ANOTAÇÕES DOS ESTUDOS DE BANCO DE DADOS (MYSQL)

FASE 1: Fundamentos de Banco de dados

Sumário

1. Introdução à Banco de dados

1.1. O que são Banco de Dados?

Um banco de dados é um sistema de armazenamento de informações que permite a coleta, o armazenamento, a recuperação e a manipulação de dados de maneira estruturada e eficiente, ou seja, trata-se de uma rede integrada de dados que serve para gerenciar e acessar informações de forma confiável.

Um banco de dados é essencial para muitas aplicações, desde empresas que gerenciam informações de clientes até aplicações científicas e governamentais que lidam com grandes volumes de dados.

Nos dias atuais, existem vários tipos de banco de dados, incluindo banco de dados Relacionais e banco de dados Não Relacional (por exemplo: NoSQL), além de outros. Cada um com suas próprias características e usos específicos.

Independentemente do tipo, os bancos de dados desempenham um papel fundamental na tomada de decisões e no suporte a operações críticas em organizações e sistemas de informação.

Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes. **Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados**. **A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL - Structured Query Language) para escrever e consultar dados.**

- **Exemplo Simples:** Imagine uma planilha do Excel onde você armazena informações de clientes. Essa planilha é um banco de dados rudimentar.
- **Por que usar?** Em cenários maiores, onde os dados crescem em quantidade e complexidade, ferramentas como MySQL permitem armazenar, organizar e consultar esses dados de maneira eficiente.

1.2. Banco de Dados Relacional e Não Relacional

Como foi visto anteriormente, nos dias atuais existem vários tipos de banco de dados. Os mais utilizados no mercado de trabalho são os dois mais principais, que são os Relacionais e os Não Relacionais.

- O que é um banco de dados relacional?

O banco de dados relacional é o tipo de banco que trabalha com tabelas relacionais, isto é, tabelas compostas por linhas e colunas, lembrando muito uma estrutura de tabela de Excel. Cada tabela representa uma entidade ou relação do mundo real.

As linhas representam registros individuais nessa entidade, e as colunas representam os atributos ou características dos registros.

A principal característica do banco relacional é a capacidade de estabelecer relacionamentos entre tabelas por meio de chaves primárias e estrangeiras. Isso permite que os dados fiquem associados e que, futuramente, sejam consultados de maneira eficiente, garantindo a integridade relacional.

Este modelo relacional é bastante utilizado em sistemas de gerenciamento de bancos de dados (SGBDs) e, **para trabalhar com ele, utilizamos SQL** (Structured Query Language, linguagem de consulta estruturada), **uma linguagem desenvolvida pela IBM na década de 70.** Bons exemplos de SGBDs mais usados no mercado são o Oracle, MySQL e Microsoft SQL Server.

- Quando utilizar um banco de dados relacional?

O uso de um banco de dados relacional é recomendado em várias situações, **principalmente quando temos um cenário que exige uma estrutura organizada e consistente**. E a linguagem SQL tem um papel fundamental para a manipulação e gerenciamento desses bancos.

Algumas situações onde o banco de dados relacional costuma ser adequado são:

1. **Estrutura de dados definida**
   - Um banco de dados relacional é adequado quando os dados possuem uma estrutura definida e há relações claras entre as entidades.

2. **Integridade dos dados críticos**
   - É recomendado utilizar um banco de dados relacional quando a integridade dos dados é crucial, especialmente em áreas como finanças, estoque ou registros de pacientes.

3. **Consultas complexas e agregações**

- Se você precisa realizar consultas complexas, como junções de tabelas, filtragem avançada, agrupamento e cálculos agregados, um banco de dados relacional é uma escolha adequada.

4. **Conformidade e segurança**
   - Se a conformidade com normas de segurança e regulamentações é fundamental, um banco de dados relacional fornece recursos avançados de segurança, como controle de acesso e criptografia.

