MEDICAL OFFICE INFORMATION SYSTEM

Team Name: L2-3E-11

Project Members:

1. Dahal Sabin
2. Dhakal Samragi
3. Nagórnik Natalia
4. Tóth András

CASE STUDY

The medical office of a company has decided to design a database for its services. Each office department is identified by a code, and it is also necessary to record the department's name, the phone number, and all the employees assigned to it. We want to know the SSN that identifies each employee, their name, phone number, category, and the department to which they belong. An employee must work in only one department.

There are two types of services: consultations and medical revisions. Medical consultations are services that can be provided daily to any company's employees in the event of an emergency, a problem, or a need of treatment. A reference number must be stored for each medical consultation to distinguish it from the other services, as well as the date and time of the consultation, the employee who was attended, the health personnel involved, the reason for the consultation, and, if used, the materials and the quantity of each material used. The health personnel belong to the company, and for each health personnel, their specialty and category (doctor or nurse) must be known. A practice must always have at least one doctor or nurse. The information on the material available in the medical office is stored in such a way that each material is assigned a code that identifies it, a description, and the number of units in the medicine cabinet.

There is other type of service: medical revisions. Medical revisions are annual services scheduled for departments to perform diagnostic tests on employees' health. Each employee is only allowed one revision per year. It is preferable to keep track of revisions and medical tests performed during the revision process. Each revision is identified by a reference number that distinguishes it from the other services, corresponds to a specific department and only to one, and consists of as many appointments as are required to make the revision to all the employees in the department. Appointments for a given revision are scheduled in groups of no more than ten employees on the same date and time.

On the other hand, we want to keep track of the medical tests that are performed at each medical revision. The code of the type of test performed and the test result will be recorded for each test performed on an employee within a revision. The code that identifies each type of test that the medical office can perform, its description, and the external physicians who can perform the test are also recorded.

TABLE OF CONTENTS

1. DESCRIPTION OF THE INFORMATION SYSTEM

This project focuses on designing a database system for a company's medical office. The sections below outline the system description, clarify ambiguous points from the original case study scenario, and introduce our proposed extensions.

1.1. Clarifications and Assumptions

While reading the assignment we ran into a few ambiguities notably about health personnel and tests. We discussed these in our group meetings and agreed on the following interpretations.

- **Health Personnel:** Health personnel are company employees, so we made HealthPersonnel a subclass of Employee. We debated instead adding a separate table but chose inheritance to avoid duplicated fields. It inherits standard data (SSN, Name, Phone) but adds specific fields for Specialty and Professional Type.
- **Services Hierarchy:** "Consultations" and "Medical Revisions" are different services but share common attributes (like an ID). To handle this, we created an abstract class called Service that holds the ReferenceNumber.
- **Medical Tests:** Tests happen inside a revision. We assumed that we need a way to link an Employee, a Revision, and a Test Type together to store the result. We called this link TestResult.
- **Materials Usage:** Simply connecting Consultations to Materials wasn't enough because we need to know how much was used. We added a Quantity attribute to this relationship.

1.2. Extended Problem Description

*Below we list the modifications our group proposed (**shown in bold**). These extensions came up after a discussion about safety and traceability.*

The medical office needs a system to manage its daily work. The company consists of **Departments** (defined by code, name, and phone). **Employees** work in these departments. For every employee, we store their SSN, name, phone number, category, and their department.

Some employees are **Health Personnel** (Doctors and Nurses). For them, we also track their specialty and category.

The system handles two main services:

1. **Medical Consultations:** These happen daily (emergencies/treatments). Every consultation has a reference number, date, time, reason, the patient (employee), and the **Health Personnel** (at least one). During the visit, **Materials** from the medicine cabinet can be used. We track the material code, description, current stock, and the

quantity used in the consultation.

[EXTENSION]: To improve the system, we added two new features:

- **Prescriptions: Doctors can now issue prescriptions.** A Prescription has a unique ID and instructions. It is connected to one or more **Medicines** (a new class with ID, name, and active ingredient).
- **Allergies: For patient safety, we added an Allergy class** (Code, Name). We track which employees have which allergies to avoid bad reactions during treatment.

