Tools & technologies used for the system development

The system is based on the Client-Server architecture model. This model predicts the existence of the client, who requests information, and the server, which provides information. In commercial systems the client is executed on each user's computer and the server is usually a host capable of providing the information requested by the connected users-users of the system.

The development of the system includes:

A) The implementation of the database, based on the relational model presented in the 1st exercise, of a car rental company. The database will be located on the server computer.

B) The implementation of a user interface application (User Interface) through which the entries in the database will be entered, processed and deleted, as well as the presentation of the results of the execution of various queries to the database. The application will be run by the client and will run on the computers of system users.

Microsoft SQL Server (SS) was selected to implement the database, which is managed through SQL Server Management Studio (SSMS). SS was selected due to previous experience in developing databases and using SSMS to execute queries. SSMS is a complete database management system implemented in SS and offers a user-friendly interface and many tools for database management and visual design. SS is a relational database suitable for high-demand applications and is one of the main options for implementing either stand-alone or online applications. The development of the system in SS is offered for development and evolution of the existing knowledge on the specific database.

Microsoft Access (MA) was chosen for the implementation of the client interface, which is an application that integrates a database and a client development environment by providing tools for creating forms, queries and reports. For this system, however, the MA will only be used to create and execute the front-end (the back-end as mentioned will be in SS). The choice of MA was made again due to the existing knowledge and experience in its use but also in the programming knowledge in Visual Basic 6. MA is programmed in Visual Basic for Applications, which has a subset of VB6 capabilities but is offered for rapid development of simpler applications and is a language that can be used to program all applications in the Microsoft Office suite. MA as a database is intended for low-demand systems.

Access communication with SQL Server

MA communicates with SS via the Open DataBase Connectivity (ODBC) interface and ActiveX Data Objects (ADO) technology for VB6 and is now called ADO.NET for the .NET framework.

You need to create an ODBC statement in Windows with the following information:

Server Name	The name of the server
Default Database	CarRental
User Name	sa
Password	sa

The following are used to send queries to the database:

- connection string: A string containing the connection information to the database via ODBC
- Sql txt: String containing the query to be executed
- Adodb.connection: An object that represents the connection to the database
- Adodb.recordset: Object representing the results of query execution

And the order of use of the above to execute a query in the database:

- Create Adodb.connection
- Creating Adodb.recordset
- Connection string statement
- Sql txt statement
- Open the connection using the connection string
- Open the recordset using the connection and send to execute sql txt

The recordset object now contains all the results of the executed query and any processing follows. Then to free up system resources we have to close the 2 items and "destroy" them

Indicatively, the code that implements the above is the following:

```
Dim with As ADODB.Connection
Dim rst As ADODB.Recordset
Dim constring As String
Dim sqltxt As String

constring = "DSN = CarRental; Uid = sa; Password = sa"
sqltxt = "SELECT ID, NAME FROM [dbo]. [tbl_Branches]"

Set con = New ADODB.Connection
Set rst = New ADODB.Recordset

con.Open constring
rst.Open sqltxt, con, adOpenKeyset, adLockReadOnly

'
' Edit recordset contents
'
rst.Close
con.Close
Set rst = Nothing
Set con = Nothing
```

Base design

The design of the base includes the implementation of the relational model presented in the 1st exercise. Below are the steps followed to implement the database. Through SSMS it is possible to create all components in 2 ways: graphic (visual) and through SQL statements. Graphic mode is the most important feature that SSMS offers to users for the full implementation, configuration and management of their database. Both methods were used to familiarize themselves with the environment and with SQL commands. Note that even the use of the graphical interface actually creates SQL statements and executes them and also allows various options to create the command and wait for the user to simply enter the appropriate parameters and execute them.

Create all database tables

The key component of a database is the tables where the records will be stored. The creation of each table presupposes the creation of the fields that will be contained and for each field its type must be declared and if necessary the size in bytes that it can occupy. It should also be stated if a field is allowed to have a NULL value and if it needs to get a default value when entering a new record.

The above are restrictions that should always be stated for each field in the table. Constraints in table fields are also referred to as column constraints. By defining an appropriate data type, we limit the value that the user will be able to give in a new registration. The definition of the appropriate type for each field serves to save storage space on the system disk and at the same time to effectively manage the field through queries eg field that gets value only int if declared as nvarchar should first be converted to int and then an act is performed. But the most important thing in such a statement is that the user is now allowed to store alphanumeric characters in a field where only numeric characters should be entered, thus losing the stability and robustness of the database.

It should be noted that the control of the price that will be given by the user should be done in the registration form, in the customer application, so that the wrong price does not reach the base.

At table field level the most important constraint, also referred to as entity integrity, is the declaration of a field as the primary PK key. In the relational model PKs were already defined in each int table because this data type is better utilized by the database than someone else.

