University Mohammed VI Polytechnic

Moroccan National Health Services/Logical Design

Data Management Course

UM6P College of Computing

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

Team Information

Team Name	AtlasDB
Member 1	Ahmed ENNASSIB
Member 2	Abdeljalil EL ACHEHAB
Member 3	Salma EL KADI
Member 4	Omar EL BOUKILI
Member 5	Adam EL MANNANI
Member 6	Housam EL GOUINA
Repository Link	https://github.com/BoukiliOmar/DBMS-AtlasDB

1 Introduction

The problem focuses on applying key concepts of database management—specifically Relational Algebra (RA), SQL querying, and Functional Dependency (FD) analysis—within the context of the Moroccan National Health Services (MNHS) database schema. The task involves translating real-world healthcare data requirements into formal relational expressions and SQL statements, ensuring both the correctness and efficiency of queries. In addition, identifying and analyzing functional dependencies allows for a deeper understanding of the underlying data relationships, which is essential for database normalization and integrity. By addressing a series of carefully designed queries, this work demonstrates how theoretical principles of database design can be applied to practical, real-world scenarios, ensuring that the MNHS database can efficiently store, retrieve, and maintain consistent healthcare information.

2 Requirements

The series of queries provided in the requirements serves as the practical foundation for applying relational database concepts to the MNHS schema. Each query is designed to test different aspects of database manipulation, from simple data retrieval to more complex operations involving joins, aggregations, and nested queries. By systematically translating these queries into Relational Algebra expressions and SQL statements, the task demonstrates how data can be efficiently extracted, analyzed, and interpreted. Additionally, the queries highlight the importance of understanding the relationships between different entities, as well as the role of functional dependencies in ensuring data consistency. Overall, this process not only validates the logical structure of the MNHS database but also provides insights into how real-world healthcare information can be accurately managed and queried.

3 Methodology

In tackling this work, we followed a step-by-step approach to get familiar with the MNHS database. We started by carefully studying the database schema, trying to understand each entity, its attributes, and how everything was connected—it felt a bit like mapping out a complex city. Next, we focused on identifying the functional dependencies to see which attributes relied on others; this helped us ensure that the data would remain consistent and reliable. For each query in the requirements, we first wrote it in Relational Algebra to fully grasp the underlying logic, and then translated it into SQL so we could actually run it and see the results in action. After executing the queries, we carefully checked the outcomes to make sure everything worked as expected. Taking this approach not only helped us understand the structure of the database better, but it also gave us confidence that our queries and analysis were accurate, meaningful, and ready for practical use.

4 Implementation

Relational Algebra and SQL Implementation

Query 1: Names of patients handled by active staff

Relational Algebra:

$\pi_{Name}(Patient_{Patient.IID=ClinicalActivity.IID}(\sigma_{Status='Active'}(Staff)_{Staff.STAFFID=ClinicalActivity.STAFFID}ClinicalActivity))$

SQL:

```sql
SELECT DISTINCT P.Name
FROM Patient P
JOIN ClinicalActivity CA ON P.IID = CA.IID
JOIN Staff S ON CA.STAFFID = S.STAFFID
WHERE S.Status = 'Active';
```

Query 2: Staff IDs who are either Active or issued at least one prescription

Relational Algebra:

$\pi_{STAFF_ID}(\sigma_{Status='Active'}(Staff)) \cup \pi_{STAFFID}(Staff_{Staff.STAFFID=Prescription.STAFFID}Prescription)$

SQL:

```sql
SELECT STAFFID
FROM Staff
WHERE Status = 'Active'
UNION
SELECT DISTINCT CA.STAFFID
FROM ClinicalActivity CA
JOIN Prescription P ON CA.CAID = P.CAID;
```

Query 3: Hospital IDs in Benguerir or with Cardiology department

Relational Algebra:

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

SQL:

```sql
SELECT HID
FROM Hospital
WHERE City = 'Benguerir'

UNION

SELECT DISTINCT D.HID
FROM Department D
WHERE D.Specialty = 'Cardiology';
```

Query 4: Hospitals with both Cardiology and Pediatrics departments

Relational Algebra:

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

SQL:

```sql
SELECT HID
FROM Department
WHERE Specialty = 'Cardiology'
INTERSECT
SELECT HID
FROM Department
WHERE Specialty = 'Pediatrics';
```

Query 5: Staff who worked in every department of hospital HID=1

Relational Algebra:

$$\pi_{STAFFID}(WorkIn) \div \pi_{DEPID}(\sigma_{HID=1}(Department))$$

SQL:

```sql
SELECT STAFFID
FROM WorkIn W
WHERE W.DEPID IN (SELECT DEPID FROM Department WHERE HID = 1)
GROUP BY STAFFID
HAVING COUNT(DISTINCT W.DEPID) =
    (SELECT COUNT(DEPID) FROM Department WHERE HID = 1);
```

