Deliverable #: Conceptual Design

Data Management Course

UM6P College of Computing

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

Team Information

Team Name	QueryMaster
Member 1	El Mehdi Regagui
Member 2	Yasser Jarboua
Member 3	Adam Ibourg-EL Idrissi
Member 4	Salma Mana
Member 5	Hiba Mhirit
Member 6	Sara Qiouame
Member 7	Douaae Mabrouk
Repository Link	`https://github.com/yasserJarboua/QueryMasters/`

---

1 Introduction

The document presents the next step on designing and implementing the MNHS DataBase, the deliverable has two main part:
Firstly, we represented the relational algebra expressions for some queries to access different data on the database in an efficient way, also the SQL code written on MySQL DBMS for those specific queries. **Secondly,** we discussed and identified the functional dependencies (FDs) in the MNHS Schema.

2 Requirements

The core objectives of the deliverable are:

1. Express non-trivial queries over MNHS using Relational Algebra (RA).
2. Translate the same queries into SQL.
3. Identify and state functional dependencies (FDs) in the MNHS schema.

3 Methodology

Our approach was built upon the foundational steps previously established, including the conceptual design and relational model. These preliminary phases enabled us to accurately identify entities and relationships, reflecting real-world practices and constraints. The relational model phase focused on defining table structures, emphasizing links between tables through foreign keys. Based on query requirements, we performed various relational algebra operations to ensure the schema supports expected use cases.

In the refinement stage, we systematically extracted functional dependencies for each relation. This process considered both explicit constraints, such as primary keys and foreign key relationships, and implicit semantic rules. We also acknowledged trivial functional dependencies, where each attribute functionally determines itself, in accordance with Armstrong's reflexivity axiom.

3.1 Part 1 - Relational Algebra For the Specific Queries

1. **Find the names of patients who have had at least one clinical activity handled by active staff.**

   In order to find the names of patients who have at least one Clinical Activity handled by active staff we need to make a natural join between the table where staff are Active and Clinical Activity and then another natural join with patient.

$$\pi_{name}(Patient \bowtie ClinicalActivity \bowtie \sigma_{Staff\_ID='Active'}(Staff))$$

2. **Find Staff IDs of staff who are either 'Active' or have issued at least one prescription.**

$$\pi_{STAFF\_ID}(\sigma_{Status='Active'}(Staff)) \cup \pi_{STAFF\_ID}(Prescription \bowtie ClinicalActivity)$$

3. **Find Hospital IDs of hospitals located in 'Benguerir' or having at least one department with the specialty 'Cardiology'.**

$$\pi_{HID}(\sigma_{City='Benguerir'}(Hospital)) \cup \pi_{HID}(\sigma_{Name='Cardiology'}(Department))$$

4. **Find Hospital IDs of hospitals that have both 'Cardiology' and 'Pediatrics' departments.**

$$\pi_{HID}(\sigma_{Name='Cardiology'}(Department)) \cap \pi_{HID}(\sigma_{Name='Pediatrics'}(Department))$$

5. **Find staff members who have worked in every department of the hospital with HID = 1.**

$$\text{Let } P = \pi_{Staff\_ID}(Work\_In)/\pi_{DEP\_ID}(\sigma_{HID=1}(Department))$$

So the final relational algebra expressing the query:

$$Staff \bowtie P$$

6. **Find staff members who participated in every clinical activity of the department with DEP_ID = 2.**

We need to select firstly the Clinical activity where DEP_ID = 2, and then we project the result to CAID column in order to perform a Division operation between Clinical Activity and the Column containing CAID and The table joining Staff and Clinical Activity.

$$\text{Let } P = Staff \bowtie ClinicalActivity$$

$$P/\pi_{CAID}(\sigma_{DEP\_ID=2}(ClinicalActivity))$$

7. **Find pairs of staff members (s_1, s_2) such that s_1 has handled more clinical activities than s_2.**

Firstly, we need to group clinical activities by Staff_ID and then apply the count method to know the number of Clinical Activities that had been done by each staff:

$$R_1 = \pi_{Staff\_ID,Total\_CL}(\gamma_{\text{Staff\_ID}:\, count(*) \to Total\_CL} ClinicalActivity)$$

Then we made a Theta-Join between R_1 and itself just we need to rename the Staff_ID and Total_CL columns:

$$\pi_{Staff\_ID,Staff\_ID_2}(\rho(R_2(1 \rightarrow Staff\_ID_2, 2 \rightarrow Total\_CL_2)\,,\, R_1 \bowtie R_2))$$