VANTAGENS	DESVANTAGENS
▪ Estrutura bem definida, ideal para dados estruturados. ▪ Alta consistência e confiabilidade. ▪ Ferramentas maduras para gerenciamento e análise.	▪ Menos flexível para dados não estruturados ou dinâmicos. ▪ Pode ter dificuldade em escalar horizontalmente (distribuir dados entre vários servidores).
Exemplos de Bancos de Dados Relacionais • MySQL • PostgreSQL • Oracle Database • Microsoft SQL Server	

- E o que são Banco de Dados Não Relacional (NoSQL)?

Agora, quando falamos de NoSQL, vale ressaltar que **NoSQL não é uma linguagem**. **NoSQL é um termo que referência tipos de bancos de dados não relacionais**, ou seja, **que não seguem o modelo de tabelas e relacionamentos utilizado pelos bancos de dados relacionais tradicionais**.

Para esses bancos de dados NoSQL**, temos uma variedade de modelos, incluindo o modelo colunar, modelo de grafos, chave-valor e modelo orientado a documentos**.

Cada um desses modelos possui suas próprias características e é adequado para diferentes tipos de aplicação e necessidades de armazenamento de dados.

Um pouco sobre cada um deles abaixo:

1. **Modelo Colunar**

- Também conhecido como armazenamento de colunas, **é uma abordagem em que os dados são armazenados como colunas em vez de linhas**.
- Esse modelo é ideal para situações que envolvem grande quantidade de dados e exigem alta performance, pois permite que apenas as colunas relevantes sejam buscadas e lidas, economizando recursos de processamento.
- Uma das empresas que utilizam esse modelo é a Netflix, que utiliza o Cassandra para gravações de volume muito alto com baixa latência.

2. **Orientado a Documentos**
   - **Nesse modelo, os dados são armazenados em documentos no formato JSON**. Cada documento é identificado por uma chave única e pode conter diversas informações, como atributos e subdocumentos.
   - Esse modelo é interessante para aplicações que exigem flexibilidade na estrutura dos dados e que lidam com grande volume de informações.
   - Na Expedia, a empresa utiliza o modelo flexível do MongoDB que facilita o armazenamento de qualquer combinação de cidades, datas e destinos.

3. **Chave-valor**
   - **Os dados são armazenados em pares de chave-valor, o que significa que cada dado é identificado por uma chave única**.
   - Esse modelo é ideal para aplicações que exigem alta performance em leitura e gravação de dados, como em aplicações de cache ou armazenamento de sessões de usuários.
   - Este modelo usado pelo Twitter, que utiliza o Redis para implementar recursos em tempo real como contadores de retweets, curtidas e seguidores.

4. **Modelo de Grafos**
   - **Neste modelo os dados são usados para armazenar dados interconectados, como em redes sociais ou sistemas de recomendação**.
   - Com o modelo de grafos, é possível fazer buscas detalhadas nas relações entre os dados, mesmo em bancos com centenas de milhares de relacionamentos.

- O Medium, por exemplo, utiliza o Neo4j para criar grafos que representam as conexões entre usuários e artigos, permitindo a montagem de um sistema de recomendação.

- Quando utilizar bancos de dados não relacionais?

Os bancos não relacionais oferecem uma flexibilidade e escalabilidade muito vantajosa, principalmente quando se trata de grandes conjuntos de dados. **Mas como as operações dos bancos NoSQL dependem do tipo de modelo escolhido, para utilizá-lo, precisamos entender a necessidade de nosso negócio**, como:

1. **Aplicações que trabalham com cache**
   - Em cenários onde o desempenho de leitura e gravação é fundamental, como em um sistema que precise de armazenar dados frequentemente acessados de forma rápida (sistema de cache) em tempo real, os modelos chave-valor dos bancos NoSQL, como o Redis, são frequentemente utilizados devido à sua alta velocidade de acesso e recuperação.

2. **Sistemas de catálogos ou estruturas flexíveis**
   - Se a aplicação requer flexibilidade na estrutura e na consulta de dados, o modelo orientado a documentos, como MongoDB, pode ser uma boa escolha pela sua capacidade de conter informações de um objeto em um único documento.

Por estas e outras razões, é muito complicado comparar um modelo com outro, já que dependemos bastante do problema que precisamos resolver.

