Politécnico
Castelo Branco
Escola Superior
de Tecnologia

CENTRO 2030 PORTUGAL 2030 Cofinanciado pela União Europeia
Os Fundos Europeus mais próximos de si.

Proposal Management System

20241130 – Diana-Patricia Toader
20241159 – Karla-Maria Boga
20241160 – Timeea Gota

Curso Técnico Superior Profissional em Redes e Sistemas Informáticos

1º Ano / 2º Semestre

UC: Bases de Dados – Parte I

Docente: Ricardo Fontes

2024/2025

General index

Figure index

1 Introduction

A database system is a tool that helps people store, organize, and work with information on a computer.

It has two main parts:

Database – This is where the information (data) is kept. Think of it like a digital filing cabinet.

DBMS (Database Management System) – This is the program that helps you add, change, find, or delete information in the database.

With a database, data can be saved—for example, information about addresses. This system allows the information to be easily found, updated, or deleted, and helps keep the data safe and secure.

In this project, the database system was used to implement the process of an administrative institution when it receives a proposal from citizens.

1.1 Objective

The aim of this work is to implement a **database system** that stores all relevant information regarding a project, including its **stages and execution phases**, starting from the moment a **proposal is submitted** to the final stage, if the project is approved and funded. From the moment the proposal is registered, the database should be able to **record and manage all opinions** related to the project, track **every stage of the process with exact dates,** and store detailed information about the **allocated budget** and the **companies responsible** for implementing each part of the project. In the final stage, the database must include a **work report** that contains a summary of the entire process, including every step the project went through from start to finish.

To implement this type of database, the first step was to **fully understand the requirements,** in order to design a structure that reflects the real-world process. This required creating a **logical network of all necessary entities,** using specific relationships that clearly show how each element of the system is connected. This network is known as an **Entity-Relationship (ER) Model.** The ER model was developed based on the information provided in the requirements and includes all essential **entities, attributes, and relationships** that allow the system to function effectively and efficiently.

The next step involved creating the **Relational Model**, which translates the ER model into a structure suitable for implementation in a database system. During this step, all the rules of relational databases were applied, including the correct use of **primary keys** to uniquely identify records and **foreign keys** to define the relationships between tables. These keys ensure data integrity and help maintain consistent links between entities.

Once the Relational Model was completed, it was used as the basis for **implementing the actual database system** in software. The goal was to ensure that all information related to the entities is **stored securely** and can be **easily accessed** when needed. To build the database, the query language **SQL (Structured Query Language)** was used. With SQL, specific commands were written to **create tables** and define the relationships between them according to the Relational Model.

A **Database Management System (DBMS)** was used to manage the implementation, allowing us to **populate the tables** with real data such as citizen information, submitted proposals, project stages, and execution phases. DBMS also provided the tools needed to manage, update, and maintain the data throughout the lifecycle of the project.

After all the necessary data was inserted into the database, we were able to begin **working with the system.** By using the defined tables and relationships, and applying specific SQL commands, we could access any required information about the project. This included checking the **current stage** of a proposal, viewing the **scheduled dates**, analyzing the **budget information,** and monitoring the **companies involved** in the execution. The system makes it easy to keep track of all aspects of the project and ensures that all important data is well organized and readily available.

1.2 Document structure

The document is composed of **four chapters**, each providing a detailed explanation of the process—from creating the Entity-Relationship (ER) model to working with the information stored in the database.

In **Chapter 1**: Introduction, a brief overview of the project development is presented. It includes essential background information, starting with the definition of a database system and DBMS, and ending with the final part of the project, which involves interacting with the registered data in the database.

Chapter 2: Modelling is divided into four sections:

- In **Section 2.1**, a short introduction to the modelling phase of the system is provided, explaining its importance in the overall development process.
- Section 2.2 presents the Entity-Relationship (ER) Model. Before introducing the actual model, all the entities that form the foundation of the ER are described. This section also includes the constraints required to ensure the system functions correctly and efficiently.
- **Section 2.3** provides the Relational Model, which consists of a complete list of all entities and their attributes, clearly indicating the primary keys and any foreign keys where applicable. This model also contains a list of relationships, including the names, keys, and attributes that connect two or more entities. Furthermore, this section gives a detailed explanation of each attribute, including its data type and any constraints applied.
- In **Section 2.4**, the **'Group Proportionality'** component is introduced. This section highlights the additional entity, relationship, and attribute that were developed as a contribution of the group to the project.

Chapter 3: Implementation consists of three sections:

- **Section 3.1** describes the creation of the tables using the SQL query language, along with the establishment of connections between the tables through primary and foreign keys.
- **Section 3.2** presents the SQL scripts used to answer specific questions. These scripts demonstrate how to access, manipulate, and retrieve data using appropriate SQL commands.
- **Section 3.3** includes the **'Group Proportionality'** questions and the corresponding SQL scripts developed to provide answers to those queries.

Finally, **Chapter 4**: Conclusion summarizes all the key aspects discussed throughout the document, offering a clear and concise overview of the project, from initial modelling to the final data operations.

2 Modelling

2.1 System Requirements

In many organizations and municipalities, proposals play a critical role in decision-making and project planning. However, without a well-structured system to manage them efficiently, tracking, evaluating, and executing proposals can become chaotic and error prone.

This system is designed to facilitate the **management of proposals**, ensuring streamlined processes from **creation** to **evaluation** and **implementation**.

2.2 Entity-Relationship Model

The E-R Model consists of **ten entities**, each playing a crucial role in the proposal management system:

- **Citizen** – Represents individuals submitting proposals.
- **Proposal** – Stores details of submitted projects.
- **Municipal Bodies** – Responsible for giving feedback about proposals.
- **Budget** – Manages financial allocations for approved proposals.
- **Work Report** – Summarises the entire journey of the proposal.
- **Council Services** – Analysis the proposal, defines dates for implementation and determines the total amount for implementation.
- **Stage** – Defines different stages a proposal goes through.
- **Execution Phase** – Represents specific phases within the execution process.
- **Consortia** – Groups of construction companies involved in proposal execution.
- **Construction Companies** – Companies involved in proposal execution.

