A11.
DDL:
CREATE INDEX
DROP INDEX
E MIGRAÇÃO DE DADOS

PROF. WILLIAM C. AUGUSTONELLI (BILLY)

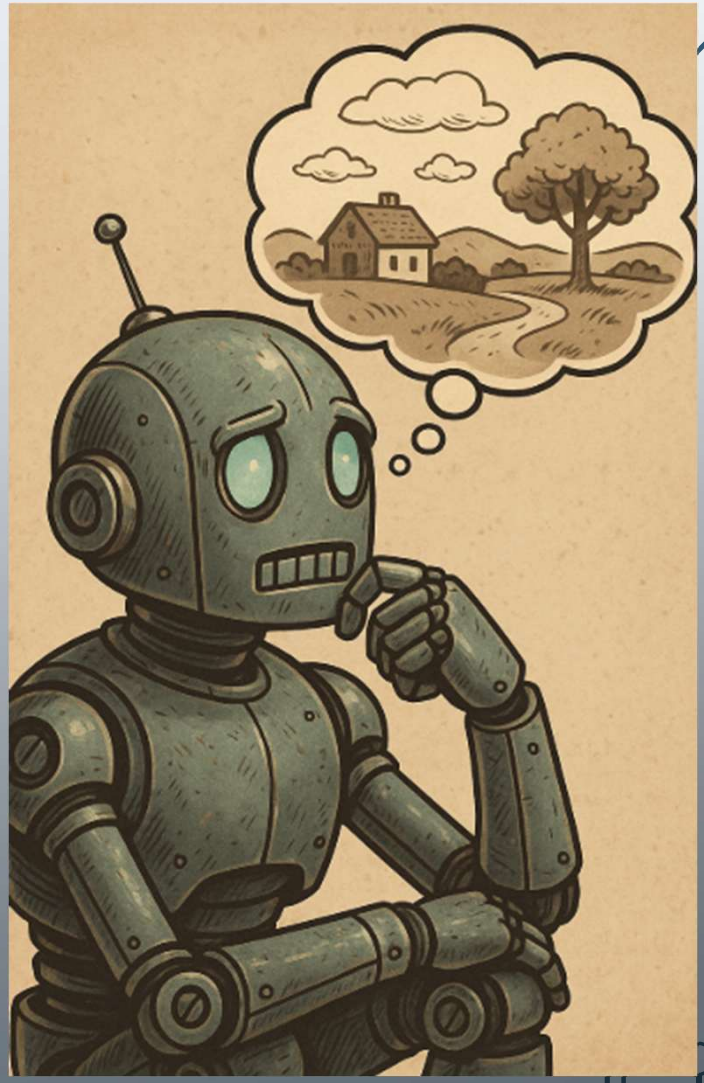
WILLIAM.AUGUSTONELLI@DOCENTE.SENAI.BR – 2S2025

OBJETIVO

- Entender o que são **índices** em SGBDs relacionais
- Demonstrar a criação e remoção de índices
- Mostrar a importância dos índices para o desempenho
- Apresentar **migração de dados** (exportação e importação)

NA ÚLTIMA AULA...

