

UFG – UNIVERSIDADE FEDERAL DE GOIÁS

EMC – ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E
COMPUTAÇÃO

Jonatas Novais Tomazini

PROJETO CONCEITUAL – MRARIS

Projeto Conceitual relativo ao
Projeto Final da disciplina de
Banco de Dados I

Docente: Ana Claudia Bastos Loureiro Moncao

1 Definição do objetivo	4
2 Descrição do cenário e do escopo	4
3 Definição dos requisitos de acordo com o cenário e o escopo	4
4 Modelo entidade-relacionamento de forma gráfica (DER)	7
5 Mapeamento e Normalização.....	7
6 Criação das tabelas e construção de SQL.....	8

DEFINIÇÃO DO OBJETIVO

Desenvolver um banco de dados para a plataforma MRARIS, que organiza o gerenciamento de professores, alunos, turmas e exercícios. O objetivo principal é oferecer um sistema que permita:

- Professores criarem turmas e adicionarem alunos.
- Professores cadastrarem e disponibilizarem exercícios para turmas específicas.
- Alunos acessarem os exercícios de suas respectivas turmas.

DESCRIÇÃO DO CENÁRIO E ESCOPO

Cenário: MRARIS é uma plataforma de ensino lógico que conecta professores e alunos em um ambiente interativo de aprendizado. Professores gerenciam suas turmas, criam e disponibilizam exercícios, enquanto os alunos acessam suas turmas e completam os exercícios atribuídos.

Escopo: O sistema deve permitir o cadastro de:

- Professores, incluindo informações pessoais e as turmas que lecionam.
- Alunos, com informações de cadastro e as turmas a que pertencem.
- Turmas, associadas a professores e contendo alunos.
- Exercícios, vinculados a turmas e realizados pelos alunos.

DEFINIÇÃO DOS REQUISITOS DE ACORDO COM O CENÁRIO E ESCOPO

Para o sistema MRARIS, os requisitos foram definidos para cobrir todas as funcionalidades relacionadas ao gerenciamento de professores, alunos, turmas e exercícios. Cada requisito foi elaborado para refletir a dinâmica do cenário apresentado e garantir que todas as operações essenciais sejam atendidas.

Requisitos Funcionais

1. Professor:

- Deve ser possível cadastrar professores no sistema com os seguintes dados:
 - Nome completo.
 - E-mail (deve ser único no sistema).
 - Senha (armazenada de forma segura).

- Código único (ID) para identificação do professor.
- Professores podem:
 - Criar turmas.
 - Gerenciar as turmas criadas (editar informações e adicionar/remover alunos).
 - Cadastrar exercícios nas turmas que lecionam.
 - Visualizar respostas submetidas pelos alunos aos exercícios das turmas que lecionam.

2. Aluno:

- Deve ser possível cadastrar alunos no sistema com os seguintes dados:
 - Nome completo.
 - E-mail (deve ser único no sistema).
 - Senha (armazenada de forma segura).
 - Código único (ID) para identificação do aluno.
- Alunos podem:
 - Participar de uma ou mais turmas (vinculadas por um professor).
 - Acessar exercícios das turmas em que estão matriculados.
 - Submeter respostas aos exercícios disponíveis.
 - Visualizar feedback e notas (se o sistema incluir funcionalidades de correção automática ou avaliação pelo professor).

3. Turma:

- Deve ser possível criar turmas com os seguintes dados:
 - Nome da turma.
 - Código único (ID) da turma.
 - Professor responsável pela turma (associado via ID).
 - Lista de alunos (relacionamento N:N entre alunos e turmas).
- Cada turma deve estar vinculada a um professor, mas um professor pode lecionar várias turmas.
- Alunos podem ser adicionados ou removidos de turmas pelo professor.

4. Exercício:

- Deve ser possível criar exercícios com os seguintes dados:
 - Código único (ID) do exercício.
 - Título do exercício.
 - Descrição do exercício.
 - Data de criação.
 - Turma associada (ID da turma).
- Cada exercício pertence a uma única turma, mas uma turma pode conter vários exercícios.
- Os exercícios podem ser editados ou excluídos pelos professores.