These entities form the foundation of the system, ensuring efficient proposal tracking, evaluation, and execution.

During the implementation of the Entity-Relationship (ER) model, several constraints were applied to ensure the validity and consistency of the data.

The main constraints used in the table creation are:

- NOT NULL: Applied to essential fields related to citizens and proposals to ensure that all required information is provided.
- CHECK (age >= 18): Ensures that a citizen can submit a proposal only if they are of legal age.
- CHECK (start_impl < end_impl), CHECK (start_date < end_date), and CHECK (start_phase < end_phase): These constraints prevent logical errors by making sure that the end date of a phase or project cannot come before the start date.

These rules help maintain data integrity and avoid registration errors in the system.

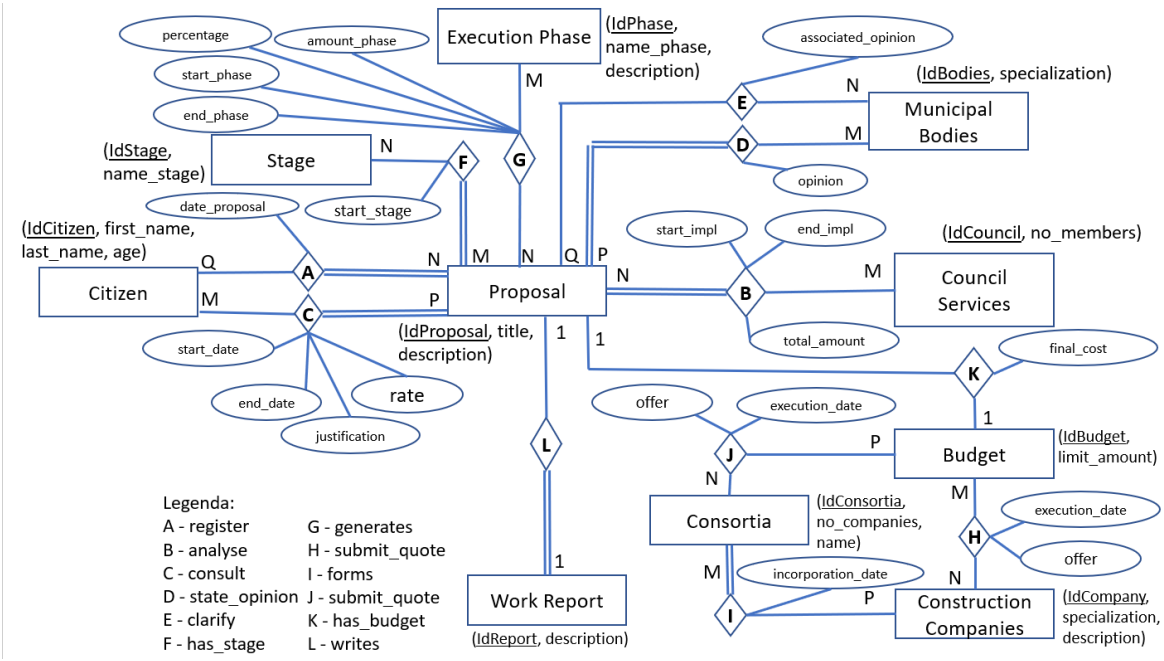

Figure 1 – Entity-Relationship Model

2.3 Relational Model

The relational model translates the Entity-Relationship model into relational tables, defining primary keys (PK), foreign keys (FK), and their relationships. This model ensures data integrity, normalization, and supports efficient querying and maintenance.

Entities:

Proposal (IdProposal, title, description,)

Citizen (IdCitizen, first_name, last_name, age)

Municipal_Bodies (IdBodies, specialization)

Council_Services (IdCouncil, no_members)

Execution_Phase(IdPhase, name_phase, description)

Stage (IdStage, name_stage)

Budget(IdBudget, limit_amount)

Construction_Companies (IdCompany, specialization, description)

Consortia(IdConsortia, no_companies, name)

Work_report (IdReport, description, IdProposal (L))

Relationships:

A: Register (IdProposal(FK), IdCitizen(FK), date_proposal)

B: Analyse (IdProposal(FK), IdCouncil(FK), start_impl, end_impl, total_amount)

C: Consult (IdProposal(FK), IdCitizen(FK), start_date, end_date, rate, justification)

D: State_opinion (IdProposal(FK), IdBodies(FK), opinion)

E: Clarification (IdProposal(FK), IdBodies(FK), associated_opinion)

F: Has (IdProposal(FK), IdStage(FK), start_stage)

G: Generates (IdProposal(FK), IdPhase(FK), start_phase, end_phase, amount_phase, percentage)

H: Submit_quote (IdBudget(FK), IdCompany(FK), execution_date, offer)

I: Forms (IdConsortia(FK), IdCompany()FK, incorporation_date)

J: Submit_quote (IdBudget()FK, IdConsortia(FK), execution_date, offer)

K: Has (IdProposal(FK), IdBudget(FK), final_cost)

2.3.1 Data Model

This section presents the data model in the form of structured tables, specifying the name, data type, description and constrains of each attribute within the system. The purpose of this model is to ensure clarity, consistency, and a standardized format for implementing the database in a relational database management system

Tabel: **Proposal** – This table is used to record the proporsals.

Campo	Data Type	Description	Observations
IdProposal	INT	Unique identifier of the proposal.	Primary Key, doesn't admit NULL values, unique
title	Varchar(30)	Title of the proposal.	doesn't admit NULL values
description	Varchar(100)	Description about the project.	Defaul NULL

Tabel: **Citizen** – This table is used to record the citizens.

Campo	Data Type	Description	Observations
IdCitizen	INT	Unique identifier of the citizen.	Primary Key, doesn't admit NULL values, unique
first_name	Varchar(30)	First name of the citizen.	
last_name	Varchar(30)	Surname of the citizen.	
age	INT	Age of the citizen.	Age>=18

Tabel: **Municipal_Bodies** – This table is used to record the municipal bodies.

Campo	Data Type	Description	Observations
IdBodies	INT	Unique identifier of the municipal bodies.	Primary Key, doesn't admit NULL values, unique
specialization	Varchar(30)	Specialization of the municipal bodie.	doesn't admit NULL values

Tabel: **Council_Services** – This table is used to record the council services.

