Database Project 23/24

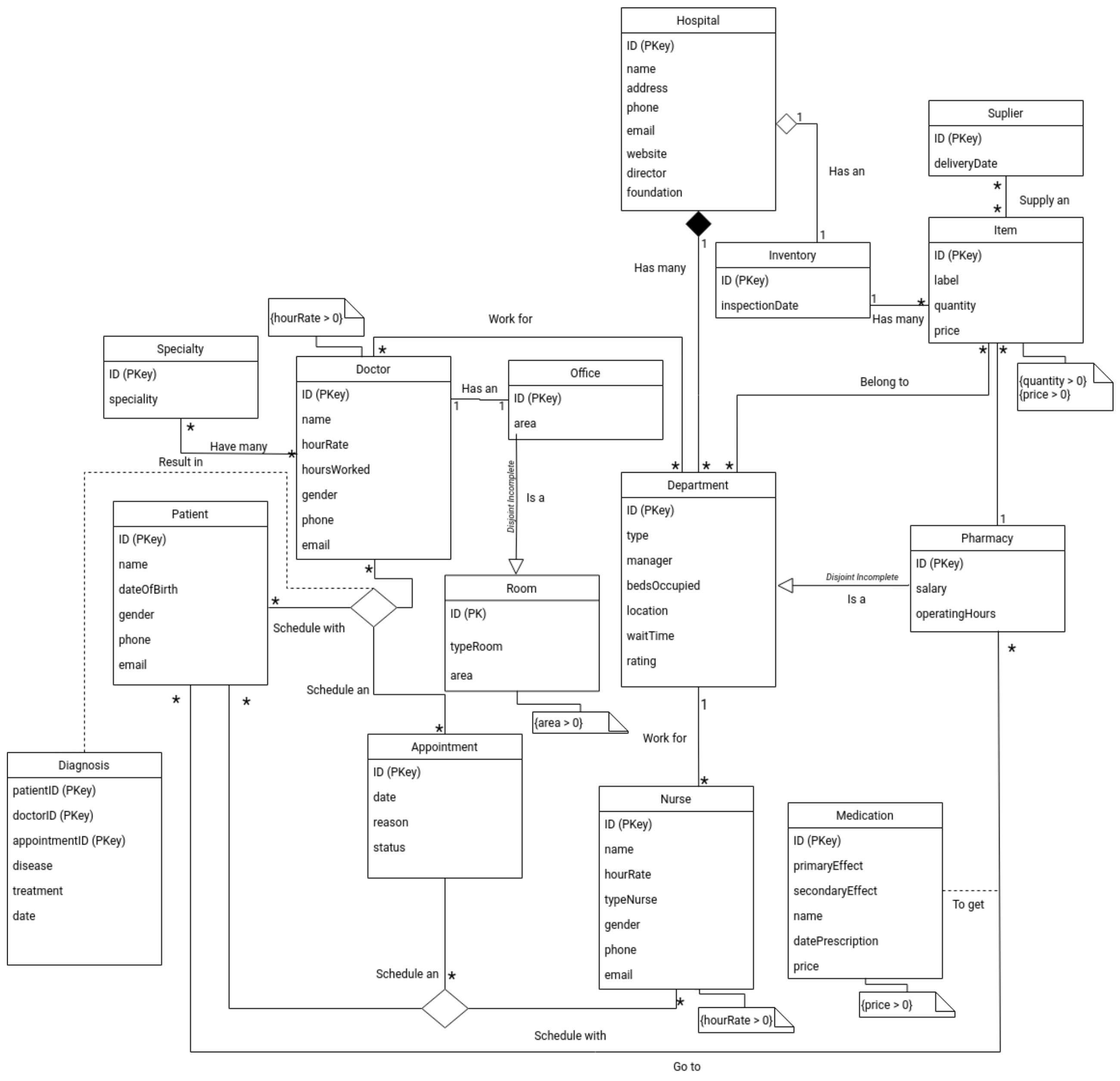
Hospital Management System

5. SECOND SUBMISSION:

RELATIONAL MODELING, DATABASE CREATION AND DATA LOADING

BD1305 Membros: Afonso Mansilha André de Sousa Rodrigo de Sousa

A. Refine the Conceptual Model



BD1305 Membros: Afonso Mansilha André de Sousa Rodrigo de Sousa

B. Define the Relational Schema

2. Relational Model for the given Conceptual Model

1. Hospital(ID, name, address, phone, email, website, director, foundation, ID → Inventory (FK))
2. Supplier(ID, deliveryDate)
3. Item(ID, label, quantity, price, ID → Inventory (FK), ID → Pharmacy (FK))
4. Inventory(ID, inspectionDate)
5. Pharmacy(ID → Department, salary, operatingHours)
6. Specialty(ID, specialty)
7. Doctor(ID, name, hourRate, hoursWorked, gender, phone, email)
8. Office(ID → Room, area, ID → Doctor (FK))
9. Department(ID, type, manager, bedsOccupied, location, waitTime, rating, ID → Hospital (FK))
10. Room(ID, typeRoom, area)
11. Patient(ID, name, dateOfBirth, gender, phone, email)
12. Appointment(ID, date, reason, status)
13. Nurse(ID, name, hourRate, typeNurse, gender, phone, email, ID → Department (FK))

14. SupplierItem(ID → Supplier, ID → Item)
15. ItemDepartment(ID → Item, ID → Department)
16. DepartmentDoctor(ID →Department, ID →Doctor)
17. DoctorSpecialty(ID →Doctor, ID →Specialty)
18. Diagnosis(ID, ID → Patient, ID → Doctor, ID → Appointment, disease, treatment)
19. Medication(ID →Pharmacy, ID →Patient, ID, primaryEffect, secondaryEffect, name, date)
20. PatientAppointementNurse(ID → Patient, ID → Appointment, ID → Nurse)

3. AI Tool Integration

21. Hospital(ID, name, address, phone, email, website, director, foundation, inventoryID → Inventory (FK))
22. Supplier(ID, deliveryDate)
23. Item(ID, label, quantity, price, inventoryID → Inventory (FK), pharmacyID → Pharmacy (FK))
24. Inventory(ID, inspectionDate)
25. Pharmacy(departmentID → Department, salary, operatingHours)
26. Specialty(ID, specialty)
27. Doctor(ID, name, hourRate, hoursWorked, gender, phone, email)
28. Office(roomID → Room, area, doctorID → Doctor (FK))
