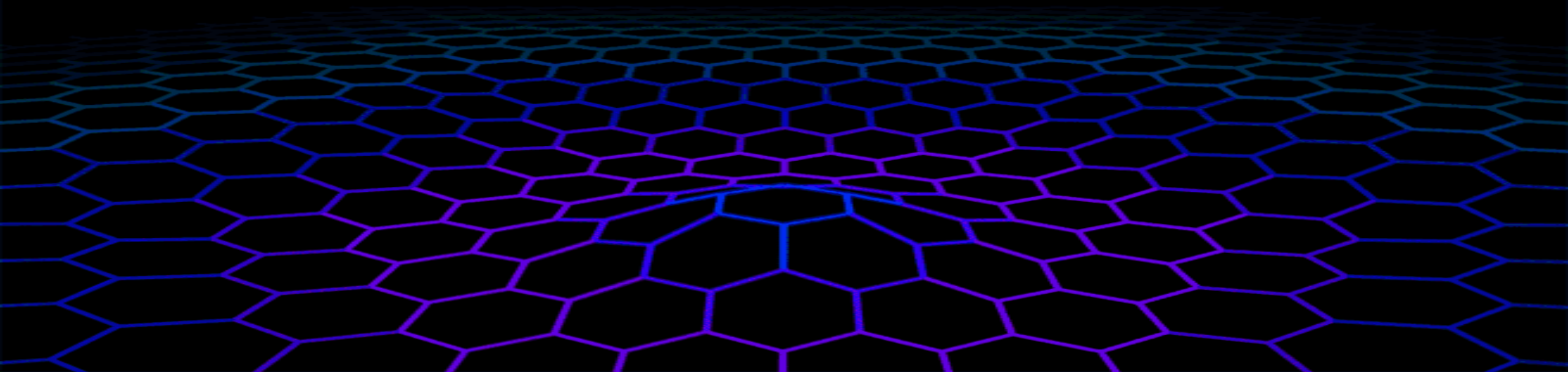
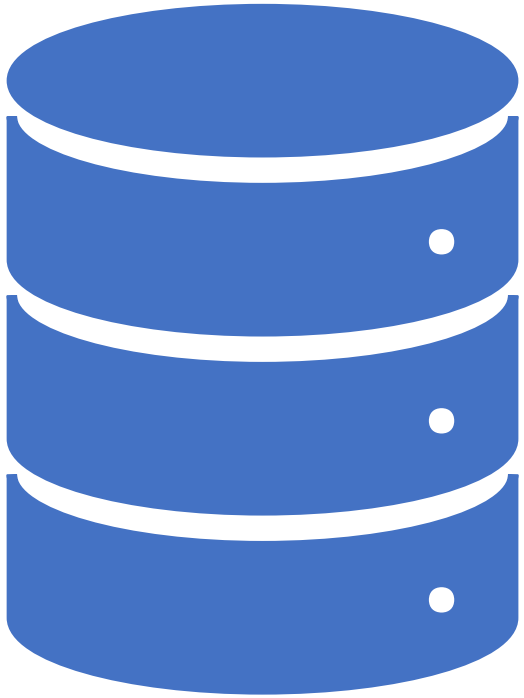
Unisenac
Campus Pelotas

Banco de Dados II





INDEXAÇÃO

ÍNDICES, ENGINES E ANÁLISE DE PERFORMANCE

- Criação e uso de índices em tabelas
- Análise de performance com EXPLAIN
- Comparativo de mecanismos de armazenamento (Storage Engines)
- Cuidados com índices redundantes ou desnecessários

MySQL e PostgreSQL oferecem formas distintas de criar, gerenciar e analisar índices. Vamos destacar essas diferenças.

Conceito de Índice

O que é um índice?

Um índice funciona como o índice de um livro — ele acelera a busca de registros em uma tabela.

Vantagens:
✅ Melhora performance de consultas com WHERE, JOIN, ORDER BY, etc.

Desvantagens:
❌ Consome mais espaço em disco
❌ Impacta INSERT, UPDATE e DELETE

MySQL e PostgreSQL possuem os mesmos conceitos, com sintaxes semelhantes.

