Escolhendo o banco de dados para minha aplicação

SQL OU NOSQL

About Me

Weder Mariano de Sousa

Specialist Senior Java - GFT

Technician **System Development**

Graduated **Computer Science**

Post Graduate in **Midias UFG**

Post Graduate in **Information Security**

GOJava

AWS User Group Goiânia

AWS certified Cloud Practitioner FOUNDATIONAL

AWS certified Developer ASSOCIATE

AWS certified Solutions Architect ASSOCIATE

AWS community builder Serverless

https://www.linkedin.com/in/wedermarianodesousa/

https://github.com/weder96

https://twitter.com/weder96

https://dev.to/weder96

AWS Technical Accredited PARTNER

AWS Cloud Economics Accredited PARTNER

AWS Business Accredited PARTNER

AWS TECHNOLOGY File Storage SPECIALIZED

AWS TECHNOLOGY Storage Data Migration SPECIALIZED

AWS TECHNOLOGY Storage Data Protection & Disaster Recovery SPECIALIZED

AWS Well-Architected PROFICIENT

AWS TECHNOLOGY Amazon Braket

AWS TECHNOLOGY Architecting

AWS TECHNOLOGY Migration Foundations

AWS TECHNOLOGY Amazon EKS

AWS TECHNOLOGY Networking Core

AWS TECHNOLOGY Serverless

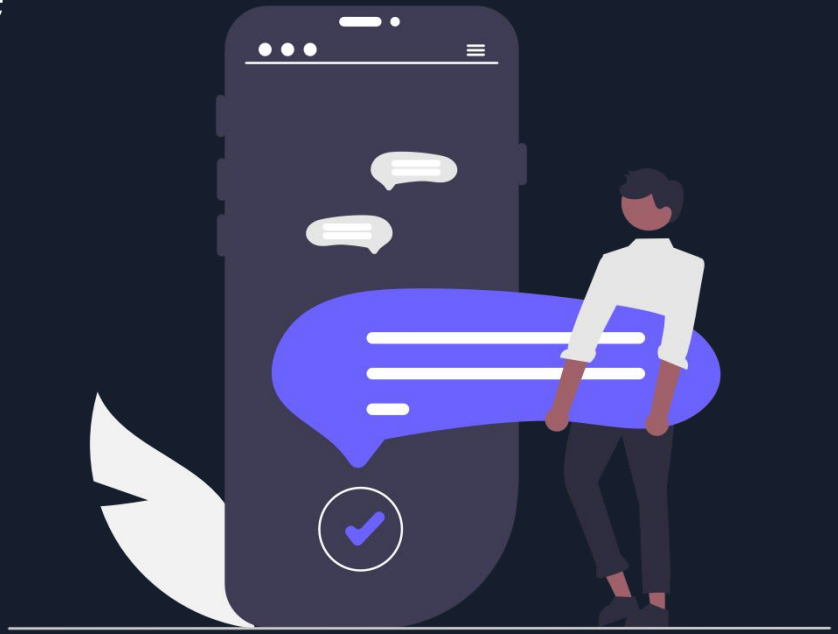
AWS TECHNOLOGY Events and Workflows

AWS TECHNOLOGY Compute

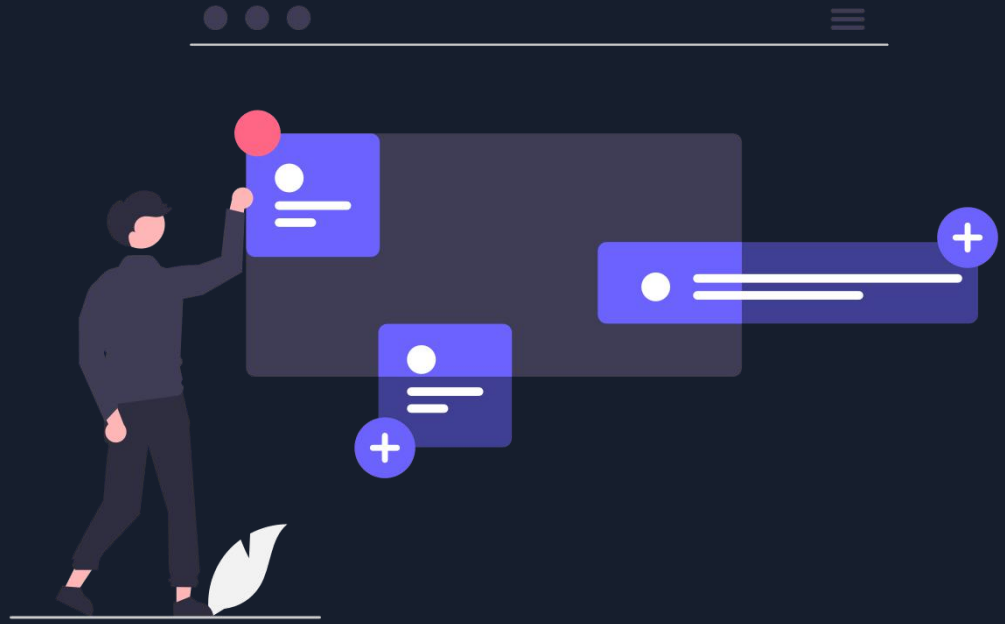
AWS TECHNOLOGY Cloud Essentials

O valor do banco de dados relacional

Persistência de Dados

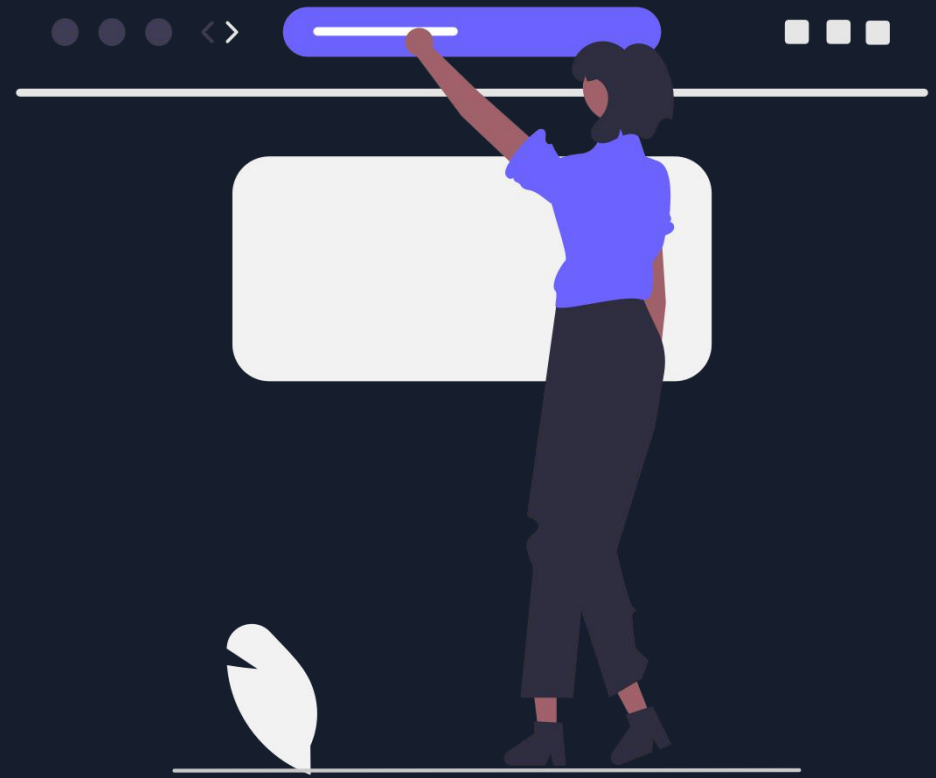


Concorrência



Suporte de transação

Suporte de Integração



Mesmo Dialeto

Incompatibilidade de impedância

Exemplos de Incompatibilidade de Impedância

- Diferenças de Estruturas de Dados