2. **Medical Revisions:** These are annual check-ups for a whole department. A Revision has a reference number and links to one Department. To organize it, we create **Appointments**. Each appointment has a date/time and a group of employees (max 10). During a revision, **Medical Tests** are done. We record the **Test Type** and the **Result** for each employee. Some tests are done by **External Physicians** (we store their license number and name).

2. ANALYSIS OF REQUIREMENTS

Here, we have listed the functional requirements for the database, divided into data updates and data queries.

2.1. Update Requirements

- **Manage employees:** Add new employees, assign them to departments, modify or delete employee data. Record SSN and contact information.
- **Manage stock & materials:** Add new materials and update stock levels. When materials are used in consultation, the system decrements stock.
- **Manage consultations:** Register a new consultation for an employee with the attending health personnel. Record reference number, date/time , employee attended, personnel attended, reason, materials used and quantities.
- **Manage prescriptions & allergies (extensions):**
    - Create a prescription for a consultation, including all medicines.
    - **Add new allergies and assign them to patients. Check for patient allergies before prescribing medicines.**
- **Manage medical revisions:** Open a new annual revision for a specific department.
- **Manage appointments:** Create time slots for the revisions and assign groups of employees (up to 10) to each appointment.
- **Manage test results:** Enter test results for employees during a revision.

2.2. Query Requirements

1. **View information about a department:** Show the department name, phone number and all the employees assigned to it.
2. **View patient history:** List all past consultations for an employee, including dates and treatments.
3. **View Safety Check (Extension): Check if a patient has any allergies before prescribing medicine.**
4. **View Material Stock:** Check remaining units of a material are left and see usage history.
5. **View Revision Progress:** See upcoming appointments and check which employees finished their tests.
6. **View External Doctors:** Show which medical tests are performed by a specific external doctor.

3. CONCEPTUAL DESIGN (UML CLASS DIAGRAM)

3.1. Classes and Attributes

- **Basic Classes:**
    - **Department:** Code (PK), Name, Phone.
    - **Employee:** SSN (PK), Name, Phone, Category.
    - **HealthPersonnel** (Inherits from Employee): Specialty, ProfessionalType (Doctor/Nurse).
    - **Material:** Code (PK), Description, StockUnits.
    - **Service** (Abstract Parent): ReferenceNumber (PK).
    - **Consultation** (Child of Service): Date, Time, Reason.
    - **Revision** (Child of Service): Year.
    - **Appointment:** Date, Time.
    - **TestType:** Code (PK), Description.
    - **ExternalPhysician:** LicenseNum (PK), Name.
    - **TestResult** (Association Class): ResultValue.
    - **UsedMaterial** (Association Class): Quantity.

