University
Mohammed VI
Polytechnic

College of Computing

Deliverable 2: Relational Schema of the Moroccan National Health Service Data Management System

Data Management Course

UM6P College of Computing

Professor: Karima Echihabi **Program:** Computer Engineering

Session: Fall 2025

Team Information

Team Name	alfari9 alkhari9
Member 1	Ilyas Rahmouni
Member 2	Malak Koulat
Member 3	Aymane Raiss
Member 4	Zakaria Harira
Member 5	Youness Latif
Member 6	Rayane Khaldi
Member 7	Younes Lougnidi
Repository Link	`https://github.com/...`

---

1 Logical Desgin Tasks 1 : Attributes, Primary Keys and Justification of Composite Keys

Patient (IID: int primary key)
Attributes: IID (int), Birth (date), CIN (varchar(50) unique, not null), Name (varchar(100), not null), Sex (enum('Male','Female')), Phone (int), Blood_Group (enum('A+','A-','B+','B-','O-','O+','AB+','AB-'))

Contact_Location (CLID: int primary key)
Attributes: CLID (int), Province (varchar(100)), Street (varchar(200)), Number (int), Postal_Code (int), Phone (int), City (varchar(100))

Have ((IID, CLID): composite primary key)
Attributes: IID (int), CLID (int)
Justification: Composite key ensures that each patient can have multiple contact locations and each contact location can belong to multiple patients.

Staff (STAFF_ID: int primary key)
Attributes: STAFF_ID (int), Name (varchar(100)), Status (varchar(50))

Practitioner (STAFF_ID: int primary key) — ISA relationship with Staff
Attributes: STAFF_ID (int), License_Number (varchar(100)), Speciality (varchar(200))

Caregiving (STAFF_ID: int primary key) — ISA relationship with Staff
Attributes: STAFF_ID (int), Grade (varchar(50)), Ward (varchar(100))

Technical (STAFF_ID: int primary key) — ISA relationship with Staff
Attributes: STAFF_ID (int), Certifications (varchar(500)), Modality (varchar(200))

Hospital (HID: int primary key)
Attributes: HID (int), Name (varchar(200)), City (varchar(200)), Region (varchar(200))

Department (DEP_ID: int primary key)
Attributes: DEP_ID (int), Name (varchar(200)), Speciality (varchar(200)), HID (int)

WorkIn ((STAFF_ID, DEP_ID): composite primary key)
Attributes: STAFF_ID (int), DEP_ID (int)
Justification: Composite key ensures that a staff member can work in multiple departments and a department can have multiple staff members.

Medication (DrugID: int primary key)
Attributes: DrugID (int), Class (varchar(200)), Name (varchar(200)), Form (varchar(200)), Strength (varchar(50)), Manufacturer (varchar(200)), ActiveIngredient (varchar(100))

Stock ((DrugID, HID): composite primary key)
Attributes: DrugID (int), HID (int), Unit_Price (int), StockTimestamp (timestamp), Qty (int), ReorderLevel (int)
Justification: Composite key ensures that each hospital can stock multiple drugs and each drug can exist in multiple hospitals.

Prescription (PID: int primary key)
Attributes: PID (int), DateIssued (date), CAID (int)

Include ((PID, DrugID): composite primary key)
Attributes: PID (int), DrugID (int), Dosage (varchar(100)), Duration (varchar(100))
Justification: Composite key ensures that each prescription can include multiple drugs and each drug can appear in multiple prescriptions.

Clinical_Activity (CAID: int primary key)
Attributes: CAID (int), Time (time), Date (date), DEP_ID (int), IID (int), STAFF_ID (int)

Generates (CAID: int primary key, ExID: int unique)
Attributes: CAID (int), ExID (int)
Justification: Each clinical activity generates exactly one expense and each expense is generated by exactly one clinical activity.