- Divergência nos Tipos de Dados
- Problemas de Consultas e Sintaxe
- Interoperabilidade de Aplicações

Consequências da Incompatibilidade:

Perda de Dados
Complexidade de Integração
Desempenho Insatisfatório

Como Resolver a Incompatibilidade de Impedância

Mapeamento de Estruturas e Tipos de Dados:

Criar um mapa que relacione os tipos de dados e estruturas entre os diferentes sistemas. Isso pode incluir a conversão de tipos de dados (por exemplo, transformar um campo de data de um formato para outro).

Middleware e Ferramentas de Integração:

Usar soluções de middleware que possam traduzir dados entre sistemas ou ferramentas ETL (Extração, Transformação e Carga) para mover e transformar dados automaticamente, adequando-os às necessidades dos sistemas de destino.

Implementação de APIs:

Desenvolver APIs que abstraiam a lógica de acesso aos dados. APIs bem projetadas podem oferecer uma interface uniforme, independentemente da origem dos dados, facilitando a interoperabilidade.

Como Resolver a Incompatibilidade de Impedância

Uso de Formatos de Dados Universais:

Adotar formatos de dados padrão como JSON ou XML para facilitar a comunicação entre diferentes sistemas, uma vez que esses formatos são amplamente suportados em várias tecnologias.

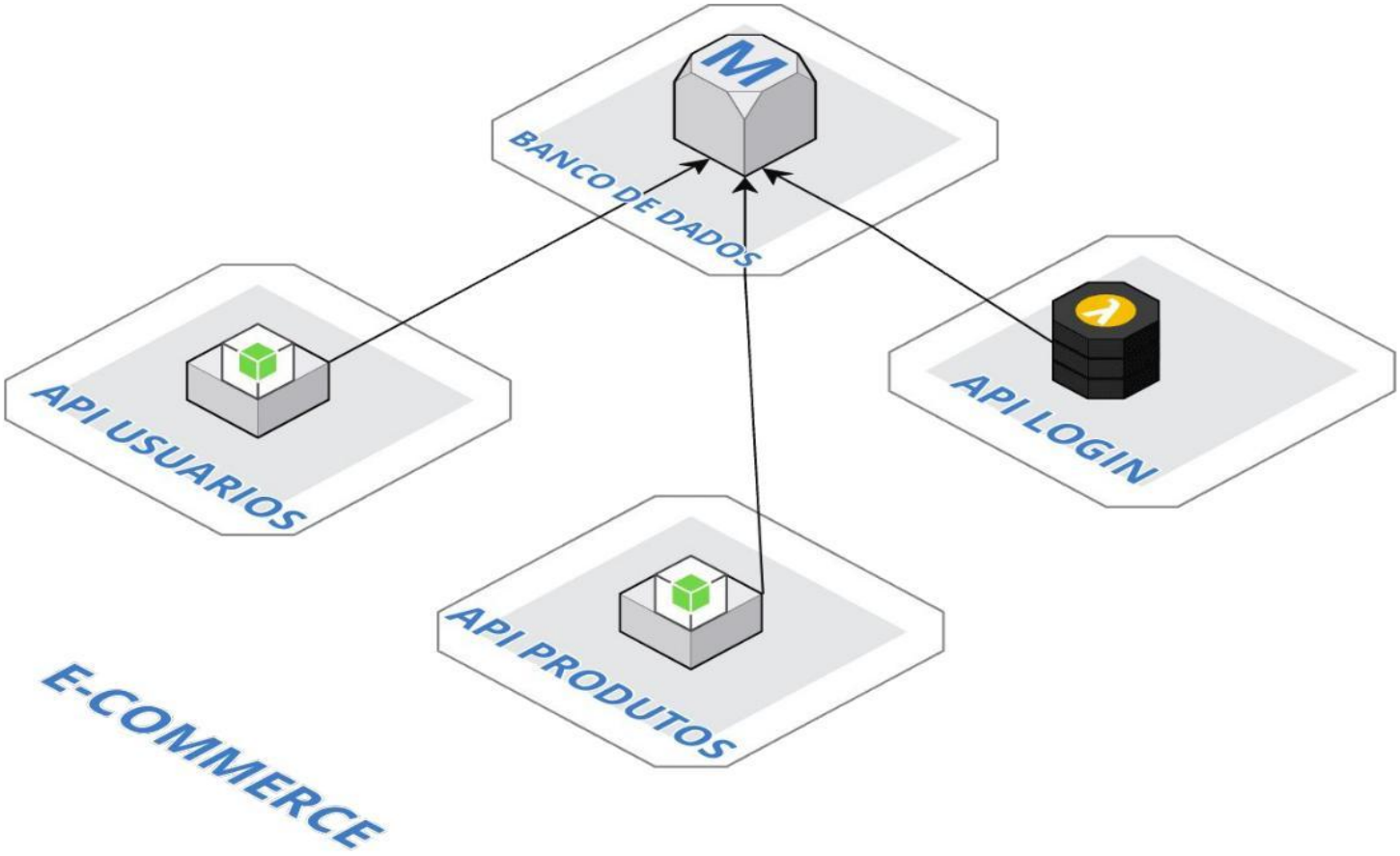
Abstração por Camadas:

Criar uma camada de abstração que isole as aplicações das particularidades dos bancos de dados. Isso pode ser feito com o uso de ORMs (Object-Relational Mappers), que ajudam a compatibilizar diferentes modelos de dados.

