

**awari**

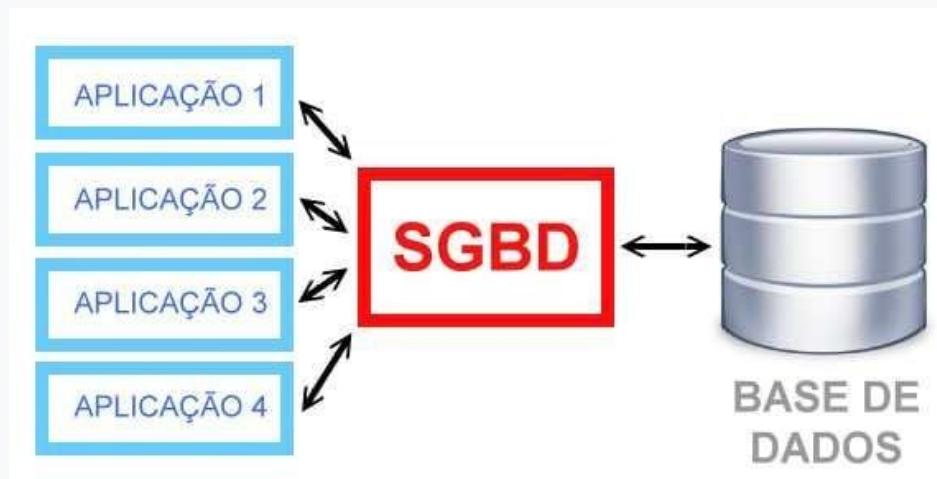
# Banco de Dados e SGBD

Onde fazemos a persistência dos dados

# O que é um SGBD?



**SGBD** é o acrônimo para Sistema de Gerenciamento de Banco de Dados (em inglês, DBMS - Database Management System). **SGBD** é um conjunto de softwares utilizado para o gerenciamento de uma ou mais bases de dados. Esse conjunto é responsável por controlar, acessar, organizar e proteger as informações de uma aplicação, tendo como principal objetivo gerenciar as bases de dados utilizadas por aplicações clientes e remover esta responsabilidade das mesmas.



# Tipos de SGBD

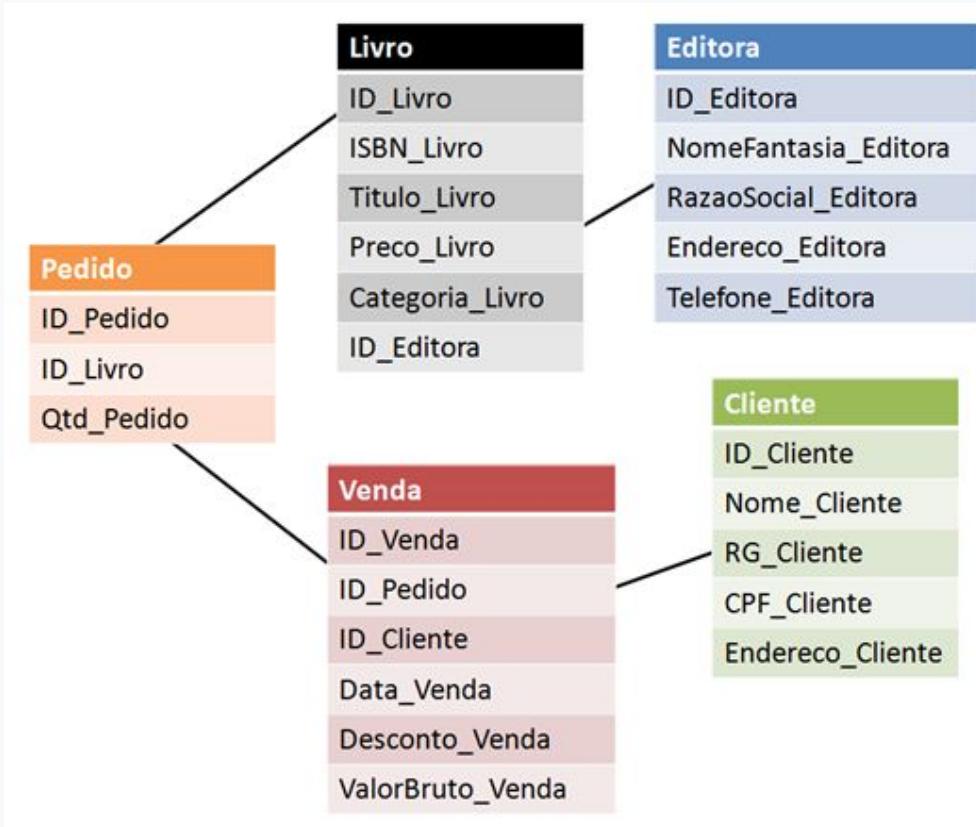


- ❖ **Relacionais** - Os SGBDS relacionais são banco de dados que modelam os dados no formato de tabelas, que podem se relacionar entre si. Cada tabela pode possuir diversos atributos, com diversos tipos de dados.
- ❖ **Não-relacionais (NoSQL)** - NoSQL (Not Only SQL) é o termo utilizado para banco de dados não relacionais de alto desempenho, onde geralmente não é utilizado o SQL como linguagem de consulta. Estes bancos utilizam diversos modelos de dados, incluindo documentos, gráficos, chave-valor e colunares. São amplamente reconhecidos pela facilidade em seu desenvolvimento, desempenho escalável, alta disponibilidade e resiliência.
- ❖ **Hierárquico** - Modelo hierárquico de banco de dados consiste em uma coleção de arquivos que são conectados entre si por meio de ligações, baseando a sua base de dados em um modelo de entidades e relacionamentos.
- ❖ **De rede** - Possuindo uma organização semelhante ao modelo de banco de dados hierárquico, o modelo de rede possui uma estrutura mais completa, possuindo como principal diferença entre eles o fato de não existir restrição hierárquica. É um modelo que permite a organização dos dados em uma estrutura formada por várias listas, formada por um conjunto complexo de ligações.
- ❖ **Orientado a objetos** - Baseado no paradigma da programação orientada a objetos, neste modelo as funcionalidades de orientação a objetos são integradas aos bancos de dados, onde cada informação é armazenada em forma de um objeto e seus registros em forma de tuplas.