5. Resposta de Exercício:

- Deve ser possível armazenar respostas submetidas pelos alunos com os seguintes dados:
 - Código único (ID da resposta).
 - Código do exercício associado.

- Código do aluno que submeteu a resposta.
- Conteúdo da resposta (texto ou outro formato, dependendo do tipo de exercício).
- Data e hora da submissão.
- Nota ou feedback (opcional, se avaliado pelo professor).
- Apenas os alunos da turma podem submeter respostas para os exercícios da turma.

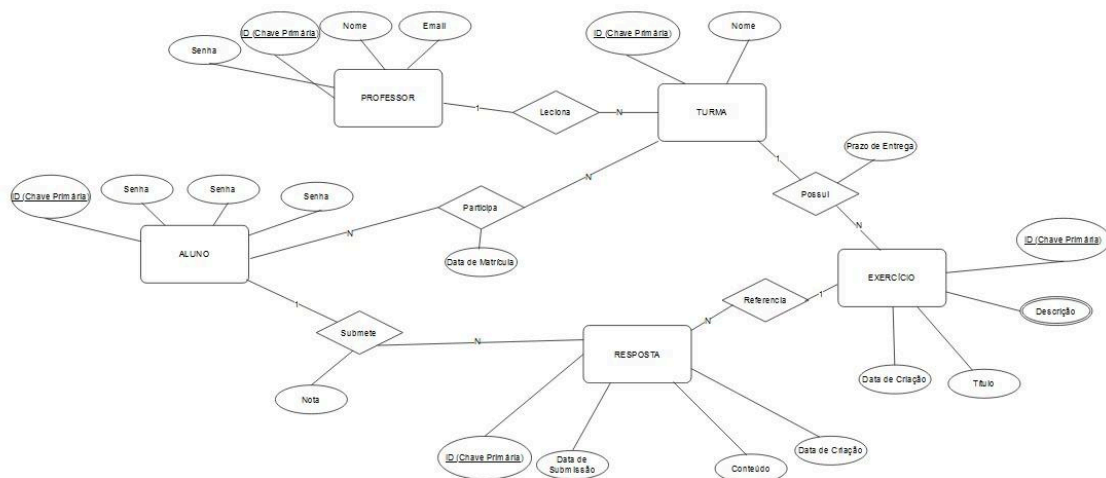
Requisitos Não Funcionais

- O sistema deve:
 - Garantir a segurança dos dados (e.g., senhas armazenadas usando criptografia).
 - Ser escalável, para suportar um grande número de alunos, professores, turmas e exercícios.
 - Ter tempo de resposta rápido para consultas frequentes (e.g., listagem de exercícios e turmas).

Requisitos de Relacionamento

- **Professor e Turmas:**
 - Um professor pode estar associado a várias turmas (1:N).
- **Aluno e Turmas:**
 - Um aluno pode estar matriculado em várias turmas, e uma turma pode conter vários alunos (N:N).
- **Turma e Exercícios:**
 - Uma turma pode ter vários exercícios, mas cada exercício pertence a uma única turma (1:N).
- **Aluno e Respostas:**
 - Um aluno pode submeter várias respostas, mas cada resposta pertence a um único exercício (N:1).

MODELO DE ENTIDADE-RELACIONAMENTO (DER)



Entidades Fortes

1. **PROFESSOR**
Professor(ID, Nome, Email, Senha)

2. **ALUNO**
Aluno(ID, Nome, Email, Senha)

3. **TURMA**
Turma(ID, Nome)

4. **EXERCICIO**
Exercicio(ID, Título, Descrição, Data)

5. **RESPOSTA**
Resposta(ID, Conteúdo, Data_Submissao, Nota)

Relacionamentos

1. Leciona

Leciona(Professor_ID, Turma_ID)

- Relaciona Professores às Turmas que criaram.
- **Cardinalidade:** 1:N (um professor pode lecionar várias turmas).

2. Participa

Participa(Aluno_ID, Turma_ID, Data_Matricula, Status)

- Relaciona Alunos às Turmas em que estão matriculados.
- **Cardinalidade:** N:N (um aluno pode participar de várias turmas e uma turma pode ter vários alunos).
- **Atributos:**
 - **Data_Matricula:** Data de ingresso na turma.