Banco de Dados exemplo

```sql
-- MySQL & PostgreSQL
DROP DATABASE IF EXISTS aula07;
CREATE DATABASE aula07;
USE aula07;

-- MySQL
CREATE TABLE cliente (
  id      MEDIUMINT UNSIGNED AUTO_INCREMENT,
  nome    VARCHAR(255) NOT NULL,
  email   VARCHAR(255) DEFAULT NULL,
  cidade  VARCHAR(255),
  PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=1;

-- PostgreSQL
CREATE TABLE cliente (
  id      SERIAL,
  nome    VARCHAR(255) NOT NULL,
  email   VARCHAR(255),
  cidade  VARCHAR(255),
  PRIMARY KEY (id)
);
```

Inserção e Consulta Sem Índice

```sql
-- MySQL e PostgreSQL
INSERT INTO cliente (nome, email, cidade) VALUES
('Ruth Boyd', 'elit.erat@aol.org', 'Sarpsborg'),
('Nayda T. Baker', 'adipiscing.elit.etiam@google.net', 'Orenburg');

-- MySQL
SELECT SQL_NO_CACHE *
FROM cliente
WHERE email LIKE "%_it.e%" AND cidade LIKE "%rg%";

-- PostgreSQL
EXPLAIN ANALYZE
SELECT *
FROM cliente
WHERE email LIKE '%_it.e%' AND cidade LIKE '%rg%';
```

Criando Índices

```sql
-- MySQL
ALTER TABLE cliente ADD INDEX (email(50));

-- PostgreSQL
CREATE INDEX idx_email_cliente ON cliente (email);
```

Análise com EXPLAIN

```sql
-- MySQL
EXPLAIN SELECT *
FROM cliente
WHERE email LIKE "%_it.e%" AND cidade LIKE "%rg%";

-- PostgreSQL
EXPLAIN ANALYZE
SELECT *
FROM cliente
WHERE email LIKE '%_it.e%' AND cidade LIKE '%rg%';

/*
Esperado:
- Redução no tempo de leitura
- Utilização do índice no plano de execução (EXPLAIN)
*/
```

Tipos de Índices

MySQL:
- BTREE (padrão)
- FULLTEXT
- HASH (em MEMORY engine)

PostgreSQL:
- BTREE (padrão)
- HASH
- GIN, GiST (para JSONB, arrays, texto)
- SP-GiST, BRIN (avançados)

Exemplos em MySQL

```sql
-- Índice BTREE (padrão)
CREATE INDEX idx_nome ON cliente (nome);

-- Índice composto
CREATE INDEX idx_nome_cidade ON cliente (nome, cidade);

-- FULLTEXT (para textos longos – InnoDB ou MyISAM)
CREATE FULLTEXT INDEX idx_texto_full ON cliente (nome, email);

-- HASH só é usado automaticamente em MEMORY engine
CREATE TABLE temp_mem (
  id INT,
  valor VARCHAR(100),
  INDEX idx_hash (valor)
) ENGINE=MEMORY;
```

Exemplos em PostgreSQL

```sql
-- Índice BTREE (padrão)
CREATE INDEX idx_nome ON cliente (nome);

-- Índice HASH
CREATE INDEX idx_hash_email ON cliente USING HASH (email);

-- GIN para busca textual (full-text search)
CREATE INDEX idx_gin_nome ON cliente
USING GIN (to_tsvector('portuguese', nome));

-- GIN para campos JSONB
CREATE INDEX idx_gin_json ON cliente
USING GIN (dados_json);

-- BRIN para dados sequenciais (muito grandes)
-- (Usado para grandes tabelas ordenadas por tempo ou ID)
CREATE INDEX idx_brin_id ON cliente
USING BRIN (id);
```

Verificando Índices Existentes

```sql
-- MySQL
SHOW INDEXES FROM cliente;

-- PostgreSQL
SELECT * FROM pg_indexes WHERE tablename = 'cliente';
```

Remoção de Índices

```sql
-- MySQL
DROP INDEX idx_email_cliente ON cliente;

-- PostgreSQL
DROP INDEX idx_email_cliente;
```

Engines no MySQL

InnoDB:
- Suporte a transações ACID
- FOREIGN KEY
- Row locking (registro)
- Alta concorrência

MyISAM:
- Sem suporte a transações
- Sem FK
- Table locking
- Rápido para leitura