8. **Find Patient IDs of patients who had clinical activities with at least two different staff members.**

$$\rho(cla1(IID, STAFF_ID)\,,\, ClinicalActivity)$$

$$ClinicalActivity \bowtie_{ClinicalActivity.IID=cla1.IID \wedge ClinicalActivity.STAFF\_ID \neq cla1.STAFF\_ID}\ cla1$$

Or, we can adopt another way using the Group by operator in order to make it easier for reading:

$$\pi_{IID}(\sigma_{DSID \geq 2}(\gamma_{\text{IID}: Count(Distinct\ \ SID) \rightarrow DSID} ClinicalActivity))$$

9. **Find CAIDs of clinical activities performed in September 2025 at hospitals located in "Benguerir".**

By joining the Department and Hospitals where $city =' Benguerir'$ we got all the Department on Benguerir city then we Join them with Clinical Activity that have been done in September 2025:

$$\text{Let } CL = \sigma_{\text{2025-09-01} \leq date \leq \text{2025-09-30}}(ClinicalActivity)$$

$$\pi_{CID}(CL \bowtie Department \bowtie \sigma_{city='Benguerir'}(Hospital))$$

10. **Find Staff IDs of staff who have issued more than one prescription.**

We adopted the Join on this query (a Group by method could also work, it's the same idea as the query number 8). Firstly we Joined Clinical Activity with Prescription.

$$\text{Let } SP = ClinicalActivity \bowtie Prescription$$

$$\rho(SP1(PID, STAFF_ID)\ , SP)$$

$$\pi_{STAFF_ID}(SP \bowtie_{SP.STAFF_ID=SP1.STAFF_ID \wedge SP.PID \neq SP1.PID} SP1)$$

11. **List IIDs of patients who have scheduled appointments in more than one department.**

$$\text{Let } R = ClinicalActivity \bowtie \sigma_{Status='Scheduled'}(Appointment)$$

So the final Query will be:

$$\gamma_{\text{IID}:\, Count(Distinct\ \ DEP\_ID) \rightarrow num} R$$

12. **Find Staff IDs who have no scheduled appointments on the day of the Green March holiday (November 6).**

$$\pi_{Staff\_ID}(\sigma_{Date=\text{09-30}}(ClinicalActivities) \bowtie \sigma_{Status \neq 'Scheduled'}(Appointment))$$

13. **Find departments whose average number of clinical activities is below the global departmental average.**

$$Let\ R1 = \gamma_{\text{Dep\_ID}:\, Count(*) \rightarrow num} ClinicalActivity$$

$$\pi_{Dep\_ID}(\sigma_{avg > num}(R1 \times \gamma_{\text{Dep\_ID}:\, Average(num) \rightarrow avg} R1))$$

14. **For each staff member, return the patient who has the greatest number of completed appointments with that staff member.**

Firstly, we select Appointment where status = 'Completed':

$$\text{Let } R_1 = \sigma_{Status='Completed'}(Appointment)$$

Then we group R_1 by Staff_ID, IID and then we perform the count on each group we store the result on num column:

$$R_2 = \gamma_{\text{Staff\_ID, IID}:\, count(*)=num} R_1$$

We regroup by Staff_ID and calculate the max num on R_2:

$$R_3 = \gamma_{\text{Staff\_ID}:\, max(num)=max\_num} R_2$$

And we Join R_2 and R_3 using a Theta-Join to have the final table:

$$R_4 = R_2 \bowtie_{R_2.Staff\_ID=R_3.STAFF\_ID \wedge num=max\_num} R_3$$

15. **List patients who had at least 3 emergency admissions during the year 2024.**

$$\text{Let } R_1 = \sigma_{Date=2024}(ClinicalActivities) \bowtie \sigma_{outcome='Admitted'}(Emergency)$$

$$R_2 = \sigma_{ADD \geq 3}(\gamma_{\text{IID}:\, Count(*) \to ADD} R_1)$$

So the final expression to list patients (i.e all patient tuples and not only names) who had at least 3 emergency admissions during the year 2024:

$$\pi_{name}(R_2) \bowtie Patient$$

3.2 Part2 - Functional Dependencies by Entity and Relation

Entities

Note: For each entity in the schema, the primary key functionally determines all other attributes within that entity. This follows the fundamental principle of relational database design where primary keys uniquely identify each tuple. Additionally, all attributes exhibit trivial functional dependencies by determining themselves. These foundational dependencies are inherent to the relational model and apply universally across all entities.