- CREATE TABLE
  - Tipos de dados
  - Restrições
- ALTER TABLE
- DROP TABLE



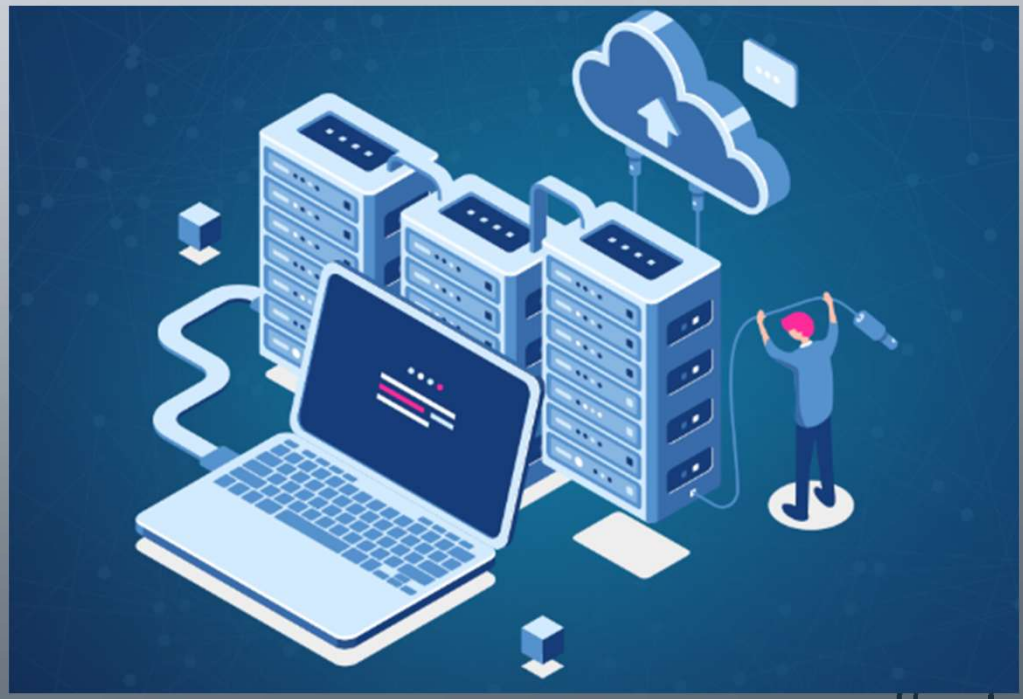
NOSSA AULA DE HOJE...

- Índices em Banco de Dados
  - O que são e para que servem
  - Tipos de índices
- DDL – CREATE INDEX e DROP INDEX
- Migração de Dados
  - Exportação
  - Importação

O QUE É UM ÍNDICE?

Estrutura de dados auxiliar que permite ao SGBD **acessar os registros de forma mais rápida**

- Funciona como um **atalho** para localizar informações em uma tabela
- Internamente, os SGBDs costumam usar estruturas como **árvores B+ (B-Trees)** ou **hashes** para armazenar os índices

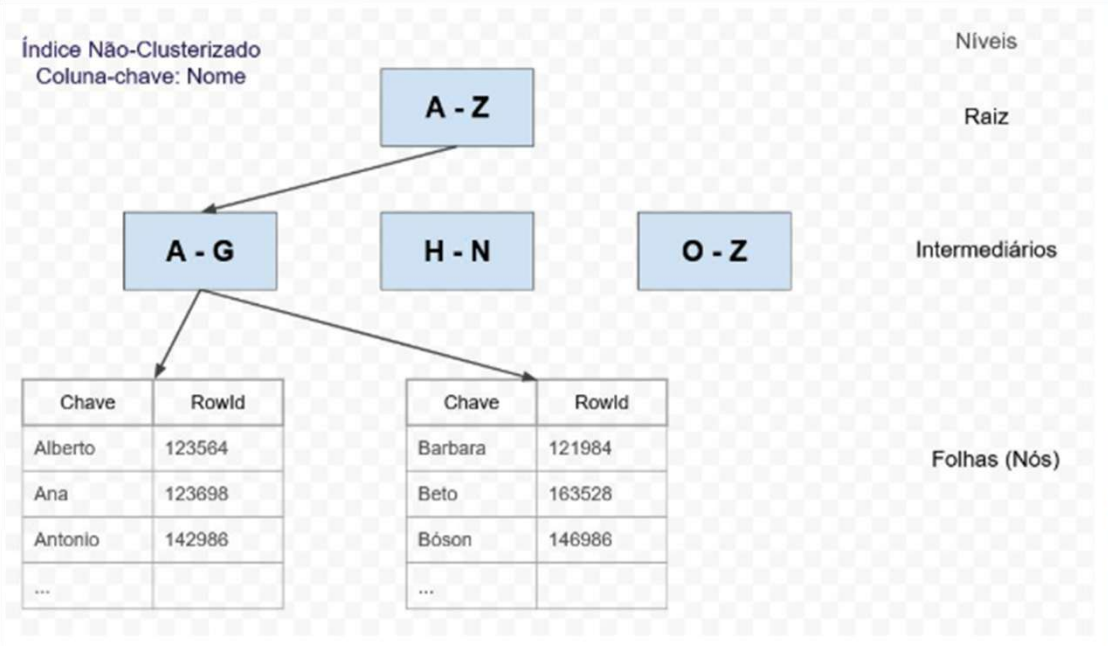


ÍNDICE: ÁRVORES B+ (B-TREES)

