

Resumão de BD

Conceitos essenciais

Dado
Informação

Banco de dados

DDL - Data Definition Language (Linguagem de Definição de Dados)

A DDL, como o próprio nome sugere, foca na **definição da estrutura** do banco de dados. É como se fosse o arquiteto do banco de dados, projetando e construindo os alicerces que armazenam seus dados. Os comandos DDL permite criar, modificar e remover elementos como:

- **Tabelas:** Definindo a estrutura das tabelas, incluindo colunas, tipos de dados e restrições.
- **Índices:** Organizando os dados para consultas mais rápidas e eficientes.
- **Visões:** Criando perspectivas personalizadas das tabelas existentes.
- **Procedimentos Armazenados:** Automatizando tarefas complexas em blocos de código reutilizáveis.

Comandos DDL comuns:

- **CREATE:** Cria tabelas, índices, views e procedures.
- **ALTER:** Modifica a estrutura de tabelas, índices e views.
- **DROP:** Remove tabelas, índices, views e procedures.

DML - Data Manipulation Language (Linguagem de Manipulação de Dados):

Enquanto a DDL define a estrutura, a DML entra em ação para **manipular os dados** armazenados dentro do banco de dados. É como se fosse o inquilino do banco de dados, inserindo, alterando e consultando as informações. Os comandos DML permitem:

- **Inserir:** Adicionar novos registros nas tabelas.
- **Atualizar:** Modificar dados existentes nas tabelas.
- **Excluir:** Remover registros das tabelas.
- **Consultar:** Recuperar dados específicos das tabelas.

Comandos DML comuns:

- **INSERT:** Insere novos registros em uma tabela.
- **UPDATE:** Atualiza dados em registros existentes.
- **DELETE:** Remove registros de uma tabela.
- **SELECT:** Recupera dados específicos de uma ou mais tabelas.

Resumindo:

- **DDL:** Define a estrutura do banco de dados (o que armazenar).
- **DML:** Manipula os dados armazenados no banco de dados (o que fazer com os dados).

Analogia:

Imagine um armário. A DDL seria responsável por projetar e construir o armário (definindo prateleiras, gavetas e compartimentos). Já a DML seria responsável por organizar e manusear os itens dentro do armário (colocando, retirando e movimentando objetos).

Integridade Referencial: a base essencial para Bancos de Dados Consistentes

No mundo dos bancos de dados relacionais, a **integridade referencial** é um conceito crucial para garantir a confiabilidade e a consistência dos dados armazenados. Ela funciona como um conjunto de regras que garantem que os relacionamentos entre as tabelas estejam sempre válidos e precisos, evitando anomalias e inconsistências.

1. Chave Primária e Chave Estrangeira:

- **Chave Primária:** Cada tabela possui uma coluna ou conjunto de colunas que identificam **unicamente** cada registro, como o ID de um cliente ou o número de um pedido.
- **Chave Estrangeira:** Uma coluna em uma tabela que **referencia a chave primária** de outra tabela, estabelecendo uma ligação entre os dados.

2. Regras de Integridade Referencial:

- **Restrição de Entidade:** Um registro na tabela filha (com chave estrangeira) **deve ter um registro correspondente** na tabela pai (com chave primária). Ou seja, não é possível ter um pedido sem um cliente associado.
- **Restrição de Referência:** Um registro na tabela pai (com chave primária) **não pode ser excluído** se existir um registro filho na tabela filha (com chave estrangeira que o referencia). Ou seja, um cliente não pode ser excluído se ainda tiver pedidos vinculados a ele.
- **Restrição de Atualização:** Um registro na tabela pai (com chave primária) **não pode ser alterado de forma a invalidar** a referência de um registro filho na tabela filha (com chave estrangeira). Ou seja, o ID de um cliente não pode ser alterado se já existir um pedido associado a esse ID.

Benefícios da Integridade Referencial:

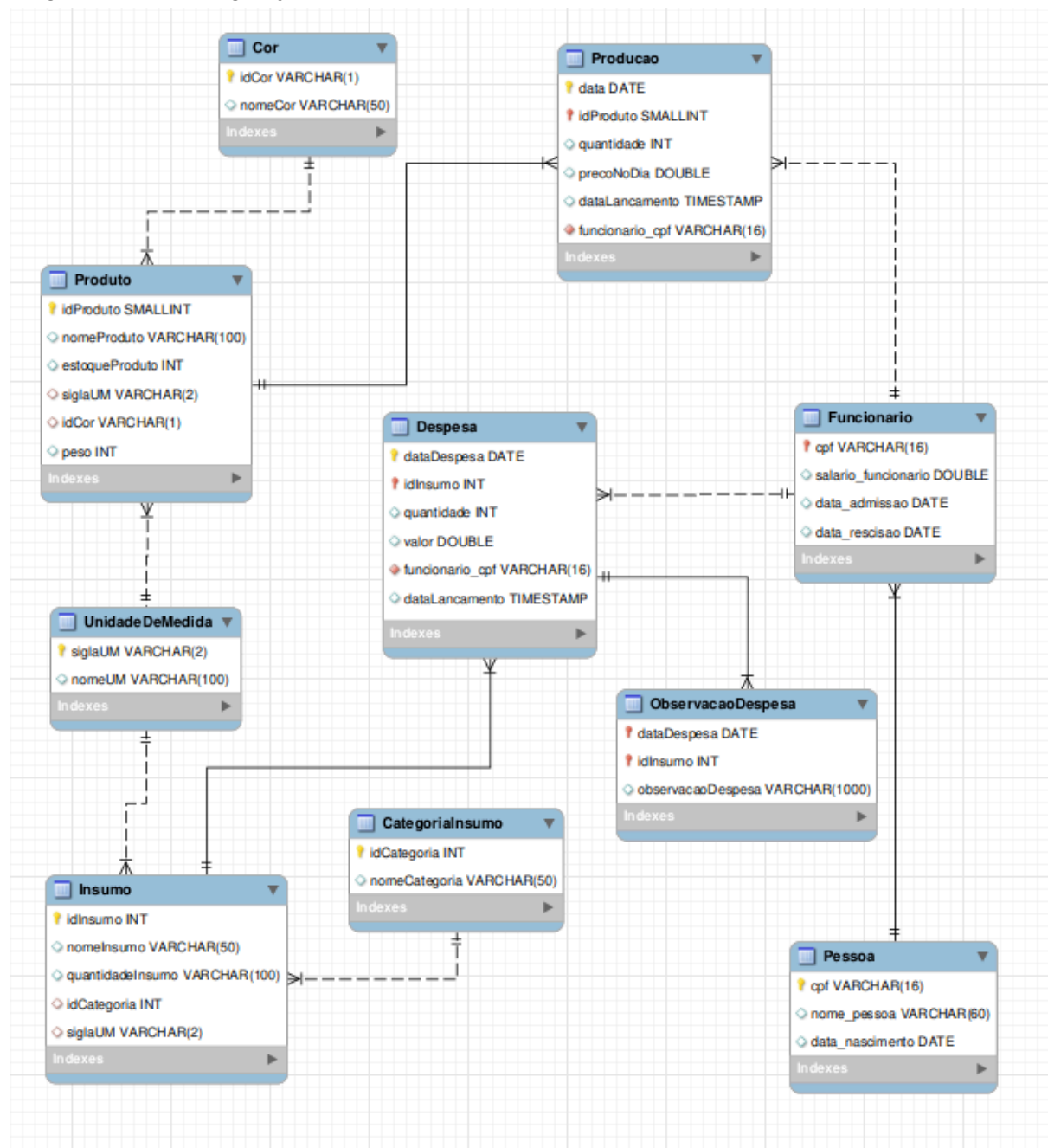
- **Dados Consistentes:** Evita anomalias e inconsistências nos dados, garantindo a confiabilidade das informações.
- **Implementação de Regras de Negócio:** Permite impor regras específicas do seu negócio, como a obrigatoriedade de um cliente ter pelo menos um pedido.

- **Redução de Erros:** Diminui a ocorrência de erros de digitação e inconsistências durante a manipulação dos dados.
- **Maior Qualidade dos Dados:** Garante que os dados armazenados sejam precisos e confiáveis para consultas e análises.

Ferramentas CASE

- Mysql Workbench
- dBeaver

Diagramas (DER) - granja



Views - Consultas - Selects

Desenvolvimento de consultas ao banco de dados. Ou seja, transformar dados em informações.

-- o comando SELECT possui (na forma básica)

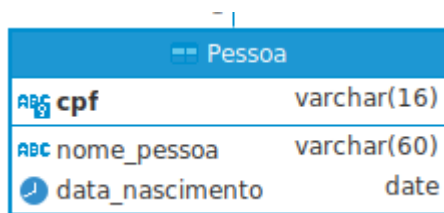
SELECT <lista de atributos>

FROM <lista de entidades>

WHERE <expressões lógicas>

Consultas em uma só entidade (tabela)

Exemplos



Pessoa	
cpf	varchar(16)
nome_pessoa	varchar(60)
data_nascimento	date

Tenha sempre em mãos o DER (Diagrama de Entidade e Relacionamento) para facilitar a sua vida.

- 1) Selecione todos os atributos da tabela pessoa e mostre os dados de todas as pessoas cadastradas

```
SELECT *  
FROM Pessoa p
```

-- Pessoa p, o p é um apelido para facilitar a digitação. Poderia ser SELECT * FROM Pessoa WHERE `Pessoa`.cpf = '123'.