29. Department(ID, type, manager, bedsOccupied, location, waitTime, rating, hospitalID → Hospital (FK))
30. Room(ID, typeRoom, area)
31. Patient(ID, name, dateOfBirth, gender, phone, email)
32. Appointment(ID, date, reason, status)
33. Nurse(ID, name, hourRate, typeNurse, gender, phone, email, departmentID → Department (FK))

34. SupplierItem(supplierID → Supplier, itemID → Item)
35. ItemDepartment(itemID → Item, departmentID → Department)
36. DepartmentDoctor(departmentID →Department, doctorID →Doctor)
37. DoctorSpecialty(doctorID →Doctor, specialtyID →Specialty)
38. Diagnosis(ID, patientID → Patient, doctorID → Doctor, appointmentID → Appointment, disease, treatment)
39. Medication(pharmacyID →Pharmacy, patientID →Patient, ID, primaryEffect, secondaryEffect, name, date)
40. PatientAppointmentNurse(patientID → Patient, appointmentID → Appointment, nurseID → Nurse)

4. AI Tool Discussion

AI did fix an error (PatientAppointementNurse → PatientAppointmentNurse).and helped us have less ambiguity with the Foreign Keys for the sql part later on.

C. Functional Dependencies and Normal Forms Analysis

1. Functional Dependencies

1. HospitalID → name, address, phone, email, website, director, foundation
2. SupplierID → deliveryDate
3. ItemID → label, quantity, price
4. InventoryID → inspectionDate
5. DepartmentID → salary, operatingHours
6. SpecialtyID → specialty
7. DoctorID → name, hourRate, hoursWorked, gender, phone, email
8. RoomID → area
9. DepartmentID → type, manager, bedsOccupied, location, waitTime, rating
10. RoomID → typeRoom, area
11. PatientID → name, dateOfBirth, gender,phone, email
12. AppointmentID → date, reason, status
13. NurseID → name, hourRate, typeNurse, gender, phone, email

2. Boyce-Codd Normal Form

Boyce-Codd Normal Form is characterized by these two fundaments:

Every non-prime attribute (attributes not part of any candidate key) must be functionally dependent on the superkeys.

No non-prime attribute should be transitively dependent on any superkey.

1. Attributes: name, address, phone, email, website, director, foundation

   All attributes are functionally dependent on the HospitalID.