VANTAGENS	DESVANTAGENS
▪ Excelente para dados dinâmicos ou que mudam frequentemente. ▪ Escalabilidade horizontal fácil e eficiente. ▪ Suporte a diversos tipos de dados.	▪ Pode ser menos consistente, dependendo do modelo (alguns NoSQL adotam o modelo BASE em vez de ACID). o BASE: Disponibilidade Básica, Estado flexível e Eventual

	consistência. ▪ Ferramentas e suporte podem ser menos maduras comparados aos bancos relacionais.
Exemplos de Bancos de Dados Não Relacionais • MongoDB (documentos) • Cassandra (colunar) • Redis (chave-valor) • Neo4j (grafo) • Amazon DynamoDB (chave-valor)	

Em resumo, tanto os bancos de dados relacionais quanto os não relacionais têm seus pontos fortes e fracos, e a escolha entre um ou outro dependerá das necessidades específicas de cada aplicação. Os bancos de dados relacionais são ideais para aplicações que exigem consistência e integridade de dados, enquanto os bancos de dados não relacionais são mais adequados para aplicações que exigem alta escalabilidade e flexibilidade no esquema de dados. Por isso, ter conhecimento dessas ferramentas é interessante para qualquer profissional de dados.

Aspecto	Relacional	Não Relacional
Estrutura	Tabelas com esquemas rígidos	Flexível (documentos, grafos, colunas)
Modelo de Dados	Estruturado	Semi-estruturado ou não estruturado
Linguagem de Consulta	SQL	APIs ou linguagens específicas
Escalabilidade	Geralmente vertical	Horizontal
Consistência	Alta (ACID)	Eventual ou configurável (BASE)
Uso Comum	Sistemas financeiros, ERPs, CRMs	Big data, IoT, redes sociais, catálogos

1.3. O que é um Sistema de Gerenciamento de Banco de Dados (SGBD)?

Um **SGBD** (como MySQL, PostgreSQL ou Oracle) **é um software que gerencia bancos de dados.** Ele fornece ferramentas para**:**

- Criar bancos de dados.

- Inserir, atualizar e excluir dados.
- Recuperar informações específicas (consultas).
- Garantir segurança e integridade dos dados.

Principais características de um SGBD:

- **Persistência:** Dados continuam existindo mesmo após desligar o sistema.
- **Controle de acesso:** Permite que apenas pessoas autorizadas acessem ou modifiquem os dados.
- **Consistência:** Garante que os dados estejam sempre corretos e atualizados.
- **Confiabilidade:** Protege os dados contra falhas (backup, transações).
- **Escalabilidade:** Suporta um grande volume de dados e múltiplos usuários

Banco de Dados	SGBD
É o local onde os dados estão armazenados.	É o software que gerencia os dados no banco.
Exemplo: Uma tabela com dados de clientes.	Exemplo: MySQL, que permite criar e gerenciar essa tabela.
Estrutura estática (dados organizados).	Ferramenta dinâmica que aplica operações sobre os dados.

1.4. Terminologia básica

1. Dados

Informações armazenadas no banco de dados, representando fatos ou números que têm significado.

Exemplo: "João", "35", "joao@gmail.com".

2. Banco de Dados

Um conjunto organizado de dados armazenados e acessados eletronicamente.

Objetivo: Facilitar o armazenamento, a recuperação e o gerenciamento de dados.

Exemplo: Banco de dados de clientes de uma loja.

3. **SGBD (Sistema de Gerenciamento de Banco de Dados)**

Software usado para criar, gerenciar e interagir com o banco de dados.

Exemplos: MySQL, PostgreSQL, MongoDB.

4. **Tabela**

Estrutura principal de um banco de dados relacional. É composta por linhas (registros) e colunas (campos).

Exemplo: Tabela Clientes com colunas ID, Nome, Email.

5. **Registro (Linha ou Tupla)**

Uma entrada na tabela que representa uma instância específica.

Exemplo: Na tabela Clientes, um registro pode ser 1, João, joao@gmail.com.