Campo	Data Type	Description	Observations
IdCouncil	INT	Unique identifier of the council service.	Primary Key, doesn't admit NULL values, unique
no_members	INT	Number of members in a council.	doesn't admit NULL values

Tabel: **Execution_Phase** – This table is used to record the execution phase.

Campo	Data Type	Description	Observations
IdPhase	INT	Unique identifier of the execution phase.	Primary Key, doesn't admit NULL values, unique
name_phase	Varchar(30)	Name of the phase.	doesn't admit NULL values
description	Varchar(100)	Description about the execusion phase.	

Tabel: **Stage** – This table is used to record the stage each proposal is in.

Campo	Data Type	Description	Observations
IdStage	INT	Unique identifier of the stage.	Primary Key, doesn't admit NULL values, unique
name_stage	Varchar(30)	Name of the stage.	

Tabel: **Budget** – This table is used to record the budget.

Campo	Data Type	Description	Observations
IdBudget	INT	Unique identifier of the budget.	Primary Key, doesn't admit NULL values, unique
limit_amount	INT	Limit of money alocated to one project.	

Tabel: **Construction_Companies** – This table is used to record the construction comanies.

Campo	Data Type	Description	Observations
IdCompany	INT	Unique identifier of the construction company.	Primary Key, doesn't admit NULL values, unique
specialization	Varchar(30)	Specialization of the company.	
description	Varchar(100)	Description about the company.	

Tabel: **Consortia** – This table is used to record the consortia.

Campo	Data Type	Description	Observations
IdConsortia	INT	Unique identifier of the consortia.	Primary Key, doesn't admit NULL values, unique
no_companies	INT	Number of construction companies that forms the consortia.	
name	Varchar(30)	Name of the consortia	

Tabel: **Work_report** – This table is used to record the work report.

Campo	Data Type	Description	Observations
IdReport	INT	Unique identifier of the work report.	Primary Key, doesn't admit NULL values, unique
description	Varchar(100)	Description of the work report.	
IdProposal	INT	Identifier number of the proposal the work report is about.	Foreign Key references to Proposal table, doesn't admit NULL values

Tabel: **Register** – This table records the submission of proposals by citizens. It captures which citizen submitted which proposal and on what date, serving as a historical record of proposal registrations.

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier of the registration.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdCitizen	INT	One of the unique identifier of the registration.	Primary Key, Foregn Key referances to Citizen table, doesn't admit NULL values
date_proposal	DATE	Date of proposal	

Tabel: **Analyse** – Tracks the analysis performed by council services on a proposal

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier of the analysation.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdCouncil	INT	One of the unique identifier of the analysation.	Primary Key, Foregn Key referances to Council table, doesn't admit NULL values
start_impl	DATE	Startin date for implementation of the project defined by the council.	start_impl<end_impl
end_impl	DATE	End date for implementation of the project defined by the council.	end_impl>start_impl
total_amount	INT	The total amount of money for financing a proposal.	

Tabel: **Consult** – enables citizen feedback through ratings and justifications

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier of the consult.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdCitizen	INT	One of the unique identifier of the consult.	Primary Key, Foregn Key referances to Citizen table, doesn't admit NULL values
start_date	DATE	Startin date for consultation by the residence.	start_date<end_date
end_date	DATE	End date for consultation by the residence.	end_date>start_date
rate	INT	The rating given by the citizen for one proposal.	
justification	Varchar(100)	The justification about one proposal.	

Tabel: **State_opinion** – ensure municipal bodies provide opinions.

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdBodies	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Municipal Bodies table, doesn't admit NULL values
opinion	Varchar(100)	The opinion given by the Municipal Bodies about a proposal.	

Tabel: **Clarify** – ensure municipal bodies provide any necessary clarifications.

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdBodies	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Municipal Bodies table, doesn't admit NULL values
associated_opinion	Varchar(100)	The associated opinion given by the Municipal Bodies about a proposal.	

Tabel: **Has_stage** - tracks proposal progression through different stages

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdStage	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Stage table, doesn't admit NULL values
start_stage	DATE	The date when the proposal reach a specific stage.	

Tabel: **Generates** - handles proposal transitions into execution phases, managing timing and budget distribution.

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdPhase	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Execusion_Phase table, doesn't admit NULL values
start_phase	DATE	The date when the proposal reach an execution phase.	start_pahse<end_phase
end_phase	DATE	The date when the proposal reach an execusion phase.	end_phase>start_phase
amount_phase	INT	The money alocated each phase.	
percentage	DECIMAL(5,2)	Percentage in terms of fines in the event of non-compliance.	

Tabel: **Submit_Quote_Company** - governs budget submissions by construction companies.

Campo	Data Type	Description	Observations
IdBudget	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Budget table, doesn't admit NULL values
IdCompany	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Construction_Companies table, doesn't admit NULL values
execution_date	DATE	The date when the proposal reach an execution phase.	
offer	INT	The offer that each company makes.	

Tabel: **Submit_Quote_Consortia** - governs budget submissions by consortia.

Campo	Data Type	Description	Observations
IdBudget	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Budget table, doesn't admit NULL values
IdConsortia	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Consortia table, doesn't admit NULL values
execution_date	DATE	The date when the proposal reach an execution phase.	
offer	INT	The offer that each consortia makes.	

Tabel: **Forms** - establishes consortia partnerships for implementation.

Campo	Data Type	Description	Observations
IdConsortia	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Consortia table, doesn't admit NULL values
IdCompany	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Construction_Companies table, doesn't admit NULL values
Incorporation_date	DATE	The date when the consortia was made	

Tabel: **Has_Budget** - tracks proposal progression through different budgets.

Campo	Data Type	Description	Observations
IdProposal	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Proposal table, doesn't admit NULL values
IdBudget	INT	One of the unique identifier.	Primary Key, Foregn Key referances to Budget table, doesn't admit NULL values
final_cost	INT	Final cost for a project.	