Expense (ExID: int primary key)
Attributes: ExID (int), Total (varchar(50)), InsID (int)

Appointment (CAID: int primary key) — ISA relationship with Clinical_Activity
Attributes: CAID (int), Reason (varchar(500)), Status (varchar(500))

Emergency (CAID: int primary key) — ISA relationship with Clinical_Activity
Attributes: CAID (int), TriageLevel (varchar(500)), Outcome (varchar(500))

Insurance (InsID: int primary key)
Attributes: InsID (int), Type (varchar(500))

Covers ((IID, InsID): composite primary key)
Attributes: IID (int), InsID (int)
Justification: Composite key ensures that one patient can have multiple insurances and one insurance can cover multiple patients.

2 Logical Design Task 2 : Foreign Keys, Participation and Domain Checks

Have
Foreign keys: IID → Patient(IID), CLID → Contact_Location(CLID)
Participation: Many-to-Many (both partial)

Practitioner, Caregiving, Technical — ISA with Staff
Foreign key: STAFF_ID → Staff(STAFF_ID)
Participation: Total (each practitioner/caregiver/technician must be a staff member)

Department
Foreign key: HID → Hospital(HID)
Participation: Each department belongs to exactly one hospital (total on Department side)

WorkIn
Foreign keys: STAFF_ID → Staff(STAFF_ID), DEP_ID → Department(DEP_ID)
Participation: Many-to-Many (both partial)

Explanation :
In our schema, the participation of Staff in the WorkIn relationship is meant to be total, meaning every staff member should belong to at least one department. However, this constraint cannot be strictly enforced using the basic SQL mechanisms we have studied, since adding a staff member does not automatically create a corresponding entry in WorkIn. Ensuring total participation would require more advanced techniques such as triggers (to automatically insert or validate records), stored procedures (to handle insertions in both tables together), application-level checks (enforcing the rule in the program's logic), or database assertions (theoretical SQL constraints not supported in most systems).

Stock
Foreign keys: DrugID → Medication(DrugID), HID → Hospital(HID)
Participation: Many-to-Many (both partial)

Prescription
Foreign key: CAID → Clinical_Activity(CAID)
Participation: Total (each prescription must be linked to a clinical activity)

Include
Foreign keys: DrugID → Medication(DrugID), PID → Prescription(PID)
Participation: Many-to-Many (both partial)

Clinical_Activity
Foreign keys: DEP_ID → Department(DEP_ID), IID → Patient(IID), STAFF_ID → Staff(STAFF_ID)
Participation: Each clinical activity must involve a patient, a staff member, and a department (total)

Generates
Foreign keys: CAID → Clinical_Activity(CAID), ExID → Expense(ExID)
Participation: One-to-One (each clinical activity generates one expense and each expense is generated by exactly one clinical activity)

Expense
Foreign key: InsID → Insurance(InsID)
Participation: Partial (not every expense must be insured)

Solution 1 :(above)
The First solution is to introduce a separate relationship table, for example Generates, which contains foreign keys referencing both Expense and Clinical Activity. One of these foreign keys is also declared as the primary key of the relationship table to enforce total participation for that specific entity. This design provides a more modular structure and can easily accommodate additional attributes specific to the relationship itself. However, it still does not guarantee total participation for both entities simultaneously—only for the one whose key serves as the primary key.

Solution 2 :
In the Second solution, the primary key of the Clinical Activity entity is declared as a foreign key in the Expense table. This ensures a one-to-one relationship, where each Expense must correspond to exactly one Clinical Activity, and no clinical activity can have more than one expense. The ON DELETE CASCADE option can be used to maintain referential integrity, so that deleting a clinical activity automatically deletes the associated expense. While this design effectively enforces a total participation of Expense in the relationship, it only guarantees partial participation on the side of Clinical Activity—it is still possible for a clinical activity to exist without an expense.

