

Banco de Dados II



aula 01

Prof. Felipe Scheidt

2023

Conteúdos

1. Paradigmas de armazenamento de dados
2. SQL
3. NoSQL
 - a. MongoDB
 - b. Projeto com Python - CRUD
 - c. Visualização dos dados

Banco de dados

Banco de dados é uma coleção de dados armazenados e organizados de acordo com uma estrutura,

Essa estrutura pode seguir um esquema rígido ou flexível.

BD é gerenciado pelo **SGBD** (*sistema de gerenciamento de banco de dados*)

Principais funcionalidades são armazenamento e recuperação de dados.

Funções do BD podem ser acessadas através de uma API

- CLI - linha de comando (command line interface)
- Driver - biblioteca programada em uma linguagem de programação
- GUI - Interface Web ou Desktop (ex.: Workbench, phpmyadmin,...)

Persistência e Recuperação

- Praticamente toda aplicação necessita armazenar dados (persistência)
- Informação armazenada pode ser futuramente *consultada* (recuperação)
- Persistir um dado é essencialmente torná-lo **não-volátil**
- Essas são operações fundamentais que todo banco de dados [**SGBD** ou DBMS] implementa.
- Recuperação (*select*) e persistência (*insert*) são operações triviais usando SQL, porém a implementação no SGBD envolve certa complexidade.

Exemplos de funcionalidades

Recursos comuns oferecidos pelo SGBD:

- Armazenamento dos dados com recuperação e atualização.
- Suporte a permissões de usuários
- Acesso remoto
- Regras de validação.
- Interface/API para manipulação dos dados

Recursos +avançados:

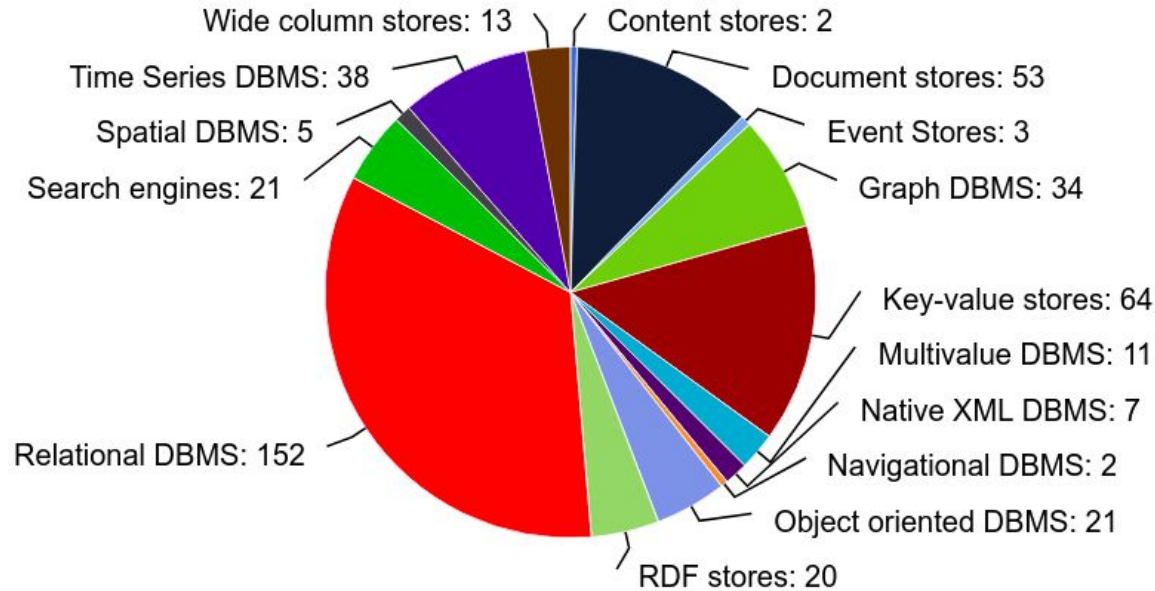
- Suporte a transações e concorrência
- Base de dados distribuída
- Suporte a esquemas de dados flexíveis e rígidos