2. Attributes: deliveryDate

   All attributes are functionally dependent on the SupplierID.

3. Attributes: label, quantity, price

   All attributes are functionally dependent on the ItemID.

4. Attributes: inspectionDate

   The only attribute is functionally dependent on the InventoryID.

5. Attributes: salary, operatingHours

   Both attributes are functionally dependent on the DepartmentID but so are the ones in 3, so it isn't in the Boyce-Codd Normal Form.

6. Attributes: specialty

   The only attribute is functionally dependent on the SpecialtyID.

7. Attributes: name, hourRate, hoursWorked, gender, phone, email

   All attributes are functionally dependent on the DoctorID.

8. Attributes: area

   The only attribute is functionally dependent on the RoomID.

9. Attributes: type, manager, bedsOccupied, location, waitTime, rating

   All attributes are functionally dependent on the DepartmentID.

10. Attributes: typeRoom, area

The attributes are functionally dependent on the RoomID but so is number 8 so this is not in the Boyce-Codd Normal Form.

11. Attributes: name, dateOfBirth, gender, phone, email

    All attributes are functionally dependent on the PatientID.

12. Attributes: date, reason, status

    All attributes are functionally dependent on the AppointmentID.

13. Attributes: name, hourRate, typeNurse, gender, phone, email

    All attributes are functionally dependent on the NurseID.

3. 3NF

All, once again, except RoomID and DepartmentID are in the 3NF, as non-prime attributes are fully functionally dependent on the primary key (ID).

4. Fixing into 3NF

Separating into different relations can be one way to fix it:

OfficeID → area
RoomID → typeRoom, area

PharmacyID → salary, operatingHours
DepartmentID → department, manager, bedsOccupied, location, waitTime, rating

5. AI Integration

The AI suggested that we separate the IDs for each, creating an OfficeID separate from the RoomID and a PharmacyID separate from the DepartmentID.

6. AI Usage

It helped us make sure what we were doing was correct and clear some hurdles.

D. SQLite Database Creation

4. Refinements made using AI

Using ChatGPT we were able to add constraints such as NOT NULL and CHECK (quantity >= 0) that improve the quality of our data base. Furthermore, the AI also helped us make sure that our code was complete and organized.

E. SQLite

3. Refinements made using AI

For the populate2.sql we checked if the syntax was correct using sqlite3 and ChatGPT so we just used the populate1.sql, since it didn’t offer us any improvement. Also, adding more tuples wasn’t something AI helped us with since it never managed to do it the way we needed it to be done.

F. AI Usage

2. Refinements made using AI

For the Functional Dependencies we provided the ones we had already made and asked “what should i improve:”. It provided this:
Your schema seems to be well-organized, but there are a few suggestions for improvement:

1. **Consistent Naming Convention:**
    - Try to maintain a consistent naming convention for your identifiers (tables and columns). For example, you have tables with names like "PatientAppointementNurse" and "Medication." Consider using a consistent spelling, such as "Appointment" instead of "Appointement," to make your schema more readable and maintainable.
2. **Foreign Key Naming Convention:**
    - It's a good practice to have a consistent naming convention for foreign keys. You have, for example, "ID → Inventory (FK)" and "ID → Pharmacy (FK)." Consider using a more standard naming convention, like "InventoryID" and "PharmacyID," for foreign keys.

For the Normal Forms, we used AI to help us check if we could just separate those two and if that would fix the issues found, providing the functional dependencies that we had and asking it to provide a fix to those it found not already in the 3rd Normal Form, giving the same idea we had.

For the refinement in D-4: “Improve this database by adding constraints such as: ex hourRate > 0 , hoursWorked > 0 and others.” after providing it with the create1.sql.

To check the populate1.sql to make sure it was correct we put the whole text on ChatGPT and asked it if there was any improvements to be made and it gave us the same thing.

3. Problems of using AI

ChatGPT really struggled to give us more data for the populate2.sql as no matter how many instructions we gave it, it would get lost in the density of the information already provided by us before.

6. PARTICIPATION OF THE VARIOUS STUDENTS OF THE GROUP

For this submission we designated topics for each of us to do: the Uml reformulation was made by André, the relational model was made by Afonso and Rodrigo, the Boyce-Codd Normal Form, the 3NF and the create1 file were made by Rodrigo, The polulate1 and populate2 files were made by Afonso. For last, the AI refinements such as the create2 were made by Rodrigo and Afonso.