PostgreSQL:
- Não possui engines configuráveis por tabela
- Usa MVCC e armazenamento gerenciado internamente

Engines no MySQL (exemplos)

```sql
CREATE TABLE exemplo_innodb (
  id INT AUTO_INCREMENT,
  nome VARCHAR(100),
  PRIMARY KEY (id)
) ENGINE=InnoDB;

CREATE TABLE exemplo_myisam (
  id INT AUTO_INCREMENT,
  nome VARCHAR(100),
  PRIMARY KEY (id)
) ENGINE=MyISAM;
```

Tabelas para Teste com “volume” (MySQL)

```sql
DROP TABLE IF EXISTS usuario;
CREATE TABLE usuario (
  id INT AUTO_INCREMENT,
  nome VARCHAR(255),
  email VARCHAR(255),
  senha VARCHAR(255),
  PRIMARY KEY (id)
) ENGINE=InnoDB;

DROP TABLE IF EXISTS atividade;
CREATE TABLE atividade (
  id INT AUTO_INCREMENT,
  idUsuario INT,
  dataHora TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB;
```

Procedure insereUsuario (MySQL)

```
DELIMITER $$
CREATE PROCEDURE insereUsuario(qtd INT)
BEGIN
  DECLARE v_contador INT DEFAULT 0;
  START TRANSACTION;
    WHILE v_contador < qtd DO
      INSERT INTO usuario (nome, email, senha)
      VALUES (
        CONCAT('usuario', v_contador),
        CONCAT('email', v_contador, '@exemplo.com'),
        MD5(RAND())
      );
      SET v_contador = v_contador + 1;
    END WHILE;
  COMMIT;
END $$
```

Procedure insereAtividade (MySQL)

```
CREATE PROCEDURE insereAtividade(qtd INT)
BEGIN
  DECLARE v_contador INT DEFAULT 0;
  SELECT MAX(id) INTO @max_id FROM usuario;
  START TRANSACTION;
    WHILE v_contador < qtd DO
      INSERT INTO atividade (idUsuario)
      VALUES (1 + FLOOR(RAND() * @max_id));
      SET v_contador = v_contador + 1;
    END WHILE;
  COMMIT;
END $$
DELIMITER ;
```

Chamando as procedures (MySQL)

```
CALL insereUsuario(5000);
CALL insereAtividade(1000);
```

Faça testes

- Cuidado para não “travar o NINTENDO”

Dicas

- ✅ Use EXPLAIN para análise de queries.
- ✅ Evite excesso de índices.
- ✅ Prefira InnoDB no MySQL.
- ✅ No PostgreSQL, foque em boas práticas de modelagem.

Criando índice para melhorar JOIN

```sql
-- MySQL e PostgreSQL
CREATE INDEX idx_idUsuario ON atividade(idUsuario);

-- MySQL
EXPLAIN SELECT u.nome, COUNT(*)
FROM usuario u
JOIN atividade a ON a.idUsuario = u.id
GROUP BY u.nome;

-- PostgreSQL
EXPLAIN ANALYZE SELECT u.nome, COUNT(*)
FROM usuario u
JOIN atividade a ON a.idUsuario = u.id
GROUP BY u.nome;
```

✅ O uso de índice em colunas de JOIN é uma boa prática para performance.

EXPLAIN FORMATADO

```sql
-- MySQL
EXPLAIN FORMAT=JSON
SELECT u.nome, COUNT(*)
FROM usuario u
JOIN atividade a ON a.idUsuario = u.id
GROUP BY u.nome;

-- PostgreSQL
EXPLAIN (ANALYZE, BUFFERS, FORMAT TEXT)
SELECT u.nome, COUNT(*)
FROM usuario u
JOIN atividade a ON a.idUsuario = u.id
GROUP BY u.nome;
```

EXPLAIN FORMATADO

Dicas:
- Use EXPLAIN ANALYZE no PostgreSQL para ver tempo real da execução
- Use FORMAT=JSON no MySQL para entender melhor os planos

Lembre-se de verificar:
- Tipo de leitura (Seq Scan, Index Scan, Index Only Scan)
- Custo estimado (cost=...)
- Quantidade de linhas estimadas vs reais

mysqlslap

mysqlslap

Pequena ferramenta de diagnóstico que acompanha o MySQL

Testar a carga de servidores de banco de dados

Pode emular um grande número de conexões de cliente que chegam ao servidor de banco de dados simultaneamente.

