which together form its composite primary key.

UM6P
University
Mohammed VI
Polytechnic

College of
Computing

Foreign Key (FK)

- Key **InsID** represents Insurance(InsID)

- Key **IID** represents Patient(IID)

Composite Key   Covers (InsID, IID): Relationship of type N:M between Patient and Insurance.

Relation [Expense_Attached] (1:M)

Patient (<u>IID</u>: INTEGER, CIN: STRING, name: STRING, sex: CHAR(1), birth: DATE, blood_group: STRING, phone: STRING)

Expense_Attached (<u>ExID</u>: INTEGER, total: DECIMAL, **InsID**: INTEGER)
   Note: Each Expense instance has **exactly one** *Insurance instance, thus the merging between Expense and Attached.*

Foreign Key (FK)

- Key **InsID** represents Insurance(InsID)

ISA [Staff] (1:1 Total Specialization)

Practitioner (**<u>STAFF_ID</u>**: INTEGER, specialty: STRING, license_number: STRING)

Caregiving (**<u>STAFF_ID</u>**: INTEGER, grade: STRING, ward: STRING)

Technical (**<u>STAFF_ID</u>**: INTEGER, modality: STRING, certifications: STRING)

Foreign Key (FK)

- Key **STAFF_ID** represents Staff(STAFF_ID)

Relation [Work_In] (N:M)

Staff (<u>STAFF_ID</u>: INTEGER, name: STRING, status: STRING)

Department (<u>DEP_ID</u>: INTEGER, name: STRING, specialty: STRING)

Work_In (**<u>STAFF_ID</u>**: INTEGER, **<u>DEP_ID</u>**: INTEGER)
   Note: Since this is an N:M relationship, a new relation (Work_In) is created. It inherits the primary keys from both related entities, which together form its composite primary key.

Foreign Key (FK)

- Key **STAFF_ID** represents Staff(STAFF_ID)

- Key **DEP_ID** represents Department(DEP_ID)

Composite Key **Work_In** (**STAFF_ID, DEP_ID**): Relationship of type N:M between Staff and Department.

Relation [Linked] (1:M)

Clinical_Activity (CAID: INTEGER, time: TIME, date: DATE, **staff_id**: INTEGER, **IID**: INTEGER, **ExID**: INTEGER, **DEP_ID**: INTEGER)

Staff (STAFF_ID: INTEGER, name: STRING, status: STRING)

Foreign Key (FK)

- Key **staff_id** represents Staff(STAFF_ID)

Relation [Has] (1:M)

Clinical_Activity (CAID: INTEGER, time: TIME, date: DATE, **staff_id**: INTEGER, **IID**: INTEGER, **ExID**: INTEGER, **DEP_ID**: INTEGER)

Patient (IID: INTEGER, CIN: STRING, name: STRING, sex: CHAR(1), birth: DATE, blood_group: STRING, phone: STRING)

Foreign Key (FK)

- Key **IID** represents Patient(IID)

Relation [Generates] (1:M)

Clinical_Activity (CAID: INTEGER, time: TIME, date: DATE, **staff_id**: INTEGER, **IID**: INTEGER, **ExID**: INTEGER, **DEP_ID**: INTEGER)

Expense_Attached (ExID: INTEGER, total: DECIMAL, **InsID**: INTEGER, Note: STRING)

Foreign Key (FK)

- Key **ExID** represents Expense_Attached(ExID)

Relation [Occurs] (1:M)

Clinical_Activity (CAID: INTEGER, time: TIME, date: DATE, **staff_id**: INTEGER, **IID**: INTEGER, **ExID**: INTEGER, **DEP_ID**: INTEGER)

Department (DEP_ID: INTEGER, name: STRING, specialty: STRING)

UM6P
University
Mohammed VI
Polytechnic

College of
Computing

Foreign Key (FK)

- Key **DEP_ID** represents Department(DEP_ID)

ISA [Clinical_Activity] (1:1 Total Specialization)

Appointment (<u>**CAID**</u>: INTEGER, reason: STRING, status: STRING)

Emergency (<u>**CAID**</u>: INTEGER, triage_level: INTEGER, outcome: STRING)

Foreign Key (FK)

- Key **CAID** represents Clinical_Activity(CAID)

Relation [Generate] (1:M)

Clinical_Activity (<u>CAID</u>: INTEGER, time: TIME, date: DATE, **staff_id**: INTEGER, **IID**: INTEGER, **ExID**: INTEGER, **DEP_ID**: INTEGER)

Prescription (<u>PID</u>: INTEGER, DateIssued: DATE, **CAID**: INTEGER)
   Note: The 1:M relationship indicates that each Prescription is generated by exactly one Clinical_Activity, and a Clinical_Activity can have 0 or 1 Prescription.