# Relacional x Não-relacional



# Relacional

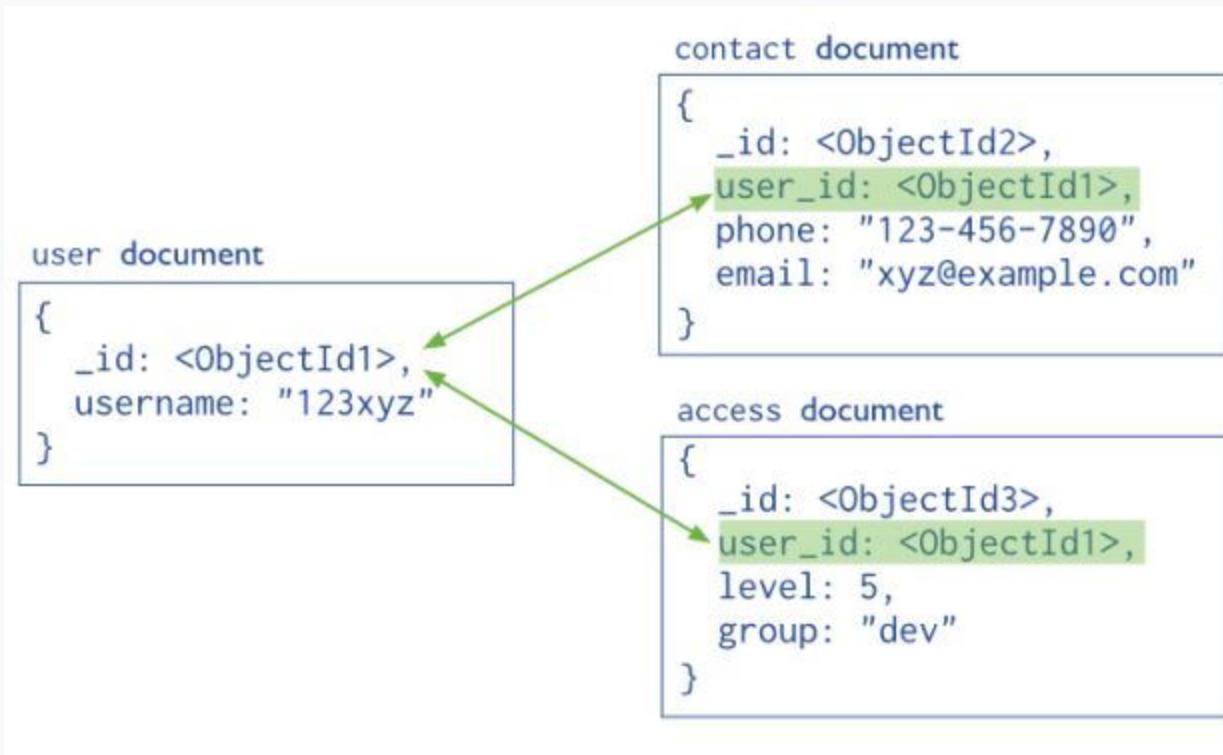


# Não-relacional



```
{  
    "id": 55,  
    "Pais": "Brasil",  
    "Regiao": "América do Sul",  
    "Populacao": 201032714,  
    "PrincipaisCidades": [  
        {  
            "NomeCidade": "São Paulo",  
            "Populacao": 1182876,  
        },  
        {  
            "NomeCidade": "Rio de Janeiro",  
            "Populacao": 6323037,  
        }  
    ]  
}
```

# Não-relacional



**awari**

**C.R.U.D.**

Operações fundamentais em banco de dados

# O que é um CRUD?

**CRUD** é o acrônimo para Create, Read, Update e Delete (Criar, Ler, Atualizar e Apagar). Com esse resumo, já dá para imaginar que o **CRUD** é uma sequência de funções de um sistema que trabalha com banco de dados, seja ele local (na sua máquina) ou online (na nuvem).

Este conjunto de operações pertence a um conjunto de operações do bancos de dados chamado de *Data Manipulation Language* (DML) ou, em português, Linguagem de Manipulação de Dados.

Este conjunto de operações da linguagem SQL é usado para recuperar, incluir, remover e modificar informações dentro de um banco de dados.

# O que é um CRUD?



CREATE



READ



UPDATE



DELETE

---

C R U D

# O que é um CRUD?

- Comandos SQL		Verbos do Protocolo HTTP	-
-			-
<hr/>			
- CREATE		POST	-
- READ		GET	-
- UPDATE		PUT	-
- DELETE		DELETE	-
<hr/>			

**awari**

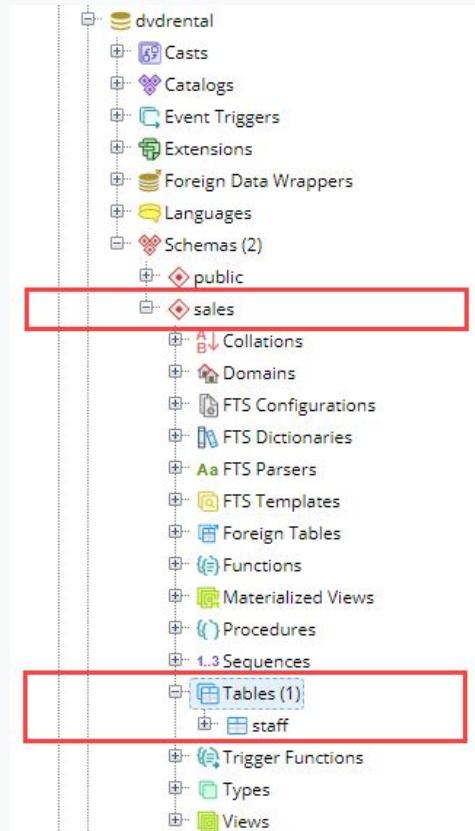
# **Esquema, Instância e Estado**

Qual a diferença entre esses termos?

