Household Bill Management Software

DATABASE PROJECT

1. Project Description

1.1. Requirements

The objective is to develop an application that provides customers with the ability to connect and view the invoices they receive. Additionally, for each registered user of the application, it is required to maintain a record of their contracts with a provider, based on which invoices can be generated. The contracts should be viewable along with the details of the purchased service.

1.2. Requirement Analysis

There is a need for knowledge about individuals, suppliers, contracts, and invoices. Thus, for an individual, essential information such as their full name, phone number, home address, and possibly email address needs to be obtained. Regarding suppliers, information about their name, services, and current rates is required. For contracts, information about the start date, end date, and the parties involved (supplier and individual) needs to be known. Lastly, for invoices, it is necessary to have knowledge of the contract upon which it is generated, the date of receipt, the due date, consumption details, and the total payment amount.

2. Used Technologies

For the user interface part, the implementation considered the use of the cx_Oracle library for establishing the connection with the database, the PyQt6 package and the Qt library for creating windows, buttons, tables, etc., and the Python programming language, specifically version 3.9.
The logical and relational model creation, as well as the generation of the necessary table codes, were accomplished through the Oracle SQL Developer Data Modeler tool. The actual creation of the tables was performed using Oracle SQL Developer by executing the previously generated script.

3. Entity-Relationship Model Development

The identified entities are:

- *Persons*
- *Providers*
- *Services*
- *Services_Descriptions*
- *Contracts*
- *Invoices*
- *Tariff_Types*

Among these entities, we distinguish all three types of relationships: 1:1 (one-to-one), 1:m (one-to-many), and m:m (many-to-many), as follows:

For the entities Services and Services_Descriptions, the established relationship is 1:1. We interpret the relationship as follows: a service can have a description, indicating optionality at the Services entity end; a service description is assigned to a service, indicating a mandatory requirement at the Services_Descriptions entity end.

The Providers entity is in an m:m relationship with the Services entity. A provider can offer one or more services - optional. A service can be offered by one or more providers; it is mandatory. Therefore, because a contract must be based on a service offered by a provider, a new entity (Providers_Services) needed to be introduced within the logical model. This entity uniquely captures a provider, a type of service offered, and an attribute called price, which becomes the property of this relationship. Consequently, the new entity will be in the second normal form since the new attributes depend on its unique identifier. Providers_Services has a 1:m relationship with the Providers and Services tables, where non-transferability is required because the data on which a contract directly depends should not be modified during the process.

From an abstract perspective, Contracts represent an intermediate entity formed based on the m:m relationship between people and the services provided by a provider. Thus, the Persons entity creates a 1:m relationship with the Contracts entity, which possesses the non-transferability property since the person of a contract cannot be changed. A person can have one or more contracts, describing optionality, and a contract must be owned by a person, describing obligatoriness. The other 1:m relationship is characteristic of the Contracts and Providers_Services entities and is described as follows: A contract must be concluded by the provider for the services offered. A provider can offer services by concluding one or more contracts. Again, this relationship has the property of non-transferability because the provider (with the associated services) who concluded the contract cannot be modified.

The relationship between the Contracts and Invoices entities is described by the following statements: A contract can generate one or more invoices; an invoice must be generated based on a contract. Therefore, it is a 1:m relationship characterized by the non-transferability property since the contract identifier cannot be modified in the generated invoice.

Within the 1:m relationships, the relationship between the Services and Tariffs_Types entities is also included. A service must have a tariff type, and a tariff type must be assigned to one or more services. The connection between them is established based on the unique identifier attribute of the Tariffs_Types entity.