- A **estrutura mais usada** em banco de dados relacionais para implementar índices
- É uma **árvore balanceada** – mantém os dados organizados em níveis
- **Características principais**
  - Todas as folhas ficam no mesmo nível
  - Os **nós internos** guardam apenas chaves de navegação
  - Os **nós folhas** guardam ponteiros para os registros reais
  - Permite **busca, inserção** e **exclusão** em tempo logarítmico O(log n)

Vantagens em banco de dados

- Ótima para consultas por intervalos
- Mantém os dados sempre ordenados



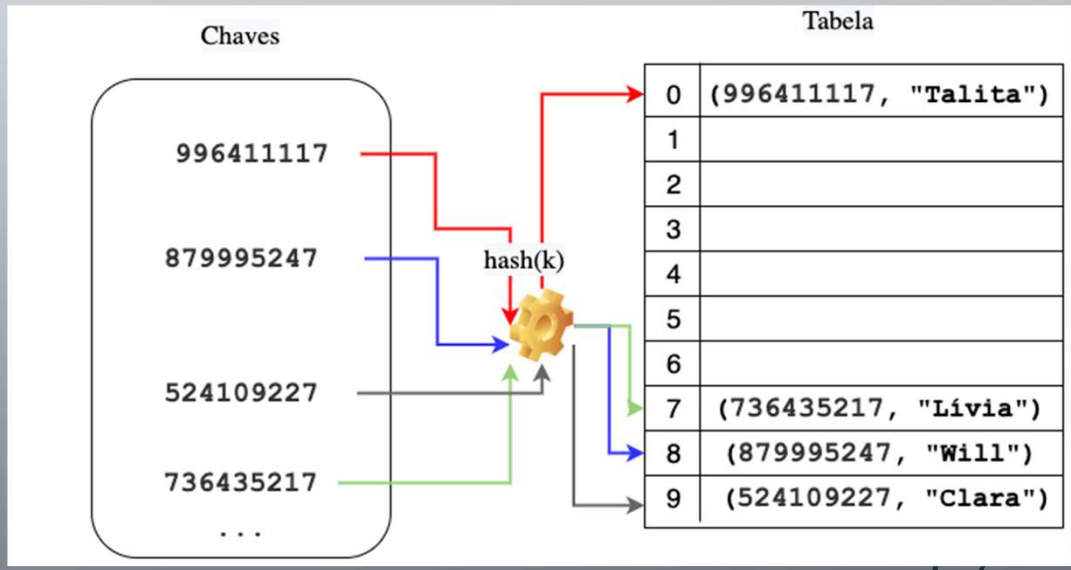
Árvore Balanceada – Índice B-Tree

ÍNDICE: HASHES

- Outra estrutura comum usada para índices, mas com aplicação **diferente** da árvore B+
- Usa **funções hash** para calcular a posição do registro

Funcionamento

1. Pega o valor da chave – aplica uma função matemática (hash)
2. Essa função retorna um **endereço na tabela de hash**
3. Vai direto ao registro (quase um atalho absoluto)



Vantagens em banco de dados

- Muito eficiente para **buscas exatas**

Limitações

- Não funciona bem para **intervalos**
- Pode haver **colisões** (dois valores diferentes caindo na mesma posição

TIPOS DE ÍNDICES

1. **Índice Primário**
   - Criado **automaticamente** quando definimos uma **chave primária (PK)**
   - Garante que cada registro seja único e não nulo
   - Usado para identificar unicamente cada linha
2. **Índice Único (unique)**
   1. Garante que não existem valores duplicados em uma coluna
   2. Diferente da PK, pode ser usado em **múltiplas colunas opcionais**
3. **Índice Composto**
   1. Criado sobre **duas ou mais colunas**
   2. Muito útil em consulta que usam filtros combinados

