



University
Mohammed VI
Polytechnic



Deliverable #4: Relational Algebra, SQL and Functional Dependencies

Data Management Course
UM6P College of Computing

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

Team Information

Team Name	Groupe2
Member 1	Abir Fakhreddine
Member 2	Malak El Assali
Member 3	Nada El Farissi
Member 4	Amine Chrif
Member 5	Anass Fertat
Member 6	Yasser Hallou
Repository Link	https://github.com/beaNoBeebea

1 Introduction

After transforming the MNHS conceptual schema into a relational model by defining tables, attributes, primary keys, and foreign keys, the next step is to identify and state the functional dependencies (FDs).

Clear FDs reduce redundancy, enforce integrity, and improve efficiency. Without them, the schema is prone to anomalies; for example, storing the same medication in multiple rows and updating only one creates an update anomaly. By specifying the correct FDs, we ensure that each fact is stored in exactly one place and that any change propagates consistently.

With this foundation in place, we can then express queries over MNHS in Relational Algebra and translate them into SQL.

2 Requirements

In this deliverable, we are required to state, for each table in the MHNS relational model, the functional dependencies. Additionally, we are required to describe, theoretically, how the data will be retrieved and manipulated within the relational model for each query (using relational algebra), before moving on to the actual implementation in a real database using SQL.

This approach shows that the requirements are both analytical and practical. The analytical part involves identifying functional dependencies and describing queries using relational algebra, which provides a theoretical understanding of the data and its relationships. The practical part consists of translating these parts into SQL, ensuring that the analytical part is correct.

3 Methodology

We choose to begin with Functional dependencies , showing how each attribute or a set of them can determine another. **As for the primary keys dependencies we did not specify them since it is trivial that it implicitly determines all the other attributes as well as the fact that each attribute determines itself, and each relation determines itself as well.** Therefore we focused only on the non-trivial functional dependencies. **Entities without relevant dependencies were omitted.**

Usually, we would have experts, or the client guide us on the FDS, but as this is not available, we had to rely on our common sense to derive FDs

We expressed each query using both Relational Algebra and SQL. For Relational Algebra we used standard operators such as :

- **Selection (σ):** Chooses rows from a relation that satisfy a specified condition.
- **Projection (π):** Chooses specific columns (attributes) from a relation.
- **Join (\bowtie):** Combines rows from two or more relations based on a common attribute or condition.
- **Grouping/Aggregation (γ):** Groups rows by one or more attributes and computes aggregate values (for example, SUM, or AVG) for each group.
- **Rename (ρ):** Renames a relation or its attributes to avoid ambiguity or to make expressions clearer in complex queries.

to describe how to retrieve data theoretically. For SQL we translated the standard operators into syntax that can be run on actual databases.

4 Implementation & Results

4.1 Functional dependencies:

Trivial dependencies: primary keys determine the whole relation, superkeys determine the whole relation, and each attribute determines itself.

- **Entity:** ContactLocation
Primary key: CLID
Attributes: Street, Number, City, Province, PostalCode, Phone.
- **Functional dependency:**
 $\text{Postalcode} \rightarrow \text{City, Province}$
- **Entity:** Patient
Primary key: IID **Attributes:** CIN, Name, Sex, Birth, BloodGroup, Phone.
- **Functional dependency:**
 $\text{CIN} \rightarrow \text{IID}$ (and subsequently the whole relation, Name, Sex, Birth, BloodGroup, Phone)
- **Entity:** Staff
Primary key: STAFF ID **Attributes:** Name, Status.
ISA subtypes:
Practitioner: LicenseNumber, Specialty.
Caregiving: Grade, Ward.
Technical: Certifications, Modality.
- **Functional dependency:**
 $\text{LicenseNumber} \rightarrow \text{STAFF ID, Name, Status, Specialty}$ (partial FD)
- **Entity:** Hospital
Primary key: HID
Attributes: Name, City, Region.
- **Functional dependency:**
 $\text{City} \rightarrow \text{Region}$
- **Entity:** Medication
Primary key: DrugID
Attributes: Name, Form, Strength, Manufacturer, Class, ActiveIngredient.
- **Functional dependency:**
 $\text{Name} \rightarrow \text{Form, Strength, Manufacturer, Class, ActiveIngredient.}$