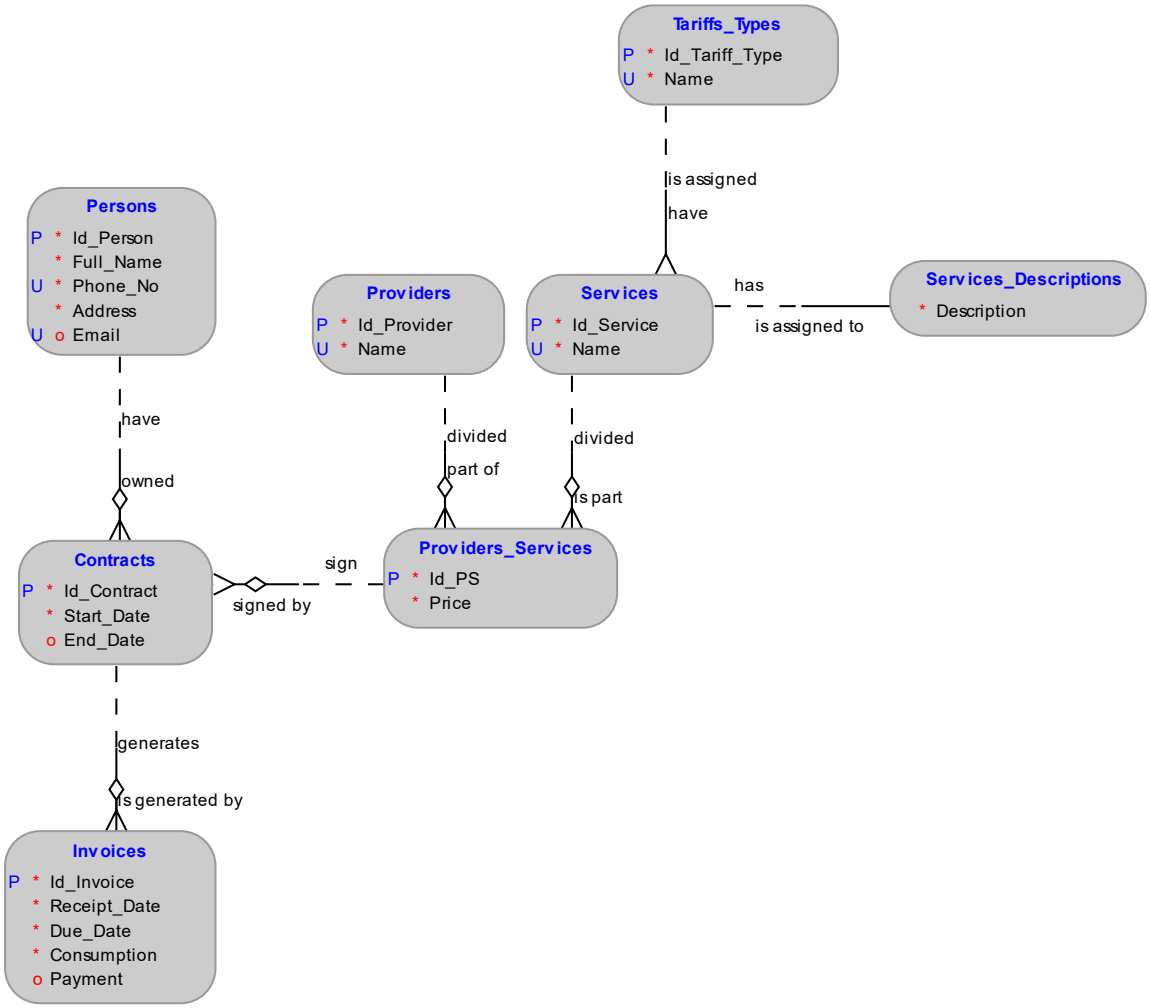


Figure 3.1. Logical model.