# O que é um Esquema de Banco de Dados?

Um esquema de banco de dados representa a configuração lógica da totalidade ou de parte de uma base de dados relacional. Ele pode existir tanto como uma representação visual, quanto como um conjunto de fórmulas conhecidas como restrições de integridade que regem um banco de dados.

Estas fórmulas são expressadas em uma linguagem de definição de dados, como o SQL, por exemplo. Como parte de um dicionário de dados, um esquema de banco de dados indica como as entidades que compõem o banco de dados se relacionam entre si, incluindo tabelas, exibições, procedimentos armazenados e muito mais.



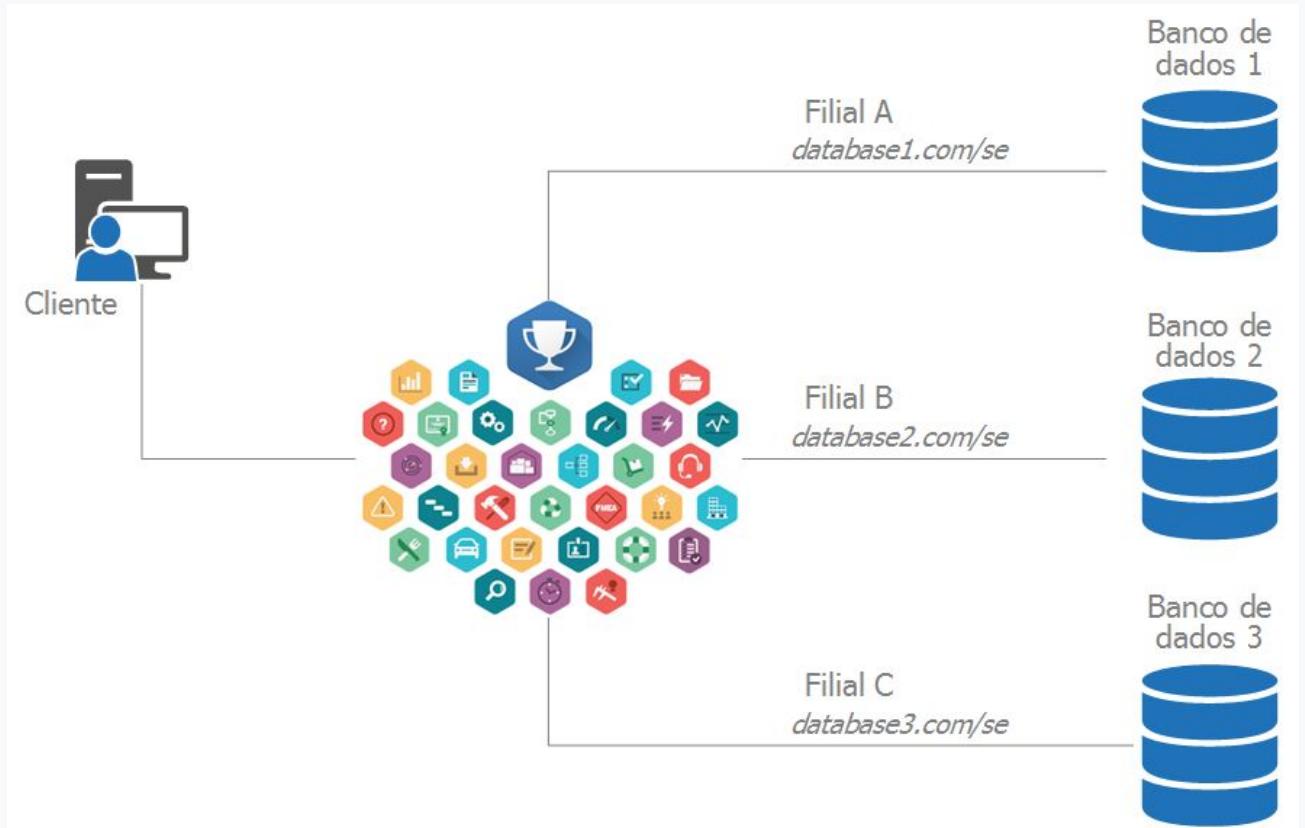
# Esquema ou Instância do Banco de Dados?

Estes termos, embora relacionados, não significam a mesma coisa. Um esquema de banco de dados é um esboço de um banco de dados planejado. Na verdade, ele não contém dados. Uma instância de banco de dados, por outro lado, é um retrato de um banco de dados da forma como existia em um determinado momento.

Sendo assim, instâncias de banco de dados podem mudar ao longo do tempo, enquanto um esquema de banco de dados é geralmente estático, já que é difícil mudar a estrutura de um banco de dados a partir do momento que estiver operacional.

Esquemas e instâncias de banco de dados podem se afetar mutuamente por meio de um Sistema de Gerenciamento de Banco de Dados (SGBD). O SGBD assegura que cada instância de banco de dados esteja em conformidade com as restrições impostas pelos designers do banco de dados no esquema de banco de dados.