Paradigmas de armazenamento de dados

Os dados podem ser armazenados em diferentes estruturas também chamados de paradigmas. Exemplos:

1. Documento (no-sql)
2. Relacional (sql)
3. Memória (in-memory - key-value pair)
4. Coluna (wide column)
5. Grafos
6. Séries temporais
7. Modelos múltiplos

BD por categorias



RELACIONAL (SQL)

Paradigma Relacional (RDBMS)

Paradigma relacional

- Dados são armazenados em diversas tabelas
- Tabelas possuem tipos de relacionamento (1-n, 1-1, n-n,...)
- Tabelas armazenam objetos (varchar,int,...) e índices (chave PK e FK)
- Existem regras e estrutura de dados que chama-se de esquema (*schema*).
- O esquema é garantido pelo DBMS*

Estrutura tabular

Conceitos principais:

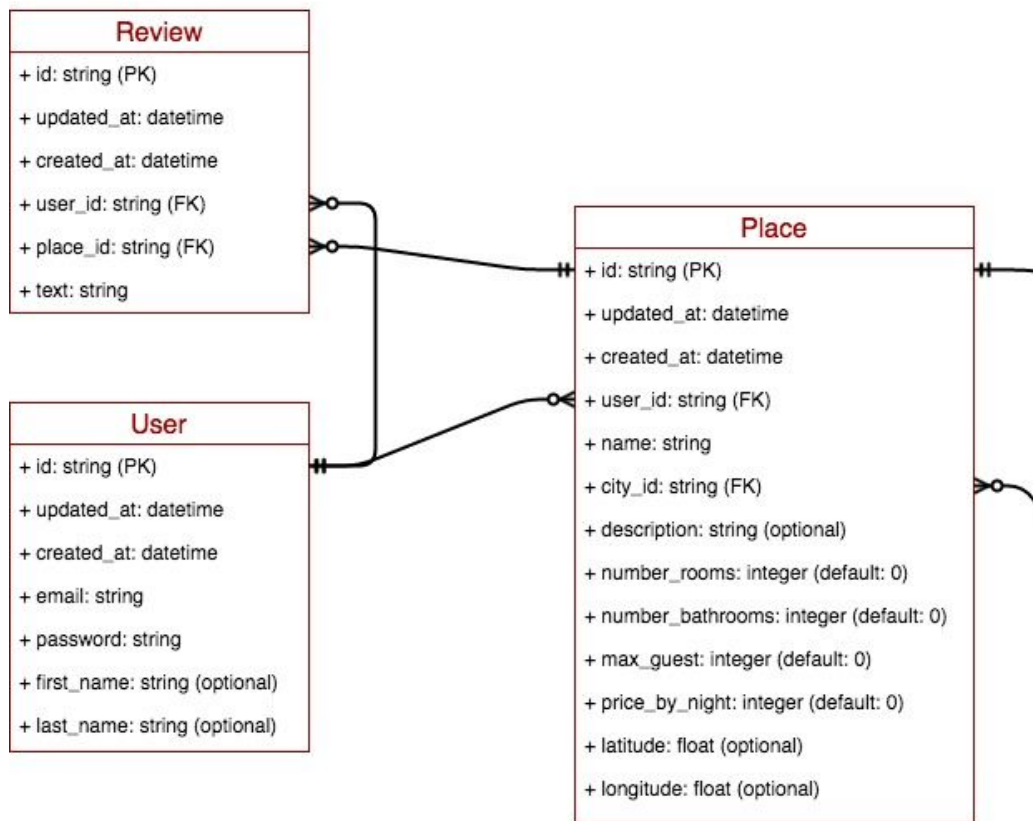
- Tabela
- Coluna
- Linha

Database Table

The diagram shows a table with three columns: ID, firstName, and email. The rows are numbered 1 to 4. A red arrow labeled 'Column' points to the first column (ID). A red arrow labeled 'Row' points to the second row (John). A red box highlights the second row (John) and the second column (firstName).

