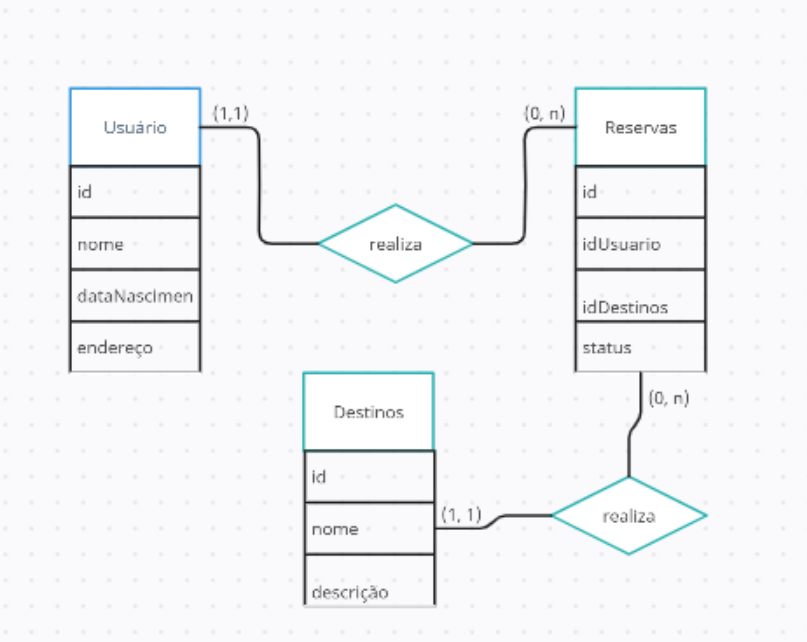
Introdução aos Bancos de Dados Relacionais

Um banco de dados é um conjunto organizado de dados que são armazenados eletronicamente em um sistema de computador. Ele é projetado para facilitar a criação, atualização, consulta e gerenciamento desses dados de forma eficiente e segura.

SQL - Structured Query Language é uma linguagem de consulta padronizada e amplamente utilizada. Surgiu em 1970. É dividido em algumas declarações:
DQL - Linguagem de Consulta de Dados - SELECT
DML - Linguagem de Manipulação de Dados - INSERT, UPDATE e DELETE
DDL - Linguagem de Definição dos Dados - CREATE, ALTER, DROP
DCL - Linguagem de Controle de Dados - GRANT, REVOKE
DTL - Linguagem de Transação de Dados - BEGIN, COMMIT, ROLLBACK

MER - Modelo Entidade Relacionemento é representado através de diagramas chamados Diagramas Entidade-Relacionamento (DER)

Entidades - são nomeados que representam de forma clara a função dentro do domínio.
Atributos - as características ou propriedades das entidades
Relacionamentos - são associações entre duas ou mais entidades
Cardinalidades - refere-se ao número de instâncias de uma entidade que podem estar associadas a uma única instância de outra entidade. Exemplos: (1:1), (1:n), (n:n), (0,n)



Modelagem de Dados Relacionais

TABELAS: cada tabela de um banco de dados relacional tem um nome único e é dividida em colunas e linhas.

COLUNA: representa um atributo específico, nome único e tipo de dado associado

TIPOS DE DADOS: Integer, Decimal, Varchar, Date/Time, Boolean, Text

RESTRIÇÕES DE VALOR:

- NOT NULL: Coluna não pode ter valores nulos.
- UNIQUE: Valores da coluna devem ser únicos.
- DEFAULT: Valor padrão para a coluna.

COMANDOS EXEMPLOS:

```
CREATE TABLE usuarios (id INT, nome VARCHAR(255) NOT NULL,);
```

```
INSERT INTO users (name, age) VALUES ('Alice', 30), ('Bob', 25);
```

```
SELECT * FROM users;
```

Para procurar:

```
SELECT * FROM usuarios WHERE nome LIKE '%Silva%';
```

Excluir:

```
DELETE FROM reservas WHERE status = 'cancelada';
```

Excluir tabela:

```
DROP TABLE usuarios;
```

Alterar informações:

```
ALTER TABLE usuarios_nova RENAME TO usuarios;
ALTER TABLE usuarios ALTER COLUMN endereco TYPE VARCHAR(150)
```

CHAVES PRIMÁRIAS (PRIMARY KEY): identifica exclusivamente, não pode conter valores nulos e uma table pode contar apenas uma chave primária

CHAVES ESTRANGEIRAS (FOREIGN KEY): usada para estabelecer e manter a integridade dos dados entre tableas relacionadas, não pode ser nula e uma tabela pode ter mais de uma chave estrangeira

RESTRIÇÕES DE CHAVE ESTRANGEIRA:

- ON DELETE: Define o comportamento quando a chave primária é excluída.
- ON UPDATE: Define o comportamento quando a chave primária é atualizada.
- CASCADE: Propaga a ação de exclusão ou atualização para as tabelas relacionadas.
- SET NULL: Define os valores da chave estrangeira como NULL.
- SET DEFAULT: Define os valores da chave estrangeira como o valor padrão.
- RESTRICT: Não permite a exclusão ou atualização da chave primária se houver registros relacionados.