Expense Will Become :

Expense (ExID: int primary key, CAID : int)
Attributes: CAID (int), ExID (int), Total (varchar(50)), InsID (int)
Foreign key: CAID → Clinical_Activity(CAID)
Participation: Total

Appointment / Emergency — ISA with Clinical_Activity
Foreign key: CAID → Clinical_Activity(CAID)
Participation: One-to-One (each clinical activity can either be an appointment or emergency)

Covers
Foreign keys: IID → Patient(IID), InsID → Insurance(InsID)
Participation: Many-to-Many (both partial)

Domain Checks

- Sex: ENUM('Male', 'Female')
- Blood_Group: ENUM('A+', 'A-', 'B+', 'B-', 'O-', 'O+', 'AB+', 'AB-')
- Phone, Postal_Code, Number, Qty, ReorderLevel, Unit_Price: must be positive integers
- Date, Time: valid SQL date/time format
- Status: varchar(50) (domain restricted by business rules if needed)

3 Logical Design Task 3 : Implementation of part of the schema in SQL

3.1 Creating the tables

```
CREATE TABLE Patient(
    IID INT PRIMARY KEY,
    Birth DATE,
    CIN VARCHAR(50) UNIQUE NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Sex ENUM('Male','Female'),
    Phone INT,
    Blood_Group ENUM('A+','A-','B+','B-','O-','O+','AB+','AB-')
);
```

Listing 1: Create Patient Table

```
CREATE TABLE Hospital (
    HID INT PRIMARY KEY,
    Name VARCHAR(200),
    City VARCHAR(200),
    Region VARCHAR(200)
);
```

Listing 2: Create Hospital Table

```
CREATE TABLE Department(
    DEP_ID INT PRIMARY KEY,
    Name VARCHAR(200),
    speciality VARCHAR(200),
    HID INT NOT NULL,
    FOREIGN KEY(HID) REFERENCES hospital(HID) ON DELETE NO ACTION
);
```

Listing 3: Create Departement Table

```
CREATE TABLE staff(
    STAFF_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    status VARCHAR(50)
);
```

Listing 4: Create Staff Table

```sql
CREATE TABLE clinical_activity(
    CAID INT PRIMARY KEY,
    Time TIME,
    Date DATE,
    DEP_ID INT NOT NULL,
    IID INT NOT NULL,
    Staff_ID INT NOT NULL,
    FOREIGN KEY (DEP_ID) REFERENCES Department(DEP_ID) ON DELETE
       NO ACTION,
    FOREIGN KEY (IID) REFERENCES patient(IID) ON DELETE NO ACTION
       ,
    FOREIGN KEY (Staff_ID) REFERENCES staff(Staff_ID) ON DELETE
       NO ACTION
);
```

Listing 5: Create Clinical Activity Table

```sql
CREATE TABLE Appointment(
    CAID INT PRIMARY KEY,
    Reason VARCHAR(500),
    Status VARCHAR(500),
    FOREIGN KEY (CAID) REFERENCES Clinical_Activity(CAID) ON
       DELETE CASCADE
);
```

Listing 6: Create Appointement Table

3.2 Inserting Tuples

```sql
INSERT INTO hospital (HID, Name, City, region) VALUES
(1, 'Hopital de Benguerir', 'Benguerir', 'Marrakech-Safi'),
(2, 'Hopital de Casablanca', 'Casablanca', 'Casablanca-Settat');
```

Listing 7: Insert records into the `hospital` table.

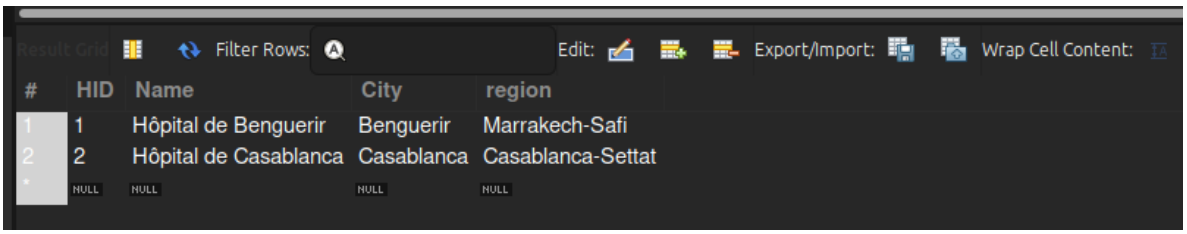

Figure 1: Hospital's table

```sql
INSERT INTO patient (IID, Birth, CIN, name, sex, phone,
   Blood_group) VALUES
(101, '1990-05-15', 'BK123456', 'Ahmed El Idrissi', 'Male',
   0611223344, 'A+'),
(102, '1985-09-22', 'BJ654321', 'Fatima Alami', 'Female',
   0655667788, 'O-');
```

Listing 8: Insert records into the `patient` table.