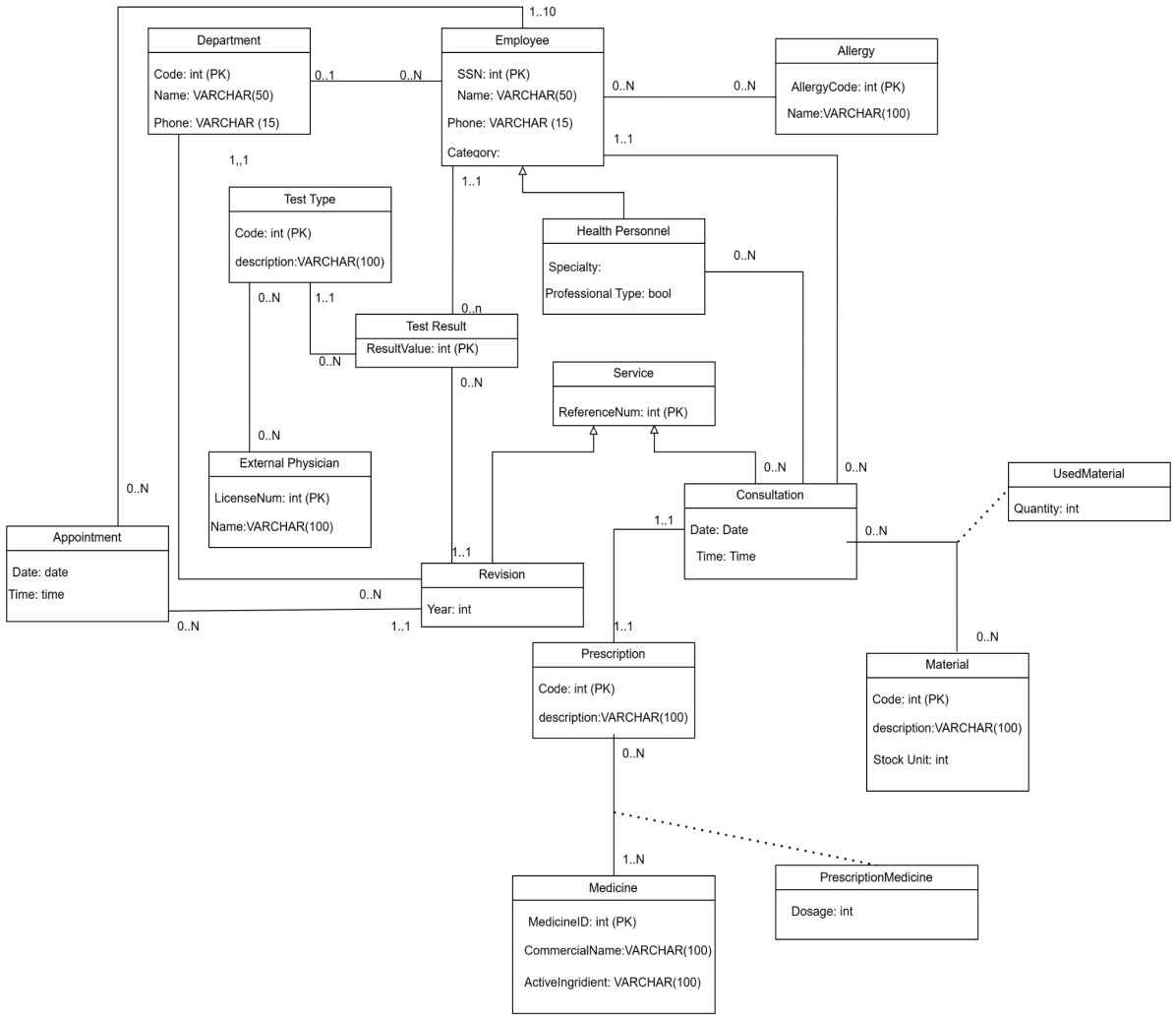
- **Extended Classes:**
    - **Prescription:** PrescriptionID (PK), Instructions.
    - **Medicine:** MedicineID (PK), CommercialName, ActiveIngredient.
    - **Allergy:** AllergyCode (PK), Name.
    - **PrescriptionMedicine(Association Class):** Dosage

3.2. Relationships

1. **Department - Employee:** 1:N (One department has many employees; an employee belongs to exactly one department).
2. **Employee - HealthPersonnel:** Generalization/Inheritance.
3. **Employee - Allergy [Extension]:** N:M (An employee can have multiple allergies; an allergy can affect multiple employees).
4. **Consultation - Employee:** N:1 (A consultation belongs to one patient).
5. **Consultation - HealthPersonnel:** N:M (A consultation is attended by one or more health personnel).
6. **Consultation - Material:** N:M (Uses). Attributes: Quantity (via UsedMaterial).
7. **Consultation - Prescription [Extension]:** 1:1 (A consultation can generate prescriptions).

8. **Prescription - Medicine [Extension]:** N:M (A prescription includes one or more medicines).
9. **Revision - Department:** N:1 (A revision corresponds to one department).
10. **Revision - Appointment:** 1:N (A revision consists of multiple appointments).
11. **Appointment - Employee:** N:M (An appointment schedules a group of employees, max 10).
12. **Revision - Employee - TestType:** Ternary relationship (via TestResult).
13. **TestType - ExternalPhysician:** N:M.

Here is the final UML class diagram we created based on our analysis using the draw.io tool.



