Banco de dados

Tipos de banco de dados

Temos dois tipos de banco de dados, sendo eles:

- Banco de dados relacional (SQL)
- Banco de dados não relacional (NoSQL)

Relacional (SQL): Um banco de dados relacional organiza as informações em tabelas, como uma planilha, onde cada coluna é um tipo de dado (como nome, idade, endereço) e cada linha é um registro (cada pessoa ou objeto que estamos armazenando). Utiliza a **linguagem SQL** para fazer as **manipulações** nos dados. **Ex:** bancos, controle de estoque.

Não relacional (NoSQL): Os bancos de dados não relacionais não seguem essa estrutura de tabelas. Eles guardam os dados de formas diferentes (como documentos (JSON), pares de chave e valor, grafos, colunas), permitindo uma estrutura mais livre. Você não precisa definir um formato fixo para os dados, o que facilita adicionar informações variadas (*). Não necessariamente utiliza a linguagem SQL para fazer as manipulações nos dados. **A linguagem para a manipulação dos dados varia** conforme a estrutura de banco não relacional escolhida. **Ex:** redes sociais, e-commerce.

(*) Para entender melhor, veja o exemplo abaixo de um formato JSON:

- Documento 1:

```
{
  "nome": "Maria",
  "idade": 30,
  "endereco": "Rua A, 123"
}
```

- Documento 2:

```
{
  "nome": "João",
  "idade": 25,
  "telefone": "1234-5678",
  "email": "joao@email.com"
}
```

Podemos notar que no documento 2 há um campo a mais e dois campos diferentes do documento 1. Note, também, que em um banco relacional isso não acontece dessa maneira.

Diferenças (SQL vs NoSQL)

Principais Diferenças

1. **Estrutura e Esquema de Dados**:
    a. **Relacional**: Tabelas com esquema rígido; ideal para dados bem definidos e estruturados.
    b. **Não Relacional**: Esquema flexível; permite armazenar dados variados, ideal para dados que mudam frequentemente.

2. **Escalabilidade**:
    a. **Relacional**: Escala verticalmente (aumentando a capacidade de um servidor), o que pode ser caro e complicado.
    b. **Não Relacional**: Escala horizontalmente (adicionando mais servidores), fácil de distribuir e ideal para aplicações que crescem rapidamente.

3. **Consistência**:
    a. **Relacional**: São focados em garantir que os dados fiquem sempre corretos e completos, essenciais para sistemas com alta necessidade de precisão.
    b. **Não Relacional**: São projetados para serem rápidos e disponíveis, mesmo que isso permita uma pequena inconsistência temporária nos dados, o que é aceitável em muitos cenários de alto volume de usuários.

4. **Manipulação de dados**:
    a. **Relacional**: Excelente para consultas complexas e joins entre várias tabelas.
    b. **Não Relacional**: Limitado para consultas complexas, e joins geralmente não são suportados; otimizado para buscas simples e rápidas.
    c. Joins: é uma operação para combinar dados de duas ou mais tabelas com base em uma condição comum entre elas. Eles são muito úteis quando os dados estão distribuídos em diferentes tabelas e precisamos unir essas informações em uma consulta para obter uma visão completa.

5. **Desempenho com Dados Não Estruturados**:
    a. **Relacional**: Melhor para dados estruturados e com esquema fixo.
    b. **Não Relacional**: Melhor para dados não estruturados, semi-estruturados e grandes volumes de dados que não seguem um padrão.

Analogia

Banco SQL (álbum de figurinhas organizado): Tudo tem seu lugar certinho e é fácil de achar, mas não é tão rápido para trocar de lugar.

Banco NoSQL (caixa mágica): Dá mais liberdade para guardar as coisas como quiser, mas pode ser mais bagunçado.

“**SQL** é como uma biblioteca bem organizada, com livros em prateleiras definidas.”

“NoSQL é mais como uma estante de ideias, onde você pode guardar dados de diferentes formas, mas sem seguir uma estrutura tão rígida.”

Database

Types of Databases

There are two types of databases:

- Relational Database (SQL)
- Non-relational Database (NoSQL)

Relational (SQL): A relational database organizes information in tables, like a spreadsheet, where each column is a data type (such as name, age, address) and each row is a record (each person or object we are storing). It uses the SQL language to manipulate data. Examples: banks, inventory control.

Non-relational (NoSQL): Non-relational databases do not follow this table structure. They store data in various formats (like documents (JSON), key-value pairs, graphs, columns), allowing a more flexible structure. You don't need to define a fixed format for the data, making it easier to add varied information (*). They don't necessarily use SQL for data manipulation; the data manipulation language varies depending on the chosen non-relational database structure. Examples: social media, e-commerce.

(*) To better understand, see the example below:

- Document 1:

```
{
"name": "Maria",
"age": 30,
"address": "Street A, 123"
}
```

- Document 2:

```
{
"name": "João",
"age": 25,
"phone": "1234-5678",
"email": "joao@email.com"
}
```

We can see that Document 2 has an additional field and two different fields from Document 1. In a relational database, this would not occur in this manner.

Differences (SQL vs NoSQL)

Main Differences

1. **Data Structure and Schema:**
    a. **Relational:** Tables with a rigid schema; ideal for well-defined and structured data.
    b. **Non-Relational:** Flexible schema; allows for storing varied data, ideal for data that changes frequently.

2. **Scalability:**
    a. **Relational:** Scales vertically (increasing the capacity of a server), which can be expensive and complicated.
    b. **Non-Relational:** Scales horizontally (adding more servers), easy to distribute and ideal for applications that grow quickly.

3. **Consistency:**
    a. **Relational:** Focused on ensuring that data is always correct and complete, essential for systems with high precision requirements.
    b. **Non-Relational:** Designed to be fast and available, even if this allows for a small temporary inconsistency in the data, which is acceptable in many high-volume user scenarios.

4. **Data manipulation:**
    a. **Relational:** Excellent for complex queries and joins between multiple tables.
    b. **Non-Relational:** Limited for complex queries, and joins are generally not supported; optimized for simple and fast searches.
    c. **Joins:** is an operation to combine data from two or more tables based on a common condition between them. They are very useful when the data is distributed across different tables and we need to join this information in a query to obtain a complete view.

5. **Performance with Unstructured Data:**
    a. **Relational:** Best for structured data with a fixed schema.
    b. **Non-Relational:** Best for unstructured, semi-structured, and large volumes of data that do not follow a standard.

Analogy

SQL database (organized sticker album): Everything has its right place and is easy to find, but it is not so quick to change places.

NoSQL database (magic box): It gives you more freedom to store things however you want, but it can be more messy.

“**SQL** is like a well-organized library, with books on defined shelves.”

“**NoSQL** is more like a bookshelf of ideas, where you can store data in different ways, but without following such a rigid structure.”