TIPOS DE ÍNDICES

4. **Índice de Texto (full-text index)**
    1. Usado em colunas **VARCHAR/TEXT** para busca de palavras
    2. Permite consultas mais complexas (`like` , buscas aproximadas)
5. **Índice Clusterizado**
    4. A tabela **é fisicamente organizada** de acordo com a ordem do índice
    5. Cada tabela pode **ter apenas um**
    6. Gerando criado na PK
6. **Não Clusterizado**
    4. O índice aponta para o registro na tabela, mas a tabela não está fisicamente ordenada
    5. Pode haver **vários índices não-clusterizados**

CREATE INDEX

```
CREATE INDEX nome_indice ON tabela(coluna);
```

- Índice em um coluna

```
CREATE INDEX idx_NomeCliente ON Clientes(nome);
```

- Índice composto (duas colunas)

```
CREATE INDEX idx_Cliente_DataVenda ON Vendas(id_Cliente, data_venda);
```

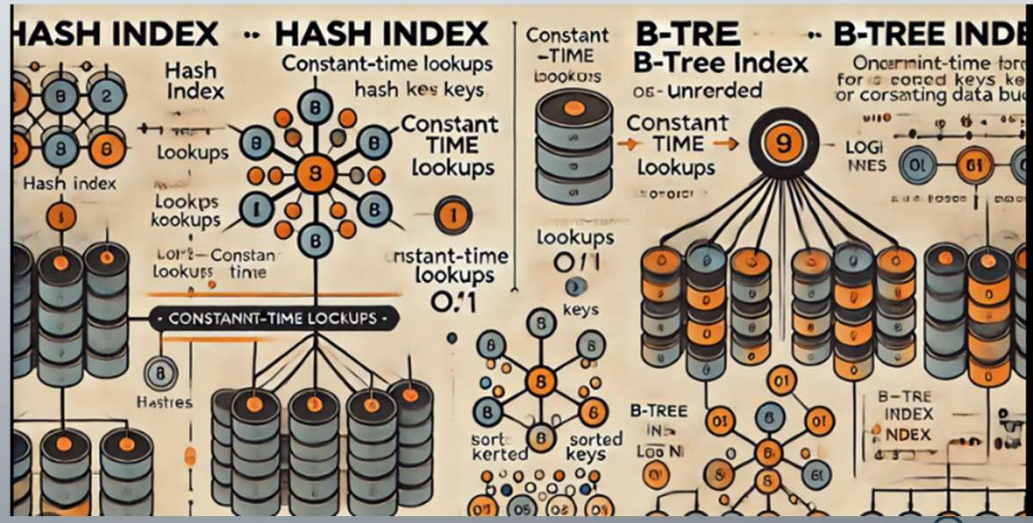
- Índice único

```
CREATE UNIQUE INDEX idx_EmailCliente ON Clientes(email);
```

TIPOS DE CREATE INDEX

Em MySQL/PostgreSQL podemos criar índices

- **BTREE (padrão)** – usado para buscas, ordenações, intervalos
- **HASH (PostgreSQL)** – ótimos para buscas exatas
- **FULLTEXT (MySQL)** – otimizado para colunas de texto grandes



Exemplo

```
CREATE FULLTEXT INDEX idx_livros_titulo ON livros(titulo);
```

QUANDO CRIAR UM ÍNDICE?

- Colunas usadas em **WHERE** com **muita frequência**
- Colunas usadas em **JOIN** entre tabelas
- Colunas em **ORDER BY e GROUP BY**

Evite índices para:

- **Tabelas pequenas** – o otimizador de consultas pode preferir ler a tabela inteira (full table scan)
- **Colunas que mudam muito** - cada modificação precisa atualizar o índice – custo extra
- **Colunas com baixa seletividade** – índice não ajuda, porque quase todas as linhas são lidas de qualquer forma
- **Muitas colunas ao mesmo tempo**

DROP INDEX

- Comando que **remove um índice** existente em uma tabela
- Útil quando
  - O índice **não é mais necessário**
  - O índice está **atrapalhando** operações de escrita (muito INSERT , UPDATE , DELETE)
  - O índice **foi criado errado** (coluna errada, redundante)

```
DROP INDEX idx_clientes_nome ON clientes;
```