- 2) Selecione o cpf e o nome de todas as pessoas cadastradas

```
SELECT p.cpf , p.nome_pessoa  
FROM Pessoa p
```

- 3) listar todos os dados da tabela pessoa ordenando pelo nome_pessoa

```
SELECT *  
FROM Pessoa  
ORDER BY nome_pessoa
```

- 4) listar os dados de UMA pessoa. A cláusula WHERE possibilita o uso de operadores lógicos =, >, <, >=, <=, !=, AND, OR sendo respectivamente igual, maior, menor, maior ou igual, menor ou igual, diferente, 'e' (lógico) e 'ou' (lógico)

```
SELECT *  
FROM Pessoa p
```

```
WHERE p.cpf = '123'
```

- 5) listar todas as pessoas que tem nomes começados pela letra "a".

```
SELECT *
FROM Pessoa p
WHERE p.nome_pessoa LIKE "A%"
ORDER BY nome_pessoa
```

É possível variar a expressão conforme o necessário. Por exemplo, todas as pessoas que possuem a sequência "ria" em alguma parte do nome.

```
SELECT *
FROM Pessoa p
WHERE p.nome_pessoa LIKE "%ria%"
ORDER BY nome_pessoa
```

- 6) listar todos os dados da tabela Producao

```
SELECT *
FROM Producao
```

- 7) listar todos os dados da tabela Producao especificando cada atributo e o nome do schema. O atributo `data` está entre crase por ser uma palavra reservada.

```
SELECT `data`, idProduto, quantidade, precoNoDia, dataLancamento,
funcionario_cpf
FROM granja.Producao;
```

Extras

--Usando apelido para facilitar a escolha dos atributos

```
SELECT prod.`data`, prod.idProduto, prod.quantidade, prod.precoNoDia,
prod.dataLancamento, prod.funcionario_cpf
FROM granja.Producao prod;
```

- 8) listar todos os dados e adicionar uma nova coluna virtual chamada subtotal(resultado de uma operação matemática)

```
SELECT prod.`data`, prod.idProduto, prod.quantidade, prod.precoNoDia,
prod.dataLancamento, prod.funcionario_cpf,
(prod.quantidade * prod.precoNoDia) AS subtotal
FROM granja.Producao prod;
```

- 9) selecionar e agrupar todos os produtos por idProduto (pois o idProduto se repete em diferentes datas) e soma as quantidades (por idProduto).Funções de agrupamento (sum, avg, etc)

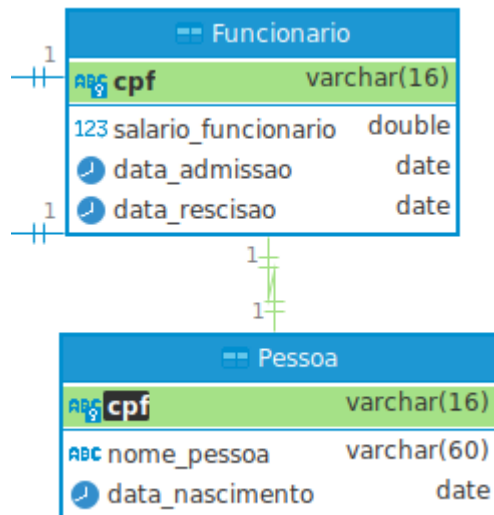
```
SELECT
```

```

    prod.idProduto,
    SUM(prod.quantidade) AS quantidadeTotal
FROM granjaAval.Producao prod
GROUP BY prod.idProduto;

```

Consultas em entidades com relacionamentos (juntando tabelas)



Uma pessoa pode “ser” funcionário (ou não). Considere que a Pessoa (cpf, nome_pessoa, data_nascimento) sendo, 111;’Berola da Silva’;’2000-10-25’ está cadastrada.

Para que essa pessoa “seja” funcionário temos que cadastrar na tabela (entidade) Funcionário. Por exemplo, cpf=’111’, salário=10000, data_admissao = ’2024-05-13’,data_recisao = NULL. Ou seja, se uma pessoa tem dados também na tabela Funcionário, ela é “além de pessoa” funcionário.

Como os dados estão em diferentes tabelas, precisamos “juntar” (JOIN).

```

SELECT *
FROM Pessoa p , Funcionario f
WHERE p.cpf = f.cpf

```

-- lendo a consulta de ‘forma humana’. Selecione todos os atributos das pessoas que são funcionários.

-- observe que é OBRIGATÓRIO igualar a PK (chave primária) com a FK (chave estrangeira) ao juntar tabelas.

```

-- listando o nome do funcionário e seu salário
SELECT p.nome_pessoa , f.salario_funcionario
FROM Pessoa p , Funcionario f
WHERE p.cpf = f.cpf ;

```