2.4 Modelling considering 'Group Proportionality'

The contributions added were:

- 1 new entity (Neigborhood)
- 1 relationship (between Neigborhood and Citizen)
- 1 attribute (added in entity Construction Companies)

2.4.1 Atualização do modelo ER

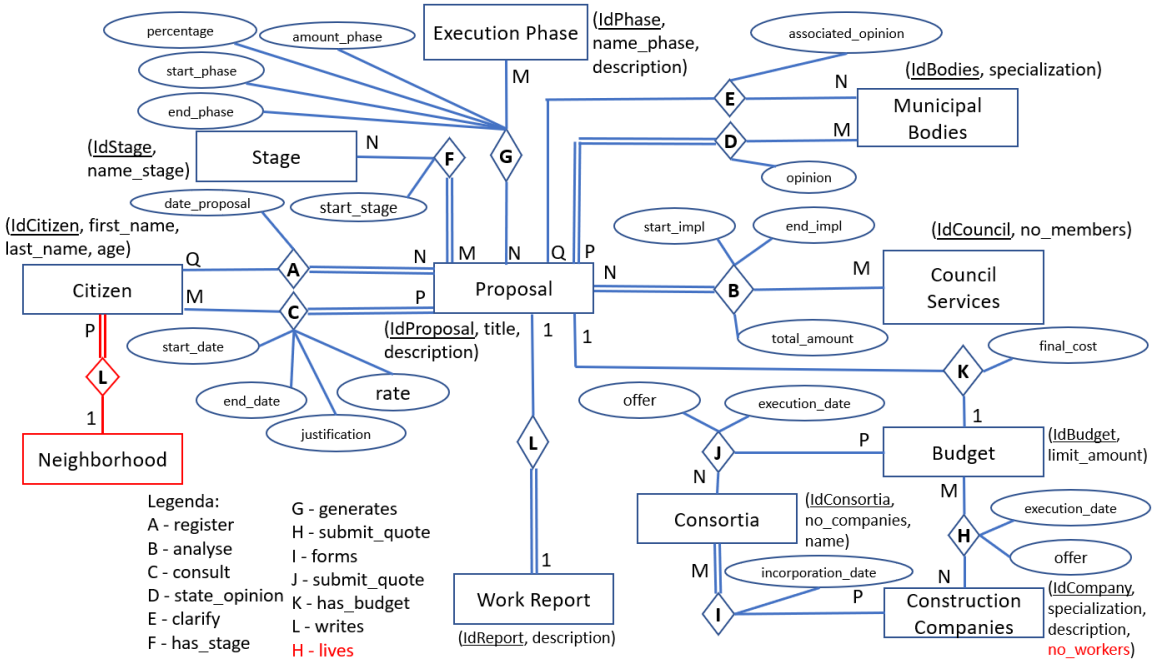

Figure 2 – Updated Entity-Relationship Model

2.4.2 Relational model actualization

Entities:

Proposal (IdProposal, title, description,)

Citizen (IdCitizen, first_name, last_name, age, IdNei(M))

Municipal_Bodies (IdBodies, specialization)

Council_Services (IdCouncil, no_members)

Execution_Phase(IdPhase, name_phase, description)

Stage (IdStage, name_stage)

Budget(IdBudget, limit_amount)

Construction_Companies (IdCompany, specialization, description, no_workers)

Consortia(IdConsortia, no_companies, name)

Work_report (IdReport, description, IdProposal (L))

Neighborhood (IdNei, street, number)

Relationships:

A: Register (IdProposal(FK), IdCitizen(FK), date_proposal)

B: Analyse (IdProposal(FK), IdCouncil(FK), start_impl, end_impl, total_amount)

C: Consult (IdProposal(FK), IdCitizen(FK), start_date, end_date, rate, justification)

D: State_opinion (IdProposal(FK), IdBodies(FK), opinion)

E: Clarification (IdProposal(FK), IdBodies(FK), associated_opinion)

F: Has (IdProposal(FK), IdStage(FK), start_stage)

G: Generates (IdProposal(FK), IdPhase(FK), start_phase, end_phase, amount_phase, percentage)

H: Submit_quote (IdBudget(FK), IdCompany(FK), execution_date, offer)

I: Forms (IdConsortia(FK), IdCompany()FK, incorporation_date)

J: Submit_quote (IdBudget()FK, IdConsortia(FK), execution_date, offer)

K: Has (IdProposal(FK), IdBudget(FK), final_cost)

2.4.2.1 Modelo de Dados Atualizado

Tabel: **Neighborhood** – This table is used to record the neighborhood.

Campo	Data Type	Description	Observations
IdNei	INT	Unique identifier of the neighborhood.	Primary Key, doesn't admit NULL values, unique
street	Varchar(30)	Name of the street.	
number	INT	Address number.	
age	INT	Age of the citizen.	Age>=18

Tabel: **Construction_Companies** – This table is used to record the construction comanies.

Campo	Data Type	Description	Observations
IdCompany	INT	Unique identifier of the construction company.	Primary Key, doesn't admit NULL values, unique
specialization	Varchar(30)	Specialization of the company.	
description	Varchar(100)	Description about the company.	
no_workers	INT	Numbers of workers in each construction company.	

Tabel: **Citizen** – This table is used to record the citizens.

Campo	Data Type	Description	Observations
IdCitizen	INT	Unique identifier of the citizen.	Primary Key, doesn't admit NULL values, unique
first_name	Varchar(30)	First name of the citizen.	
last_name	Varchar(30)	Surname of the citizen.	
age	INT	Age of the citizen.	Age>=18
IdNei	INT	Indicates the address of the citizen.	Foregn Key references to Neighborhood table, doesn't admit NULL values.

3 Implementation

In this part is presented the implementation phase of the database system. It is designed to manage and track proposals, citizen participation, execution of the process in municipal environment.

The implementation follows the relational model strictly. During the design stage there are a wide range of constrains to ensure consistency and integrity:

- Domain constrains (data types and value limits)
- Primary and Foreign key constrains
- General constrains (CHECK conditions for specific fields)

3.1 Tabel Creation Script

This section begins with the creation of scripts for all the tables existing, highlighting how the relational model is translated into SQL code. At the same time the population of the table with relevant initial data is made to simulate realistic usage scenarios.