Finally, depending on the correlations of the table that it will have (and its type) with the rest, fields should be declared as external FKs keys, which in the next step will create the PK-FK references.

The DataDictionary.xlsx file lists the fields of all arrays with their constraints (data type, size, NULL usage, PK) as well as a description of each field.

Creating a database diagram

As mentioned above in the tables PKs and FKs should be declared. When creating the base diagram we declare the correlation of each table with another external table. Each FK should state which PK it refers to. These statements are referred to as referential restrictions. The type of restriction that should apply should be regulated with each statement. In essence, we declare in the database in each possible update or delete in the PK table what we want to happen in the FK table.

The following 4 statements can be made:

- No action: Do not take any action
- Cascade: Do the same for FK's panel
- Set Null: Replace the FK value with NULL
- Set default: Give a default value

In the database, after defining all the PK-FK reports provided by the relational diagram, in essence the following relations are created:

Relationships 1 to many

Table PK	**Table FK**
Tbl_Clients	Tbl_Reservations
Tbl_Employees	Tbl_Rentals (Deliverer, Receiver)
Tbl_Branches	Tbl_Employees
Tbl_Branches	Tbl_Cars
Tbl_Cars	Tbl_Damages
Tbl_Cars	Tbl

All these references have been declared as No Action, ie not to delete any entry in the PK record if there is a reference to a record in the FK table. Also do not run the Update command if the PK value is listed in the FK table. The latter was chosen because the IDs of the PKs arrays, apart from the int data type, have been declared to automatically take values in each new entry (Indentity) with ascending number, so in a possible update to a larger number the same ID may occur with the automatic process.

Now in order for a record to be deleted from the PK table there must be no reference to the specific ID in the FK table. Otherwise orphaned entries would be created in the FK boards.

1 to 1 relationships

PK	**Table FK Table**
Tbl_Clients	Tbl_Individuals
Tbl_Clients	Tbl_Organizations
Tbl_Clients	Tbl_Foreign
Tbl_Reservations	Tbl_Rentals

The above 1 to 1 ratios should have different settings than the one to many ratios. In essence, the relationship between Client, Individual, Organization and Foreign 1 to 1 could be a table from the beginning. It does not make sense to have to delete the registration from Individual first and then go to delete from the Client as well. So these relationships will be declared as Cascade and by deleting the registration in the Client to delete the associated registration by the other / others / others. The same goes for Reservation and Rental. This is for Delete because the Update should not be Cascade for the reason mentioned earlier with the key indentity.

The DataBase Diagram with all the links is presented in the CarRental.xps file

Indexes

By declaring a field (or combination of fields) as PK, an index is created directly on that field (or a combination of them). So for PKs of all boards indexes have already been created. To improve the performance of complex queries that use array arrays, it is advisable to index all FKs in the database.

Also because the TIN field is used in many forms to find the customer, another index was created on this field.

In the DataDictionary file there is a list of indexes that exist in the database separately for PKs and FKs and other fields.

Constraints

During the design of the base some restrictions were created as mentioned above. These constraints have to do with the structure of the array fields and the relationship of the arrays to each other.

Apart from these restrictions there are also user-defined or business rules restrictions. These are not related to the structure and operation of the database but are business rules that should apply. These constraints can be integrated into the client application but if they can be implemented on the database it is preferable.

Business rules:

- The type of vehicles can be: Mini Van, Truck, Car, ATV, Morocycle
- ; The end date of the rental can not be less than the start date.
- The car must be available upon delivery to another customer.
- Upon delivery it should be checked that the rent payment has already been made.
- Upon delivery, the mileage of the car should not have exceeded the mileage of the next service.

- Upon delivery, the rental expiration date should not exceed the security expiration date

Some of these rules seem self-evident and as mentioned, they should be checked in the client application as well. But if for some reason entries are made directly to the database, these restrictions will be very useful.

These restrictions apply to tbl_Cars, tbl_Reservations, and tbl_Rentals. Restrictions on the type of car and the start and end dates of the booking are very easy to set up as the fields are in the same table. Regarding the 3 constraints in the table tbl_Rentals, because there is a need to check fields in other tables, the use of functions is required to verify the constraint

The constraint on the table tbl_Cars is as follows:

- CK_CarType

The constraint on the table tbl_Reservations is as follows:

- CK_StartAndEndDates

And the constraints in the table tbl_Rentals

; CK_tbl_CarForAvailability using the function fn_CheckCarAvailability
; CK_tbl_CarsCheckKilometers using the function fn_CheckKilometersForService
; CK_tbl_CarsInsuranceDate using the fn_CheckInsuranceEndDate function
; CK_tbl_ReservationsForPayment using the fn_CheckForPayment function

Line 740 of the DDL starts the statements of the User-defined constraints. They have been commented so they will not be executed from the beginning because after the registrations are passed they may

issue messages of violation of the restrictions. declared as NOCHECK.