Um Sistema de Gerenciamento de Banco de Dados (SGBD) é uma peça fundamental no mundo da tecnologia da informação. Trata-se de um software projetado para armazenar, organizar, recuperar e gerenciar dados de maneira eficiente e segura. Nesse sentido, as principais características de um Sistema de Gerenciamento de Banco de Dados são:

* **Armazenamento Estruturado**: Os SGBDs são projetados para armazenar dados de forma estruturada, o que significa que os dados são organizados em tabelas, com colunas que representam tipos específicos de informações. Essa estrutura facilita a busca e a recuperação de dados.
* **Segurança**: Os SGBDs oferecem recursos avançados de segurança, incluindo controle de acesso, autenticação e criptografia, para proteger os dados contra acesso não autorizado.
* **Recuperação de Dados**: Eles fornecem mecanismos para recuperar dados de maneira rápida e eficiente, usando consultas que permitem localizar informações específicas com facilidade.
* **Concorrência**: Os SGBDs são projetados para permitir o acesso concorrente aos dados, o que significa que várias pessoas ou aplicativos podem acessar e modificar os dados ao mesmo tempo, sem corrompê-los.
* **Integridade dos Dados**: Garantem a integridade dos dados por meio de restrições e validações, evitando a entrada de informações inconsistentes ou inválidas no banco de dados.
* **Recuperação de Falhas**: Os SGBDs incluem mecanismos de recuperação que garantem que os dados não sejam perdidos em caso de falhas no sistema, como panes de hardware.
* **Escalabilidade**: Permitem que os sistemas cresçam à medida que a quantidade de dados e o número de usuários aumentam, sem comprometer o desempenho.
* **Suporte a Múltiplos Usuários**: Podem atender a várias solicitações de usuários simultaneamente, gerenciando transações de forma eficiente.
* **Backup e Restauração**: Facilitam a criação de cópias de segurança regulares dos dados, tornando possível a restauração em caso de perda de dados.

Existem vários Sistemas de Gerenciamento de Banco de Dados (SGBD) disponíveis no mercado, cada um com suas características e usos específicos. Aqui estão alguns exemplos populares:

* **MySQL**: Um SGBD de código aberto amplamente utilizado, conhecido por sua velocidade e confiabilidade. É comumente usado em aplicativos da web e é compatível com várias linguagens de programação.
* **Oracle Database**: Um SGBD comercial poderoso que é frequentemente usado em empresas para gerenciar grandes volumes de dados. Ele oferece recursos avançados de segurança, escalabilidade e recuperação.
* **Microsoft SQL Server**: Desenvolvido pela Microsoft, é uma escolha popular para aplicativos Windows e é amplamente utilizado em ambientes corporativos. Oferece integração com tecnologias Microsoft.
* **PostgreSQL**: Outro SGBD de código aberto, conhecido por sua extensibilidade e suporte a recursos avançados. É usado em uma ampla gama de aplicativos, incluindo sistemas geoespaciais.
* **SQLite**: Um SGBD incorporado que é leve e não requer um servidor separado. É comumente usado em aplicativos móveis e navegadores da web.

### Principais Comandos

**SELECT**

O SELECT, realmente vai selecionar as informações que queremos visualizar da nossa tabela. Próxima coisa que colocamos será especificando, poderíamos colocar uma coluna, mas no caso, temos o asterisco que significa que queremos selecionar todos os dados. E de onde queremos que selecione esses dados? Para mostrar de onde queremos, colocamos a cláusula FROM, que em Inglês também quer dizer de onde, seguido pelo nome da tabela, ou de onde queremos que venham esses dados.

SELECT \* FROM tabelapedidos;

**WHERE**

Filtrando dados com algumas especificações usa-se a cláusula where. O WHERE vem de "onde". Vamos especificar onde alguma informação específica seja tal coisa.

**SELECT** \* **FROM** tabelafornecedores **WHERE** pais\_de\_origem = 'China'

**DISTINCT**

Como evitar duplicidades com a cláusula distinct. Como fazemos para filtrar, para não trazer esse cliente várias vezes repetidamente na nossa consulta

**SELECT** **DISTINCT** cliente **from** tabelapedidos;

**CREATE TABLE**

Para criar tabelas usa-se o comando CREATE TABLE. Para especificar como queremos que essa tabela seja, quais serão os nomes das colunas, quais os tipos de dados que cada coluna vai conter etc.

**CREATE** **TABLE** nometabela (

Campo **INT** **PRIMARY** KEY,

campo **VARCHAR** (250),

SQL é uma linguagem de consulta estruturada que permite criar, modificar e gerenciar bancos de dados e seus objetos. Para criar tabelas, bancos de dados e esquemas, utilizamos o comando CREATE, mas vamos entender a diferença de cada uma dessas estruturas que esse comando nos possibilita criar.