3.1.1 Implemented Relational Model

For implementation we used SQL. It is a reliable tool that helps to store and organize the data correctly. It makes sure that all the information follows the rules we set, like using the right types of data and keeping connections between tables valid.

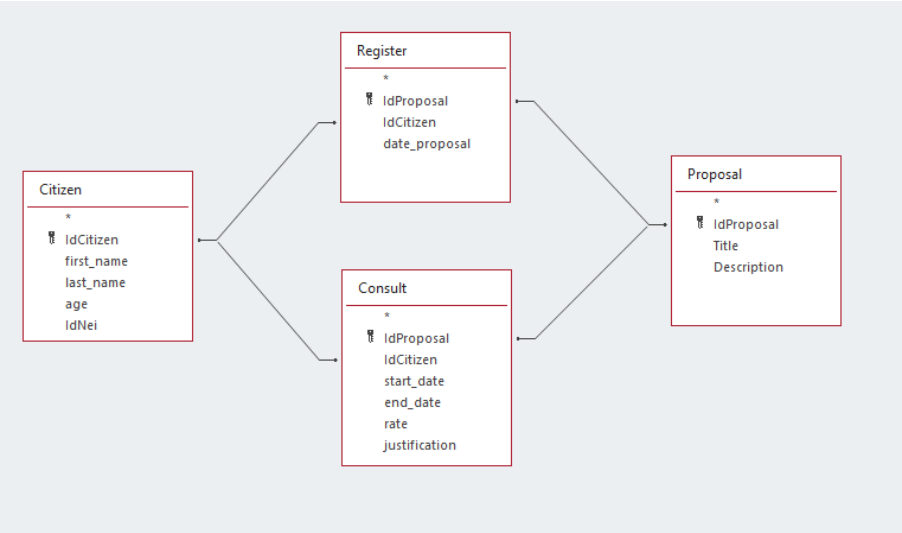

Figure 1 – relations A (Register) and C (Consult)

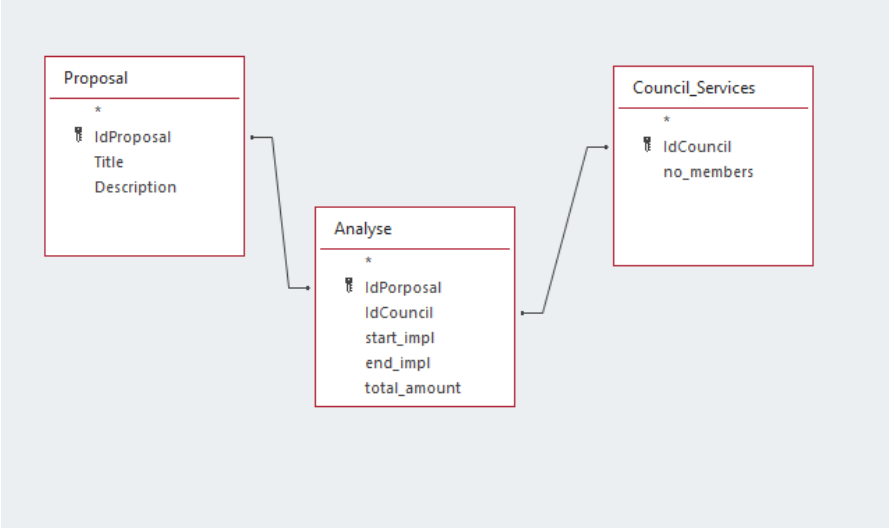

Figure 2 – relation B (Analyse)

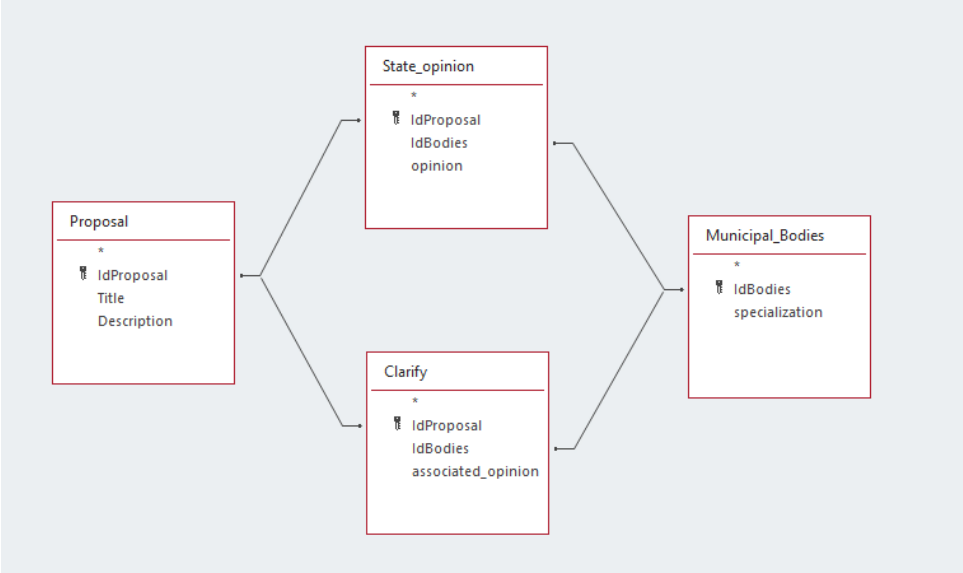

Figure 3 – relations D (State_opinion) and E (Clarify)

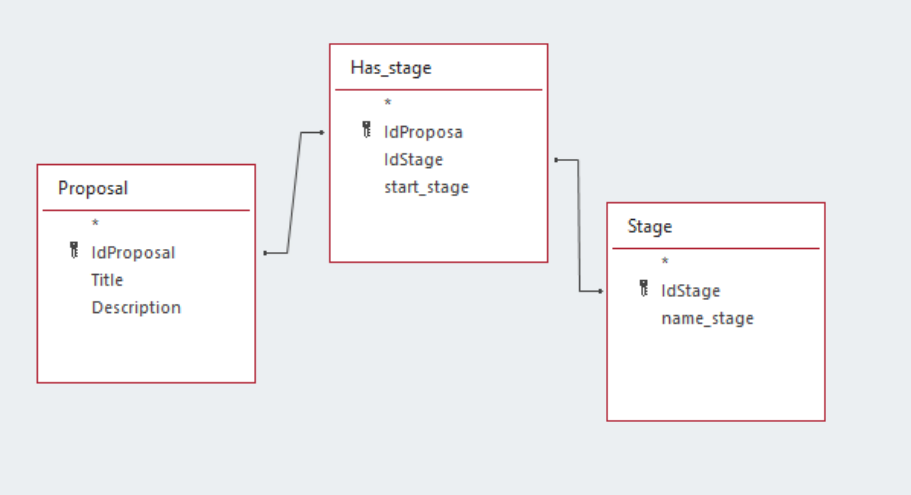

Figure 4 – relation F (Has_Stage)