4.2 Queries:

- Find the names of patients who have had at least one clinical activity handled by active staff:

$$\pi_{\text{Name}}(\text{Patient} \bowtie_{\text{IID}} (\text{Clinical_activity} \bowtie_{\text{staff_ID}} (\sigma_{\text{status}=\text{'active'}}(\text{Staff}))))$$

```

1  SELECT Name
2  FROM
3  Patient P
4  JOIN
5  Clinical_activity C
6  ON
7  P.IID=C.IID
8  JOIN
9  Staff S
10 ON
11 S.staff_ID =C.staff_ID
12 WHERE
13 status='active'

```

- Find Staff IDs of staff who are either 'Active' or have issued at least one prescription:

$$\pi_{\text{staff_ID}}((\text{Clinical_activity} \bowtie_{\text{CAID}} \text{Prescription}) \cup (\sigma_{\text{status}=\text{'active'}}(\text{Staff})))$$

```

1  SELECT staff_ID FROM Staff WHERE status='active'
2  UNION
3  SELECT staff_ID FROM Clinical_Activity C JOIN Prescription P
    ON P.CAID=C.CAID

```

- Find Hospital IDs of hospitals located in 'Benguerir' or having at least one department with the specialty 'Cardiology':

$$\pi_{\text{HID}}(\sigma_{\text{city}=\text{'Benguerir'}}(\text{Hospital}))$$

U

$$\pi_{\text{HID}}(\text{Hospital} \bowtie_{\text{HID}} (\sigma_{\text{specialty}=\text{'Cardiology'}}(\text{Department})))$$

```

1  SELECT H.HID
2  FROM Hospital AS H
3  WHERE H.city = 'Benguerir'
4  UNION
5  SELECT DISTINCT H.HID
6  FROM Hospital AS H
7  JOIN Department AS D
8  ON D.HID = H.HID
9  WHERE D.specialty = 'Cardiology';

```

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

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

```

1  SELECT HID
2  FROM Department
3  WHERE name IN ('Cardiology', 'Pediatrics')
4  GROUP BY HID
5  HAVING COUNT(DISTINCT name) = 2;

```

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

$$R1 = \pi_{Dep_ID}(\sigma_{HID=1}(Department))$$

$$R2 = \pi_{STAFF_ID, Dep_ID}(Work_in)$$

So the desired result is $R = R2/R1$

```

1  SELECT STAFF_ID
2  FROM Work_in
3  WHERE Dep_ID IN (
4      SELECT Dep_ID
5      FROM Department
6      WHERE HID = 1
7  )
8  GROUP BY STAFF_ID
9  HAVING COUNT(DISTINCT Dep_ID) = (
10     SELECT COUNT(*)
11     FROM Department
12     WHERE HID = 1
13 );

```

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

$$R1 = \pi_{CAID}(\sigma_{Dep_ID=2}(ClinicalActivity))$$

$$R2 = \pi_{STAFF_ID, CAID}(ClinicalActivity)$$

So the desired result is $R = R2/R1$

```

1  SELECT STAFF_ID
2  FROM ClinicalActivity
3  WHERE Dep_ID = 2
4  GROUP BY STAFF_ID
5  HAVING COUNT(DISTINCT CAID) = (
6      SELECT COUNT(DISTINCT CAID)
7      FROM ClinicalActivity
8      WHERE Dep_ID = 2
9 );

```

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

$$\rho(R, \gamma_{STAFF_ID; COUNT(CAID) \rightarrow n}(Staff \bowtie ClinicalActivity))$$

$$\rho(R1(STAFF_ID \rightarrow s1, n \rightarrow n1), R)$$

$$\rho(R2(STAFF_ID \rightarrow s2, n \rightarrow n2), R)$$

$$\pi_{s1,s2}(\sigma_{n1 > n2}(R1 \times R2))$$

```

1  SELECT R1.STAFF_ID AS s1,
2    R2.STAFF_ID AS s2
3  FROM (
4    SELECT S.STAFF_ID, COUNT(CA.CAID) AS n
5    FROM Staff AS S
6    LEFT JOIN ClinicalActivity AS CA
7      ON S.STAFF_ID = CA.STAFF_ID
8    GROUP BY S.STAFF_ID
9  ) AS R1,
10 (
11   SELECT S.STAFF_ID, COUNT(CA.CAID) AS n
12   FROM Staff AS S
13   LEFT JOIN ClinicalActivity AS CA
14     ON S.STAFF_ID = CA.STAFF_ID
15   GROUP BY S.STAFF_ID
16 ) AS R2
17 WHERE R1.n > R2.n;