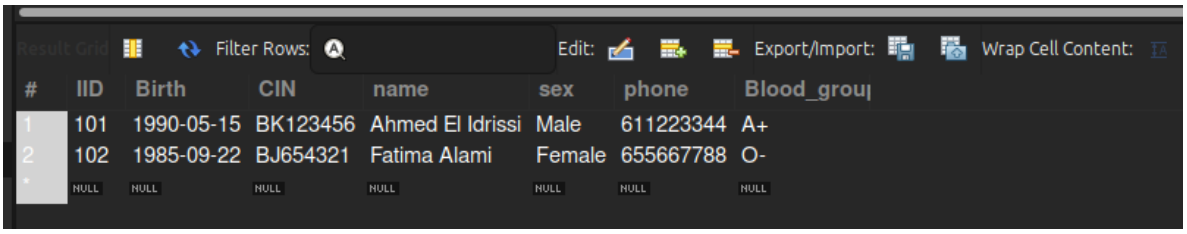

Figure 2: Patients's table

```sql
INSERT INTO staff (STAFF_ID, Name, status) VALUES
(1, 'Dr. Ali Bennani', 'Active');
```

Listing 9: Insert a record into the `staff` table.

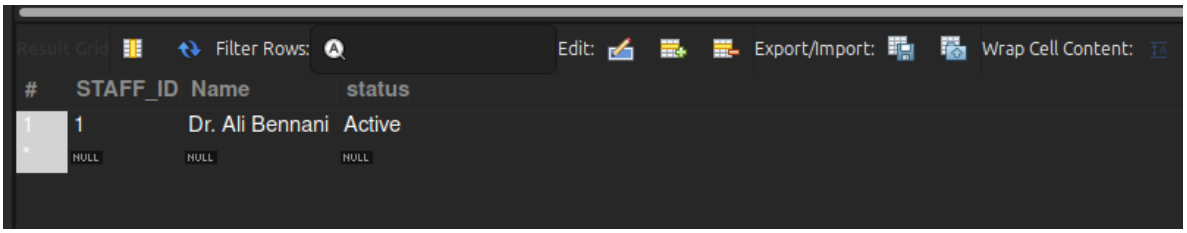

Figure 3: Staff's table

```sql
INSERT INTO Department (DEP_ID, Name, speciality, HID) VALUES
(1, 'Cardiology', 'Cardiology', 1),
(2, 'Neurology', 'Neurology', 2);
```

Listing 10: Insert records into the `Department` table.

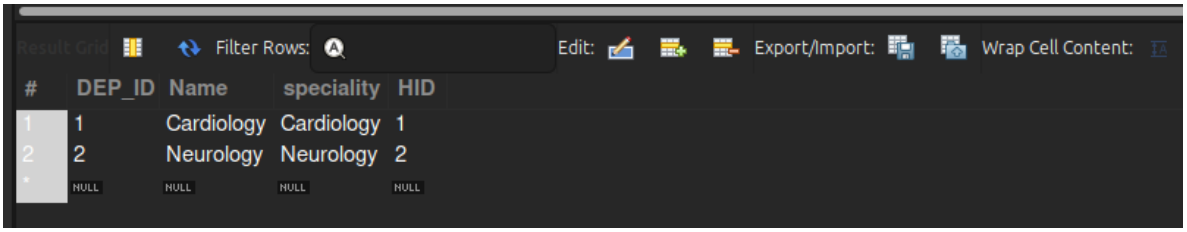

Figure 4: Departement's table

```
INSERT INTO clinical_activity (CAID, Time, Date, DEP_ID, IID,
   Staff_ID) VALUES
(1, '09:00:00', '2025-10-20', 1, 101, 1),
(2, '10:30:00', '2025-10-20', 2, 102, 1);
```

Listing 11: Insert records into the `clinical_activity` table.