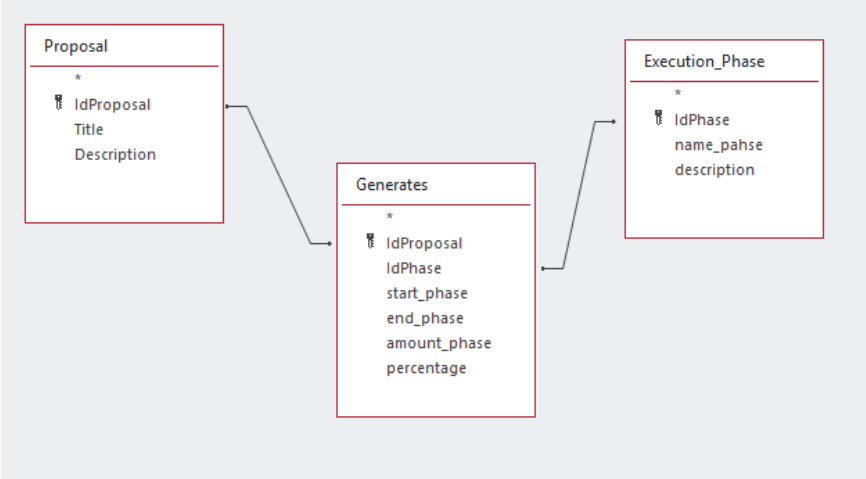

Figure 5 – relation G (Generates)

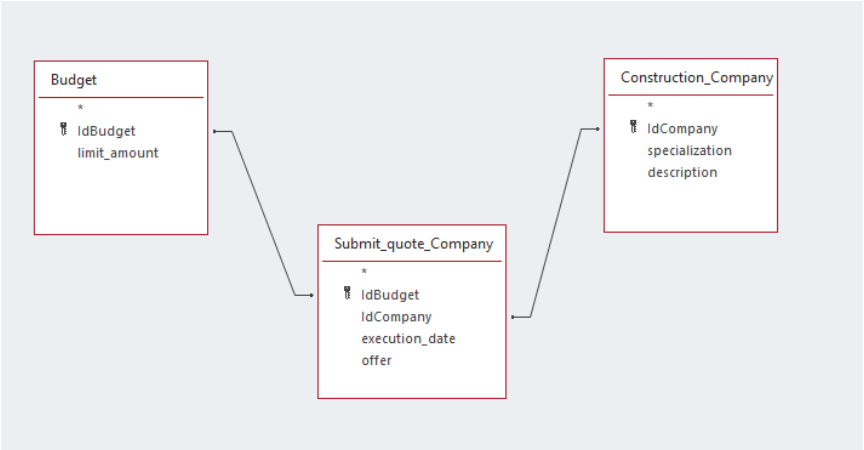

Figure 6 – relation H (Submit_quote_Company)

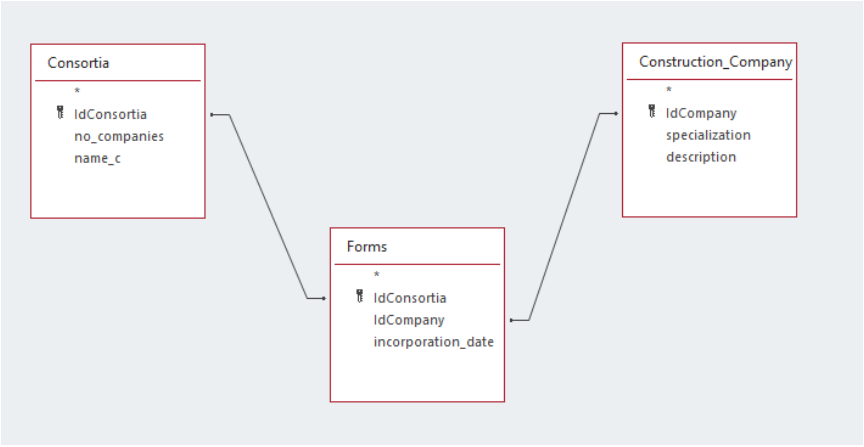

Figure 7 – relation I (Forms)

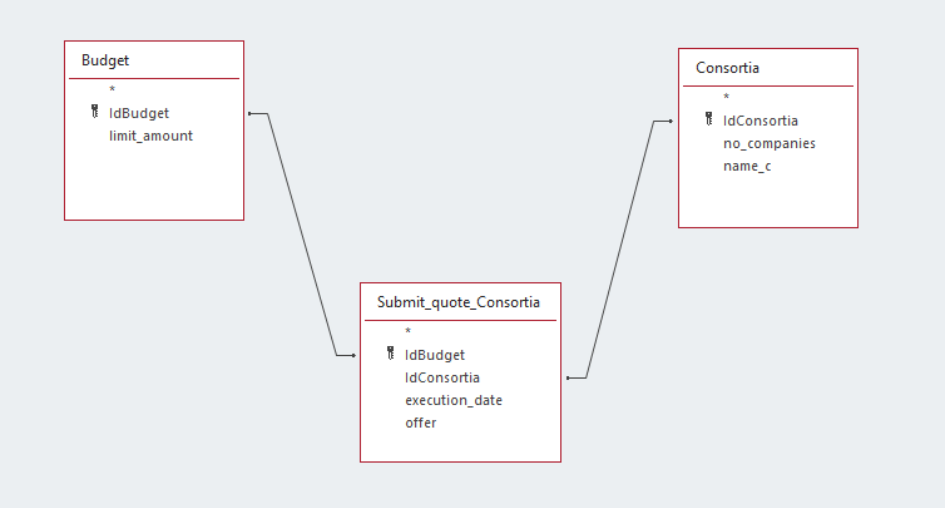

Figure 8 – relation J (Submit_quote_Consortia)