```

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

$$\pi_{IID}(\sigma_{n \geq 2}(\gamma_{IID; COUNT(STAFF_ID) \rightarrow n}(\pi_{IID, STAFF_ID}(ClinicalActivity))))$$

```

1  SELECT R1.IID
2  FROM (
3    SELECT R2.IID, COUNT(R2.STAFF_ID) AS n
4    FROM (
5      SELECT DISTINCT CA.IID, CA.STAFF_ID
6      FROM ClinicalActivity AS CA
7    ) AS R2
8    GROUP BY R2.IID
9  ) AS R1
10 WHERE R1.N >= 2;

```

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

$$\rho(R, ClinicalActivity \bowtie_{DEP_ID} Department \bowtie_{HID} Hospital)$$

$$\pi_{CAID}(\sigma_{City='Benguerir' \wedge Date \geq '2025-09-01' \wedge Date < '2025-10-01'}(R))$$

```

1  SELECT CA.CAID
2  FROM ClinicalActivity AS CA
3  JOIN Department AS D
4      ON D.DEP_ID = CA.DEP_ID
5  JOIN Hospital AS H
6      ON H.HID = D.HID
7  WHERE H.City = 'Benguerir',
8      AND CA.Date >= '2025-09-01',
9      AND CA.Date < '2025-10-01';

```

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

$$\pi_{\text{STAFF_ID}} \left(\sigma_{\text{COUNT}(\text{PID}) > 1} \left(\gamma_{\text{STAFF_ID}; \text{COUNT}(\text{DISTINCT PID})} \left(\pi_{\text{STAFF_ID}, \text{PID}} (\text{StaffActivity} \bowtie_{\text{CAID}} \text{Prescription})) \right) \right) \right)$$

```

1  SELECT sa.STAFF_ID
2  FROM StaffActivity sa
3  JOIN Prescription p ON sa.CAID = p.CAID
4  GROUP BY sa.STAFF_ID
5  HAVING COUNT(DISTINCT p.PID) > 1;

```

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

$$\pi_{\text{IID}} \left(\sigma_{\text{COUNT}(\text{DEP_ID}) > 1} \left(\gamma_{\text{IID}; \text{COUNT}(\text{DISTINCT DEP_ID})} \left(\pi_{\text{IID}, \text{DEP_ID}} (\sigma_{\text{Status}='Scheduled'} (\text{ClinicalActivity} \bowtie_{\text{CAID}} \text{Appointment})) \right) \right) \right)$$

```

1  SELECT ca.IID
2  FROM ClinicalActivity ca
3  JOIN Appointment a ON ca.CAID = a.CAID
4  WHERE a.Status = 'Scheduled'
5  GROUP BY ca.IID
6  HAVING COUNT(DISTINCT ca.DEP_ID) > 1;

```

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

$$\pi_{\text{STAFF_ID}} (\text{Staff}) - \pi_{\text{STAFF_ID}} \left(\sigma_{\text{Date}='2024-11-06' \wedge \text{Status}='Scheduled'} (\text{StaffActivity} \bowtie_{\text{CAID}} \text{ClinicalActivity} \bowtie_{\text{CAID}} \text{Appointment})) \right)$$

```

1  SELECT STAFF_ID
2  FROM Staff
3  WHERE STAFF_ID NOT IN (
4      SELECT sa.STAFF_ID
5      FROM StaffActivity sa
6      JOIN ClinicalActivity ca ON sa.CAID = ca.CAID
7      JOIN Appointment a ON ca.CAID = a.CAID
8      WHERE ca.Date = '2024-11-06' AND a.Status = 'Scheduled'
9  );