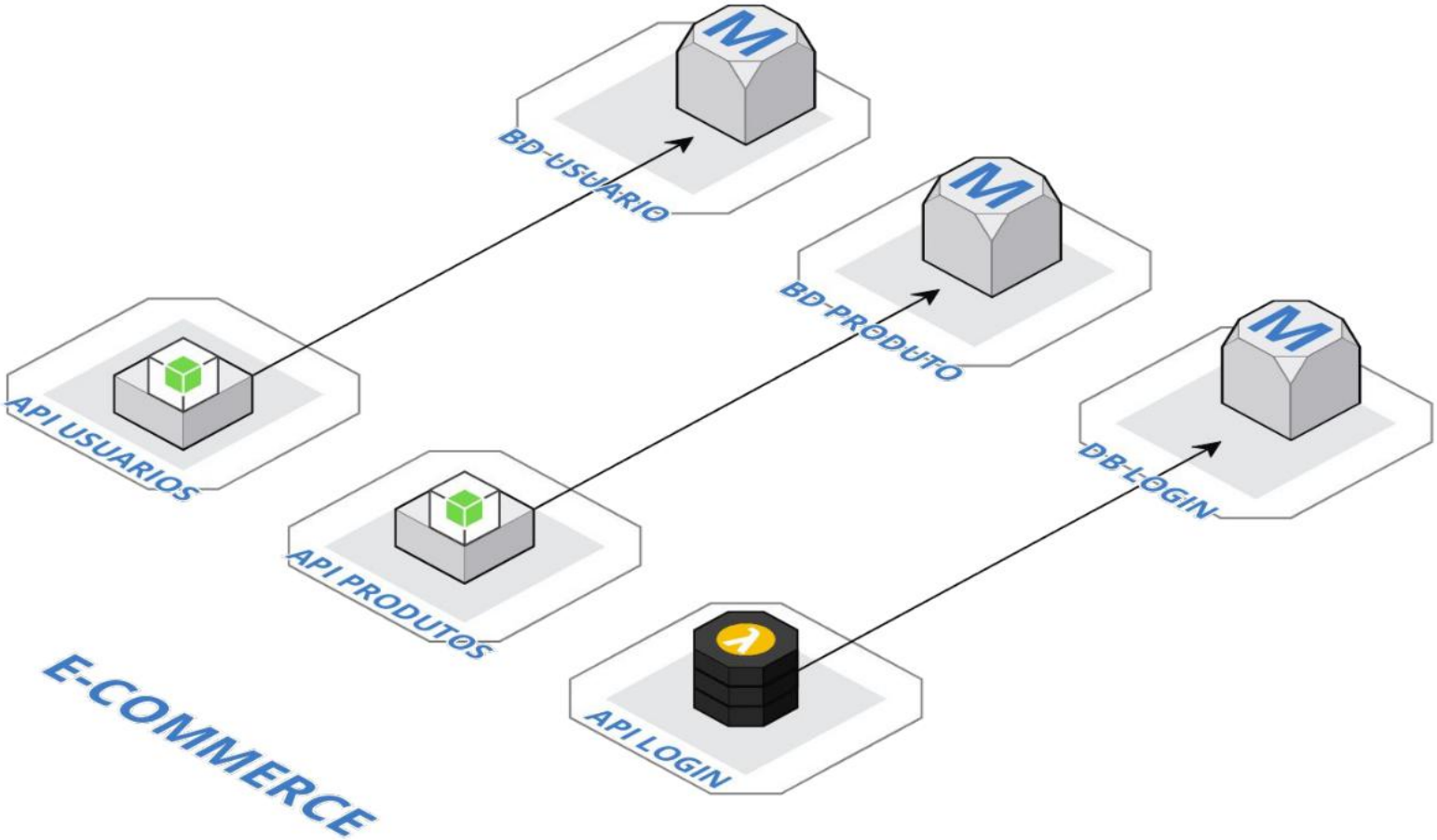
Documentação e Padrões:

Manter uma documentação clara e seguir padrões de desenvolvimento, garantindo que todos os desenvolvedores estejam cientes das limitações e das peculiaridades dos sistemas que estão integrando.

Banco de Dados de Integração


BANCO DE DADOS
API USUARIOS
API LOGIN
API PRODUTOS
E-COMMERCE

Banco de Dados de Aplicaçao


BD-USUARIO
BD-PRODUTO
DB-LOGIN
API USUARIOS
API PRODUTOS
API LOGIN
E-COMMERCE

Modelo de Dados Agregados

Os Modelos de Dados Agregados em bancos de dados referem-se a uma técnica de modelagem que agrupa ou combina dados relacionados em uma única estrutura ou entidade para simplificar o acesso e melhorar a eficiência na recuperação de informações.