3. Possui

Possui(Turma_ID, Exercicio_ID, Data_Disponibilizacao, Prazo_Entrega)

- Relaciona Turmas aos Exercícios cadastrados.
- **Cardinalidade:** 1:N (uma turma pode ter vários exercícios).
- **Atributos:**
 - **Prazo_Entrega:** Data limite para entrega.

4. Submete

Submete(Aluno_ID, Resposta_ID)

- Relaciona Alunos às Respostas que submeteram.
- **Cardinalidade:** 1:N (um aluno pode submeter várias respostas).
- Atributos do relacionamento são incluídos diretamente em **Resposta**.

5. Referência

Referência(Exercicio_ID, Resposta_ID)

- Relaciona Exercícios às Respostas submetidas.
- **Cardinalidade:** 1:N (um exercício pode ter várias respostas).

DICIONÁRIO DE DADOS

Professor

Atributo	Tipo de Dado	Descrição	Restrições
ID	SERIAL	Identificador único do professor	Chave primária, auto-incremento
Nome	VARCHAR(100)	Nome completo do professor	Não nulo
Email	VARCHAR(100)	E-mail único do professor	Não nulo, único
Senha	VARCHAR(100)	Senha para acesso ao sistema	Não nulo

Aluno

Atributo	Tipo de Dado	Descrição	Restrições
ID	SERIAL	Identificador único do aluno	Chave primária, auto-incremento
Nome	VARCHAR(100)	Nome completo do aluno	Não nulo
Email	VARCHAR(100)	E-mail único do aluno	Não nulo, único
Senha	VARCHAR(100)	Senha para acesso ao sistema	Não nulo

Turma

Atributo	Tipo de Dado	Descrição	Restrições
ID	SERIAL	Identificador único da turma	Chave primária, auto-incremento
Nome	VARCHAR(100)	Nome da turma	Não nulo

Exercício

Atributo	Tipo de Dado	Descrição	Restrições
ID	SERIAL	Identificador único do exercício	Chave primária, auto-incremento
Título	VARCHAR(200)	Título do exercício	Não nulo
Descrição	TEXT	Detalhes do exercício	Não nulo
Data	DATE	Data de criação do exercício	Não nulo

Resposta

Atributo	Tipo de Dado	Descrição	Restrições
ID	SERIAL	Identificador único da resposta	Chave primária, auto-incremento
Conteúdo	TEXT	Resposta submetida pelo aluno	Não nulo
Data_Submissao	TIMESTAMP	Data e hora da submissão	Não nulo
Nota	NUMERIC(5, 2)	Nota atribuída pelo professor	Pode ser nulo

Normalização

1. Professor

- **Análise:** Todos os atributos são atômicos, não há dependências parciais ou transitivas.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

PROFESSOR(ID, Nome, Email, Senha)

2. Aluno

- **Análise:** Todos os atributos são atômicos, não há dependências parciais ou transitivas.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

ALUNO(ID, Nome, Email, Senha)

3. Turma

- **Análise:** não há dependências transitivas.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

SCSS

Copiar código

TURMA(ID, Nome)

4. Exercício

- **Análise:** Não há dependências transitivas.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

EXERCICIO(ID, Título, Descrição, Data)

5. Resposta

- **Análise:**
 - Todos os atributos são atômicos.
 - **Nota** pertence à resposta, não ao relacionamento entre aluno e exercício.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

RESPOSTA(ID, Conteúdo, Data_Submissao, Nota)

6. Participa (Relacionamento N:N entre Aluno e Turma)

- **Análise:**
 - Todos os atributos são atômicos.
 - `Data_Matricula` pertence exclusivamente à relação entre Aluno e Turma.
- **Forma Normal:** Está em **3FN**.

