Para que serve um banco de dados?

Bancos de dados têm diversas funcionalidades, sendo comumente usados em sistemas de software.

Bancos de dados são conjuntos de arquivos que dialogam entre si, armazenando uma vasta gama de dados: nomes, documentos, pagamentos, endereços, clientes, serviços etc. São configurados e gerenciados por meio das linguagens de programação, como Javascript, SQL, PL/SQL, entre outras.

SGDB
Um Sistema de Gerenciamento de Banco de Dados (SGBD) – do inglês Data Base Management System (DBMS) – é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais a interface é constituída pelas APIs (Application Programming Interface) ou drivers do SGBD, que executam comandos na linguagem SQL (Structured Query Language).”
Fonte: Wikipédia

Se o banco de dados armazena informações, para que ele funcione é necessário muita programação e dedicação, e o responsável por isso é o DBA — Database Administrator.

Características de um banco de dados

Bancos de dados devem executar procedimentos chamados de **transações**. A integridade de uma transação deve ser regida por quatro propriedades:

- **Atomicidade**

- Todas as ações devem ser concluídas com sucesso, ou o processo falha como um todo e toda a ação é desfeita (rollback). Se há sucesso em todas as ações a informação é mantida no banco (commit);

- **Consistência**

- Deve-se obedecer regras e restrições definidas em um banco, como por exemplo uso de chaves estrangeiras ou uso de campos únicos;

- **Isolamento**
- Cada transação deve ser independente de outras transações;
- **Durabilidade**
- Os resultados de uma transação devem ser permanentes, exceto se outra transação a desfizer.

  - Controle de Redundância;
  - Compartilhamento de Dados;
  - Controle de Acesso aos Dados;
  - Múltiplas Interfaces;
  - Representação de associações complexas;
  - Garantia de restrições de Integridade;
  - Recuperação de falhas.

Banco de Dados Relacional

Um **banco de dados relacional** é um banco de dados que modela os dados de uma forma que eles sejam percebidos pelo usuário como tabelas, ou mais formalmente relações.

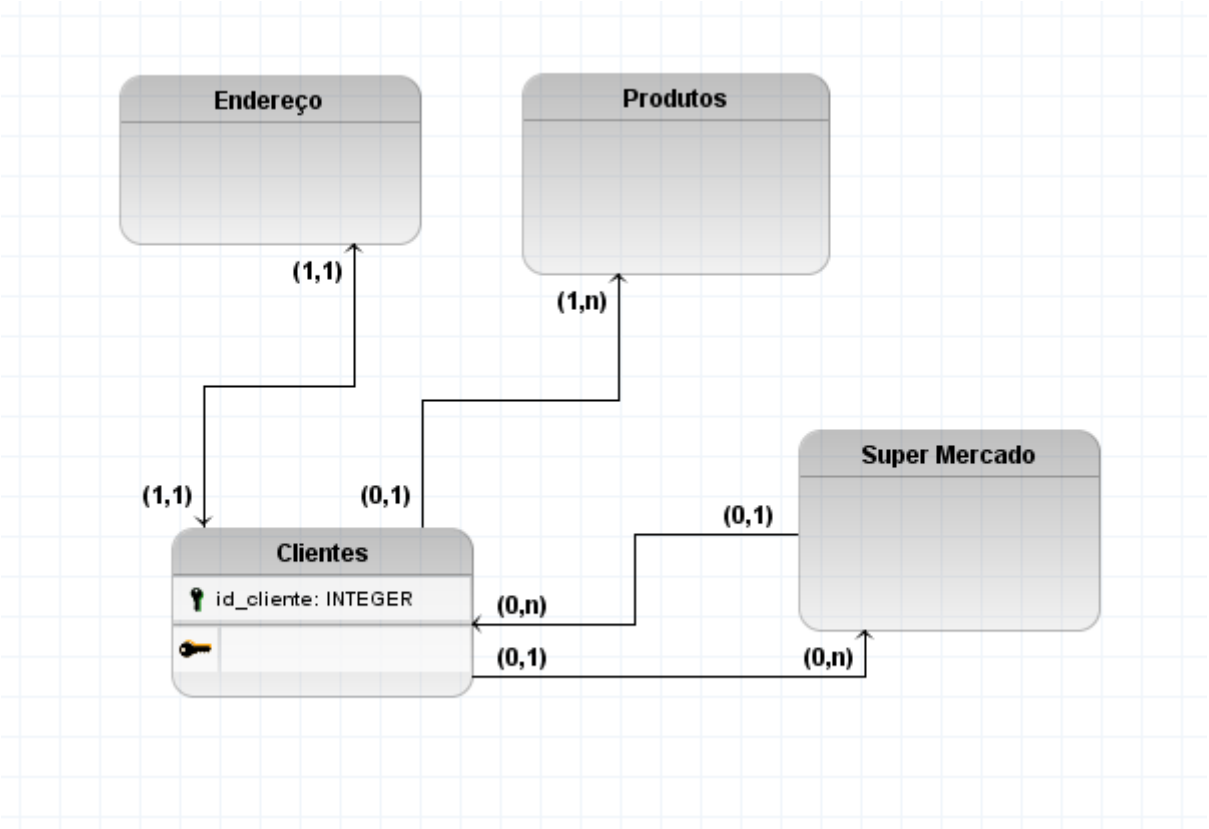
O termo é aplicado aos próprios dados, quando organizados dessa forma, ou a um **Sistema Gerenciador de Banco de Dados Relacional (SGBDR)**