6. **Campo (Coluna ou Atributo)**

Um elemento que define o tipo de dado armazenado em cada registro da tabela.

Exemplo: A coluna Nome armazena nomes dos clientes.

7. **Chave**

Identificadores ou referências em um banco de dados.

7.1. **Chave Primária (Primary Key)**: Um identificador único para cada registro em uma tabela.
Não pode conter valores duplicados ou nulos.
Exemplo: O campo ID em Clientes.

7.2. **Chave Estrangeira (Foreign Key)**
Um campo que cria uma relação entre tabelas.
Referência uma chave primária de outra tabela.
Exemplo: A tabela Pedidos pode ter uma chave estrangeira Cliente_ID que referência ID na tabela Clientes.

8. Relacionamento

A conexão entre tabelas baseada em chaves.

Tipos:

1:1 (Um para Um)	Um registro em uma tabela corresponde a um registro em outra.
1:N (Um para Muitos)	Um registro em uma tabela corresponde a vários na outra.
N:M (Muitos para Muitos	Vários registros em uma tabela correspondem a vários na outra.

9. Esquema (Schema)

A estrutura do banco de dados, incluindo tabelas, colunas, tipos de dados e relações.

Exemplo: Definição das tabelas Clientes, Pedidos e suas relações.

10. Consulta (Query)

Os tipos de consultas no SQL são classificados com base na finalidade de suas operações.

Comando utilizado para interagir com o banco de dados.

Tipos de Consultas:

DDL (Data Definition Language)	Define a estrutura do banco (Ex.: CREATE, ALTER, DROP).
DML (Data Manipulation Language)	Manipula os dados (Ex.: SELECT, INSERT, UPDATE, DELETE).
DCL (Data Control Language)	Controla permissões (Ex.: GRANT, REVOKE).
TCL (Transaction Control Language)	Gerencia transações (Ex.: COMMIT, ROLLBACK).

11. Índice

Estrutura que melhora a velocidade de consultas em uma tabela. Funciona como um índice em um livro.

Exemplo: Índice no campo Nome da tabela Clientes para buscar clientes mais rápido.

12. Normalização

Processo de organizar os dados para reduzir redundâncias e melhorar a integridade.

Exemplo: Dividir uma tabela grande em tabelas menores relacionadas.

13. Transação

Conjunto de operações realizadas como uma única unidade.
Segue as propriedades ACID:

- **A**tomicidade: Tudo ou nada.
- **C**onsistência: Mantém a validade dos dados.
- **I**solamento: Uma transação não interfere em outra.
- **D**urabilidade: Dados persistem mesmo após falhas.

14. Backup

Cópia de segurança do banco de dados, usada para recuperação em caso de falhas.

Exemplo: Realizar backups diários do banco de dados loja.

15. Banco de Dados Relacional x Não Relacional

Relacional: Organizado em tabelas com esquemas rígidos.

Não Relacional: Estruturas flexíveis, como documentos ou grafos.

2. Linguagem SQL – Introdução

2.1. O que é SQL?

A Linguagem de consulta estruturada (SQL) **é uma linguagem de programação para armazenar e processar informações em um banco de dados relacional**.

Um banco de dados relacional armazena informações em formato tabular, com linhas e colunas representando diferentes atributos de dados e as várias relações entre os valores dos dados.

Você pode usar instruções SQL para armazenar, atualizar, remover, pesquisar e recuperar informações do banco de dados. Também pode usar SQL para manter e otimizar a performance do banco de dados.

2.2. Por que a SQL é importante?

A Linguagem de consulta estruturada (SQL) é uma linguagem de consulta popular que é frequentemente usada em todos os tipos de aplicações. Analistas e desenvolvedores de dados aprendem e utilizam a SQL porque ela se integra bem a diferentes linguagens de programação.

Por exemplo, eles podem incorporar consultas SQL com a linguagem de programação Java para criar aplicações de processamento de dados de alta performance com os principais sistemas de banco de dados SQL, como Oracle ou MS SQL Server.

A SQL também é bastante fácil de aprender, pois usa palavras-chave comuns em inglês em suas instruções.