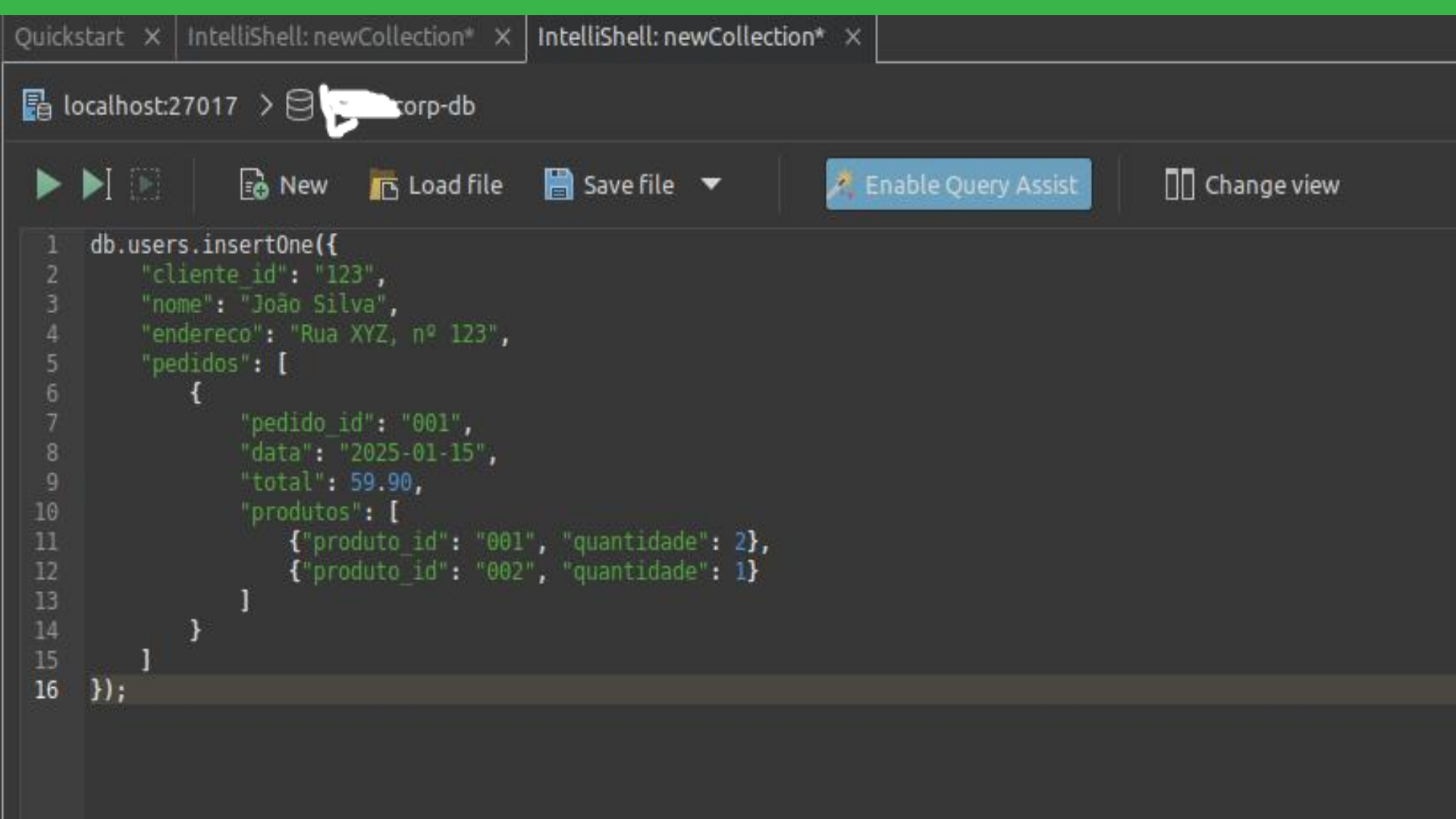
Este conceito é comum em sistemas de bancos de dados NoSQL, onde os dados são frequentemente modelados de maneira a facilitar o acesso rápido e a consulta de informações relacionadas, sem a necessidade de complexas operações de junção (JOINs) típicas de bancos de dados relacionais.

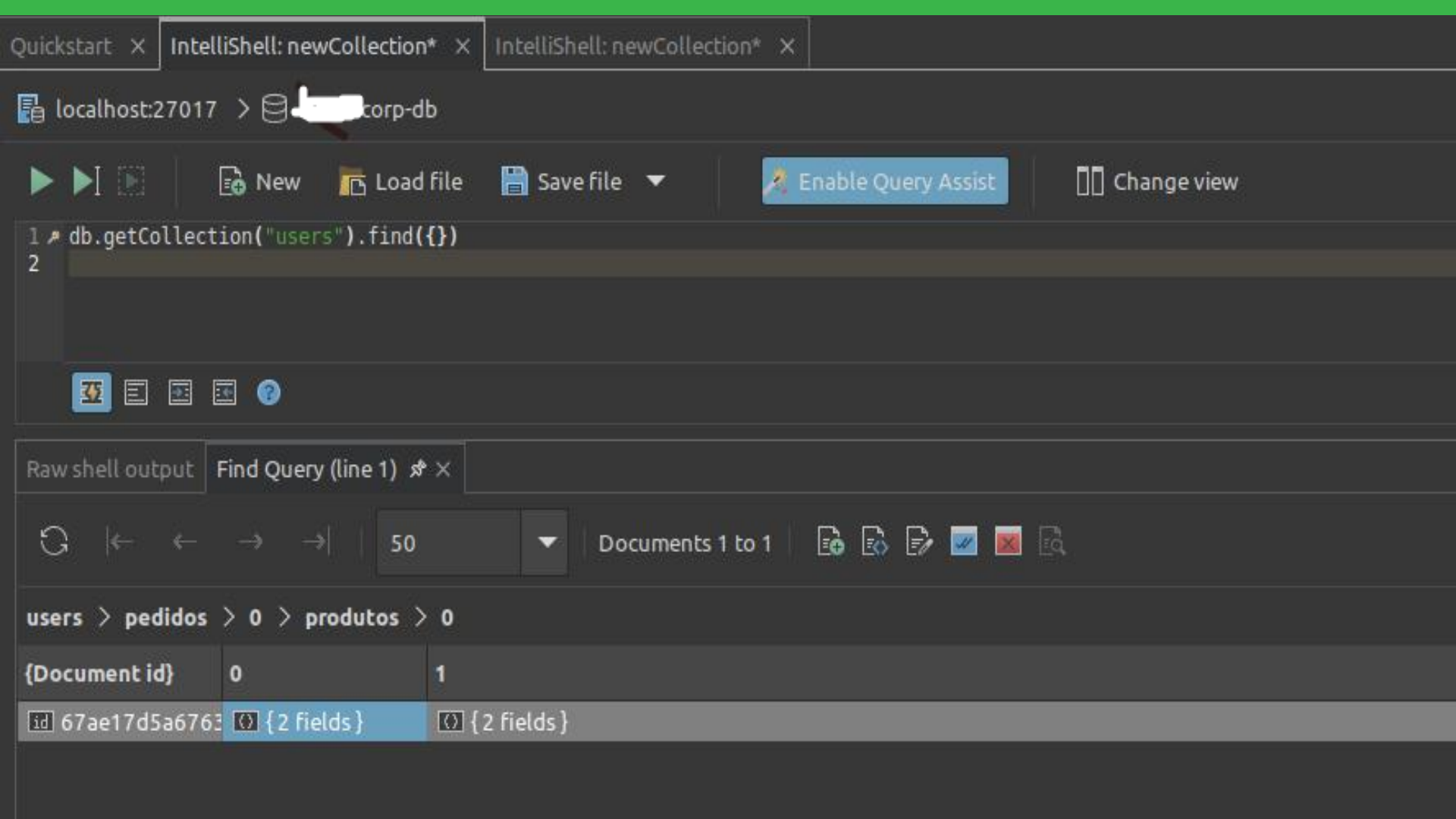
Surgimento do NoSQL

Não Utilizam modelo Relacional

Tem uma boa execuçao em cluster
Seu cdigo aberto(OpenSource)
So criados para o sculo XXI
No tem Esquema