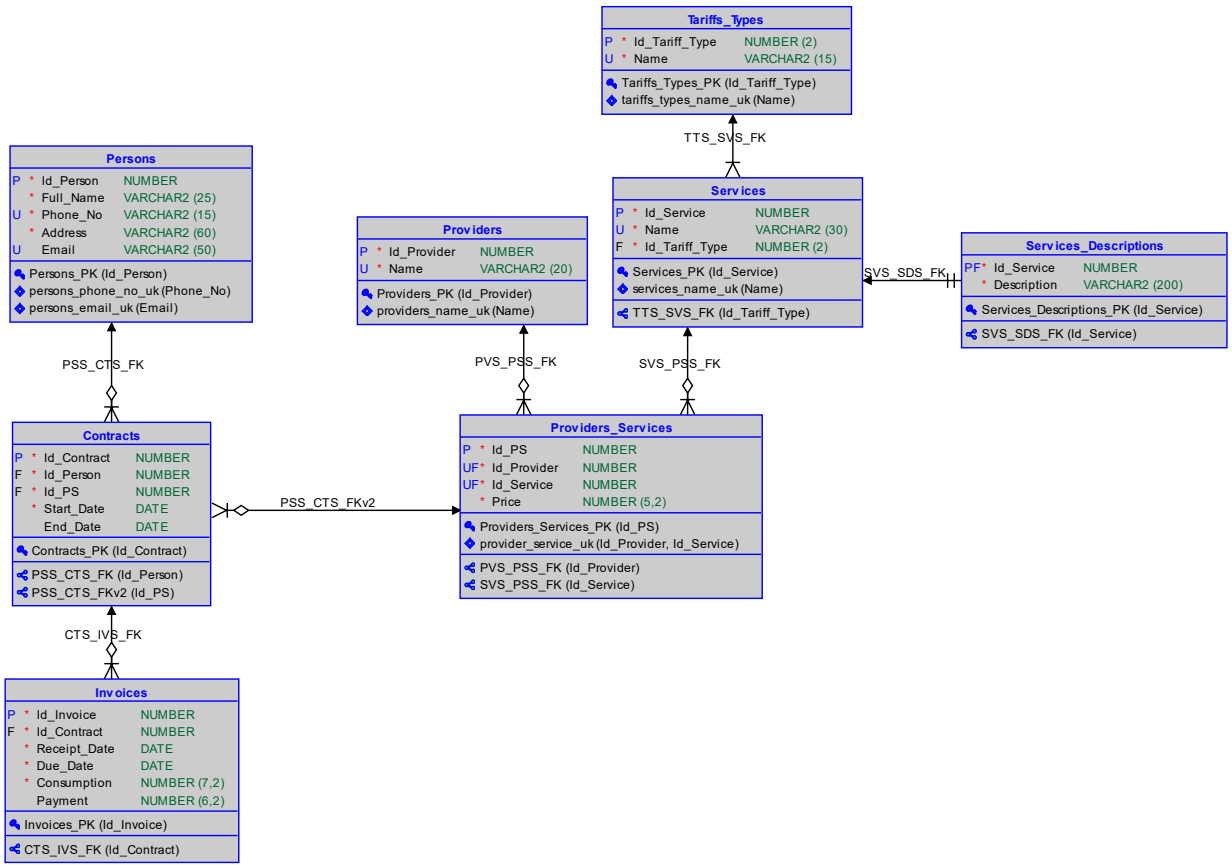


Figure 3.2. Relational model.

4. Description of Constraints

Each attribute of the entities presents at least one constraint on its value. Therefore, "not null" constraints are identified within all tables and serve to determine the necessity of a value for the newly created instance.

It is desired that the unique identifier of an entity represents a natural value, which should be incremented starting from 1. To achieve this, a check constraint will be applied to ensure that the corresponding value is greater than 0.

Since it is not possible for a price, consumption, or total payment to have a negative or zero value, it is necessary for their values to be greater than 0. Therefore, constraints are applied to verify the numeric value for the "price" attribute in the "Providers_Services" entity, as well as the "payment" and "consumption" attributes in the "Invoices" entity.

For a contract, it is essential that its start date does not exceed the current date, meaning that instances of future nature cannot be created. Additionally, the end date should be greater than the start date, as a contract cannot be concluded before it has started. Similarly, concerning a received invoice, it is important that the receipt date does not exceed the current date, and the due date, as naturally expected, must be greater than the receipt date. Hence, the need for introducing check constraints on the date attributes such as "start_date," "end_date" (Contracts), "receipt_date," and "due_date" (Invoices).

The values of certain attributes can be valid only if they adhere to a specific imposed format or are composed of certain character combinations. Consequently, it is identified that a person's name cannot contain special characters (numbers, punctuation marks, etc.), but only letters and separators like space or a hyphen (-). An email address must follow standardized rules, and for a phone number, it is necessary to have at least 10 characters, containing only digits and at most the plus character (+). Therefore, constraints based on regular expressions are established for the attributes "full_name," "email," and "phone_no" of the "Persons" entity.