IMPACTOS AO REMOVER UM ÍNDICE

Benefícios

- Menos espaço usado em disco
- Inserções/ atualizações/ exclusões ficam mais rápidas

Riscos

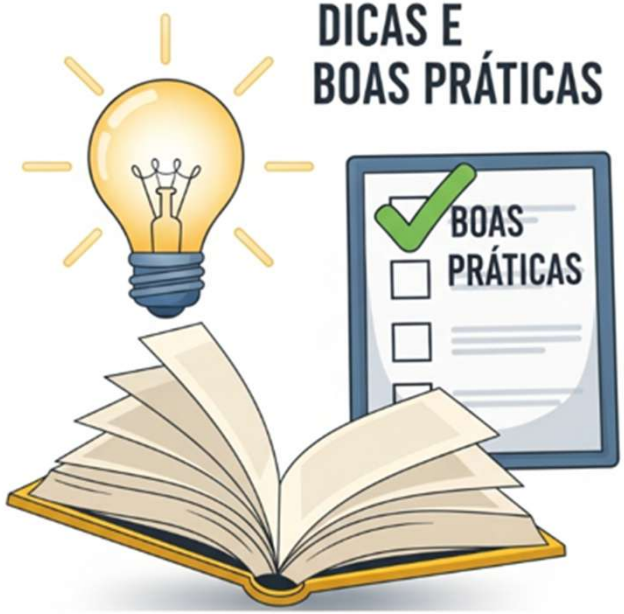
- Consultas que dependiam do índice podem ficar **bem mais lentas**
- Se o índice for **UNIQUE,** ao remover ele perde a **restrição de unicidade**

BOAS PRÁTICAS COM ÍNDICES

- Sempre verificar se o índice é realmente necessário antes de criar.
- Evitar manter **índices duplicados**.
- Antes de remover: usar **EXPLAIN** para entender se a consulta está usando o índice.
- Documentar a exclusão → pode afetar outros sistemas ou relatórios.

Heurística de DBA

- Índices valem a pena em colunas com **alta seletividade**
- Índices ajudam em **tabelas grandes**
- Para colunas com **poucos valores distintos,** é melhor não criar índices



MIGRAÇÃO DE DADOS

O que é Migração de Dados?

- Processo de **transferir informações** de um banco de dados para outro
- Pode envolver
  - Backup e restore (mesmo SGBD)
  - Exportação e importação (estrutura + dados)
  - ETL (Extract, Transform, Load) em projetos de integração
- **Objetivo** – garantir que os dados sejam movidos com segurança, integridade e consistência

Situações do Mundo Real

- **Troca de servidor** – mudar o banco de servidor de testes para produção
- **Atualização de versão** – migrar de MySQL 5.7 para MySQL 8
- **Mudança de SGBD** – migrar dados de PostgreSQL para MySQL
- **Backup** – criar cópia de segurança para recuperação futura
- **Análise de dados** – exportar dados do sistema para planilhas/ BI

FERRAMENTAS E MÉTODOS COMUNS

Linha de comando

- MySQL: `mysqldump` e `mysql`

Ferramentas gráficas

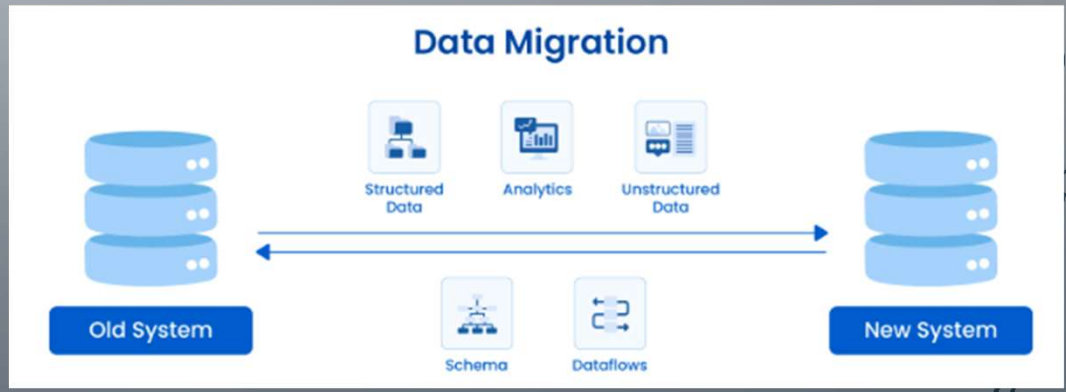
- MySQL Workbench (Data Export/ Data Import)
- pgAdmin (Backup/ Restore)