Banco de Dados Sem Esquema


Quickstart
IntelliShell: newCollection*
IntelliShell: newCollection*
localhost:27017
corp-db
New
Load file
Save file
Enable Query Assist
Change view
db.users.insertOne({
"cliente_id": "123",
"nome": "João Silva",
"endereco": "Rua XYZ, nº 123",
"pedidos": [
{
"pedido_id": "001",
"data": "2025-01-15",
"total": 59.90,
"produtos": [
{"produto_id": "001", "quantidade": 2},
{"produto_id": "002", "quantidade": 1}
]
}
]
});


Quickstart
IntelliShell: newCollection*
IntelliShell: newCollection*
localhost:27017
corp-db
New
Load file
Save file
Enable Query Assist
Change view
1 db.getCollection("users").find({})
2
Raw shell output
Find Query (line 1)
50
Documents 1 to 1
users > pedidos > 0 > produtos > 0
{Document id}
0
1
67ae17d5a676
{2 fields}
{2 fields}

Banco Orientado Chave-Valor

Utilizado quando o acesso aos dados é feito por meio da chave primária

Aceita qualquer dado no seu valor para a chave

Desvantagens

Armazenardiferentes tipos de
agregados com chances
de ocorrer conflitos

Cenário

Vamos armazenar informações sobre um usuário, como o nome e a idade, usando pares de chave-valor.
Suponha que queremos armazenar a seguinte informação para um usuário com ID user:1000.

Chave: user:1000:name
Valor: "João Silva"

Chave: user:1000:age
Valor: "30"

Utilize

Armazenamento Informação de Sessão

Perfis de Usuário

Dados de Carinhos de Compras

Não Utilize

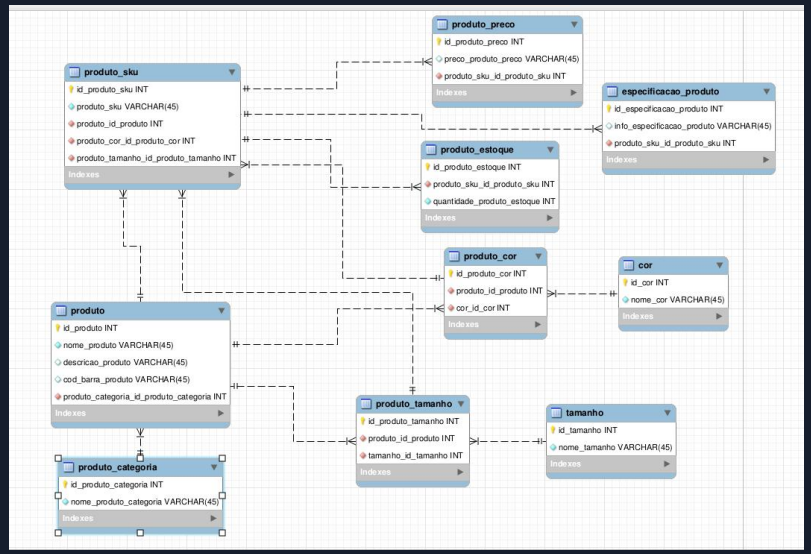
Relacionamento entre dados

Transações com multiplas Operações

Consulta de dados

Banco Orientado a Documentos

Exemplo de Documento



VS

```
{
  "cliente_id": "123",
  "nome": "João Silva",
  "endereco": "Rua XYZ, n° 123",
  "pedidos": [
    {
      "pedido_id": "001",
      "data": "2025-01-15",
      "total": 59.90,
      "produtos": [
        {"produto_id": "001", "quantidade": 2},
        {"produto_id": "002", "quantidade": 1}
      ]
    }
  ]
}
```

Relacional ⟶ Não Relacional

Utilize

Registro de Logs
CMS
Ánalise web em tempo Real
Aplicativos de comércio eletrônico