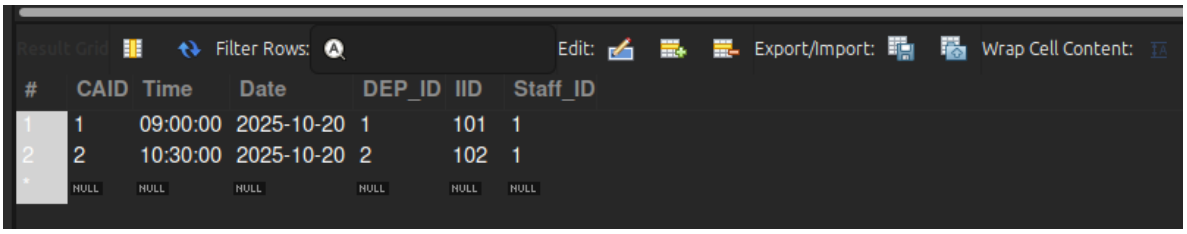

Figure 5: Clinical Activity's table

```
INSERT INTO Appointment (CAID, reason, status) VALUES
(1, 'Consultation', 'Scheduled'),
(2, 'Consultation', 'Scheduled');
```

Listing 12: Insert records into the `Appointment` table.

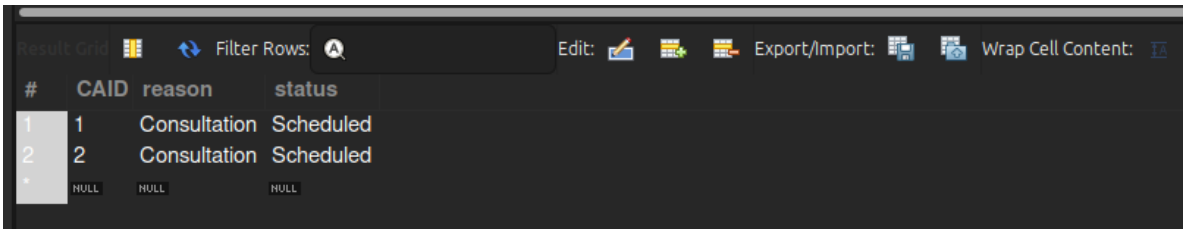

Figure 6: Appointements's table

3.3 Retrieving Patients with appointements in Benguerir

```sql
SELECT p.*
FROM MNHS.patient p
JOIN clinical_activity ca ON p.IID = ca.IID
JOIN Appointment a ON ca.CAID = a.CAID
JOIN Department d ON ca.DEP_ID = d.DEP_ID
JOIN hospital h ON d.HID = h.HID
WHERE h.City = 'Benguerir';
```

Listing 13: Query to retrieve patients with appointments in Benguerir.

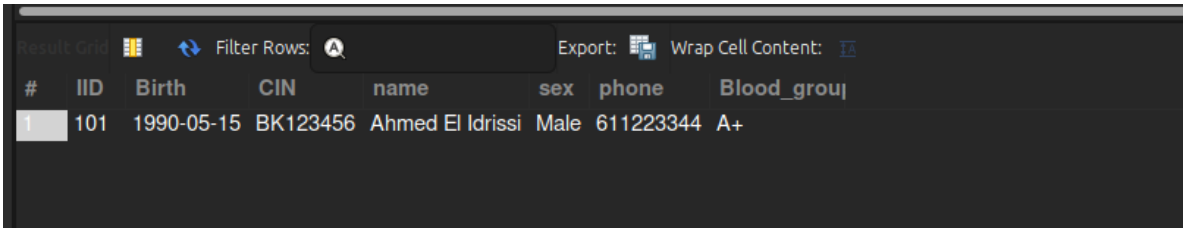

Figure 7: Query's output