Foreign Key (FK)

- Key **CAID** represents Clinical_Activity(CAID)

Relation [Include] (N:M)

Prescription (<u>PID</u>: INTEGER, DateIssued: DATE, CAID: INTEGER)

Medication (<u>drug_id</u>: INTEGER, class: STRING, name: STRING, form: STRING, strength: STRING, active_ingredient: STRING, manufacturer: STRING)

Include (<u>**PID**</u>: INTEGER, <u>**drug_id**</u>: INTEGER, dosage: STRING, quantity: INTEGER, frequency: STRING)
   Note: Since this is an N:M relationship, a new relation (Include) is created. It inherits the primary keys from both related entities, which together form its composite primary key.

Foreign Key (FK)

- Key **PID** represents Prescription(PID)

- Key **drug_id** represents Medication(drug_id)

Composite Key   Include (PID, drug_id): Relationship of type N:M between Prescription and Medication.

Relation [Stock] (N:M)

Hospital (<u>HID</u>: INTEGER, name: STRING, city: STRING, region: STRING)

Medication (<u>drug_id</u>: INTEGER, class: STRING, name: STRING, form: STRING, strength: STRING, active_ingredient: STRING, manufacturer: STRING)

Stock (**<u>HID</u>**: INTEGER, **<u>drug_id</u>**: INTEGER, unit_price: DECIMAL, stock_timestamp: TIMESTAMP, qty: INTEGER, reorder_level: INTEGER)
   Note: Since this is an N:M relationship, a new relation (Stock) is created. It inherits the primary keys from both related entities, which together form its composite primary key.

Foreign Key (FK)

- Key **HID** represents Hospital(HID)

- Key **drug_id** represents Medication(drug_id)

Composite Key   Stock (HID, drug_id): Relationship of type N:M between Hospital and Medication.

Relation [Belongs] (1:M)

Hospital (<u>HID</u>: INTEGER, name: STRING, city: STRING, region: STRING)

Department (<u>DEP_ID</u>: INTEGER, name: STRING, specialty: STRING, **HID**: INTEGER)
   Note: The constraint "each hospital has at least one department" is a participation constraint enforced by application logic or database triggers.

Foreign Key (FK)

- Key **HID** represents Hospital(HID)

3   Implementation & Results

Code SQL:

```
CREATE DATABASE MNHS;
USE MNHS;

CREATE TABLE Patients (
    iid INT PRIMARY KEY AUTO_INCREMENT,
    cin VARCHAR(50),
```