An attribute that contains a character sequence will be considered valid only if its length is greater than 1. Hence, a constraint of this type will be applied to attributes such as "address" (Persons), "name" (Providers, Services, and Tariffs_Types), and "description" (Services_Descriptions).

Next, the uniqueness of entity attributes will be analyzed. It is evident that two people cannot have the same email address or phone number. For providers, their name must be unique, and the services they offer should not be repetitive since a service is expected to be sold at the same price. In the case of variations among user types (customers), the introduction of a new service type will be necessary. Additionally, it is required that a tariff type not be repetitive in order to maintain the entity in its normal form. These unique constraints will be applied to attributes such as "phone_no," "email" (Persons), and "name" (Providers, Services, and Tariffs_Types).

Considering that an invoice is generated based on a contract, it will be necessary to verify the validity of the contract, ensuring that it has not yet been concluded.

5. Connecting to the application's database

Regarding the database connection, a dialog interface is presented where the user is required to input the username and password.

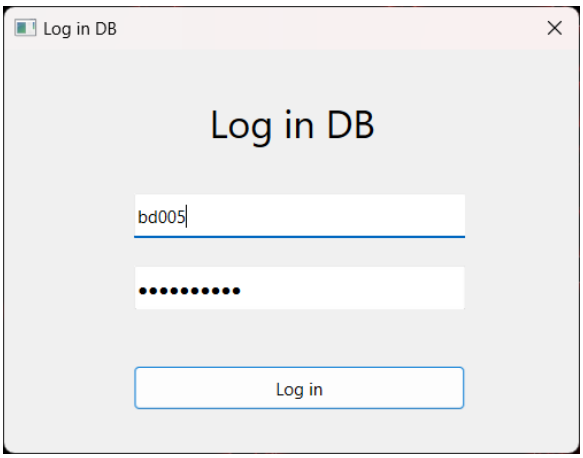
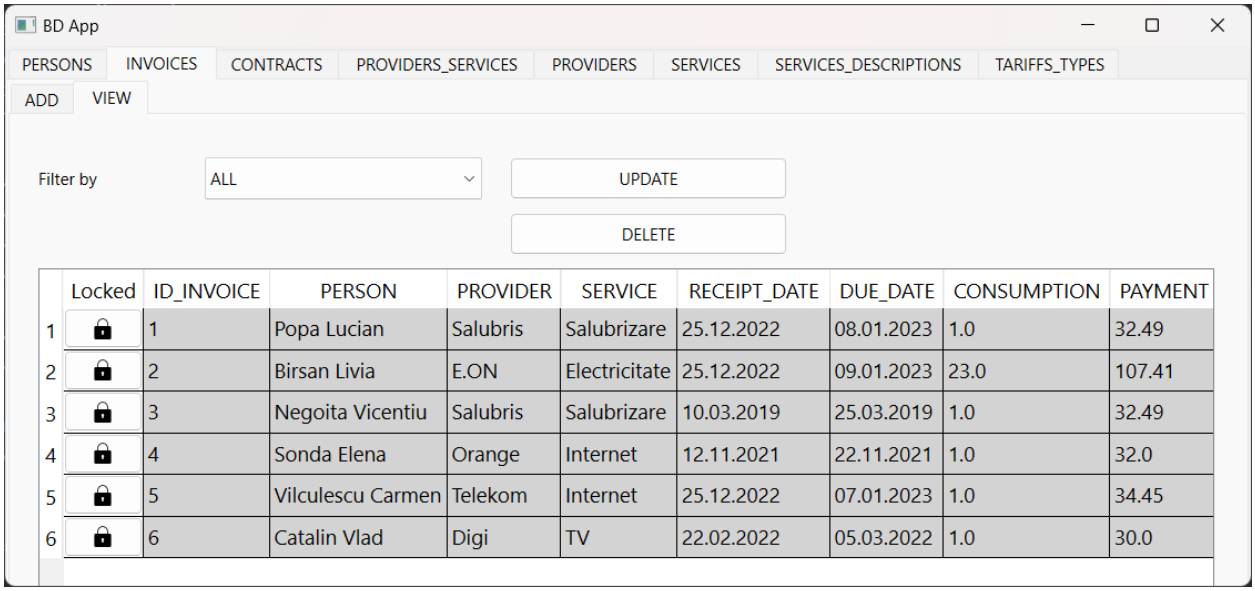


Figure 5.1. Dialog interface for connecting to the database.