```

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

$$\begin{aligned} \text{DeptCount} &\leftarrow \gamma_{\text{DEP_ID}; \text{count(CAID)} \rightarrow \text{cnt}}(\text{ClinicalActivity}) \\ \text{GlobalAvg} &\leftarrow \gamma(\text{avg(cnt)} \rightarrow \text{gavg})(\text{DeptCount}) \\ \text{Answer} &\leftarrow \pi_{\text{DEP_ID}}(\sigma_{\text{cnt} < \text{gavg}}(\text{DeptCount} \times \text{GlobalAvg})) \end{aligned}$$

```

1  WITH DeptCount AS (
2      SELECT CA.DEP_ID, COUNT(*) AS cnt
3      FROM ClinicalActivity CA
4      GROUP BY CA.DEP_ID
5  ),
6  GlobalAvg AS (
7      SELECT AVG(cnt) AS gavg
8      FROM DeptCount
9  )
10 SELECT D.DEP_ID
11 FROM DeptCount DC
12 JOIN Department D ON D.DEP_ID = DC.DEP_ID
13 CROSS JOIN GlobalAvg GA
14 WHERE DC.cnt < GA.gavg;

```

14. For each staff member, return the patient(s) with the greatest number of completed appointments with that staff member:

$$\begin{aligned} R_1 &\leftarrow \sigma_{\text{Status}='completed'}(\text{ClinicalActivity} \bowtie \text{Appointment}) \\ R_2 &\leftarrow \pi_{\text{AID}, \text{STAFF_ID}, \text{IID}}(R_1) \\ R_3 &\leftarrow \gamma_{\text{STAFF_ID}, \text{IID}; \text{COUNT(*)} \rightarrow n}(R_2) \\ R_{\max} &\leftarrow \gamma_{\text{STAFF_ID}; \text{max}(n) \rightarrow \text{max_n}}(R_3) \\ \text{Answer} &\leftarrow \pi_{\text{STAFF_ID}, \text{IID}} \left(R_3 \bowtie_{R_3.\text{STAFF_ID} = R_{\max}.\text{STAFF_ID}} R_{\max} \right. \\ &\quad \left. R_3.n = R_{\max}.\text{max_n} \right) \end{aligned}$$

```

1  WITH Completed AS (
2      SELECT CA.STAFF_ID, CA.IID, CA.CAID
3      FROM ClinicalActivity CA

```

```

4   JOIN Appointment A ON A.CAID = CA.CAID
5   WHERE A.Status = 'completed'
6 ),
7 Counts AS (
8   SELECT STAFF_ID, IID, COUNT(*) AS n
9   FROM Completed
10  GROUP BY STAFF_ID, IID
11 ),
12 MaxCounts AS (
13   SELECT STAFF_ID, MAX(n) AS max_n
14   FROM Counts
15  GROUP BY STAFF_ID
16 )
17 SELECT C.STAFF_ID, C.IID
18 FROM Counts C
19 JOIN MaxCounts M
20  ON C.STAFF_ID = M.STAFF_ID
21 AND C.n = M.max_n;

```

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

$E2024 \leftarrow \sigma_{Date \geq '2024-01-01' \wedge Date \leq '2024-12-31'}(\text{ClinicalActivity}) \bowtie_{CAID} \text{Emergency}$
 $\text{Counts} \leftarrow \gamma_{IID; \text{count}(CAID) \rightarrow \text{cnt}}(E2024)$
 $\text{Answer} \leftarrow \pi_{IID}(\sigma_{\text{cnt} \geq 3}(\text{Counts}))$

```

1 SELECT CA.IID, P.FullName
2 FROM ClinicalActivity CA
3 JOIN Emergency E ON E.CAID = CA.CAID
4 JOIN Patient P ON P.IID = CA.IID
5 WHERE CA.Date >= '2024-01-01',
6   AND CA.Date < '2025-01-01',
7 GROUP BY CA.IID, P.FullName
8 HAVING COUNT(*) >= 3;

```

5 Discussion

Stepping back from the mechanics, a few themes stood out. Our treatment of naming in *Medication*, especially whether *Name* determines dosage and related attributes, hinged on domain input; client practices shaped what we considered redundancy. We also felt the gap between RA's relative brevity and clean structure and SQL's verbosity: what reads as a single operator can turn into layered joins and HAVING clauses. Wrapping our heads around the translation was certainly a good exercise in noticing the stark difference between practical and theoretical models.

This dual representation strengthened our understanding of how abstract data models translate into executable logic.

6 Conclusion

By identifying functional dependencies, we clarified that attributes may depend not only on the primary key but also on other attributes.

For querying, the RA expressions and the SQL statements highlighted the link between theory and practice: RA specifies the operations needed to retrieve data, while SQL turns those operations into commands that run on the database workbench.

This two-level approach, formal and operational, helped bridge the gap between database theory and real execution, fostering a clearer and more disciplined way of reasoning about data.