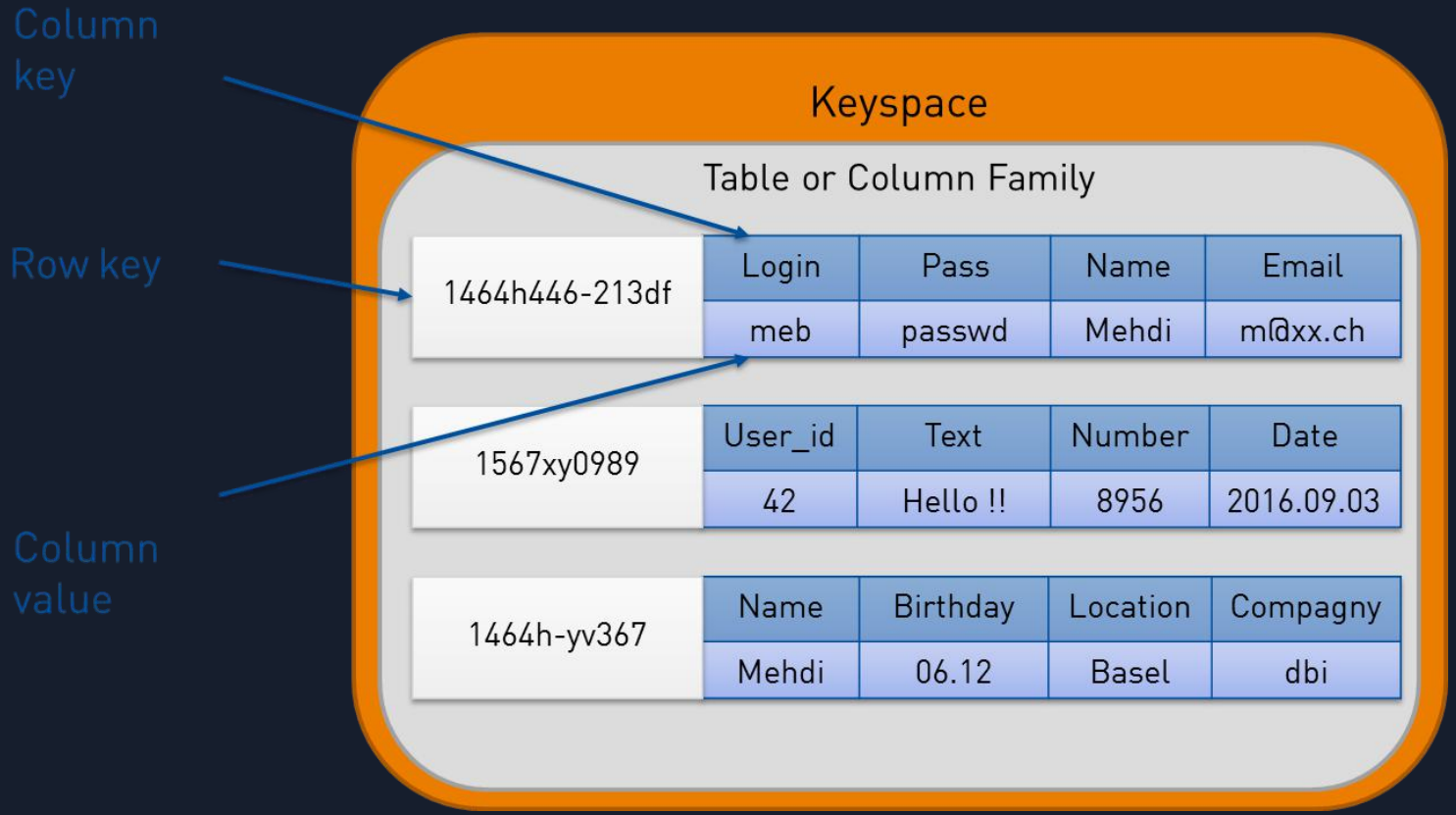
Não Utilize

Transaçoes complexas

Consulta com muitos agregados

Banco Orientado a Colunas

Exemplo de Colunas



Utilize

Registro de Logs
CMS
Ánalise web em tempo Real
Aplicativos de comércio eletrônico

Não Utilize

Transaçoes complexas

Consulta com muitos agregados

Qual a diferença de um database NoSQL **Coluna** por **Documento**

Bancos de Dados de Coluna:

Armazenam dados em formato de **colunas** em vez de **linhas**. Isso significa que, em vez de armazenar todos os dados de uma única **entrada** (ou linha) juntos, eles agrupam dados semelhantes por coluna.

Cada **coluna** pode ter um tipo de dado diferente e, muitas vezes, as colunas de uma única linha podem ser organizadas de forma independente.

Exemplos: Apache Cassandra, HBase. DynamoBD,

VS

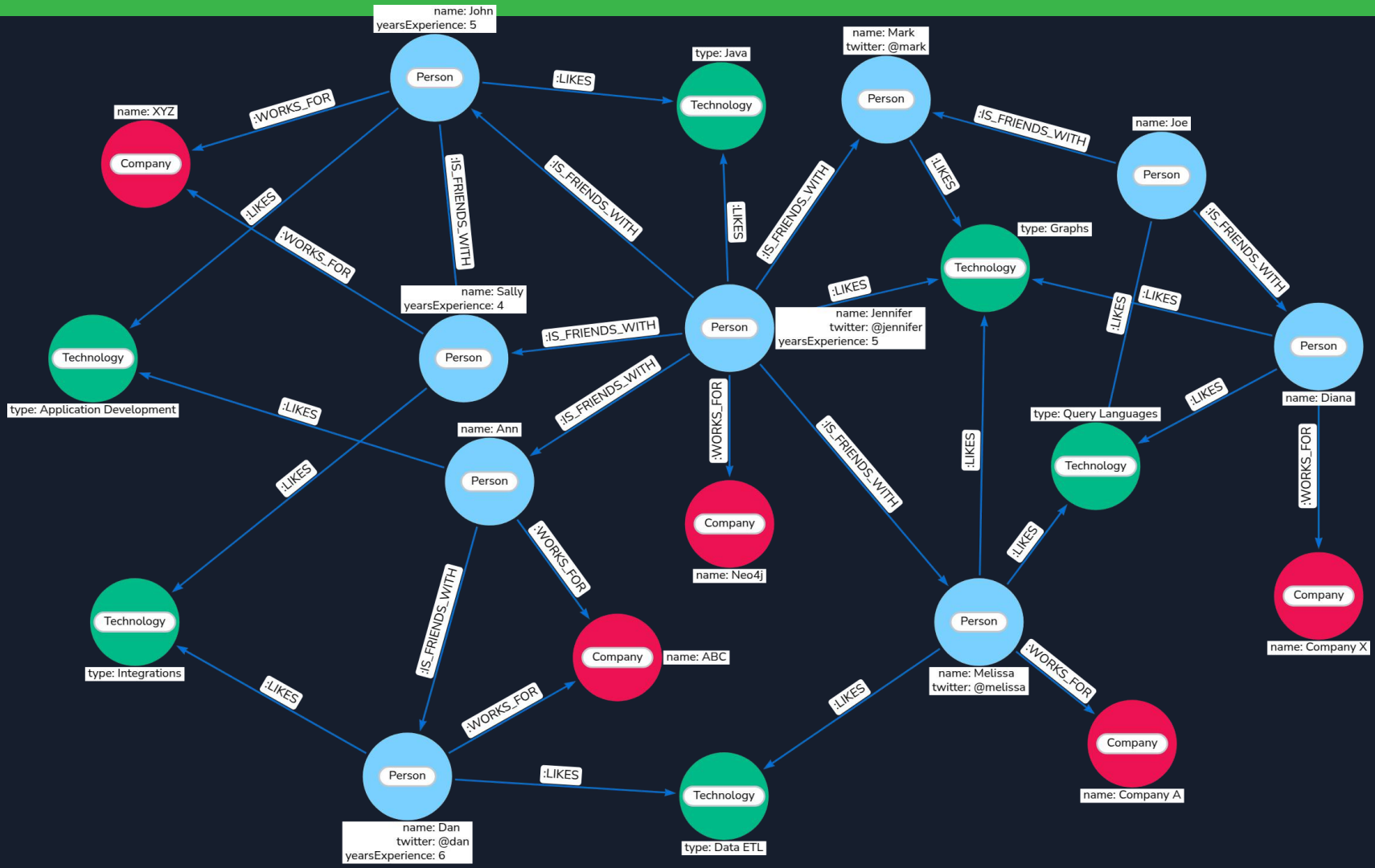
Bancos de Dados de Documento:

Armazenam dados em documentos, geralmente no formato JSON, BSON ou XML. Cada documento pode ter uma estrutura diferente e ser auto-descritivo.

Os documentos agrupam dados que pertencem a um único objeto, possibilitando que dados complexos e aninhados sejam facilmente representados.

Exemplos: MongoDB, CouchDB.