Patient:

IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
CIN → FullName, Birth, Sex, BloodGroup, Phone

Hospital:

HID → Name, City, Region
City → Region

Department:

DEP_ID → HID, Name, Specialty
Given: DEP_ID → HID and HID → Name, City, Region
Derived: DEP_ID → Name, City, Region (Hospital attributes)

Staff:

STAFF_ID → FullName, Status
STAFF_ID → licenceNumber, Speciality (if Practitioner)
STAFF_ID → Grade, Ward (if Caregiving)
STAFF_ID → Certifications, Modality (if Technical)

ClinicalActivity:

CAID → IID, STAFF_ID, DEP_ID, Date, Time
CAID → (Reason, Status) if Appointment
CAID → (TriageLevel, Outcome) if Emergency

Given: CAID → IID and IID → CIN, Name, Sex, Birth, BloodGroup, Phone
Derived: CAID → CIN, Name, Sex, Birth, BloodGroup, Phone (Patient attributes)

Given: CAID → DEP_ID and DEP_ID → Name, Specialty, HID
Derived: CAID → Name, Specialty, HID (Department and Hospital attributes)

Given: CAID → STAFF_ID and STAFF_ID → FullName, Status
Derived: CAID → FullName, Status

Prescription:

PID → CAID, DateIssued
CAID → PID

Given: PID → CAID and CAID → IID, DEP_ID, Date, Time
Derived: PID → IID, DEP_ID, Date, Time

Given: PID → IID and IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
Derived: PID → CIN, FullName, Birth, Sex, BloodGroup, Phone

Medication:

DrugID → Name, Form, Strength, Manufacturer, Class, ActiveIngredient

Note: The functional dependency Name → Class does not hold, as evidenced by common medications like 'Aspirin' which can belong to different therapeutic classes.

ContactLocation:

CLID → City, Province, Street, Number, PostalCode, Phone
City → Region (as each Moroccan city belongs to exactly one administrative region)

Insurance:

InsID → Type

Expense:

ExpID → InsID, CAID, Total

Given: ExID → CAID and CAID → IID, DEP_ID, Date, Time
Derived: ExID → IID, DEP_ID, Date, Time

Given: ExID → IID and IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
Derived: ExID → CIN, FullName, Birth, Sex, BloodGroup, Phone

Relationships

Stock (Hospital - Medication)

(HID, MID, StockTimestamp) → UnitPrice, Qty, ReorderLevel

Have (Patient - ContactLocation)

(IID, CLID) → CLID, IID

Given CLID → City, Province, Street, Number, PostalCode, Phone and (IID, CLID) → CLID
Derived (IID, CLID) → City, Province, Street, Number, PostalCode, Phone

Given (IID, CLID) → IID and IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
Derived (IID, CLID) → CIN, FullName, Birth, Sex, BloodGroup, Phone

Covers (Insurance - Patient)

(InsID, IID) → IID, InsID

Given (InsID, CLID) → IID and IID → CIN, FullName, Birth, Sex, BloodGroup, Phone
Then (InsID, IID) → CIN, FullName, Birth, Sex, BloodGroup, Phone

Given (InsID, CLID) → InsID and InsID → Type
Then (InsID, CLID) → Type

Work_in (Staff - Department)

(STAFF_ID, DEP_ID) → DEP_ID, STAFF_ID

Belongs (Department - Hospital)

DEP_ID → HID (because each department belongs to exactly one hospital)

Include (Prescription - Medication)

(PID, Drug_ID) → Duration, Dosage

Occurs (ClinicalActivity - Department)

CAID → DEP_ID (each activity occurs in exactly one department)

Has (Patient - ClinicalActivity)

CAID → IID (each activity belongs to exactly one patient)

Linked (Staff - ClinicalActivity)

CAID → STAFF_ID (each activity is linked to exactly one staff member)