# Esquema ou Instância do Banco de Dados?



# Estados do Banco de Dados



Podemos dizer que estados são determinados momentos em que o banco de dados se encontra. Como se fossem “fotografias” do banco de dados em um determinado momento. Sempre que é realizada uma nova inserção, alteração ou exclusão de dados, dizemos que um novo estado foi criado.

The screenshot shows three Microsoft Access tables stacked vertically:

- Produtos**: Shows two rows with products "1 Chai" and "2 Chang" from supplier "Exotic Liquids".
- Clientes**: Shows three rows with company names: "Alfreds Futterkiste", "Ana Trujillo Emparedados y helados", and "Maria Anders".
- Pedidos**: Shows three rows of order details. The first row has order ID 10248, customer "Wilman Kala", and employee "Buchanan, Steve". The second row has order ID 10249, customer "Tradição Hipermercados", and employee "Suyama, Mich". The third row has order ID 10250, customer "Hanari Carnes", and employee "Peacock, Marg".

**awari**

# Índices

Mais rapidez nas consultas

# O que é um Índice?

O **índice**, no contexto de banco de dados, é uma das ferramentas de otimização mais conhecida e utilizada por desenvolvedores em geral.

O uso de indexação em algumas colunas das tabelas pode aumentar consideravelmente a performance em consultas ao banco de dados. Mas, pode diminuir a velocidade de transações em *inserts* e *updates*. Portanto, saber quais tarefas são mais importantes no sistema em questão – se de leitura ou de escrita – nos dá um ponto de partida para saber como lidar com a indexação.

Codigo = 3? Não. Desconsidere. →

Codigo = 3? Não. Desconsidere. →

Codigo = 3? Sim. Retornar registro. →

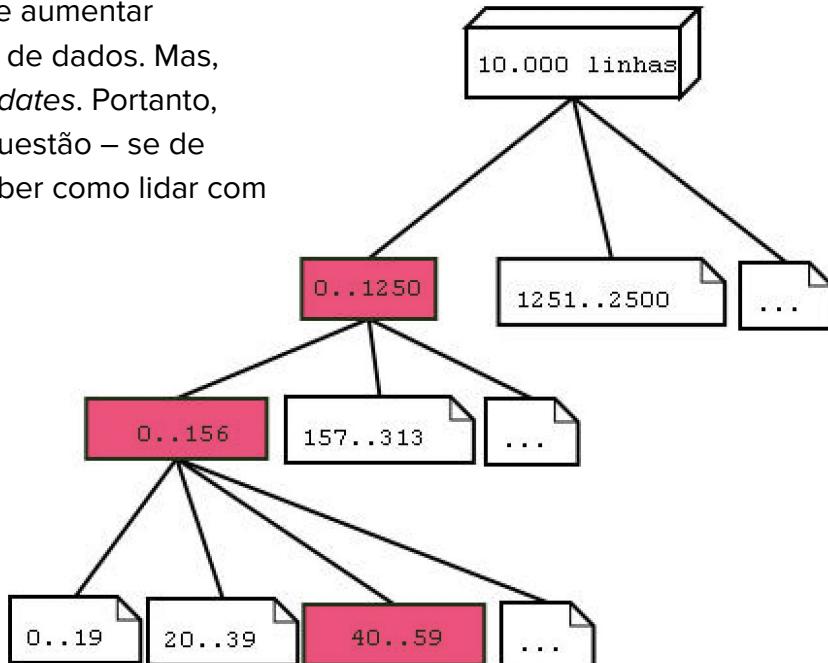
Codigo	Nome
1	Joao
2	Antonio
3	Maria

# O que é um Índice?



O **índice**, no contexto de banco de dados, é uma das ferramentas de otimização mais conhecida e utilizada por desenvolvedores em geral.

O uso de indexação em algumas colunas das tabelas pode aumentar consideravelmente a performance em consultas ao banco de dados. Mas, pode diminuir a velocidade de transações em *inserts* e *updates*. Portanto, saber quais tarefas são mais importantes no sistema em questão – se de leitura ou de escrita – nos dá um ponto de partida para saber como lidar com a indexação.