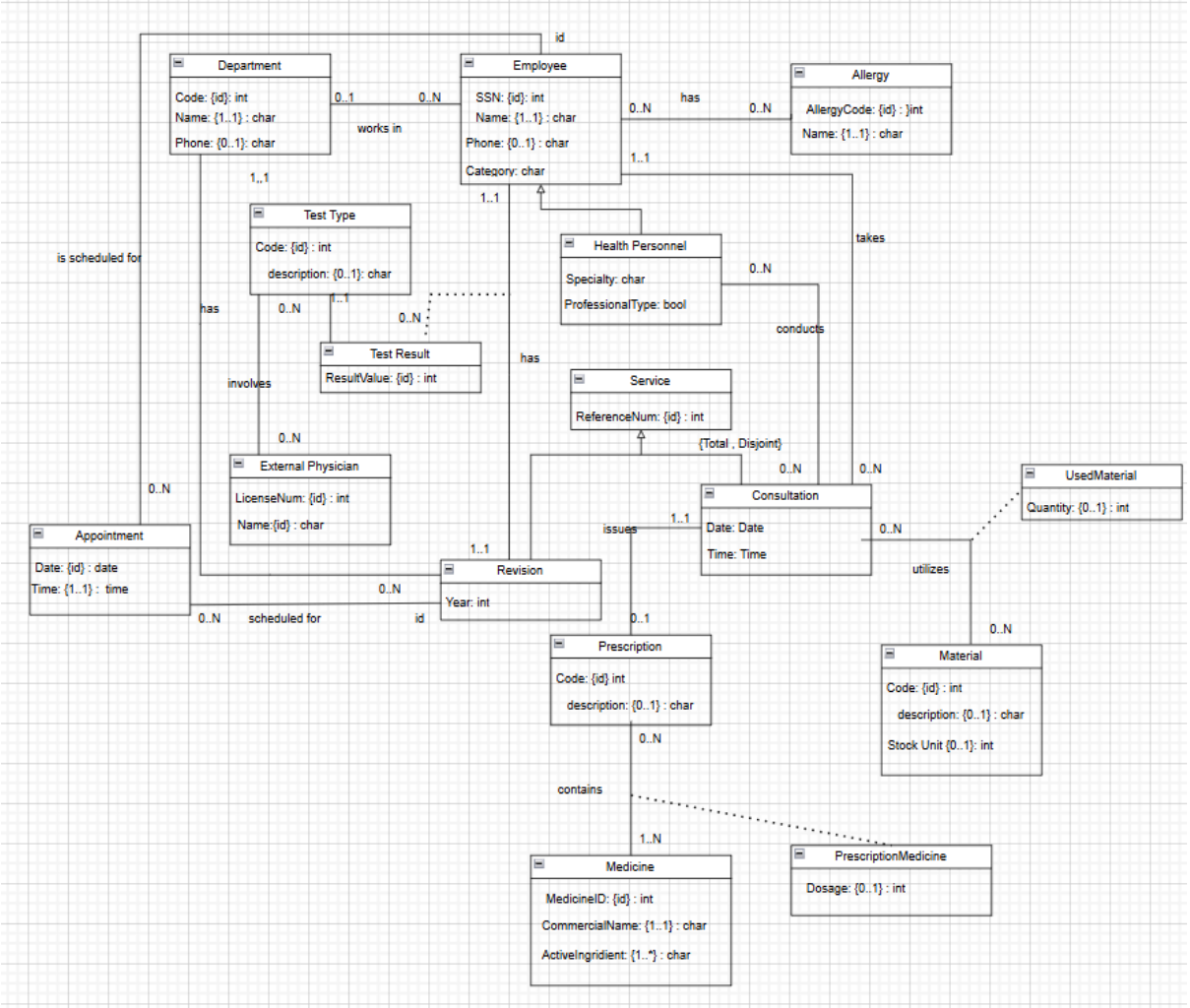
4. LIST OF INTEGRITY CONSTRAINTS

We identified the following integrity constraints from the problem description and must be enforced in the database system some of which appears in the diagram and other to be enforced in the logic.

- **Unique Identification:**
    - The Department Code uniquely identifies a department.
    - The Employee SSN uniquely identifies an employee.
    - The Service Reference Number uniquely identifies both Consultations and Revisions.
    - The Material Code, Medicine ID, and Allergy Code uniquely identify their respective objects.
- **Departmental Assignment:**
    - An employee must work in exactly one department.
- **Consultation Rules:**
    - A medical consultation must define the Date, Time, and Reason.
    - A consultation must always have at least one doctor or nurse associated with it .
- **Revision and Appointment Rules:**
    - A Medical Revision corresponds to exactly one Department.
    - Appointments for a revision are scheduled in groups of **no more than ten** employees on the same date and time.
    - An employee is only allowed one revision per year.
- **Data Integrity:**
    - Material stock cannot be negative (StockUnits >= 0). We'll enforce this with a DB check constraint.
    - Test Results must be associated with a valid Test Type, a valid Employee, and a valid Revision context.