Banco Orientado a Grafo


name: John
yearsExperience: 5
Person
:LIKES
type: Java
Technology
name: Mark
twitter: @mark
Person
:WORKS_FOR
name: XYZ
Company
:IS_FRIENDS_WITH
:IS_FRIENDS_WITH
:LIKES
:IS_FRIENDS_WITH
name: Joe
Person
:LIKES
:LIKES
:WORKS_FOR
type: Graphs
Technology
:IS_FRIENDS_WITH
name: Sally
yearsExperience: 4
:LIKES
name: Jennifer
twitter: @jennifer
yearsExperience: 5
Technology
type: Application Development
Person
:IS_FRIENDS_WITH
Person
:LIKES
:LIKES
type: Query Languages
Technology
:IS_FRIENDS_WITH
:IS_FRIENDS_WITH
:WORKS_FOR
name: Diana
:LIKES
name: Ann
Person
:WORKS_FOR
:LIKES
:IS_FRIENDS_WITH
Company
name: Neo4j
:LIKES
:WORKS_FOR
Company
name: Company X
Technology
type: Integrations
:WORKS_FOR
Company
name: ABC
Person
name: Melissa
twitter: @melissa
:LIKES
:LIKES
:WORKS_FOR
:LIKES
Company
name: Company A
Person
name: Dan
twitter: @dan
yearsExperience: 6
:WORKS_FOR
:LIKES
Technology
type: Data ETL

Características do Neo4 j:

I. Modelo de Grafo
II. Vertices (Nós)
III. Relacionamentos
IV. Propriedades
V. Consulta com Cypher
VI. Desempenho em Consultas de Relações
VII. Escalabilidade
VIII. Transações ACID
IX. Suporte a Grafos de Vários Tipos
X. Visualização de Grafos
XI. Ecossistema e Integrações
XII. Análise de Grafo

Utilize

Redes Sociais
Recomendações

Não Utilize

Atualizar todas as entidades
ou um subconjunntos delas

Outras Possibilidades NOSQL

Bancos de Dados de Objetos

db4o

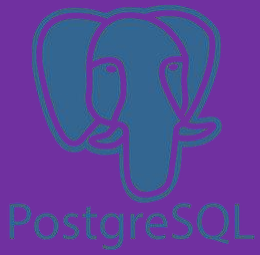
ObjectDB

Bancos de Dados de Séries Temporais

Banco de Dados NoSQL em Nuvem

Banco de Dados Distribuido

Banco Relacional



Sistemas Gerenciadores de Banco de Dados (SGBD)

Definição de SGBD

Sistemas que permitem a criação, manipulação e administração de bancos de dados.

Facilitam a gestão de dados e asseguram a integridade, segurança e eficiência na recuperação de informações.

Importância dos SGBDs

I. Organizar e gerenciar grandes volumes de dados.
II. Facilitar acesso simultâneo por múltiplos usuários.
III. Garantir a segurança dos dados.
IV. Suportar transações confiáveis e integridade referencial.
V. Prover funcionalidades de backup e recuperação.

Componentes de um SGBD

- **Modelo de Dados**

Estrutura que define como os dados são armazenados e organizados.

- **Linguagem de Consulta**

Linguagem utilizada para interagir com o banco de dados (ex.: SQL).

- **Sistema de Gerenciamento de Transações**

Controla a correta execução de operações no banco de dados.

- **Interface com o Usuário**

Ferramentas e interfaces para interação fácil e intuitiva com o SGBD.

Vantagens dos SGBDs

- **Consistência:**
  - Asseguram que os dados estejam corretos e disponíveis.

- **Integridade:**
  - Garantem a integridade de dados através de chaves primárias e estrangeiras.

- **Segurança:**
  - Controle de acesso e autenticação.

- **Escalabilidade:**
  - Capacidade de lidar com crescimentos de dados e usuários.

Casos de Uso

Empresas: Armazenamento de dados de clientes e transações.

Aplicações Web: Gerenciamento de dados de usuários e conteúdo.

IoT: Coleta e análise de dados de dispositivos conectados.

Análise de Dados: Processamento de grandes volumes de dados para insights.

Tendências Atuais

Banco de Dados em Nuvem: Crescimento do uso de soluções em nuvem para armazenamento e escalabilidade.

Big Data: Integração de SGBDs com ferramentas de análise e processamento de Big Data (ex.: Hadoop, Spark).

Machine Learning e AI: Integração de SGBDs com algoritmos de aprendizado de máquina para análise avançada de dados.

Banco de dados serverless: (ou sem servidor) é um modelo de banco de dados que permite que os desenvolvedores utilizem serviços de banco de dados sem a necessidade de gerenciar a infraestrutura subjacente,

Outras Tendências Atuais

1. Conectividade com a Nuvem
2. Adoção de Dados em Tempo Real
3. Suporte a Big Data e Dados Não Estruturados
4. Integração com A.I. e Machine Learning
5. Automação e Gerenciamento de Bancos de Dados
6. Segurança Aprimorada
7. Modelagem de Dados Avançada
8. SQL Polivalente e Suporte a Vários Modelos de Dados
9. Desempenho e Otimização de Consultas
10. Criação e Manutenção em DataLakes

SQL StructuredQuery Language

O que é SQL?

Donald D. Chamberlin

Raymond F. Boyce

SQL é uma linguagem padrão utilizada para gerenciar e manipular bancos de dados relacionais.

Permite a realização de operações como:
consulta, inserção, atualização e exclusão de dados.

Desenvolvido nos anos 1970 por **Donald D. Chamberlin e Raymond F. Boyce** na IBM.
Tornou-se um padrão ANSI em 1986 e ISO em 1987.

Importância do SQL

Universalidade:
É a linguagem padrão para a maioria dos bancos de dados relacionais (MySQL, PostgreSQL, Oracle, SQL Server, etc.).

Facilidade de uso:
Possui uma sintaxe intuitiva e fácil de entender.

Poderosa ferramenta de consulta:
Permite realizar consultas complexas e análises de dados de forma eficiente.

Componentes do SQL

DDL (Data Definition Language): Definição de estruturas de dados.
Comandos: CREATE, ALTER, DROP.

DML (Data Manipulation Language): Manipulação de dados.
Comandos: SELECT, INSERT, UPDATE, DELETE.

DCL (Data Control Language): Controle de acesso aos dados.
Comandos: GRANT, REVOKE.