**awari**

# **Modelo Entidade Relacionamento**

Como estruturar seu banco relacional

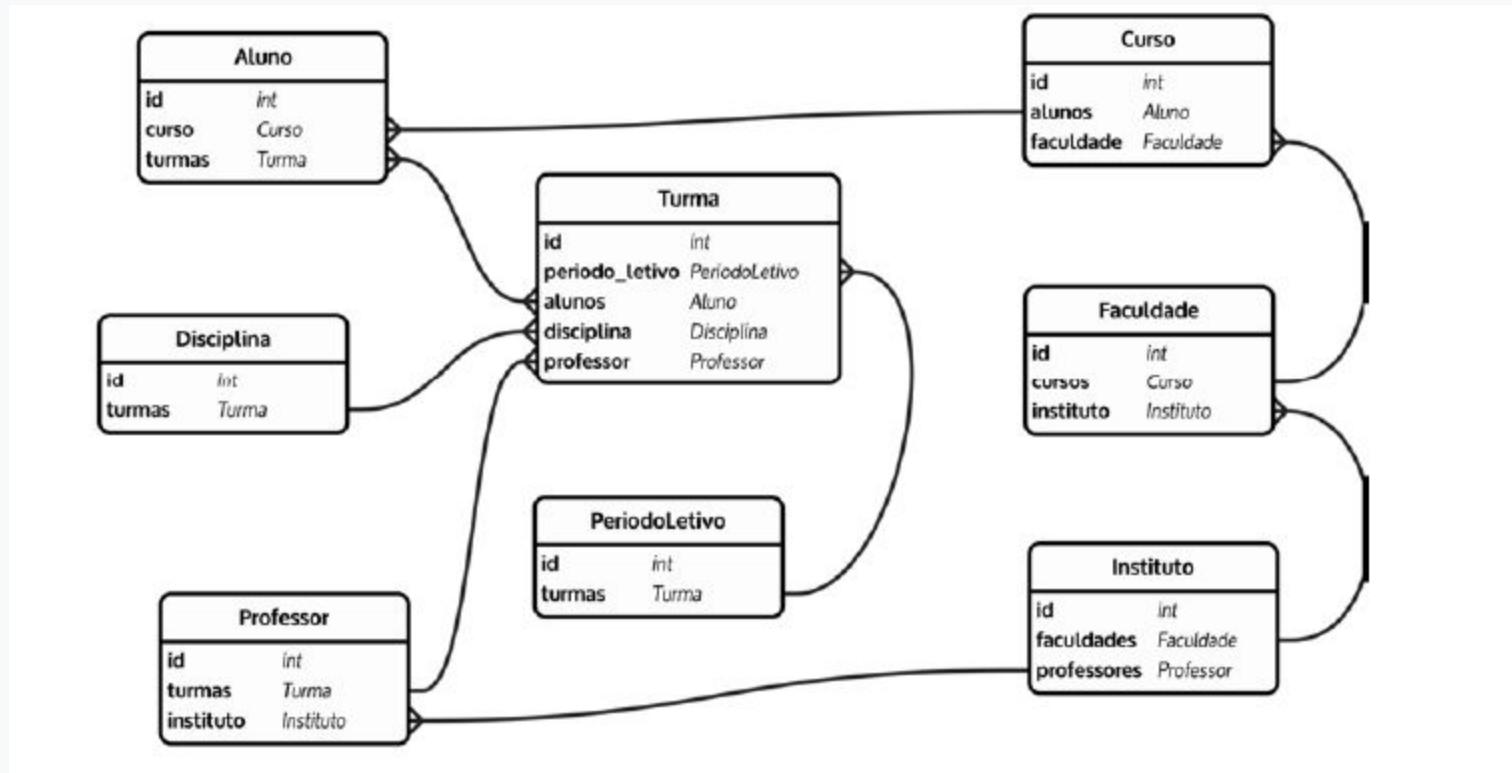
# O que é o Modelo Entidade Relacionamento?



O **Modelo Entidade Relacionamento** (também chamado **Modelo ER**, ou simplesmente **MER**), como o nome sugere, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

Em geral, este modelo representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação. Obviamente, o banco de dados poderá conter várias outras entidades, tais como chaves e tabelas intermediárias, que podem só fazer sentido no contexto de bases de dados relacionais.

# O que é o Modelo Entidade Relacionamento?



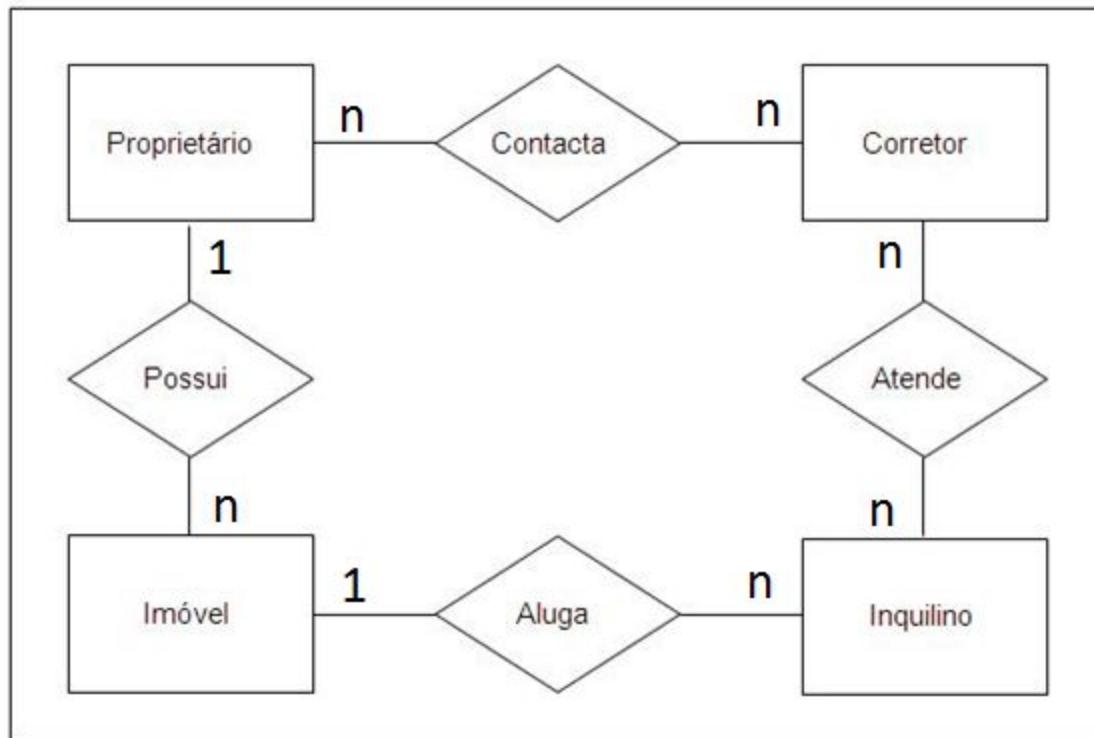
# O que é o Diagrama Entidade Relacionamento?