Normalização de Dados

A Normalização de Dados é o processo de organizar os dados em um banco de dados relacional de **forma eficiente e livre de redundâncias**.

FORMAS NORMAIS

1FN: Atomicidade de dados - cada coluna deve ter um valor único e cada célula da tabela deve ser atômica

Para dividir uma tabela no banco de dados, use o comendo SPLIT_PART que divide uma string em partes com base em um delimitador e retorna a parte desejada.

Exemplo: transformar a coluna Endereço em nas colunas Rua, Numero, Cidade e Estado

```
-- Adicionar colunas de endereço à tabela "Usuarios"

ALTER TABLE Usuarios
ADD rua VARCHAR(100),
ADD numero VARCHAR(10),
ADD cidade VARCHAR(50),
ADD estado VARCHAR(50);
```

```
-- Atualizar os dados das novas colunas com base na coluna "endereco" existente

UPDATE Usuarios
SET rua = SPLIT_PART(endereco, ', ', 1),
    numero = SPLIT_PART(endereco, ', ', 2),
    cidade = SPLIT_PART(endereco, ', ', 3),
    estado = SPLIT_PART(endereco, ', ', 4);
```

```
-- Excluir a coluna "endereco" da tabela original

ALTER TABLE Usuarios
DROP COLUMN endereco;
```

2FN: a tabela deve estar na 1FN e se todas as colunas que não fazem parte da chave primária dependem totalmente da chave primária.

3FN: a tabela deve estar na 2FN e não deve conter dependências transisitvas entre as colunas não chave.

Consultas Avançadas

As junções são utlizadas para combinar linhas de duas ou mais tabelas com base em uma condição relacionada entre elas.

- JOIN: Utilizado para combinar linhas de duas ou mais tabelas com base em uma condição relacionada entre elas.
- INNER JOIN: Retorna registros quando houver pelo menos uma correspondência em ambas as tabelas.
- LEFT JOIN: Retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita.
- RIGHT JOIN: Retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda.
- FULL JOIN: Retorna todos os registros quando houver uma correspondência em qualquer uma das tabelas.

Subconsultas: Consultas aninhadas dentro de uma consulta externa.

- Utilizadas em várias cláusulas, como WHERE, FROM, etc.
- Podem ser correlacionadas, referenciando colunas da consulta externa dentro da subconsulta.
-

FUNÇÕES AGREGADAS E AGRUPAMENTO DE RESULTADOS:

- SELECT COUNT(*): Retorna o número de registros em uma tabela.
- AVG(): Retorna a média de valores em uma coluna.
- SUM(): Retorna a soma dos valores em uma coluna.
- MIN(): Retorna o menor valor em uma coluna.
- MAX(): Retorna o maior valor em uma coluna.
- GROUP BY: Agrupa registros com base em valores em uma ou mais colunas, permitindo a aplicação de funções de agregação, como COUNT, SUM, AVG, MIN e MAX, a cada grupo.
- LIMIT: Limita o número de registros retornados por uma consulta.
- OFFSET: Especifica o número de registros a serem ignorados antes de começar a retornar resultados em uma consulta.
- ORDER BY: Ordena os resultados de uma consulta com base em uma ou mais colunas, podendo ser ascendente (ASC) ou descendente (DESC).

ANÁLISE DO PLANO DE EXECUÇÃO: Permite examinar as operações realizadas, as tabelas acessadas, os índices utilizados e outras informações importantes para identificar possíveis melhorias de desempenho.

EXPLAIN SELECT * FROM usuarios WHERE nome = 'Maria';

Bônus: Exportar arquivo SQL

Usando o `pg_dump`.

Exportar o banco de dados inteiro:

pg_dump -U usuario -d nome_do_banco -f caminho_para_arquivo.sql

Exportar apenas a estrutura do banco de dados (sem os dados):

pg_dump -U usuario -d nome_do_banco -s -f caminho_para_arquivo.sql

Exportar apenas dados de uma tabela específica:

pg_dump -U usuario -d nome_do_banco -t nome_da_tabela -a -f caminho_para_arquivo.sql

Exportar apenas a definição da estrutura de uma tabela específica:

pg_dump -U usuario -d nome_do_banco -t nome_da_tabela -s -f caminho_para_arquivo.sql