ID	firstName	email
1	John	john@email.com
2	Paul	paul@email.com
3	Peter	peter@email.com
4	James	james@email.com

Esquema relacional



Integridade dos dados (RDBMS)

BD Relacional “garante” integridade referencial:

- Consistência dos dados (validação valores nulos, tipo de dados)
- Integridade referencial (FK e PK são válidas)
- Garantir o esquema (regras ou constraints definidas na criação da tabela)
- Prevenir que o BD entre em estado inconsistente

RDBMS

Vantagens:

- Consistência dos dados
- Organização (estrutura fixa)
- Linguagem SQL para manipulação dos dados
- Garantias [ACID](#)

Desvantagem:

- Problema com escalabilidade (difícil escalar horizontalmente)
- Consome mais recursos - single server based
- Complexidade de atualizar o esquema (sistemas em desenvolvimento)
- Consulta a múltiplas relações gera queries complexas e mais processamento para fazer join entre tabelas.

ACID

Atomicidade:

Deve finalizar completamente uma tarefa ou caso contrário não ter nenhum efeito.

Consistência:

Deve respeitar as regras de esquema de dados definidas no BD

Isolamento:

Deve executar uma tarefa sem que esta interfira em outras tarefas que estão executando em paralelo

Durabilidade:

Deve escrever/salvar os dados permanentemente

NO-SQL

Paradigma NoSQL

Non-relational database, **Non structured data**, Not only sql

Dados não precisam seguir o mesmo esquema!

De fato nem precisam de um esquema (*schema-less* == sem restrições do DBMS)

“Não há relacionamentos”

A ideia é agrupar todas as informações necessárias em um só lugar.

Ganho de performance em operações de leitura pois não é preciso buscar em outras tabelas.

Desvantagem é que há um certo nível de dados duplicados.

Operações de update devem levar isso em consideração.

NoSQL

Flexibilidade para realizar atualizações no esquema do database sem quebrar o código.

Simplicidade do design

Robustez para aplicações big data e real-time.

Escalabilidade (horizontal e vertical)

Entretanto não há garantia que determinado campo exista ou o tipo de dados.

Exige um tratamento extra na programação.

Overhead em situação de escrita em várias collections.

NoSQL

Vantagens:

- Flexibilidade
- Escalabilidade
- “Velocidade”

Desvantagens:

- Documentos de uma mesma collection podem ter campos diferentes, o que pode gerar resultados inesperados (tratar no código)
- Duplicação de dados

Orientado a Documento (NoSQL)

Paradigma orientado a documento.

Os dados são armazenados no formato JSON (*javascript object notation*)

Analogia com o esquema Relacional: cada registro de uma tabela seria equivalente a um documento JSON.

Entretanto cada documento pode ter uma estrutura diferente, campos e atributos que podem existir em alguns casos e em outros não.

Exemplo:

Documento: Usuario

Campo: comentarios



















Paradigma NoSQL

Exemplo de uma coleção de usuários.

Nessa coleção há dois documentos ou usuários.

```
{
  "id": 1,
  "nome": "Marcos Soares",
  "email": "mars@email.com"
},
{
  "id": 2,
  "nome": "Andrea Silva",
  "login": "ad1000",
  "email": "andrea77@email.com",
  "posts": [
    {
      "titulo": "Primeiro Post",
      "conteudo": "Conteudo do primeiro post"
    },
    {
      "titulo": "Segundo Post",
      "conteudo": "Conteudo do segundo post"
    }
  ]
}
```