Complete List of Functional Dependencies

FD = {IID → CIN, FullName, Birth, Sex, BloodGroup, Phone,

CIN → FullName, Birth, Sex, BloodGroup, Phone,

HID → Name, City, Region,

City → Region,

DEP_ID → HID, Name, Specialty,

DEP_ID → Name, City, Region,

STAFF_ID → FullName, Status,

STAFF_ID → licenceNumber, Speciality,

STAFF_ID → Grade, Ward,

STAFF_ID → Certifications, Modality,

CAID → IID, STAFF_ID, DEP_ID, Date, Time,

CAID → Reason, Status,

CAID → TriageLevel, Outcome,

CAID → CIN, Name, Sex, Birth, BloodGroup, Phone,

CAID → Name, Specialty, HID,

CAID → FullName, Status,

PID → CAID, DateIssued,

CAID → PID,

PID → IID, DEP_ID, Date, Time,

PID → CIN, FullName, Birth, Sex, BloodGroup, Phone,

DrugID → Name, Form, Strength, Manufacturer, Class, ActiveIngredient,

CLID → City, Province, Street, Number, PostalCode, Phone,

City → Region,

InsID → Type,

ExpID → InsID, CAID, Total,

ExpID → IID, DEP_ID, Date, Time,

ExpID → CIN, FullName, Birth, Sex, BloodGroup, Phone, (HID, MID, StockTimestamp) → UnitPrice, Qty, ReorderLevel,