Enquanto o MER é um modelo conceitual, o **Diagrama Entidade Relacionamento** (Diagrama ER ou ainda DER) é a sua representação gráfica e

principal ferramenta. Em situações práticas, o diagrama é tido muitas vezes como sinônimo de modelo, uma vez que sem uma forma de visualizar as informações, o modelo pode ficar abstrato demais para auxiliar no desenvolvimento do sistema. Dessa forma, quando se está modelando um domínio, o mais comum é já criar sua representação gráfica, seguindo algumas regras.

O diagrama facilita ainda a comunicação entre os integrantes da equipe, pois oferece uma linguagem comum utilizada tanto pelo analista, responsável por levantar os requisitos, e os desenvolvedores, responsáveis por implementar aquilo que foi modelado.

# O que é o Diagrama Entidade Relacionamento?



**awari**

# **Modelo Relacional**

Entendendo o relacionamento na prática

# O que é o Modelo Relacional?



O **Modelo Relacional** se divide, basicamente, nos seguintes itens:

- ❖ Entidades e atributos
- ❖ Chave primária
- ❖ Relacionamentos entre entidades
- ❖ Integridade Referencial
- ❖ Normalização de tabelas

# Entidades e Atributos



O início de uma modelagem acontece a partir das **entidades**. Uma **entidade** é uma representação de um conjunto de informações sobre determinado conceito do sistema. Toda entidade possui **atributos**, que são as informações que referenciam a entidade.

Por exemplo, em um sistema de controle para uma biblioteca, partimos do princípio que a principal responsabilidade do sistema é a de empréstimo de livros por usuários da biblioteca. A partir disso, vamos abrindo as possibilidades e descobrindo novos conceitos.

Podemos iniciar nosso raciocínio da seguinte forma:

Uma biblioteca possui livros que podem ser tomados em empréstimo pelos usuários matriculados.

- ❖ Cadastro de Livros
- ❖ Cadastro de Usuários
- ❖ Registro de Empréstimos

# Entidades e Atributos



- ❖ LIVROS
  - Título
  - Autor
  - Editora
  - Ano
  - Edição
  - ISBN
- ❖ USUÁRIOS
  - Nome
  - Matrícula
  - Documento de Identificação
  - Endereço
- ❖ EMPRÉSTIMOS
  - Livro
  - Usuário
  - Data do Empréstimo
  - Data de Devolução

# Chave Primária



- ❖ LIVROS
  - ISBN (PK)
  - Título
  - Autor
  - Editora
  - Ano
  - Edição
- ❖ USUÁRIOS
  - Matrícula (PK)
  - Nome
  - Documento de Identificação
  - Endereço
- ❖ EMPRÉSTIMOS
  - Livro (PK)
  - Usuário (PK)
  - Data do Empréstimo (PK)
  - Data de Devolução

# Relacionamentos



- ❖ **Relacionamento Um-Para-Um (1:1)** - Uma instância da entidade X relaciona-se a uma instância da entidade Y
- ❖ **Relacionamento Um-Para-Muitos (1:N)** - Uma instância da entidade X relaciona-se a várias instâncias da entidade Y
- ❖ **Relacionamento Muitos-Para-Muitos (N:M)** - Várias instâncias da entidade X relacionam-se a várias instâncias da entidade Y

# Relacionamento Um-Para-Um (1:1)



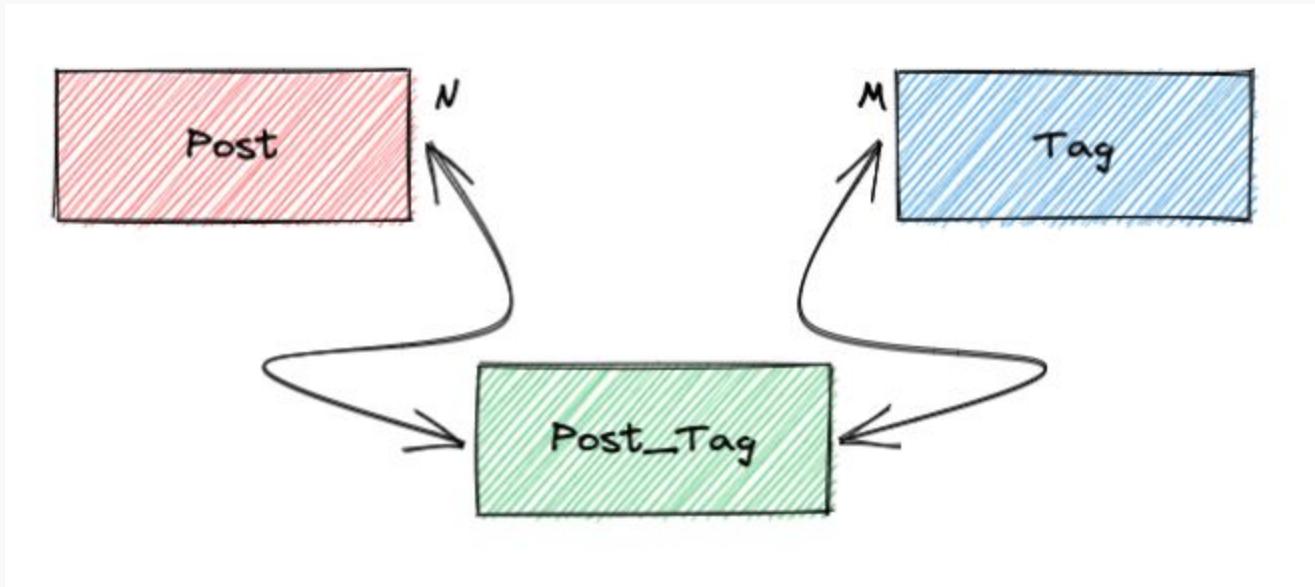
# Relacionamento Um-Para-Muitos (1:N)