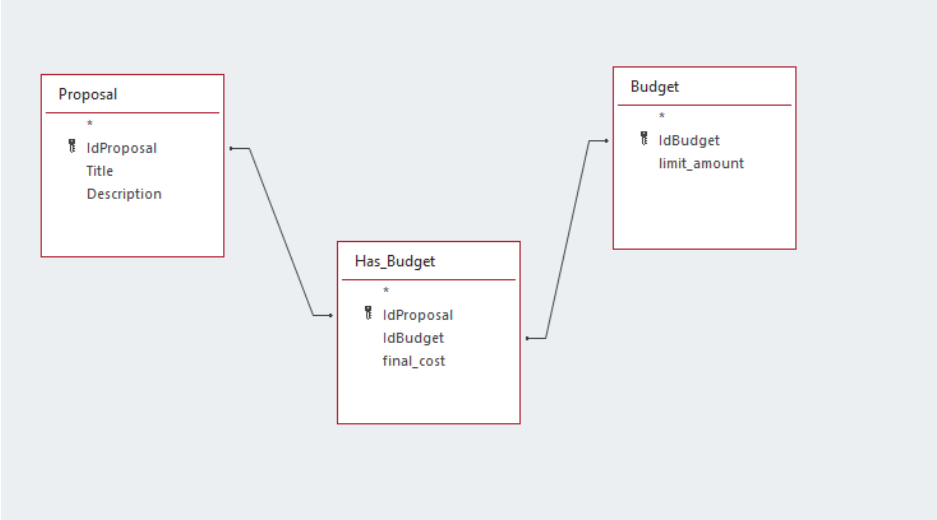

Figure 8 – relation K (Has_Budget)

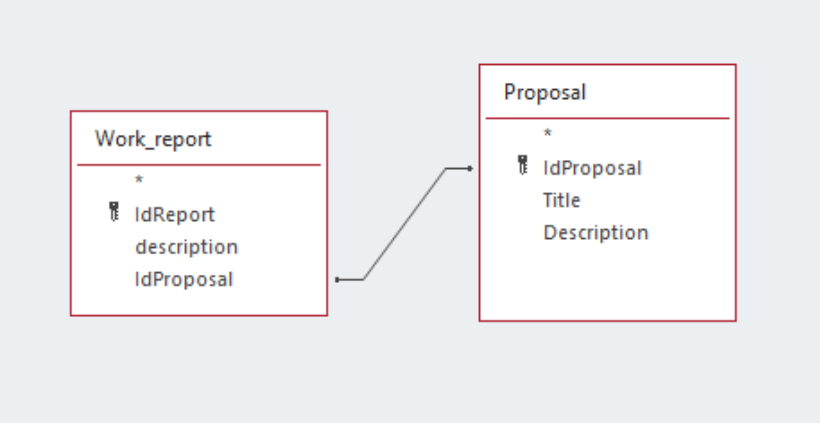

Figure 9 – relation L (the Id from Proposal travels to Work_report)

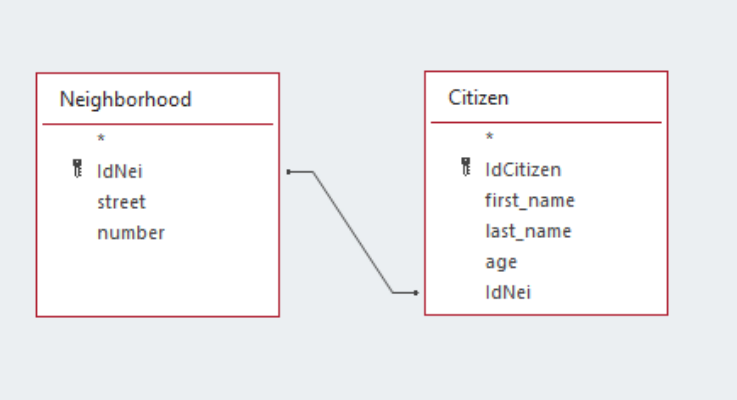

Figure 9 – relation M (the Id from Neighborhood travels to Citizen)

3.2 *Script* de Resposta às Questões

1. Present a general list for each of the tables..

SELECT * FROM Citizen

SELECT * FROM Proposal

SELECT * FROM Municipal_Bodies

SELECT * FROM Council_Services

SELECT * FROM Execution_Phase

SELECT * FROM Stage

SELECT * FROM Budget

SELECT * FROM Construction_Companies

SELECT * FROM Consortia

SELECT * FROM Work_report

SELECT * FROM Neighborhood

SELECT * FROM Register

SELECT * FROM Analyse

SELECT * FROM Consult

SELECT * FROM State_opinion

SELECT * FROM Clarify

SELECT * FROM Has_Stage

SELECT * FROM Generates

SELECT * FROM Submit_Quote_Company

SELECT * FROM Forms

SELECT * FROM Submit_Quote_Consortia

SELECT * FROM Has_Budget

2. List all the information about the citizens(residents), with 'Rita' in their name, and who made proposals in the second quarter of 2024. Display the result sorted by name from Z to A.

```
SELECT *
FROM Citizen C INNER JOIN Register R ON C.IdCitizen=R.IdCitizen
WHERE C.first_name='Rita'  AND MONTH(R.date_proposal) IN (4,5,6)
ORDER BY C.first_name DESC;
```

3. For each resident's name, list the number of proposals made. Bear in mind that there may be residents who have not made any proposals and that these should also be included in the answer. Sort the results starting with the resident with the most proposals.

```
SELECT C.first_name, COUNT(P.IdProposal) AS Prop_Number
FROM (Citizen C LEFT OUTER JOIN Register R ON C.IdCitizen=R.IdCitizen)
    LEFT OUTER JOIN Proposal P ON R.IdProposal=P.IdProposal
GROUP BY C.first_name
ORDER BY Prop_Number DESC
```