Em **bancos de dados relacionais,** a **relação entre as tabelas** de dados é relacional. Bancos de dados relacionais conectam dados em tabelas diferentes, **usando elementos comuns de dados ou um campo chave**. Dados em bancos de dados relacionais **são armazenados em tabelas diferentes**, cada uma com um **campo chave que identifica cada linha ou registro**.

Cardinalidade

Em modelagem de dados a cardinalidade é um dos princípios fundamentais sobre relacionamento de um banco de dados relacional. Nela são definidos os graus de relação entre duas entidades ou tabelas.

No modelo relacional, podemos ter os seguintes níveis de relacionamento: **1:N**, **N:N**, **1:1**.



Tipos de Dados

char(n)	Tamanho fixo, completado com espaços em brancos	caracteres
varchar(n)	Tamanho variável com limite	8,000 caracteres
varchar(max)	Tamanho variável com limite	1,073,741,824 caracteres
bit	Número Inteiro que pode ser 0, 1 ou NULL	
tinyint	Permite números inteiros de 0 a 255	1 byte
smallint	Permite números inteiros entre -32,768 e 32,767	2 bytes

int	Permite números inteiros entre -2,147,483,648 e 2,147,483,647	4 bytes
float(n)	Precisão de número flutuante de -1.79E + 308 a 1.79E + 308.	
datetime	De 1 de janeiro de 1753 a 31 de dezembro de 9999 com uma precisão de 3,33 milisegundos	8 bytes
datetime2	De 1º de janeiro de 0001 a 31 de dezembro de 9999 com precisão de 100 nanossegundos	6-8 bytes
smalldatetime	De 1 de janeiro de 1900 a 6 de junho de 2079 com precisão de 1 minuto	4 bytes
date	Armazena apenas uma data. De 1 de janeiro de 0001 a 31 de dezembro de 9999	3 bytes
time	Armazena um tempo apenas para uma precisão de 100 nanosegundos	3-5 bytes
datetimeoffset	O mesmo que datetime2 com a adição de um deslocamento de fuso horário	8-10 bytes
timestamp	Armazena um número único que é atualizado sempre que uma linha é criada ou modificada. O valor do timestamp é baseado em um relógio interno e não corresponde ao tempo real. Cada tabela pode	

	ter apenas uma variável timestamp	

Linguagem SQL

SQL *(Structured Query Language)* quer dizer Linguagem de Consulta Estruturada. Permite a manipulação de tabelas do banco de dados.

Ela é a linguagem de busca de informações em bancos de dados relacionais. A linguagem SQL é dividida em:

DML - Linguagem de Manipulação de Dados

Permite manipulação de dados, como exclusão, inclusão e alterações. Exemplos de comandos:

- INSERT (permite adicionar dados)
- UPDATE (permite atualizar dados)
- DELETE (permite apagar dados)

DDL - Linguagem de Definição de Dados

Permite a criação e alteração de dados. Exemplos de comandos:

- CREATE TABLE (cria tabelas)
- ALTER TABLE (altera tabelas)

- DROP TABLE (apaga tabelas).