Tabela Normalizada:

`PARTICIPA(Aluno_ID, Turma_ID, Data_Matricula)`

7. Possui (Relacionamento 1:N entre Turma e Exercício)

- **Análise:**
 - Todos os atributos são atômicos.
 - `Data_Disponibilizacao` e `Prazo_Entrega` pertencem exclusivamente à relação entre Turma e Exercício.
- **Forma Normal:** Está em **3FN**.
- **Tabela Normalizada:**
`POSSUI(Turma_ID, Exercicio_ID, Data_Disponibilizacao, Prazo_Entrega)`

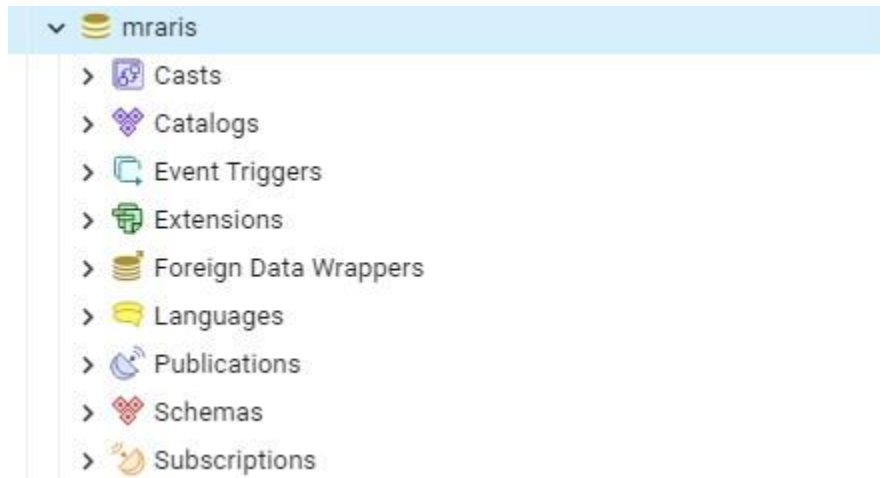
Após o processo de normalização, todas as tabelas foram ajustadas para atender à **3ª Forma Normal (3FN)**, garantindo que:

1. Todos os atributos são atômicos.
2. Não há dependências parciais (nas tabelas com chaves compostas).
3. Não há dependências transitivas.

SCRIPT SQL

-- Criação do Banco de Dados

```
CREATE DATABASE mraris;
```



Criação das Tabelas

```
CREATE TABLE professor (  
    ID SERIAL PRIMARY KEY,  
    Nome VARCHAR(100) NOT NULL,  
    Email VARCHAR(100) UNIQUE NOT NULL,  
    Senha VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE aluno (
```

```
    ID SERIAL PRIMARY KEY,  
    Nome VARCHAR(100) NOT NULL,  
    Email VARCHAR(100) UNIQUE NOT NULL,  
    Senha VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE turma (  
    ID SERIAL PRIMARY KEY,  
    Nome VARCHAR(100) NOT NULL,  
    Professor_ID INT REFERENCES Professor(ID) NOT NULL  
);
```

```
CREATE TABLE exercicio (  
    ID SERIAL PRIMARY KEY,  
    Titulo VARCHAR(200) NOT NULL,  
    Descricao TEXT NOT NULL,  
    Data DATE NOT NULL,  
    Turma_ID INT REFERENCES Turma(ID) NOT NULL  
);
```

```
CREATE TABLE resposta (  
    ID SERIAL PRIMARY KEY,  
    Conteudo TEXT NOT NULL,  
    Data_Submissao TIMESTAMP NOT NULL,
```

```
Nota NUMERIC(5,2),  
Feedback TEXT,  
Aluno_ID INT REFERENCES Aluno(ID) NOT NULL,  
Exercicio_ID INT REFERENCES Exercicio(ID) NOT NULL  
);
```



Inserts

```
INSERT INTO Professor (Nome, Email, Senha) VALUES  
('Prof. João', 'joao@mraris.com', 'senha123'),  
('Prof. Maria', 'maria@mraris.com', 'senha456');
```

	id [PK] integer	nome character varying (100)	email character varying (100)	senha character varying (100)
1	1	Prof. João	joao@mraris.com	senha123
2	2	Prof. Maria	maria@mraris.com	senha456

INSERT INTO Aluno (Nome, Email, Senha) VALUES

('Aluno 1', 'aluno1@mraris.com', 'senha789'),

('Aluno 2', 'aluno2@mraris.com', 'senha987');

	id [PK] integer	nome character varying (100)	email character varying (100)	senha character varying (100)
1	1	Aluno 1	aluno1@mraris.com	senha789
2	2	Aluno 2	aluno2@mraris.com	senha987