5. CORRECTED UML DIAGRAM

Here, is the corrected version of our previous UML diagram with necessary changes made to it:



6. RELATIONAL SCHEMA DESIGN

The following relational schema was obtained by translating the UML class diagram into relations using the notation defined in the our course of relational schema designs:

- **Department Relation**

  Department (Code: INT, Name: VARCHAR(50), Phone: VARCHAR(15))

  PK: {Code}

- **Employee Relation**

  Employee (SSN: INT, Name: VARCHAR(50), Phone: VARCHAR(15), Category: VARCHAR(50), DeptCode: INT)

  PK: {SSN}

  FK: {DeptCode} → Department(Code)

- **HealthPersonnel Relation**

  HealthPersonnel (SSN: INT, Specialty: VARCHAR(50), ProfessionalType: BOOLEAN)

  PK: {SSN}

  FK: {SSN} → Employee(SSN)

- **Allergy Relation**

  Allergy (AllergyCode: INT, Name: VARCHAR(100))

  PK: {AllergyCode}

- **EmployeeAllergy Relation**

  EmployeeAllergy (SSN: INT, AllergyCode: INT)

  PK: {SSN, AllergyCode}

  FK: {SSN} → Employee(SSN)

  FK: {AllergyCode} → Allergy(AllergyCode)

- **Service Relation**

  Service (RefNum: INT)

  PK: {RefNum}

- **Consultation Relation**

  Consultation (RefNum: INT, Date: DATE, Time: TIME, Reason: VARCHAR(100), PatientSSN: INT)

  PK: {RefNum}

  FK: {RefNum} → Service(RefNum)

  FK: {PatientSSN} → Employee(SSN)

- **Revision Relation**

  Revision (RefNum: INT, Year: INT, DeptCode: INT)

  PK: {RefNum}

  FK: {RefNum} → Service(RefNum)

  FK: {DeptCode} → Department(Code)

- **Appointment Relation**

  Appointment (RefNum: INT, Date: DATE, Time: TIME, SSN: INT)

  PK: {RefNum, Date, Time, SSN}

  FK: {RefNum} → Revision(RefNum)

  FK: {SSN} → Employee(SSN)

- **TestType Relation**

  TestType (Code: INT, Description: VARCHAR(100))

  PK: {Code}

- **ExternalPhysician Relation**

  ExternalPhysician (LicenseNum: INT, Name: VARCHAR(100))

  PK: {LicenseNum}

- **TestType_ExternalPhysician Relation**

  TestType_ExternalPhysician (TestCode: INT, LicenseNum: INT)

  PK: {TestCode, LicenseNum}

  FK: {TestCode} → TestType(Code)

  FK: {LicenseNum} → ExternalPhysician(LicenseNum)

- **TestResult Relation**

  TestResult (SSN: INT, RefNum: INT, TestCode: INT, ResultValue: INT)

  PK: {SSN, RefNum}

  FK: {SSN} → Employee(SSN)

  FK: {RefNum} → Revision(RefNum)

  FK: {TestCode} → TestType(Code)

- **Material Relation**

  Material (Code: INT, Description: VARCHAR(100), StockUnit: INT)

  PK: {Code}

- **UsedMaterial Relation**

  UsedMaterial (RefNum: INT, MaterialCode: INT, Quantity: INT)

  PK: {RefNum, MaterialCode}

  FK: {RefNum} → Consultation(RefNum)

  FK: {MaterialCode} → Material(Code)

- **Prescription Relation**

  Prescription (Code: INT, Description: VARCHAR(100), RefNum: INT)

  PK: {Code}

  FK: {RefNum} → Consultation(RefNum)

- **Medicine Relation**

  Medicine (MedicineID: INT, CommercialName: VARCHAR(100), ActiveIngredient: VARCHAR(100))

  PK: {MedicineID}

- **PrescriptionMedicine Relation**

  PrescriptionMedicine (PrescriptionCode: INT, MedicineID: INT, Dosage: INT)