DQL - Linguagem de Consulta de Dados

Permite a realização de buscas nas tabelas dos bancos de dados. Exemplo de comando:

- SELECT (comando mais importante usado para realizar buscas)

Buscas podem ser melhoradas com cláusulas:

`SELECT * FROM TABELA`

Realiza uma busca por todos os dados `*` em uma tabela chamada TABELA. A clásula `FROM` indica a tabela.

Ainda há outras cláusulas:

- WHERE: indica as condições
- GROUP BY: realiza agrupamentos
- ORDER BY: ordena os dados

Ainda podemos combinar buscas com operadores lógicos.

- AND: avalia se duas condições são verdadeiras
- OR: avalia se uma condição é verdadeira
- NOT: negação

Operadores relacionais permitem fazer comparações nas consultas:

- < Menor

- > Maior
- <= Menor ou igual
- >= Maior ou igual
- = Igual
- <> Diferente

CREATE: CRIAR

Na linguagem SQL, podemos utilizar o comando CREATE para criar tabelas e bases de dados. CREATE é um comando do tipo DDL (*Data Definition Language*; ou Linguagem de Definição de Dados). Veja os exemplos a seguir:

```
CREATE DATABASE BASEDEDADOS
```

```
CREATE TABLE Cadastro(
  id_cad INT NOT NULL AUTO_INCREMENT,
  Nome VARCHAR(100) NOT NULL,
  Sobrenome VARCHAR(100) NOT NULL,
  CPF varchar(11) NOT NULL,
  PRIMARY KEY ( id_cad )
);
```

DROP versus DELETE

DROP: do inglês derrubar, soltar, jogar

DELETE: do inglês apagar, deletar

Na linguagem SQL, DROP é um comando do tipo DDL, ou seja, comando de definição de dados; enquanto, DELETE é um comando do tipo DML, ou seja, manipulação de dados.

- Use DROP para excluir tabelas e bases de dados;
- Use DELETE para deletar dados em tabelas.

drop table cadastro;

DELETE FROM cadastro WHERE id_cad = 1

delete from cadastro where Nome = "Joao"

Comando de inserção

INSERT: INSERIR

Na linguagem SQL, podemos utilizar o comando INSERT para inserir dados em uma tabela. INSERT é um comando do tipo **DML** (*Data Manipulation Language* - Linguagem de Manipulação de Dados). Veja os exemplos a seguir:

INSERT INTO TABELA (COLUNAS) VALUES("VALORES")

("INSERIR DENTRO DA TABELA (NOME DAS COLUNAS QUE QUER FAZER INSERÇÃO) OS VALORES ('COLOCAR VALORES ENTRE ASPAS OU APÓSTROFOS')").

* Observe que números não precisam de aspas.

insert into cadastro (Nome,Sobrenome) Values('Joao','Silva')

Comando de seleção

SELECT: SELECIONAR

Na linguagem SQL, podemos utilizar o comando SELECT para ler dados de tabelas. SELECT é um comando do tipo DQL (*Data Query Language* - Linguagem de Consulta de dados). Veja os exemplos a seguir:

```
SELECT * FROM MINHA_TABELA
```

(“SELECIONAR * DA MINHA_TABELA", onde asterisco representa "TUDO")

```
select *  from cadastro
```

Você poderia especificar colunas que deseja fazer a seleção, alterando o *. Por exemplo:

```
SELECT NOME FROM ALUNOS
```

("SELECIONAR (coluna) NOME DA (tabela) ALUNOS")

```
select Nome  from cadastro
```

Comando de atualização

UPDATE: ATUALIZAR

Na linguagem SQL, podemos utilizar o comando UPDATE para atualizar dados em uma tabela. UPDATE é um comando do tipo **DML** (*Data Manipulation Language* - Linguagem de Manipulação de Dados). Veja os exemplos a seguir:

```
UPDATE MINHA_TABELA SET CAMPO="NOVO VALOR" WHERE ID =1
```

("ATUALIZAR MINHA TABELA DEFINA O CAMPO = "NOVO VALOR" ONDE ID=1)

* Neste exemplo, o ID representa qual linha será atualizada. No caso estamos atualizando a coluna chamada CAMPO na primeira linha.

```
update cadastro set Nome = 'Jose' where Nome = 'Renato'
```

```
ALTER TABLE cadastro ADD CEP int(9) AFTER CPF
```