INSERT INTO Turma (Nome, id) VALUES

('Turma Lógica 1', 1),

('Turma Lógica 2', 2);

	id [PK] integer	nome character varying (100)
1	1	Turma Lógica 1
2	2	Turma Lógica 2

INSERT INTO Exercicio (Titulo, Descricao, Data, Turma_ID) VALUES

('Exercicio 1', 'Resolva este problema lógico', '2024-12-01', 1),

('Exercicio 2', 'Analise este desafio lógico', '2024-12-05', 2);

	id [PK] integer	titulo character varying (200)	descricao text	data date	turma_id integer
1	1	Exercicio 1	Resolva este problema lógico	2024-12-01	1
2	2	Exercicio 2	Analise este desafio lógico	2024-12-05	2

INSERT INTO Resposta (Conteudo, Data_Submissao, Nota, Feedback, Aluno_ID, Exercicio_ID) VALUES

('Resposta ao exercício 1', '2024-12-10 10:00:00', 8.5, 'Boa tentativa', 1, 1),

('Resposta ao exercício 2', '2024-12-11 14:30:00', 9.0, 'Muito bom', 2, 2);

	id [PK] integer	conteudo text	data_submissao timestamp without time zone	nota numeric (5,2)	feedback text	aluno_id integer	exercicio_id integer
1	1	Resposta ao exercício 1	2024-12-10 10:00:00	8.50	Boa tentativa	1	1
2	2	Resposta ao exercício 2	2024-12-11 14:30:00	9.00	Muito bom	2	2

Deletes

DELETE FROM Resposta WHERE ID = 1;

	id [PK] integer	conteudo text	data_submissao timestamp without time zone	nota numeric (5,2)	feedback text	aluno_id integer	exercicio_id integer
1	2	Resposta ao exercício 2	2024-12-11 14:30:00	9.00	Muito bom	2	2

Updates

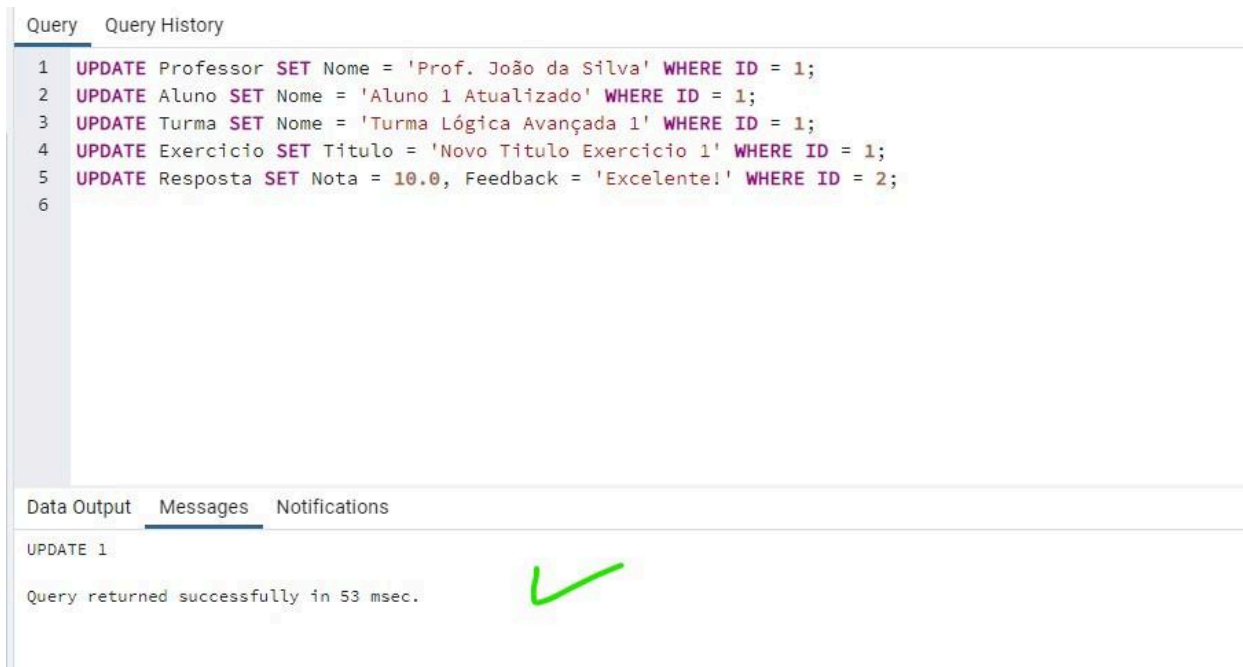
UPDATE Professor SET Nome = 'Prof. João da Silva' WHERE ID = 1;