História da SQL

A SQL foi inventada na década de 1970 com base no modelo de dados relacionais. Ela era inicialmente conhecida como linguagem de consulta estruturada em inglês (SEQUEL). Mais tarde, o termo foi abreviado para SQL. A Oracle, antes conhecida como Relational Software, tornou-se o primeiro fornecedor a oferecer um sistema de gerenciamento de banco de dados relacional SQL em nível comercial.

2.3. Categorias de Comandos SQL

Os tipos de consultas (ou categorias de comandos) no SQL são classificados com base na finalidade de suas operações.

Elas se dividem em quatro categorias principais:
DDL (Data Definition Language),
DML (Data Manipulation Language),
DCL (Data Control Language) e
TCL (Transaction Control Language).

2.3.1. DDL (Data Definition Language)

DDL é usada para definir ou modificar a estrutura do banco de dados, como tabelas, colunas, índices e relacionamentos.

- **Principais comandos:**
  - **CREATE**: Cria novos objetos no banco de dados (base de dados, tabelas, etc).
    Exemplo: Criar uma tabela chamada Clientes:

```
CREATE TABLE Clientes (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100),
    Email VARCHAR(100)
);
```

  - **ALTER**: Modifica a estrutura de um objeto existente, como adicionar ou excluir colunas.
    Exemplo: Adicionar a coluna Telefone à tabela Clientes:

```
ALTER TABLE Clientes ADD Telefone VARCHAR(15);
```

  - **DROP**: Remove um objeto do banco de dados (tabela, índice, etc.).
    Exemplo: Excluir a tabela Clientes:

```
DROP TABLE Clientes;
```

  - **TRUNCATE**: Remove todos os registros de uma tabela, mas mantém sua estrutura.
    Exemplo:

```
TRUNCATE TABLE Clientes;
```

```sql
-- Criando uma tabela
CREATE TABLE Alunos (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Idade INT,
    Curso VARCHAR(50)
);

-- Alterando a tabela para adicionar uma nova coluna
ALTER TABLE Alunos ADD Email VARCHAR(100);

-- Excluindo a tabela
DROP TABLE Alunos;
```

2.3.2. DML (Data Manipulation Language)

DML é usada para manipular os dados armazenados no banco de dados, incluindo inserção, consulta, atualização e exclusão de registros.

- **Principais comandos:**
    - **INSERT**: Insere novos registros em uma tabela.
      Exemplo: Inserir um cliente na tabela Clientes:

```sql
INSERT INTO Clientes (ID, Nome, Email) VALUES (1, 'João', 'joao@gmail.com');
```

    - **UPDATE**: Atualiza registros existentes.
      Exemplo: Alterar o e-mail de um cliente:

```sql
UPDATE Clientes SET Email = 'joao.novo@gmail.com' WHERE ID = 1;
```

    - **DELETE**: Remove registros de uma tabela.
      Exemplo: Excluir o cliente com ID 1:

```sql
DELETE FROM Clientes WHERE ID = 1;
```

```sql
-- Inserindo registros
INSERT INTO Alunos (Nome, Idade, Curso)
VALUES ('João Silva', 20, 'TI');

-- Atualizando registros
UPDATE Alunos
SET Idade = 21
WHERE Nome = 'João Silva';

-- Deletando registros
DELETE FROM Alunos
WHERE Idade < 18;
```

2.3.3. DQL (Data Query Language)

Usada para consultar dados de uma tabela. O principal comando é o SELECT.

- **SELECT**: Recupera dados do banco de dados (ou seja, faz uma consulta no banco de dados para recuperar registros de uma ou mais tabelas).
  Exemplo: Selecionar todos os clientes:

```sql
-- Selecionando todos os registros
SELECT * FROM Alunos;

-- Selecionando colunas específicas
SELECT Nome, Curso FROM Alunos;

-- Filtrando registros
SELECT * FROM Alunos WHERE Curso = 'TI';

-- Ordenando resultados
SELECT * FROM Alunos ORDER BY Nome ASC;

-- Limitando resultados
SELECT * FROM Alunos LIMIT 5;
```

2.3.4. DCL (Data Control Language)

DCL é usada para gerenciar permissões e controlar o acesso dos usuários ao banco de dados.