Formatos comuns

- SQL (`.sql`) – scripts de criação e inserção
- CSV (`.csv`) - tabelas para análise no Excel
- JSON (`.json`) – integração com APIs

TIPOS DE MIGRAÇÃO

1. **Lógica**
   - Exporta apenas a estrutura e os dados via SQL
   - Ex.: `mysqldump` – gera arquivo `.sql` com `CREATE` + `INSERT`
2. **Física**
   - Copia os arquivos binários do banco diretamente
   - Mais rápida, mas depende do SGBD e do SO
3. **Parcial (seletiva)**
   - Apenas algumas tabelas/ dados
   - Ex.: exportar só a tabela de clientes
4. **Total**
   - Banco inteiro (estrutura + dados)



EXPORTAÇÃO

- **Exportar** significa gerar um **arquivo externo** com a estrutura e/ou os dados de um banco
- É como *tirar uma fotografia* do banco para guardar ou levar para outro ambiente

Exportação completa (estrutura + dados)

```
mysqldump –u root –p nome_banco > backup.sql
```

- `-u root` – usuário do banco
- `-p` – pede senha
- `nome_banco` – banco que será exportado
- `> backup.sql` – arquivo gerado
- opção `--no-data` – gera apenas a estrutura sem os dados
- opção `--no-create-info` – gera apenas os `INSERT`, sem `CREATE TABLE`

EXPORTAÇÃO EM FORMATO CSV

```
SELECT * FROM clientes
INTO OUTFILE '/tmp/clientes.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

IMPORTAÇÃO

- **Importar** significa **carregar em um banco de dados** um arquivo externo previamente exportado (geralmente .sql ou .csv)
- É o processo inverso da exportação

Importação de Arquivo SQL (backup completo)