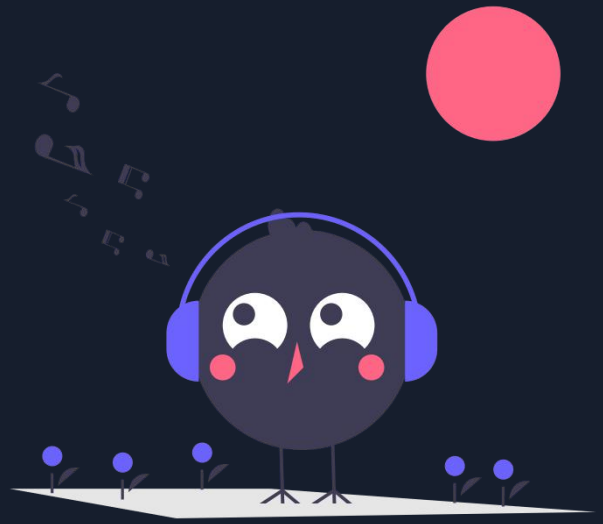
TCL (Transaction Control Language): Controle de transações.
Comandos: COMMIT, ROLLBACK, SAVEPOINT.

Criando um Database

```
create database [databasename];
```

Comando:

```
create database empresa;
```



Criando uma tabela SQL

```
create table tb_client(
        client_id serial, -- Identificador
        codigo varchar(10), -- codigo referencia
        nome varchar(150),
        razao varchar(150),
        data date,
        cnpj varchar(150),
        fone varchar(150),
        cidade varchar(80),
        estado varchar(80),
        primary key (client_id));
```

Inserindo dados em uma tabela SQL

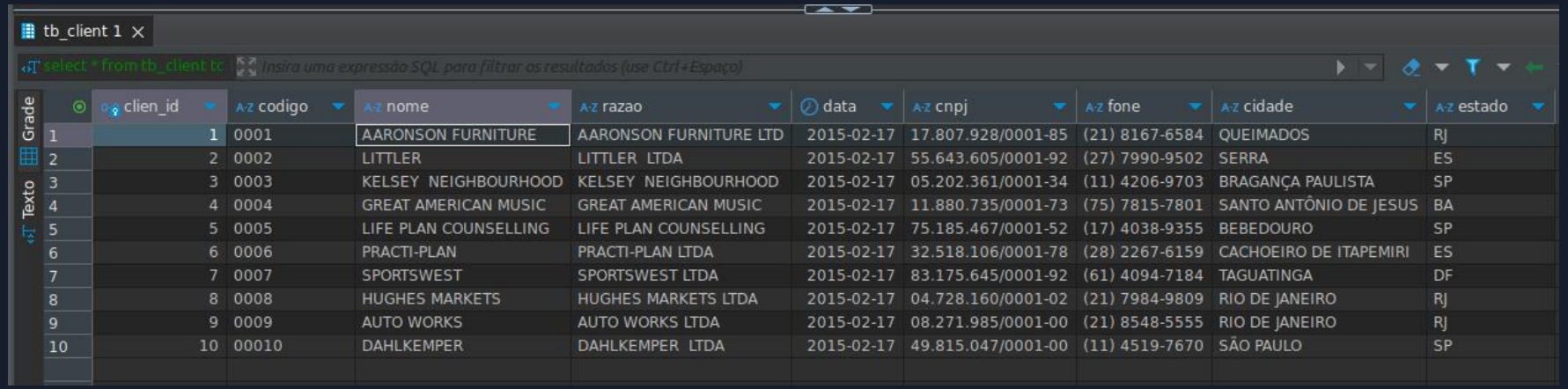
```
insert into tb_client(codigo, nome, razao, "data", cnpj, fone, cidade, estado)
values('0001','AARONSON FURNITURE','AARONSON FURNITURE LTD','2015-02-17 23:14:50',
       '17.807.928/0001-85', '(21) 8167-6584' ,'QUEIMADOS','RJ' );
```

```
insert into tb_client(codigo, nome, razao, "data", cnpj, fone, cidade, estado)
values('0002','LITTLER ,'LITTLER  LTDA','2015-02-17 23:14:50','55.643.605/0001-92',
       '(27) 7990-9502' ,'SERRA','ES' );
```

```
insert into tb_client values('0001','AARONSON FURNITURE' ,'AARONSON FURNITURE LTD',
'2015-02-17 23:14:50',     '17.807.928/0001-85', '(21) 8167-6584' ,'QUEIMADOS'          ,'RJ' );
```

Selecionando dados de uma tabela(SQL)

```
select * from tb_client tc;
```



tb_client 1

select * from tb_client tc — Insira uma expressão SQL para filtrar os resultados (use Ctrl+Espaço)

Grade | Texto

	clien_id	codigo	nome	razao	data	cnpj	fone	cidade	estado
1	1	0001	AARONSON FURNITURE	AARONSON FURNITURE LTD	2015-02-17	17.807.928/0001-85	(21) 8167-6584	QUEIMADOS	RJ
2	2	0002	LITTLER	LITTLER LTDA	2015-02-17	55.643.605/0001-92	(27) 7990-9502	SERRA	ES
3	3	0003	KELSEY NEIGHBOURHOOD	KELSEY NEIGHBOURHOOD	2015-02-17	05.202.361/0001-34	(11) 4206-9703	BRAGANÇA PAULISTA	SP
4	4	0004	GREAT AMERICAN MUSIC	GREAT AMERICAN MUSIC	2015-02-17	11.880.735/0001-73	(75) 7815-7801	SANTO ANTÔNIO DE JESUS	BA
5	5	0005	LIFE PLAN COUNSELLING	LIFE PLAN COUNSELLING	2015-02-17	75.185.467/0001-52	(17) 4038-9355	BEBEDOURO	SP
6	6	0006	PRACTI-PLAN	PRACTI-PLAN LTDA	2015-02-17	32.518.106/0001-78	(28) 2267-6159	CACHOEIRO DE ITAPEMIRI	ES
7	7	0007	SPORTSWEST	SPORTSWEST LTDA	2015-02-17	83.175.645/0001-92	(61) 4094-7184	TAGUATINGA	DF
8	8	0008	HUGHES MARKETS	HUGHES MARKETS LTDA	2015-02-17	04.728.160/0001-02	(21) 7984-9809	RIO DE JANEIRO	RJ
9	9	0009	AUTO WORKS	AUTO WORKS LTDA	2015-02-17	08.271.985/0001-00	(21) 8548-5555	RIO DE JANEIRO	RJ
10	10	00010	DAHLKEMPER	DAHLKEMPER LTDA	2015-02-17	49.815.047/0001-00	(11) 4519-7670	SÃO PAULO	SP

"A **única limitação** é aquela que você impõe a si mesmo.

Vá em frente!"