- **Principais comandos:**

- **GRANT**: Concede permissões a usuários ou roles.
  Exemplo: Permitir que o usuário admin tenha acesso à tabela Clientes:

```
GRANT SELECT, INSERT ON Clientes TO 'admin';
```

- **REVOKE**: Remove permissões anteriormente concedidas.
  Exemplo: Revogar a permissão de inserção para o usuário admin:

```
REVOKE INSERT ON Clientes FROM 'admin';
```

```
-- Concedendo permissão de SELECT a um usuário
GRANT SELECT ON BancoDeDados.Alunos TO 'usuario'@'localhost';

-- Revogando permissão
REVOKE SELECT ON BancoDeDados.Alunos FROM 'usuario'@'localhost';
```

2.3.5. TCL (Transaction Control Language)

TCL é usada para gerenciar transações no banco de dados, garantindo que as operações sejam concluídas com sucesso ou revertidas em caso de erro.

- **Principais comandos:**
  - **COMMIT**: Finaliza uma transação e salva as alterações no banco de dados.
    Exemplo:

```
BEGIN;
INSERT INTO Clientes (ID, Nome, Email) VALUES (2, 'Maria', 'maria@gmail.com');
COMMIT;
```

  - **ROLLBACK**: Reverte uma transação para o estado anterior.
    Exemplo:

```
BEGIN;
INSERT INTO Clientes (ID, Nome, Email) VALUES (3, 'Carlos', 'carlos@gmail.com');
ROLLBACK;
```

- **SAVEPOINT**: Define um ponto de salvamento dentro de uma transação. Permite reverter para esse ponto em vez de reverter toda a transação.
  Exemplo:

```
BEGIN;
INSERT INTO Clientes (ID, Nome, Email) VALUES (4, 'Ana', 'ana@gmail.com');
SAVEPOINT Ponto1;
INSERT INTO Clientes (ID, Nome, Email) VALUES (5, 'Lucas', 'lucas@gmail.com');
ROLLBACK TO Ponto1;
COMMIT;
```

```
-- Início de uma transação
START TRANSACTION;

-- Atualizando registros
UPDATE Alunos SET Curso = 'Engenharia' WHERE Nome = 'João Silva';

-- Revertendo a alteração
ROLLBACK;

-- Confirmando a transação
COMMIT;
```

Resumo dos Tipos de Consultas:

Categoria	Objetivo Principal	Exemplos de Comandos
DDL	Definir e estruturar o banco de dados	`CREATE`, `ALTER`, `DROP`, `TRUNCATE`
DML	Manipular os dados armazenados	`SELECT`, `INSERT`, `UPDATE`, `DELETE`
DCL	Controlar permissões e acessos	`GRANT`, `REVOKE`
TCL	Gerenciar transações	`COMMIT`, `ROLLBACK`, `SAVEPOINT`

3. Referências

3.1. Sites

- https://www.alura.com.br/artigos/banco-de-dados?srsltid=AfmBOoqgQxxuH4Bwi4CemMMeYxhH0DTr2a-O_S42qppq7vTaJYhoa9Uu
- https://www.oracle.com/br/database/what-is-database/
- https://aws.amazon.com/pt/compare/the-difference-between-relational-and-non-relational-databases/#:~:text=Voc%C3%AA%20pode%20vincular%20as%20tabelas,para%20acessar%20e%20gerenciar%20dados.
- https://www.alura.com.br/artigos/sql-nosql-bancos-relacionais-nao-relacionais?srsltid=AfmBOorflvatYWXGSuhPX_xu81l5cRprvksvtzHg0DrsLSD64p1JN1c4
- https://www.alura.com.br/artigos/o-que-e-sql?srsltid=AfmBOooo6_bvMirJ8TkVynYIplzXZZDFT492870eoZQKUb51VN2NS4QT
- https://aws.amazon.com/pt/what-is/sql/

3.2. Livros / Cursos

-