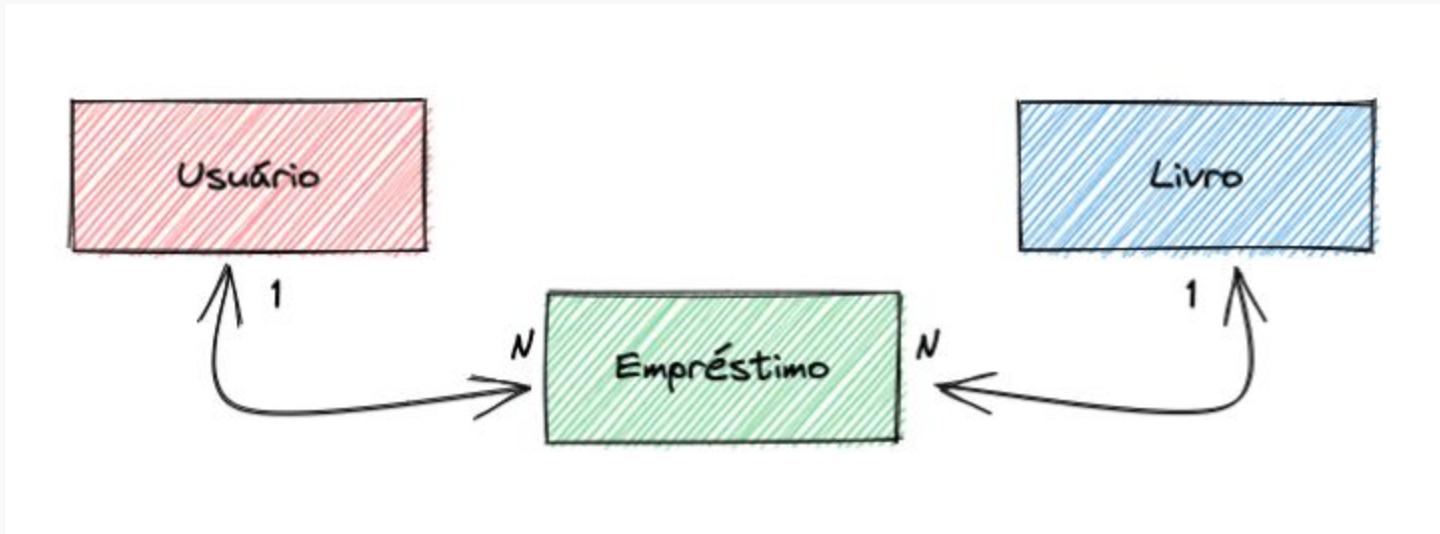
# Relacionamento Muitos-Para-Muitos (N:M)



# Relacionamento Muitos-Para-Muitos (N:M)



# Sistema para controle de Biblioteca



awari

SQL

Structured Query Language

# O que é o SQL?

**SQL** é uma linguagem padrão para trabalhar com **bancos de dados relacionais**. É uma linguagem declarativa e que não necessita de profundos conhecimentos de programação para que alguém possa começar a escrever **queries** (consultas), que trazem resultados de acordo com o que você está buscando.

**SQL** é utilizado de maneira relativamente parecida entre os principais bancos de dados relacionais do mercado: Oracle, **MySQL**, PostgreSQL, Microsoft SQL Server, entre muitos outros. Todos usam **SQL**, mas com pequenas diferenças entre si.



**SQL**

# Entendendo o SQL



id	titulo	data_cadastro	paginas
1	Entendendo o SQL	2022-01-15	150
2	Javascript para Leigos	2022-01-21	220
3	TypeScript do Zero	2022-01-22	84
4	Node.js com Express.js	2022-01-25	120
5	MongoDB	2022-02-09	50
6	Lógica de Programação	2022-02-10	35

# Entendendo o SQL (SELECT)



```
SELECT * FROM livros WHERE paginas > 100
```

# Entendendo o SQL (SELECT)



id	titulo	data_cadastro	paginas
1	Entendendo o SQL	2022-01-15	150
2	Javascript para Leigos	2022-01-21	220
4	Node.js com Express.js	2022-01-25	120

# Entendendo o SQL (INSERT)



```
INSERT INTO livros (titulo, dataCadastro, paginas) VALUES ('JS Design Patterns',  
CURRENT_DATE(), 135);
```

# Entendendo o SQL (INSERT)



id	titulo	data_cadastro	paginas
1	Entendendo o SQL	2022-01-15	150
2	Javascript para Leigos	2022-01-21	220
4	Node.js com Express.js	2022-01-25	120
7	JS Design Patterns.	2022-01-28	135

# Entendendo o SQL (UPDATE)



```
UPDATE livros SET titulo = 'Javascript: Design Patterns' WHERE id = 7;
```

# Entendendo o SQL (UPDATE)



id	titulo	data_cadastro	paginas
1	Entendendo o SQL	2022-01-15	150
2	Javascript para Leigos	2022-01-21	220
4	Node.js com Express.js	2022-01-25	120
7	Javascript: Design Patterns	2022-01-28	135

# Entendendo o SQL (DELETE)



```
DELETE from livros WHERE id = 2;
```

# Entendendo o SQL (DELETE)



id	titulo	data_cadastro	paginas
1	Entendendo o SQL	2022-01-15	150
4	Node.js com Express.js	2022-01-25	120
7	Javascript: Design Patterns	2022-01-28	135

awari



NoSQL

Not Only SQL

# O que é o NoSQL?

O termo **NoSQL** foi originalmente criado em 1998 por Carlo Strozzi e, posteriormente, reintroduzido por **Eric Evans** em 2009, quando participou da organização de um evento para discutir bancos de dados open source e distribuídos. Por falar em distribuídos, esse é um conceito amplamente utilizado pelos bancos NoSQL: basicamente, esses bancos operam em computação distribuída, um conceito que aumenta muito sua escalabilidade e performance.