To establish the connection, the Python library cx_Oracle is used, for which the folder containing the Oracle Instant Client files needs to be configured (the folder is downloaded locally according to the usage instructions). When the Log In button is pressed, the values of the fields will be retrieved and an attempt to connect will be made. If the data is invalid, a message with an appropriate error message will be displayed; otherwise, the application will be opened.

6. Application Interface Description

The application presents itself as a window that displays tabs for each table. These tabs, in turn, contain two sub-tabs (ADD and VIEW) through which the necessary operations can be performed. The VIEW sub-tab provides options for viewing (either in its entirety or filtered), deleting, and updating data within the table.



	Locked	ID_INVOICE	PERSON	PROVIDER	SERVICE	RECEIPT_DATE	DUE_DATE	CONSUMPTION	PAYMENT
1		1	Popa Lucian	Salubris	Salubrizare	25.12.2022	08.01.2023	1.0	32.49
2		2	Birsan Livia	E.ON	Electricitate	25.12.2022	09.01.2023	23.0	107.41
3		3	Negoita Vicentiu	Salubris	Salubrizare	10.03.2019	25.03.2019	1.0	32.49
4		4	Sonda Elena	Orange	Internet	12.11.2021	22.11.2021	1.0	32.0
5		5	Vilculescu Carmen	Telekom	Internet	25.12.2022	07.01.2023	1.0	34.45
6		6	Catalin Vlad	Digi	TV	22.02.2022	05.03.2022	1.0	30.0

Figure 6.1. Displaying all data from the Invoices table.

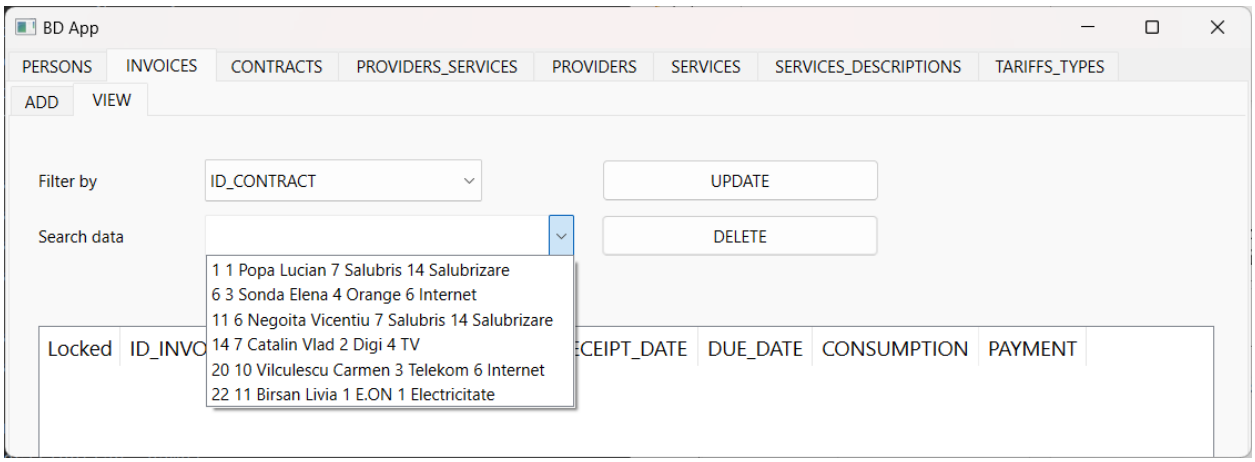


Figure 6.2. Filtering by selecting the foreign key.

Initially, all entries are locked (Figure 6.1). To edit or delete a record, the row will be unlocked using the button in the Locked column, and upon clicking either of the two buttons, an attempt will be made to update or delete the unlocked records. If this operation cannot be performed, an appropriate message will be displayed above the table.