```
mysql –u root –p nome_banco < backup.sql
```

- -u root – usuário
- -p – pede senha
- nome_banco – banco que vai recebe os dados
- < backup.sql – arquivo exportado antes

IMPORTAÇÃO

- Se o banco não existir ainda

  mysql –u root –p –e “CREATE DATABASE loja_backup;”

  mysql –u root –p loja_backup < loja_completo.sql

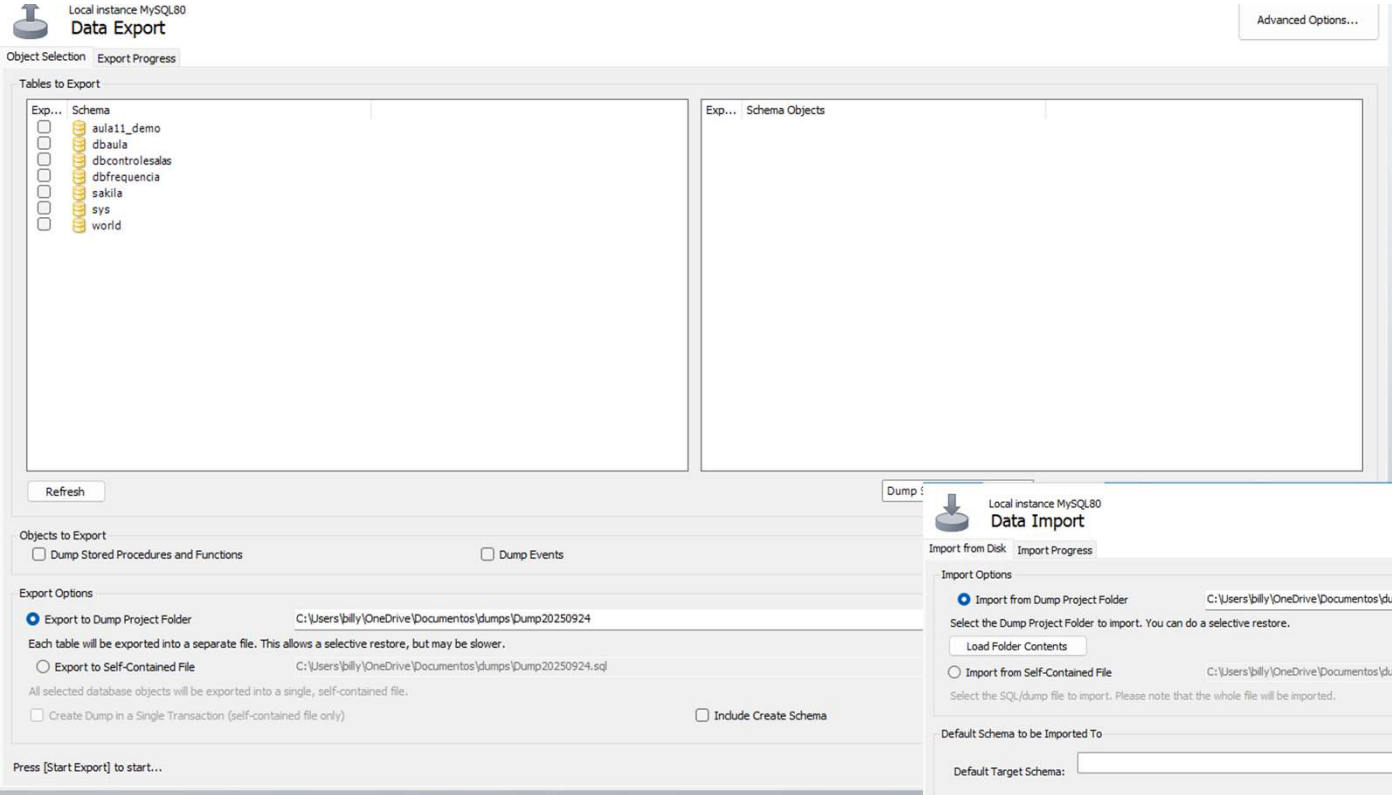
- Importação de Apenas uma Tabela
  - Se você exportou apenas a tabela clientes

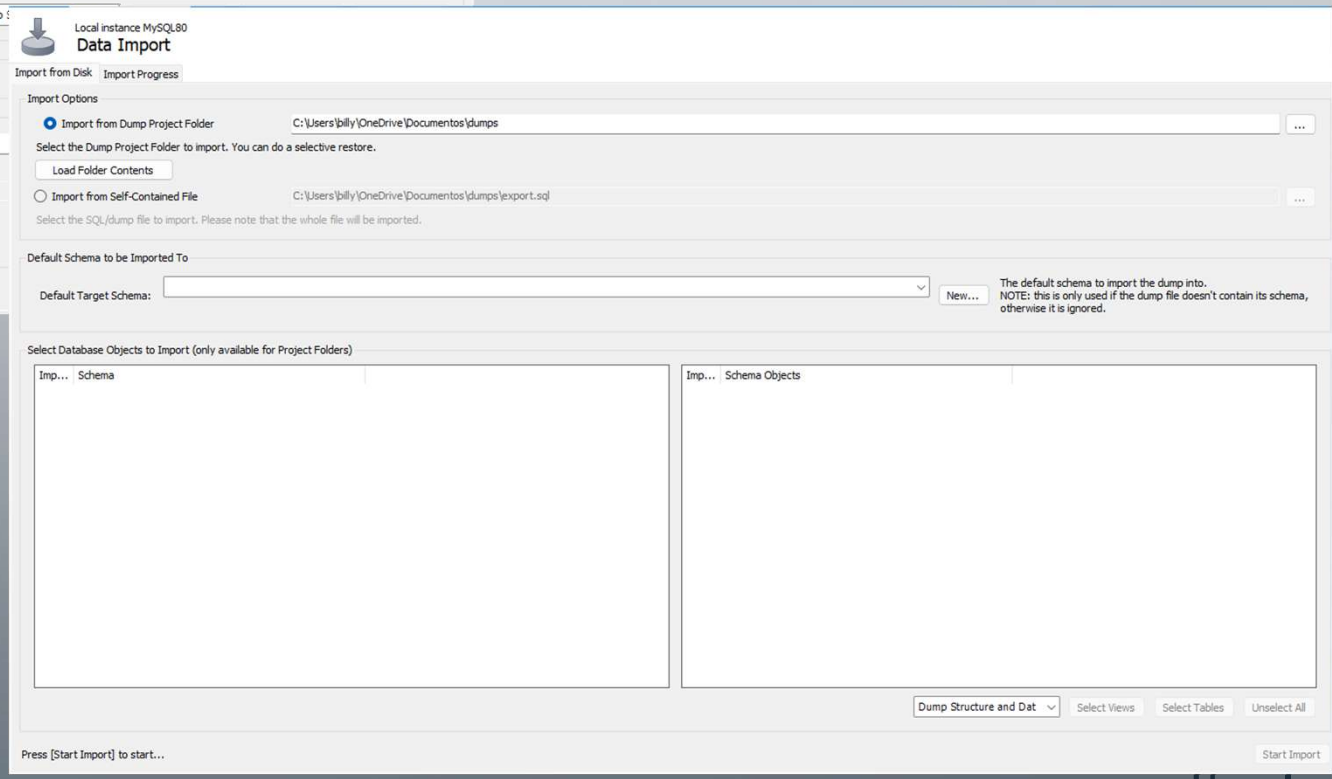
  mysql –u root –p loja < clientes.sql

- Importação de Arquivo CSV
  - Se os dados forma exportados para .csv