Exemplos de Banco de dados No-SQL

Rank			DBMS	Database Model	Score
Oct 2021	Sep 2021	Oct 2020			
1.	1.	1.	MongoDB 	Document, Multi-model 	493.55
2.	2.	2.	Amazon DynamoDB 	Multi-model 	76.55
3.	3.	3.	Microsoft Azure Cosmos DB 	Multi-model 	40.29
4.	4.	4.	Couchbase 	Document, Multi-model 	27.91
5.	5.	 6.	Firebase Realtime Database	Document	19.02
6.	6.	 5.	CouchDB	Document, Multi-model 	15.79
7.	7.	7.	MarkLogic 	Multi-model 	9.43
8.	8.	8.	Realm 	Document	9.29
9.	9.	9.	Google Cloud Firestore	Document	8.37
10.	 11.	 20.	Virtuoso 	Multi-model 	4.69

<https://db-engines.com/en/ranking/document+store>

Melhor solução?

Cada paradigma de armazenamento deve ser analisado levando-se em consideração os **requisitos** da aplicação.

Não existe o melhor paradigma.

E nos casos onde múltiplos banco de dados atendem aos requisitos da aplicação?

Fique com o BD “mais simples”.

Fato é que muitos BD já implementam diferentes paradigmas de armazenamento, que podemos chamar de multi-model databases.

Comparação SQL vs. NoSQL

Não há uma melhor solução. Tudo depende dos requisitos do sistema.

Responder ?	SQL	NoSQL
Esquema de dados dinâmico/em evolução?		✓
Operações de update são frequentes	✓	
Possui muitos dados em constante crescimento		✓
Garantia de consistência dos dados?	✓	

Processo de decisão

Preciso de um mecanismo ACID?

Controle de transações é mandatório?

Suporte a linguagem de programação do sistema em desenvolvimento?

Qual a estrutura/formato dos dados que preciso armazenar?

Velocidade de leitura dos dados é crítico?

Performance de escrita/update?

Atividade: Configurar conta

Criar conta nos seguintes sites:

- DBHub - <https://dbhub.io>
- Colab (requer conta gmail) - <https://colab.research.google.com/>
- MongoDB - <https://www.mongodb.com/cloud/atlas>

Atividade - Teste do Colab

Acessar: <https://colab.research.google.com/>

Criar um Notebook para testar o banco de dados **chinook.db**

Link do arquivo: <https://github.com/fscheidt/bd2-23/tree/master/data>

Atividade - Teste do DBHub

Acessar: <https://dbhub.io>

Criar um novo banco de dados no botão upload -> enviar o arquivo **chinook.db**

Link do arquivo: <https://github.com/fscheidt/bd2-23/tree/master/data>

Exercício 1 – DBHub – sqlite

No site do <https://dbhub.io>, após criar o banco de dados chinook.db

Escreva o SQL que responde cada questão abaixo (anexar também a saída gerada pelo SQL)

1. Nome do Artista com id = 50 (`artists`)
2. Lista de Artistas cujo nome inicia com a letra “F”
3. Quantidade total de clientes residentes no USA (`customers`)
4. Lista do nome dos países em ordem alfabética existentes na tabela clientes uma única vez.
5. Lista de nome dos clientes cujo valor da fatura (`invoices.total`) foi superior a \$20
6. O ranking dos países com maior número de clientes, ordenado decrescente.

Salvar em um arquivo do google docs, exportar para pdf e enviar pelo moodle.

Exercício 2

Faça a comparação entre 3 sistemas de banco de dados: (1) PostgreSQL, (2) Memchaced e (3) MongoDB.

Crie uma tabela (google planilhas ou libreoffice) para comparar esses 3 banco de dados em relação as seguintes características:

1. Possui integridade referencial de chave estrangeira?
2. Faz validação do tipo de dados?
3. Segue qual o paradigma de armazenamento de dados?
4. Suporta a linguagem SQL?

Exportar a planilha para **pdf** e enviar pelo moodle.

Ver material para consulta: <https://db-engines.com/en/system/Redis>