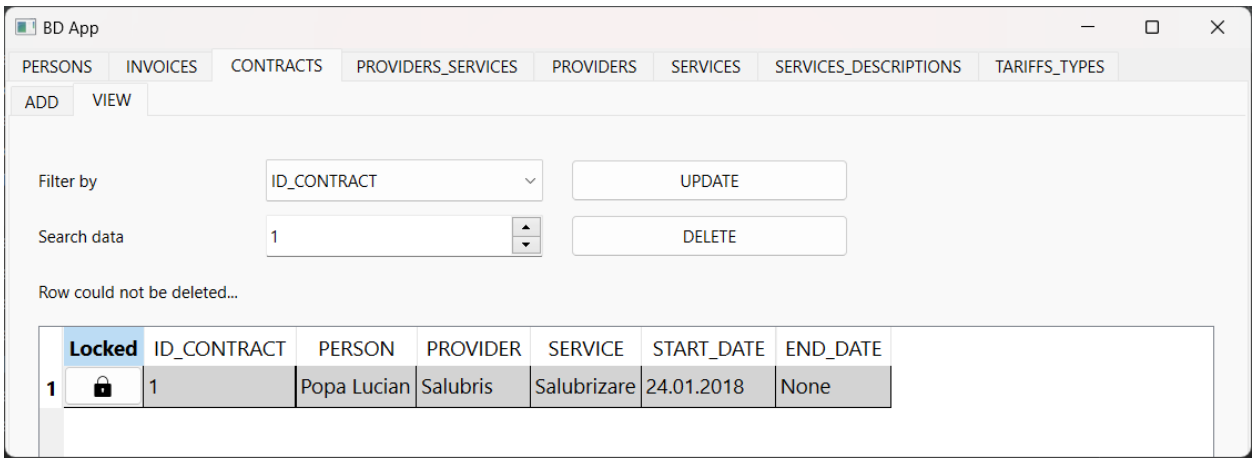


Figure 6.3. Attempting to delete a record from the Contracts table.

Adding records to the table is done through the ADD tab, which contains corresponding fields for each data type. For foreign keys, selection boxes are displayed showing the associated name of the key.

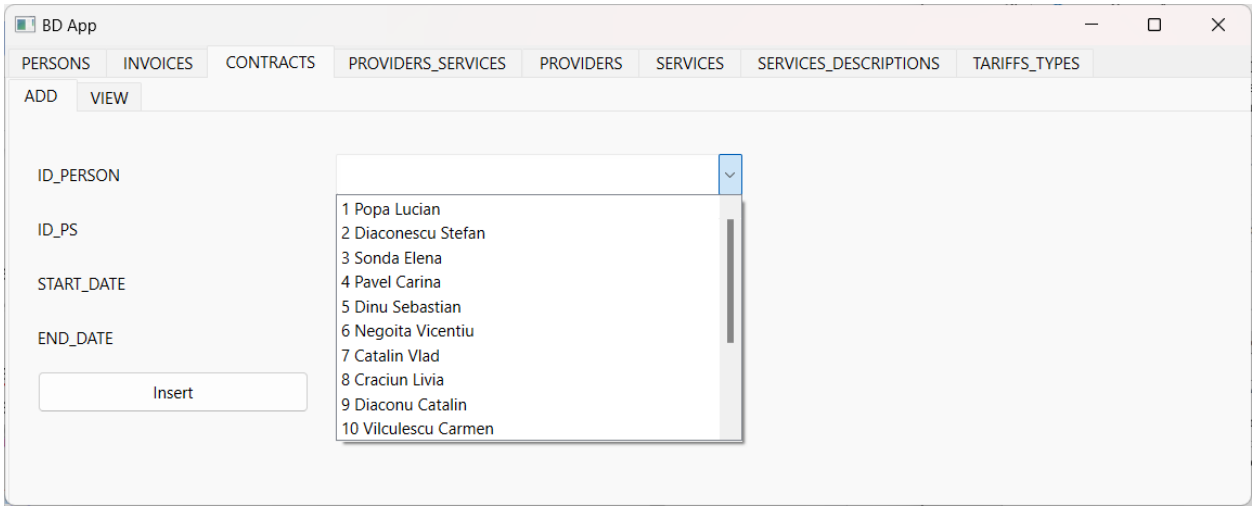


Figure 6.4. Selecting the person for whom a new contract is desired to be added.