Não existe uma definição oficial para o que esse termo significa de fato, mas uma das versões mais difundidas é: **Not Only SQL** (Não Somente SQL). Essa definição enfatiza que esses bancos podem utilizar linguagens semelhantes ao SQL para realizar consultas e demais operações, e não somente o SQL em si.

# NoSQL e suas classes



- ❖ Chave / Valor (Key / Value)
- ❖ Família de Colunas (Column Family)
- ❖ Documentos (Document)
- ❖ Grafos (Graph)

# NoSQL: Chave/Valor



Essa primeira classe é considerada a mais simples. Os dados são armazenados num esquema de registros compostos por uma chave (identificador do registro) e um valor (todo o conteúdo pertencente àquela chave). Você consegue recuperar um registro do seu banco de dados através da chave. Consultas por algum conteúdo através do valor não são permitidas.

A maioria dos bancos de dados **Chave/Valor** utilizam-se do recurso de armazenamento *in-memory* (memória RAM). Com isso, o acesso aos dados é extremamente rápido. Alguns cuidados, porém, devem ser tomados na questão da persistência desses dados, uma vez que eles estarão em uma área de memória volátil, não fazendo persistência em disco.

Essa volatilidade se dá porque a memória RAM é totalmente apagada quando os computadores são reiniciados ou desligados, ou seja, é uma área temporária.

Sistemas que requerem algum tipo de cache utilizam bastante essa classe de bancos de dados. Um exemplo de banco de dados da classe **Chave/Valor** é o **Redis**.



# NoSQL: Família de Colunas



Subindo um pouco mais a complexidade dos dados armazenados, essa segunda classe armazena os dados como um conjunto de três chaves: linha, coluna e timestamp. As linhas e colunas concentram os dados, e as diferentes versões desses dados são identificadas pelo timestamp.

Destaque para o conceito de masterless, ou seja, não existe um único servidor no cluster que concentra a escrita; essas operações são atendidas pelo servidor que estiver mais "próximo" de onde a operação vier. Sistemas em que dados analíticos em grande escala são o ponto-chave, o uso dessa classe é altamente recomendável.

Um exemplo de banco de dados da classe **Família de Colunas** é o **Cassandra**.



# NoSQL: Grafos



Essa é a classe que consegue armazenar dados muito complexos. Os dados são compostos por **nós** (vértices do grafo), **relacionamentos** (arestas do grafo), e as **propriedades** (ou atributos) dos **nós** ou **relacionamentos**. Note que o relacionamento é o ponto central dessa classe. Nesses bancos de dados, o relacionamento é físico, sendo persistido como qualquer outro dado dentro do banco. Dessa forma, as consultas que requerem esses relacionamentos são extremamente performáticas.

Os **Grafos** estão muito mais presentes em seu dia a dia do que você possa imaginar. Empresas e aplicativos de transporte ou GPS, por exemplo, utilizam os algoritmos e bancos de dados de **Grafos** para diversas de suas operações. Como encontrar o motorista mais perto de você, calcular o menor caminho de um ponto a outro e até mesmo fazer recomendações de produtos em sites de comércio eletrônico. Sistemas de recomendação e antifraude também têm encaixe perfeito para essa classe.

Um exemplo de banco de dados da classe **Grafos** é o **Neo4j**.



# NoSQL: Documentos



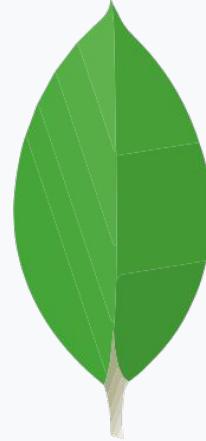
A classe mais flexível e com ampla aderência em vários casos de uso. Os dados são armazenados em estilo JSON, podendo ter vários níveis e subníveis, o que confere aos dados armazenados possibilidade de ter maior complexidade.

A estrutura de um **documento** é muito parecida com o que armazenamos na classe Chave/Valor. Porém, com **Documentos**, não temos apenas uma chave e sim um conjunto de chaves e valores. Por mais que *schemaless* (sem o uso de schema) seja um ponto presente na maioria dos bancos de dados NoSQL, na classe de **Documentos** temos esse conceito mais presente justamente pelo uso do JSON como padronização. Isso porque a inclusão, remoção ou alteração de tipos de dados são muito mais simples e fluídos utilizando JSON.

Sistemas que requerem uma gama de informações com diversos layouts e esquemas se encaixam muito bem nessa classe. Um exemplo de banco de dados da classe **Documentos** é o **MongoDB**.



# NoSQL e MongoDB



mongoDB

# JSON: Dados em Javascript



# Galeria de ícones



## Logo

awari awari

awari

# Cores

<span style="color: #1673FA;">█</span>	#1673FA
<span style="background-color: #F7F9FB; display: inline-block; width: 1em; height: 1em;"></span>	#F7F9FB
<span style="background-color: #FFFFFF; display: inline-block; width: 1em; height: 1em;"></span>	#FFFFFF
<span style="background-color: #202124; display: inline-block; width: 1em; height: 1em;"></span>	#202124
<span style="background-color: #595959; display: inline-block; width: 1em; height: 1em;"></span>	#595959

# Fontes

## Título

Highlight

Body - Title

Body - Regular

## Sombra