4. List the names of the residents who have never had a proposal associated with them. Present the results in alphabetical order by municipality name.

```
SELECT C.first_name, COUNT(P.IdProposal) AS Prop_Number
FROM (Citizen C LEFT OUTER JOIN Register R ON C.IdCitizen=R.IdCitizen)
    LEFT OUTER JOIN Proposal P ON R.IdProposal=P.IdProposal
GROUP BY C.first_name
HAVING COUNT(P.IdProposal)=0
ORDER BY C.first_name ASC
```

5. List the average number of evaluations received for each proposal name. The result should only include proposals that were made in the first half of 2024 and that have had at least 10 evaluations and an average of 8 or more. Present the answer in order of the average obtained, starting with the highest.

```
SELECT P.title, AVG(CC.rate) AS AVG_RATE
FROM Proposal P INNER JOIN Consult CC ON P.IdProposal=CC.IdProposal
GROUP BY P.title
HAVING AVG(CC.rate)>=8 AND COUNT(CC.IdCitizen)>=10
ORDER BY AVG_RATE
```

6. For each construction company name, list the name of the first consortium it was part of.

```
SELECT CC.IdCompany, CC.specialization, CC.description, C.name AS Consortium_name
FROM Construction_Companies CC
JOIN Forms F ON CC.IdCompany = F.IdCompany
JOIN (
   SELECT IdCompany, MIN(incorporation_date) AS min_date
   FROM Forms
   GROUP BY IdCompany
) Fmin ON F.IdCompany = Fmin.IdCompany AND F.incorporation_date = Fmin.min_date
JOIN Consortia C ON F.IdConsortia = C.IdConsortia;
```

7. List the name of the residents who have submitted at least two proposals for the same overall amount.

```
SELECT C.first_name, COUNT(R.IdProposal) AS proposal_count
FROM Citizen C
JOIN REGISTER R ON C.IdCitizen = R.IdCitizen
WHERE R.IdProposal IN (
   SELECT A1.IdProposal
   FROM Analyse A1
   CROSS JOIN Analyse A2
   WHERE A1.total_amount = A2.total_amount AND A1.IdProposal <> A2.IdProposal
)
GROUP BY C.IdCitizen, C.first_name
HAVING COUNT(R.IdProposal) >= 2;
```

8. Create a view (TotalProposalsMunicipe_PerSemestre) that presents the number of proposals made by each resident, in each semester (regardless of the year). The solution should only include citizens who have made proposals in the first and second semesters.

```
SELECT  C.first_name,
   SUM(CASE WHEN MONTH(R.date_proposal) BETWEEN 1 AND 6 THEN 1 ELSE 0
END) AS FIRST_SEMESTER,
   SUM(CASE WHEN MONTH(R.date_proposal) BETWEEN 7 AND 12 THEN 1 ELSE 0
END) AS SECOND_SEMESTER
FROM Citizen C JOIN  Register R ON C.IdCitizen = R.IdCitizen
GROUP BY  C.first_name
HAVING
   SUM(CASE WHEN MONTH(R.date_proposal) BETWEEN 1 AND 6 THEN 1 ELSE 0
END) > 0
   AND
   SUM(CASE WHEN MONTH(R.date_proposal) BETWEEN 7 AND 12 THEN 1 ELSE 0
END) > 0;
```

3.3 Issues regarding “Group Proportionality”

1. How many citizens live on the same street?

```
SELECT N.street, COUNT(C.IdCitizen) AS NO_CITIZENS
FROM Neighborhood N LEFT OUTER JOIN Citizen C ON N.IdNei=C.IdNei
GROUP BY N.street
ORDER BY N.street DESC
```

2. Display the name of the consortias which have more than 200 workers and also the number of the workers.

```
SELECT C.name, SUM(CC.no_workers)
```

```
FROM (Consortia C JOIN Forms F ON C.IdConsortia=F.IdConsortia) JOIN
Construction_Companies CC ON F.IdCompany=CC.IdCompany

GROUP BY C.name

HAVING SUM(CC.no_workers)>200
```

Script de resposta

3. Display the names of the streets where citizens that submitted proposals live and the number of citizens that submitted proposals ordered Z to A.

```
SELECT N.street, COUNT(C.IdCitizen) AS STREET_CITIZEN

FROM Neighborhood N LEFT OUTER JOIN Citizen C ON N.IdNei=C.IdNei

WHERE C.IdCitizen IN

(

   SELECT C1.IdCitizen

   FROM Citizen C1 JOIN Register R ON C1.IdCitizen=R.IdCitizen

)

GROUP BY N.street

ORDER BY N.street DESC
```

4 Conclusion

This project successfully demonstrates the design and implementation of a database system that manages the full lifecycle of a project—from the initial proposal to the final report. Starting with the creation of the Entity-Relationship Model, followed by the Relational Model, the system was developed using SQL and a DBMS to ensure structured data storage and easy access.

The database includes detailed information about project stages, budgets, companies involved, and final outcomes. The 'Group Proportionality' section highlights the team's contribution to the system's design.

Overall, the project shows how real-world processes can be effectively translated into a functional and reliable database system.

Bibliografia

[1] – Alberto Manuel Rodrigues da Silva / Carlos Alberto Escaleira Videira, "UML, Metodologias e Ferramentas CASE", Centro Atlântico, ISBN: 978-8426-36-4 [2] – http://en.wikipedia.org/wiki/Unified_Modeling_Language

[2] – http://en.wikipedia.org/wiki/Unified_Modeling_Language

[3] - http://blogdoabu.blogspot.pt/2009/04/requisitos-de-sistemas-e-casos-de-uso.html

[4] - http://infohades.wordpress.com/2009/12/09/diagrama-de-casos-de-uso/

NOTA1: Quando se faz referência a um autor no texto principal do documento, deve-se usar a numeração que consta na bibliografia. Por exemplo: "Segundo [1] as ferramentas CASE…"

Notas:

As notas de rodapé[1] e a bibliografia[2] deverão ter um corpo de 8 pontos, com avanço especial pendente de 0,5 e entrelinha simples.

[1] Exemplo de nota de rodapé

[2] BIBLIO