UM6P
University
Mohammed VI
Polytechnic

College of
Computing

```
    p_name VARCHAR(50),
    sex VARCHAR(10),
    birth DATE,
    blood_group VARCHAR(3),
    phone VARCHAR(15)
);

CREATE TABLE Departement (
    did INT PRIMARY KEY AUTO_INCREMENT,
    d_name VARCHAR(50),
    specialty VARCHAR(50)
);

CREATE TABLE Clinical_Activities (
    caid INT PRIMARY KEY AUTO_INCREMENT,
    ca_time TIME,
    ca_date DATE,
    iid INT NOT NULL,
    did INT NOT NULL,
    FOREIGN KEY (iid) REFERENCES Patients(iid),
    FOREIGN KEY (did) REFERENCES Departement(did)
);

CREATE TABLE Appointment (
    caid INT PRIMARY KEY AUTO_INCREMENT,
    reason VARCHAR(50),
    a_status VARCHAR(100),
    FOREIGN KEY (caid) REFERENCES Clinical_Activities(caid)
);

CREATE TABLE Emergency (
    caid INT PRIMARY KEY AUTO_INCREMENT,
    triage_level VARCHAR(100),
    outcome VARCHAR(100),
    FOREIGN KEY (caid) REFERENCES Clinical_Activities(caid)
);

CREATE TABLE Hospital (
    hid INT PRIMARY KEY AUTO_INCREMENT,
    h_name VARCHAR(100),
    city VARCHAR(50),
    region VARCHAR(50),
    did INT NOT NULL,
    FOREIGN KEY (did) REFERENCES Departement(did)
);

INSERT INTO Patients (cin, p_name, sex, birth, blood_group, phone) VALUES
    ('CD908479', 'Youssef', 'Male', '2000-05-15', 'A+', '1234567890'),
```

```
    ('AB123456', 'Sara', 'Female', '1995-08-22', 'B-', '0987654321');

INSERT INTO Departement (d_name, specialty) VALUES
    ('Cardiology', 'Heart-related treatments'),
    ('Neurology', 'Brain and nervous system treatments');

INSERT INTO Clinical_Activities (ca_time, ca_date, iid, did) VALUES
    ('10:00:00', '2023-10-01', 1, 1),
    ('14:30:00', '2023-10-02', 2, 2);

INSERT INTO Appointment (caid, reason, a_status) VALUES
    (1, 'Routine checkup', 'Scheduled'),
    (2, 'Specialist consultation', 'Completed');

INSERT INTO Emergency (caid, triage_level, outcome) VALUES
    (1, 'Level 3', 'Admitted'),
    (2, 'Level 1', 'Discharged');



INSERT INTO Hospital (h_name, city, region, did) VALUES
    ('UM6P Hospital', 'Benguerir', 'Rehamna', 1),
    ('Moulay Ismail', 'Meknès', 'Fès-Meknès', 2);

SELECT p_name FROM Patients
    JOIN Clinical_Activities ON Patients.iid = Clinical_Activities.iid
    JOIN Appointment ON Clinical_Activities.caid = Appointment.caid
    JOIN Hospital ON Clinical_Activities.did = Hospital.did
    WHERE Appointment.a_status = 'Scheduled' AND Hospital.city = 'Benguerir';
```
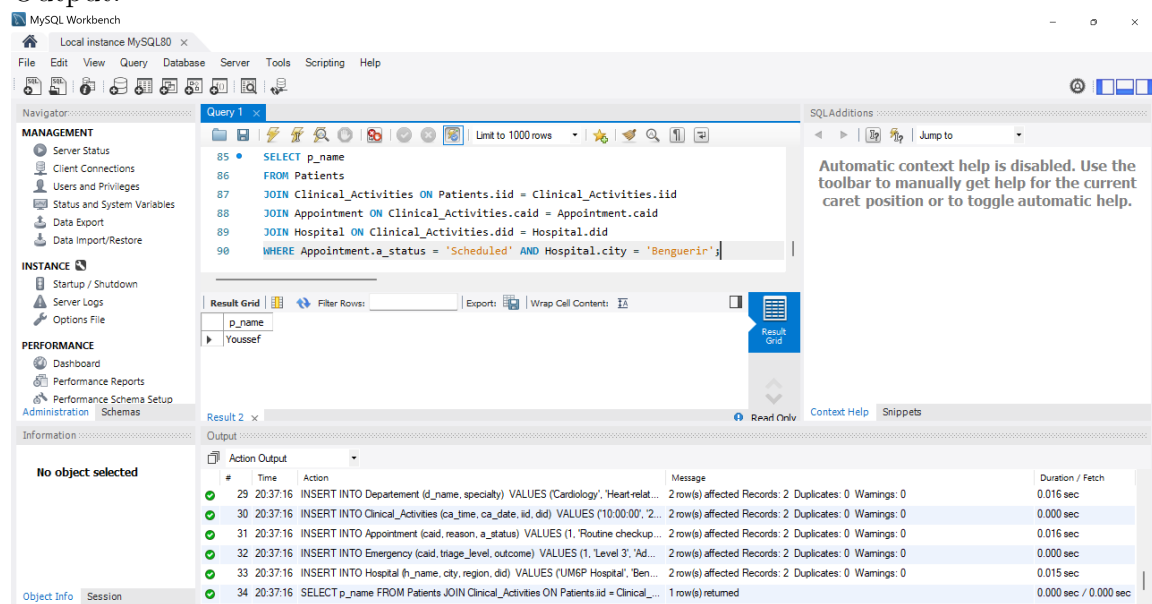
Output:

UM6P
University
Mohammed VI
Polytechnic

College of
Computing

4 Discussion

- **Challenges faced:**

  - Turning the ER diagram into relational tables was sometimes tricky, especially when deciding how to represent N:M relationships and which attributes should go where.

  - Mapping ISA hierarchies like *Staff* and *Clinical Activity* into tables needed careful thought to make sure no data was lost and that foreign keys were correct.

  - Some constraints from the conceptual design, like "each hospital must have at least one department," cannot be fully represented in the tables, so we had to think about how they would be handled in the actual database.

- **Observations:**

  - This step helped clarify parts of the ER diagram that were ambiguous. Translating everything into tables forced us to think more carefully about keys and relationships.

  - We noticed that some rules from the real world cannot always be fully enforced in the schema itself and will need extra checks in the application or with database triggers.

- **Lessons learned:**

  - Understanding the relationships between entities is key before creating tables.

  - Choosing the right primary keys, foreign keys, and composite keys is really important for keeping the data correct.

  - ISA hierarchies require careful planning so that the design stays efficient and normalized.

5 Conclusion

In this lab, we successfully converted the conceptual ER model of the **Moroccan National Health Services (MNHS)** into a logical database schema. We defined all the tables, attributes, keys, and handled N:M relationships and ISA hierarchies. This logical design gives us a clear plan for building the database, keeping data organized and consistent. Overall, this lab showed how important logical design is in turning ideas from an ER diagram into something we can actually implement in a database.