Banco de Dados:

* Um banco de dados é uma coleção de dados organizados e relacionados.
* Ele atua como um contêiner para todos os objetos relacionados a dados, como tabelas, índices, procedimentos armazenados, visões e esquemas.
* Pode conter múltiplos esquemas, que são usados para organizar objetos de banco de dados.
* É uma entidade de nível superior que armazena e gerencia informações em um sistema de gerenciamento de banco de dados (SGBD).

**CREATE** DATABASE BibliotecaDB;

Esquema (Schema):

* Um esquema é um contêiner lógico para objetos de banco de dados, como tabelas, visões, procedimentos armazenados, etc.
* É usado para organizar e segmentar objetos dentro de um banco de dados.
* Vários esquemas podem existir dentro de um único banco de dados, permitindo a separação de objetos e a aplicação de permissões específicas em cada esquema.
* É uma estrutura de nível inferior em relação ao banco de dados.

**CREATE** SCHEMA LivrosSchema;

**FOREIGN KEY**

A chave estrangeira é responsável por relacionar as tabelas, sendo a coluna que faz a ligação entre duas tabelas diferentes. O comando para especificar chave estrangeira é FOREIGN KEY. Entre parênteses, vamos colocar o nome da coluna que será a chave estrangeira dentro dessa tabela. Em seguida, escreveremos REFERENCES para referenciar qual tabela e coluna queremos fazer a ligação.

**FOREIGN** KEY (Categoria) **REFERENCES** tabelacategorias (id\_categoria),

Importante: A chave estrangeira sempre se relacionará com uma chave primária de outra coluna.

### Tipos de dados

Os bancos de dados armazenam uma variedade de tipos de dados para atender às necessidades de diferentes tipos de informações e aplicativos. A escolha dos tipos de dados a serem usados em um banco de dados depende da natureza dos dados que serão armazenados e processados. Aqui estão alguns dos tipos de dados mais comuns em um banco de dados:

**Texto (String):**

* CHAR: Armazena strings de tamanho fixo. Usado quando os valores têm um comprimento constante.
* VARCHAR: Armazena strings de tamanho variável. Apropriado para valores com comprimentos variáveis.
* TEXTO (TEXT): Armazena strings muito longas, como documentos ou descrições.

**Numérico:**

* INTEGER (INT): Armazena números inteiros.
* FLOAT: Armazena números de ponto flutuante, geralmente usados para valores com casas decimais.
* NUMERIC (DECIMAL): Armazena números com uma precisão específica, geralmente usados em aplicações financeiras.

**Data e Hora:**

* DATE: Armazena datas sem informações de horário.
* TIME: Armazena informações de horário.
* TIMESTAMP: Combina data e horário em um único tipo.

**Booleano:**

* BOOLEAN (BOOL): Armazena valores verdadeiros ou falsos.

**Binário:**

* BLOB (Binary Large Object): Armazena dados binários, como imagens, vídeos ou arquivos.
* BIT: Armazena valores binários, como 0 ou 1.

### Chave Primária

A "Primary Key" (Chave Primária) é um conceito fundamental no contexto de bancos de dados. Ela desempenha um papel crucial na organização e estruturação dos dados em um banco de dados relacional.

A Primary Key é um campo ou conjunto de campos em uma tabela de banco de dados que serve para identificar de forma exclusiva cada registro nessa tabela. Em outras palavras, é o meio pelo qual o sistema de gerenciamento de banco de dados (SGBD) garante que cada linha de dados seja única e possa ser acessada de forma eficiente.

Características da Primary Key:

* **Unicidade**: Cada valor na coluna de chave primária deve ser único em relação a todos os outros valores na mesma coluna. Isso garante que nenhum registro duplicado seja inserido na tabela.
* **Não nulo**: A chave primária não pode conter valores nulos. Cada registro deve ter um valor na coluna da chave primária.
* **Eficiência de pesquisa**: A chave primária é usada para acelerar a pesquisa de registros no banco de dados. Os SGBDs criam índices automaticamente nas colunas de chave primária para otimizar a recuperação de dados.

A chave primária é fundamental para manter a integridade dos dados e garantir que as operações de consulta e atualização sejam eficientes. Veja:

* **Identificação única**: Ela permite identificar cada registro de forma única, o que é essencial em muitos cenários de negócios. Por exemplo, em um banco de dados de clientes, a chave primária pode ser o número de identificação de cada cliente.
* **Integridade referencial**: A chave primária é usada em relacionamentos entre tabelas para garantir a integridade referencial. Isso significa que os registros relacionados em tabelas diferentes podem ser conectados de maneira confiável.
* **Eficiência**: Como mencionado anteriormente, as chaves primárias são indexadas automaticamente pelos SGBDs, tornando as consultas mais rápidas e eficientes.