UPDATE Aluno SET Nome = 'Aluno 1 Atualizado' WHERE ID = 1;

UPDATE Turma SET Nome = 'Turma Lógica Avançada 1' WHERE ID = 1;

UPDATE Exercicio SET Titulo = 'Novo Titulo Exercicio 1' WHERE ID = 1;

UPDATE Resposta SET Nota = 10.0, Feedback = 'Excelente!' WHERE ID = 2;



The screenshot shows a SQL query editor with a 'Query' tab selected. The query history displays five SQL UPDATE statements. Below the editor, the 'Messages' tab is active, showing a confirmation message: 'UPDATE 1' and 'Query returned successfully in 53 msec.' A large green checkmark is drawn over the success message.

```
Query    Query History
1  UPDATE Professor SET Nome = 'Prof. João da Silva' WHERE ID = 1;
2  UPDATE Aluno SET Nome = 'Aluno 1 Atualizado' WHERE ID = 1;
3  UPDATE Turma SET Nome = 'Turma Lógica Avançada 1' WHERE ID = 1;
4  UPDATE Exercicio SET Titulo = 'Novo Titulo Exercicio 1' WHERE ID = 1;
5  UPDATE Resposta SET Nota = 10.0, Feedback = 'Excelente!' WHERE ID = 2;
6

Data Output  Messages  Notifications
UPDATE 1
Query returned successfully in 53 msec.
```

Selects básicos

SELECT * FROM Professor;

SELECT Nome, Email FROM Aluno;

```

1
2 SELECT * FROM Professor;
3 SELECT Nome, Email FROM Aluno;
4

```

Data Output Messages Notifications

	nome character varying (100) 🔒	email character varying (100) 🔒
1	Aluno 2	aluno2@mraris.com
2	Aluno 1 Atualizado	aluno1@mraris.com

Renomeando campos

```
SELECT Nome AS Professor_Nome, Email AS Professor_Email FROM Professor;
```

```
SELECT Titulo AS Exercício_Titulo, Data AS Data_Criacao FROM Exercício;
```

Query

Query History

1

SELECT Nome AS Professor_Nome, Email AS Professor_Email FROM Professor;

2

SELECT Titulo AS Exercício_Titulo, Data AS Data_Criacao FROM Exercicio;

3

Data Output

Messages

Notifications

	<div>exercício_titulo</div> <div>character varying (200)</div> <div></div>	<div>data_criacao</div> <div>date</div> <div></div>
1	Exercicio 2	2024-12-05
2	Novo Titulo Exercicio 1	2024-12-01

Inner Joins com 2 tabelas

Query Query History

```
1 SELECT Aluno.Nome, Turma.Nome AS Turma_Nome
2 FROM Aluno
3 INNER JOIN Turma ON Turma.Professor_ID = 1;
4
5 SELECT Exercicio.Titulo, Professor.Nome AS Professor_Nome
6 FROM Exercicio
7 INNER JOIN Turma ON Exercicio.Turma_ID = Turma.ID
8 INNER JOIN Professor ON Turma.Professor_ID = Professor.ID;
```

Data Output Messages Notifications



	titulo character varying (200) 🔒	professor_nome character varying (100) 🔒
1	Exercicio 1	Prof. João da Silva
2	Exercicio 2	Prof. Maria