  PK: {PrescriptionCode, MedicineID}

  FK: {PrescriptionCode} → Prescription(Code)

  FK: {MedicineID} → Medicine(MedicineID)

7. NORMALIZATION

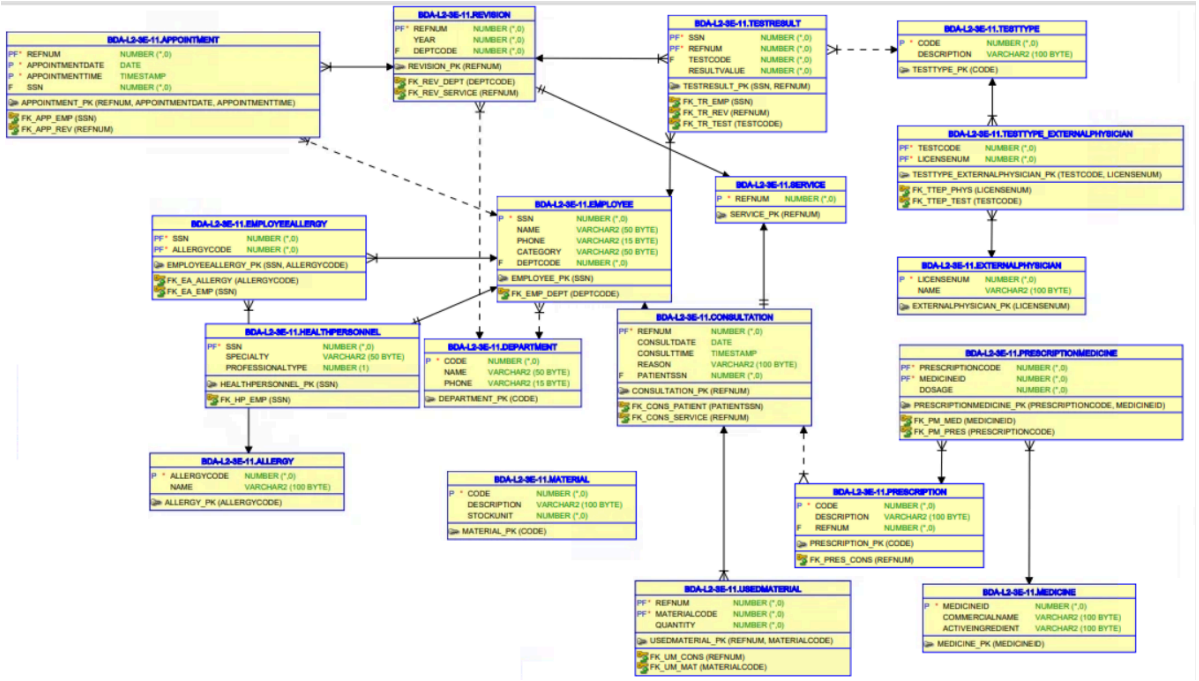
All the relations we have proposed schema satisfy Third Normal Form (3NF).

- All tables are in First Normal Form because all the attributes contain atomic values.
- The above schema is in Second Normal Form since all non key attributes depend only on the whole primary key.
- The above schema is in Third Normal form because there are no transitive dependencies meaning non key attributes depend only on the primary key.

Hence, no further decomposition of the tables were necessary.

8. ORACLE RELATIONAL SCHEMA

After we created the tables in Oracle SQL Developer, the below relational schema was generated using the Oracle Data Modeler tool.The below schema diagram visually confirms the correct creation of the tables, primary keys, foreign keys, and integrity constraints.



9. DATA LOADING PROCESS

We inserted the data for the relational schema using SQL insert statements within a single transaction. We choose this approach to ensure consistency and to avoid intermediate integrity violations during the insertion process.

Before loading the data, the foreign key constraints were defined as Deferrable during table creation. This allowed the enforcement of referential integrity constraints to be temporarily deferred until the end of the transaction.

All INSERT statements were enclosed within a BEGIN..END block and finalized with a COMMIT. The insertion followed a logical order that respected dependencies between the relations. Independent entities were inserted first followed by the dependent entities which ensured that all references were valid at the end of the transaction.

10. ORACLE USER CREDENTIALS

Here is the user credentials used for the database:

User: BDA-L2-3E-11

Password: hambrepersonas