Query 6: Staff who participated in every clinical activity of DEPID=2

Relational Algebra:

$$\pi_{STAFFID}(StaffActivity) \div \pi_{CAID}(\sigma_{DEPID=2}(ClinicalActivity))$$

SQL:

```sql
SELECT STAFFID
FROM Staff
WHERE CAID IN (SELECT CAID FROM ClinicalActivity WHERE DEPID = 2)
GROUP BY STAFFID
HAVING COUNT(DISTINCT CAID) =
    (SELECT COUNT(CAID) FROM ClinicalActivity WHERE DEPID = 2);
```

Query 7: Pairs of staff where one handled more clinical activities than the other

Relational Algebra:

$$\{(s_1, s_2) \mid countCAID(\sigma_{STAFF_ID=s1}(ClinicalActivity)) > countCAID(\sigma_{STAFF_ID=s2}(ClinicalActivity))$$

SQL:

```sql
SELECT s1.STAFF_ID AS s1, s2.STAFF_ID AS s2
FROM Staff s1
CROSS JOIN Staff s2
WHERE (
    SELECT COUNT(*)
    FROM ClinicalActivity ca
    WHERE ca.STAFF_ID = s1.STAFF_ID
) > (
    SELECT COUNT(*)
    FROM ClinicalActivity ca
    WHERE ca.STAFF_ID = s2.STAFF_ID
);
```

Query 8: Patient IDs with clinical activities by at least two different staff members

Relational Algebra:

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

SQL:

```sql
SELECT IID
FROM (
    SELECT IID, COUNT(STAFF_ID) AS StaffCount
    FROM (
        SELECT DISTINCT IID, STAFF_ID
        FROM ClinicalActivity
    ) AS distinctCA
    GROUP BY IID
) AS countTable
WHERE StaffCount >= 2;
```

Query 9: CAIDs of clinical activities in Sept 2025 at hospitals in Benguerir

Relational Algebra:

$$\pi_{CAID}(\sigma_{Date \geq '2025-09-01' \wedge Date \leq '2025-09-30'}(ClinicalActivityDepartment\sigma_{City='Benguerir'}(Hospital)))$$

SQL:

```sql
SELECT CA.CAID
FROM ClinicalActivity CA
JOIN Department D ON CA.DEPID = D.DEPID
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: Staff IDs who issued more than one prescription

Relational Algebra:

$$\pi_{STAFFID}(\sigma_{count>1}(\gamma_{STAFFID;COUNT(PID)\rightarrow count}(ClinicalActivityPrescription)))$$

SQL:

```sql
SELECT CA.STAFFID
FROM ClinicalActivity CA
JOIN Prescription P ON CA.CAID = P.CAID
GROUP BY CA.STAFFID
HAVING COUNT(P.PID) > 1;
```

Query 11: IIDs of patients with appointments in more than one department

Relational Algebra:

$$\pi_{IID}(\sigma_{count>1}(\gamma_{IID;COUNT(DISTINCTDEP_ID)\rightarrow count}(\sigma_{Status='Scheduled'}(AppointmentClinicalActivity))))$$

SQL:

```sql
SELECT CA.IID
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
GROUP BY CA.IID
HAVING COUNT(DISTINCT CA.DEPID) > 1;
```

Query 12: Staff IDs with no scheduled appointments on Nov 6, 2025

Relational Algebra:

$$\pi_{STAFFID}(Staff - \pi_{STAFFID}(\sigma_{Date='2025-11-06' \wedge Status='Scheduled'}(ClinicalActivityAppointment)))$$

SQL:

```sql
SELECT STAFFID
FROM Staff
WHERE STAFFID NOT IN (
  SELECT CA.STAFFID
  FROM ClinicalActivity CA
  JOIN Appointment A ON CA.CAID = A.CAID
  WHERE A.Status = 'Scheduled' AND A.Date = '2025-11-06'
);
```

Query 13: Departments with average clinical activities below global department average

Relational Algebra:

$$\pi_{DEP\_ID}\Big(\sigma_{activity\_count<global\_avg}(\gamma_{DEP\_ID;COUNT(CAID)\to activity\_count}(ClinicalActivity))\Big)$$

SQL:

```sql
WITH activity_counts AS (
    SELECT DEP_ID, COUNT(CAID) AS activity_count
    FROM ClinicalActivity
    GROUP BY DEP_ID
)
SELECT DEP_ID
FROM activity_counts
WHERE activity_count < (SELECT AVG(activity_count) FROM
   activity_counts);
```

Query 14: For each staff, patient with greatest completed appointments

Relational Algebra: $\mathrm{RA}_1 = \sigma_{Status='Completed'}(AppointmentClinicalActivity)$
$RA_2 = RA_1 StaffPatient$
$RA_3 = \gamma_{Staff\_ID,IID;COUNT(CAID)\rightarrow appointment\_count}(RA_2)$
$RA_4 = \sigma_{appointment\_count=MAX(appointment\_count)}(RA_3)$
$Result = \pi_{Staff\_ID,Staff.Name,IID,Patient.Name,appointment\_count}(RA_4)$

SQL:

```sql
WITH AppCounts AS (
  SELECT CA.STAFFID, CA.IID, COUNT(*) AS CompletedCount
  FROM ClinicalActivity CA
  JOIN Appointment A ON CA.CAID = A.CAID
  WHERE A.Status = 'Completed'
  GROUP BY CA.STAFFID, CA.IID
),
MaxCounts AS (
  SELECT STAFFID, MAX(CompletedCount) AS MaxCount
  FROM AppCounts
  GROUP BY STAFFID
)
SELECT A.STAFFID, A.IID
FROM AppCounts A
JOIN MaxCounts M ON A.STAFFID = M.STAFFID AND A.CompletedCount =
   M.MaxCount;
```

Query 15: Patients with at least 3 emergency admissions during 2024

Relational Algebra:

$$\pi_{IID}\Big(\sigma_{admission\_count\geq 3}\Big(\gamma_{IID;COUNT(CAID)\rightarrow admission\_count}(\sigma_{Year(Date)=2024}(EmergencyClinicalActivity))\Big)\Big)$$

SQL:

```sql
SELECT IID
FROM ClinicalActivity CA
JOIN Appointment A ON CA.CAID = A.CAID
WHERE A.Reason = 'Emergency'
AND CA.Date BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY IID
HAVING COUNT(*) >= 3;
```

Functional Dependencies

¿ Patient IID → CIN, Name, Sex, Birth, BloodGroup, Phone
ContactLocation CLID → Street, Number, City, Province, PostalCode, Phone
Staff STAFF$_ID \rightarrow$ Name, Status
Department DEP$_ID \rightarrow$ Name, Specialty, HID
Hospital HID → Name, City, Region
ClinicalActivity CAID → Date, Time, IID, STAFF$_ID, DEP_ID$
$AppointmentCAID \rightarrow$ Reason, Status
Emergency CAID → TriageLevel, Outcome
Prescription PID → DateIssued, CAID
Medication MID → Name, Form, Strength, Manufacturer, Class, ActiveIngredient
Expense ExID → Total, InsID, CAID
Insurance InsID → Type
WorkIn (STAFF$_ID, DEP_ID) \rightarrow$ none
Belongs DEP$_ID \rightarrow$ HID
Stock (HID, MID, StockTimestamp) → UnitPrice, Qty, ReorderLevel
Generate PID → CAID
Include (PID, MID) → Dosage, Duration
Have (IID, CLID) → none
Covers (InsID, IID) → none

Derived Functional Dependencies: ¿
From Department and Hospital DEP$_ID \rightarrow$ City, Region
From ClinicalActivity and Department CAID → City, Region
From ClinicalActivity and Patient CAID → CIN, Name, Sex, Birth, BloodGroup, Phone
From Prescription and ClinicalActivity PID → STAFF$_ID$
$FromPrescription, ClinicalActivity, andPatientPID \rightarrow$ CIN, Name, Sex, Birth, BloodGroup, Phone
From Include and Prescription (PID, MID) → IID, Dosage, Duration
From Stock and Hospital (HID, MID, StockTimestamp) → Name, City, Region, UnitPrice, Qty, ReorderLevel
From WorkIn, Department and Hospital (STAFF$_ID, DEP_ID) \rightarrow$ Name, City, Region
From ClinicalActivity and Staff CAID → Name, Status
From Prescription and Include (PID, MID) → IID, Dosage, Duration

5 Discussion

During this lab, we encountered several challenges that tested our understanding of the database and our problem-solving skills. One of the first difficulties was fully grasping the relationships between the different entities in the MNHS schema, especially when it came to handling complex joins and nested queries. Writing the relational algebra expressions was also tricky at times, particularly for queries that involved aggregation, division, or multiple layers of joins, since a small mistake could lead to incorrect results. Translating these expressions into SQL presented its own set of hurdles, such as ensuring that the syntax was correct, handling grouping and counting properly, and verifying that the results matched the intended logic. Additionally, understanding and applying functional dependencies required careful attention, because overlooking even one dependency could introduce redundancy or inconsistencies in the data. Despite these obstacles, working through the problems helped us deepen our understanding of relational databases and improved our ability to think logically about data.

6 Conclusion

In conclusion, this lab gave us a valuable opportunity to apply the concepts of relational algebra, SQL, and functional dependency analysis in a real-world context. By working through the MNHS database, we not only strengthened our technical skills but also learned how to approach complex data problems methodically. Despite facing challenges with complex queries and understanding the relationships between entities, we were able to systematically break down each problem, write accurate relational algebra expressions, and translate them into working SQL statements. This process reinforced the importance of careful planning, attention to detail, and collaboration when working with databases. Overall, the lab enhanced our understanding of database design and querying, and gave us practical experience that will be useful in future projects.