```
71 •    LOAD DATA INFILE '/tmp/clientes.csv'
72      INTO TABLE clientes
73      FIELDS TERMINATED BY ','
74      ENCLOSED BY '"'
75      LINES TERMINATED BY '\n';
76
```

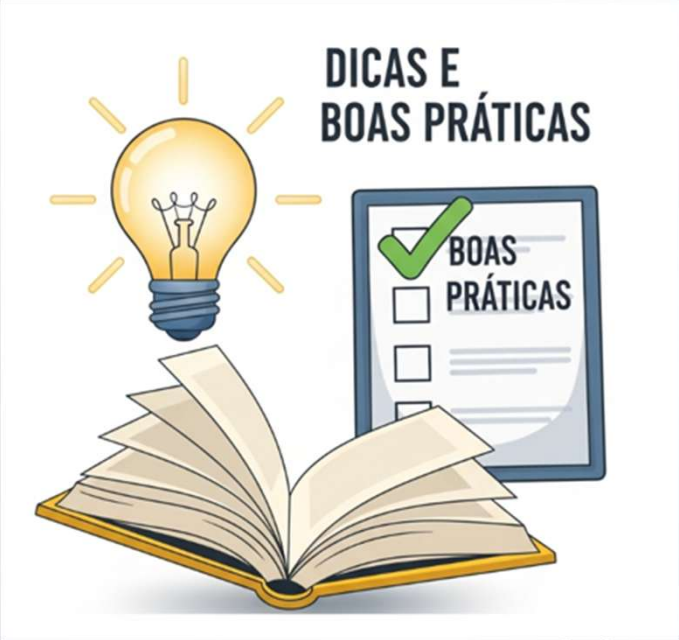
IMPORTAÇÃO VIA MYSQL WORKBENCH (GUI)





BOAS PRÁTICAS PARA EXPORTAÇÃO / IMPORTAÇÃO

- **Exportação**
  - Sempre incluir **data no nome do arquivo –** backup_2025_07_24.sql
  - Conferir se o backup está integro **tentando importar em outro banco**
  - Guardar o arquivo em local seguro (e não só no mesmo servidor)
  - Usar **compressão** (.zip , .tar.gz) para economizar espaço
- **Importação**
  - Conferir se o banco de destino está **vazio** (senão pode dar conflito de chaves primárias)
  - Conferir se o **charset/encoding** está correto (UTF-8 para acentuação)
  - Fazer um **backup do destino** antes de importar (se não for vazio)



Perguntas?! Dúvidas?!?