Inner Joins com mais de 2 tabelas

```
SELECT Resposta.Conteudo, Aluno.Nome AS Aluno_Nome, Exercicio.Titulo AS  
Exercicio_Titulo
```

```
FROM Resposta
```

```
INNER JOIN Aluno ON Resposta.Aluno_ID = Aluno.ID
```

```
INNER JOIN Exercicio ON Resposta.Exercicio_ID = Exercicio.ID
```

```
INNER JOIN Turma ON Exercicio.Turma_ID = Turma.ID;
```

The screenshot shows a SQL IDE interface. The top pane displays a SQL query with line numbers 1 through 6. The query performs an inner join between Resposta, Aluno, Exercicio, and Turma tables. The bottom pane shows the results of the query in a table format with columns: conteudo, aluno_nome, and exercicio_titulo. The first row of data shows 'Resposta ao exercício 2', 'Aluno 2', and 'Exercicio 2'.

```
1 SELECT Resposta.Conteudo, Aluno.Nome AS Aluno_Nome, Exercicio.Titulo AS Exercicio_Titulo
2 FROM Resposta
3 INNER JOIN Aluno ON Resposta.Aluno_ID = Aluno.ID
4 INNER JOIN Exercicio ON Resposta.Exercicio_ID = Exercicio.ID
5 INNER JOIN Turma ON Exercicio.Turma_ID = Turma.ID;
6
```

	conteudo text	aluno_nome character varying (100)	exercicio_titulo character varying (200)
1	Resposta ao exercício 2	Aluno 2	Exercicio 2

Outer Join e Left Join

```
SELECT Aluno.Nome, Turma.Nome AS Turma_Nome  
FROM Aluno  
LEFT JOIN Turma ON Turma.Professor_ID = 1;
```

```
SELECT Turma.Nome AS Turma_Nome, Exercicio.Titulo AS Exercicio_Titulo  
FROM Turma  
FULL OUTER JOIN Exercicio ON Turma.ID = Exercicio.Turma_ID;
```

QueryQuery History

1SELECT Aluno.Nome, Turma.Nome AS Turma_Nome

2FROM Aluno

3LEFT JOIN Turma ON Turma.Professor_ID = 1;

4

5SELECT Turma.Nome AS Turma_Nome, Exercicio.Titulo AS Exercicio_Titulo

6FROM Turma

7FULL OUTER JOIN Exercicio ON Turma.ID = Exercicio.Turma_ID;

8

Data OutputMessagesNotifications

turma_nome

character varying (100)

exercicio_titulo

character varying (200)

1	Turma Lógica 1	Exercicio 1
2	Turma Lógica 2	Exercicio 2

Agrupamento

```
SELECT Turma.Nome, COUNT(Exercicio.ID) AS Total_Exercicios  
  
FROM Turma  
  
LEFT JOIN Exercicio ON Turma.ID = Exercicio.Turma_ID  
  
GROUP BY Turma.Nome;
```

```
SELECT Professor.Nome, COUNT(Turma.ID) AS Total_Turmas  
FROM Professor  
LEFT JOIN Turma ON Professor.ID = Turma.Professor_ID  
GROUP BY Professor.Nome;
```


Query

Query History

1

2

3

4

5

6

7

8

9

10

SELECT Turma.Nome, COUNT(Exercicio.ID) AS Total_Exercicios

FROM Turma

LEFT JOIN Exercicio ON Turma.ID = Exercicio.Turma_ID

GROUP BY Turma.Nome;

SELECT Professor.Nome, COUNT(Turma.ID) AS Total_Turmas

FROM Professor

LEFT JOIN Turma ON Professor.ID = Turma.Professor_ID

GROUP BY Professor.Nome;

Data Output

Messages

Notifications

nome

character varying (100)

total_turmas

bigint

1

Prof. João da Silva

1

2

Prof. Maria

1

Ordenação

SELECT Nome, Email FROM Aluno ORDER BY Nome ASC;

SELECT Nome, Email FROM Professor ORDER BY Email DESC;

Query

Query History

1

SELECT Nome, Email FROM Aluno ORDER BY Nome ASC;

2

SELECT Nome, Email FROM Professor ORDER BY Email DESC;

3

Data Output

Messages

Notifications

	nome character varying (100)	email character varying (100)
1	Prof. Maria	maria@mraris.com
2	Prof. João da Silva	joao@mraris.com