(IID, CLID) → CLID, IID,

(IID, CLID) → City, Province, Street, Number, PostalCode, Phone,

(IID, CLID) → CIN, FullName, Birth, Sex, BloodGroup, Phone,

(InsID, IID) → IID, InsID,

(InsID, IID) → CIN, FullName, Birth, Sex, BloodGroup, Phone,

(InsID, IID) → Type,

(STAFF_ID, DEP_ID) → DEP_ID, STAFF_ID,

DEP_ID → HID,

(PID, Drug_ID) → Duration, Dosage,

CAID → DEP_ID,

CAID → IID,

CAID → STAFF_ID}

4 Implementation & Results

```sql
-- Query 1: Find the names of patients who have had at least
    one clinical activity handled by active staff.
SELECT DISTINCT p.FullName
FROM Patient p
JOIN ClinicalActivity CA ON p.IID = CA.IID
JOIN Staff s ON s.STAFF_ID = CA.STAFF_ID
WHERE s.Status = 'Active';

-- Query 2: Find Staff IDs of staff who are either 'Active'
   or have issued at least one prescription.
SELECT CA.STAFF_ID
FROM ClinicalActivity CA
JOIN Prescription Pre ON CA.CAID = Pre.CAID
```

```sql
UNION
SELECT s.STAFF_ID
FROM Staff s
WHERE s.Status = 'Active';

-- Query 3: Find Hospital IDs of hospitals located in 'Benguerir' or having at least one department with the specialty 'Cardiology'.
SELECT H.HID
FROM Hospital H
WHERE H.City = 'Benguerir'
UNION
SELECT D.HID
FROM Department D
WHERE D.Name = 'Cardiology';

-- Query 4: Find Hospital IDs of hospitals that have both 'Cardiology' and 'Pediatrics' departments.
SELECT DISTINCT D1.HID
FROM Department D1
JOIN Department D2 ON D1.HID = D2.HID
WHERE D1.Name = 'Cardiology'
  AND D2.Name = 'Pediatrics';

-- Query 5: Find staff members who have worked in every department of the hospital with HID = 1.
SELECT s.STAFF_ID, s.FullName
FROM Staff s
WHERE NOT EXISTS (
    SELECT d.DEP_ID
    FROM Department d
    WHERE d.HID = 1
    AND NOT EXISTS (
        SELECT 1
        FROM ClinicalActivity ca
        WHERE ca.STAFF_ID = s.STAFF_ID
```

```sql
        AND ca.DEP_ID = d.DEP_ID
    )
);

-- Query 6: Find staff members who participated in every
   clinical activity of the department with DEP_ID = 2.
SELECT s.STAFF_ID, s.FullName
FROM Staff s
WHERE NOT EXISTS (
    SELECT ca.CAID
    FROM ClinicalActivity ca
    WHERE ca.DEP_ID = 2
    AND NOT EXISTS (
        SELECT 1
        FROM ClinicalActivity ca2
        WHERE ca2.STAFF_ID = s.STAFF_ID
        AND ca2.CAID = ca.CAID
    )
);

-- Query 7: Find pairs of staff members (s1, s2) such that
   s1 has handled more clinical activities than s2.
WITH R1 AS (
    SELECT STAFF_ID, COUNT(*) AS num
    FROM ClinicalActivity
    GROUP BY STAFF_ID
)
SELECT R1.STAFF_ID AS Staff1, R2.STAFF_ID AS Staff2
FROM R1
JOIN R1 AS R2 ON R1.num > R2.num AND R1.STAFF_ID != R2.
   STAFF_ID;

-- Query 8: Find Patient IDs of patients who had clinical
   activities with at least two different staff members.
SELECT CA.IID
FROM ClinicalActivity CA
```

```sql
GROUP BY CA.IID
HAVING COUNT(DISTINCT CA.STAFF_ID) >= 2;

-- Query 9: Find CAIDs of clinical activities performed in
   September 2025 at hospitals located in "Benguerir".
SELECT CA.CAID
FROM ClinicalActivity CA
JOIN Department D ON CA.DEP_ID = D.DEP_ID
JOIN Hospital H ON D.HID = H.HID
WHERE H.City = 'Benguerir'
  AND CA.Date BETWEEN '2025-09-01' AND '2025-09-30';

-- Query 10: Find Staff IDs of staff who have issued more
   than one prescription.
SELECT CA.STAFF_ID
FROM ClinicalActivity CA
JOIN Prescription Pre ON CA.CAID = Pre.CAID
GROUP BY CA.STAFF_ID
HAVING COUNT(*) > 1;

-- Query 11: List IIDs of patients who have scheduled
   appointments in more than one department.
SELECT CA.IID
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Status = 'Scheduled'
GROUP BY CA.IID
HAVING COUNT(DISTINCT CA.DEP_ID) >= 2;

-- Query 12: Find Staff IDs who have NO scheduled
   appointments on the day of the Green March holiday (
   November 6).
SELECT s.STAFF_ID
FROM Staff s
WHERE NOT EXISTS (
    SELECT 1
```

```sql
    FROM ClinicalActivity CA
    JOIN Appointment A ON A.CAID = CA.CAID
    WHERE CA.STAFF_ID = s.STAFF_ID
      AND CA.Date = '2025-11-06'
      AND A.Status = 'Scheduled'
);

-- Query 13: Find departments whose average number of clinical activities is below the global departmental average.
WITH R1 AS (
    SELECT DEP_ID, COUNT(*) AS num
    FROM ClinicalActivity
    GROUP BY DEP_ID
),
R1_Avg AS (
    SELECT AVG(num) AS Avr FROM R1
)
SELECT R1.DEP_ID
FROM R1
CROSS JOIN R1_Avg
WHERE R1.num < R1_Avg.Avr;

-- Query 14: For each staff member, return the patient who has the greatest number of completed appointments with that staff member.
WITH R1 AS (
    SELECT CA.STAFF_ID, CA.IID, COUNT(*) AS num_completion
    FROM Appointment A
    JOIN ClinicalActivity CA ON A.CAID = CA.CAID
    WHERE A.Status = 'Completed'
    GROUP BY CA.STAFF_ID, CA.IID
),
R2 AS (
    SELECT STAFF_ID, MAX(num_completion) AS max_num
    FROM R1
```

```sql
    GROUP BY STAFF_ID
)
SELECT R1.STAFF_ID, R1.IID
FROM R1
JOIN R2 ON R1.STAFF_ID = R2.STAFF_ID AND R1.num_completion =
    R2.max_num;

-- Query 15: List patients who had at least 3 emergency
   admissions during the year 2024.
SELECT P.FullName
FROM ClinicalActivity CA
JOIN Patient P ON P.IID = CA.IID
JOIN Emergency E ON E.CAID = CA.CAID
WHERE E.Outcome = 'Admitted'
  AND YEAR(CA.Date) = 2024
GROUP BY CA.IID, P.FullName
HAVING COUNT(*) >= 3;
```

5 Discussion

Throughout this lab, we encountered several challenges that tested our understanding of database design and refinement. One significant difficulty was interpreting and formulating complex queries. This required repeated review of the schema.

During the refinement phase, we invested considerable time in analyzing functional dependencies. This process was especially demanding in some relations related to the medical domain, where our limited background knowledge made it challenging to discern meaningful semantic relationships between attributes. For instance, determining whether certain medication properties were functionally dependent required additional research into pharmaceutical classifications and real-world clinical practices. These challenges highlighted the importance of domain knowledge in database design.

6 Conclusion

This lab offered a focused exploration of relational algebra and schema refinement, building directly on our prior work in conceptual and relational modeling. We applied relational algebra operations to formulate queries and analyze data relationships, while the refinement phase deepened our understanding of functional dependencies. Through this process, we developed crucial skills in logical schema optimization and dependency analysis, strengthening our foundation in relational theory and its practical application in complex real-world scenarios.