BATENDO FORTE NO BANCO com mysqlslap

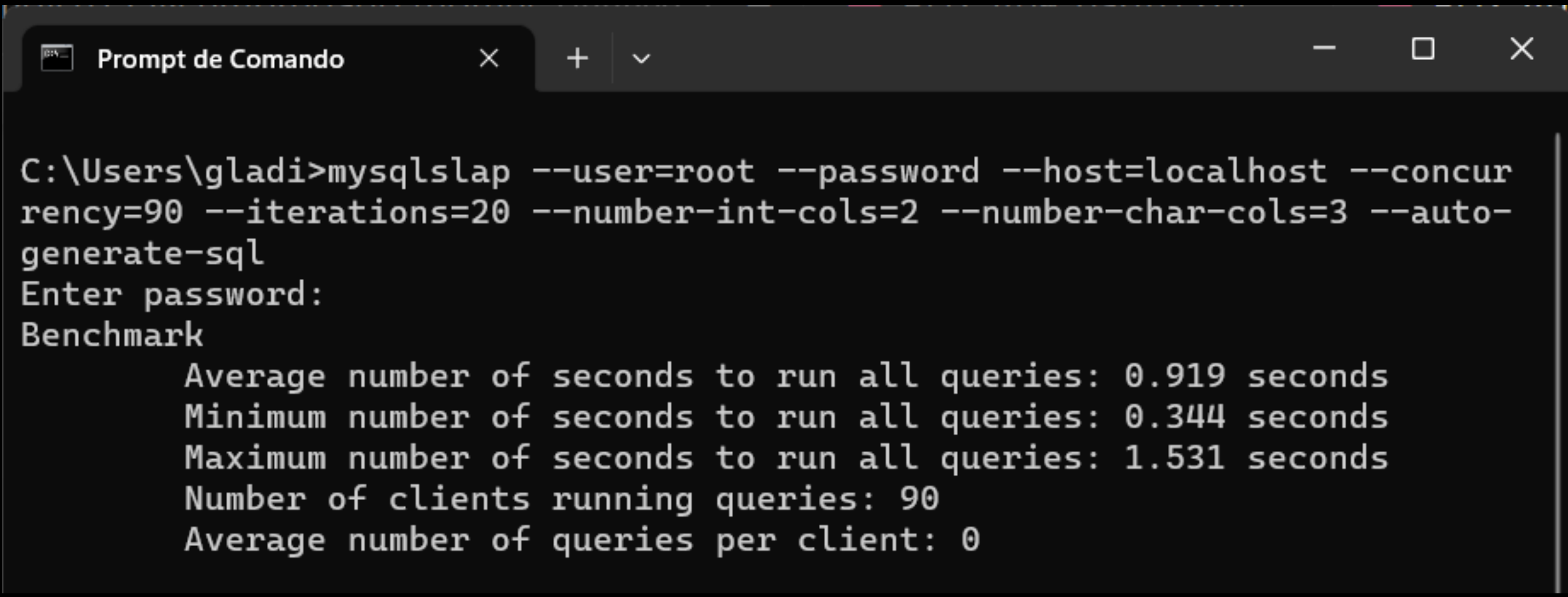
```
mysqlslap
    --user=root
    --password=
    --auto-generate-sql
    --concurrency=100
    --iterations=10
    --number-char-cols=10
    --number-int-cols=5
    --engine=innodb
```

100 conexões concorrentes, com comandos auto gerados (você pode também testar seus próprios comandos SQL), com 10 execuções, em uma tabela com 5 colunas INT, e com 10 coluna CHAR, e apenas com o Engine InnoDB

BATENDO FORTE NO BANCO com mysqlslap

```
mysqlslap --user=root --password --host=localhost --concurrency=88 --
iterations=100 --create-schema=aula07 --query="SELECT * FROM usuario;"

mysqlslap --user=root --password --host=localhost --concurrency=90 --iterations=10
--create-schema=aula07 --query="SELECT MAX(id) FROM usuario;"

mysqlslap --user=root --password --host=localhost --concurrency=90 --iterations=20
--number-int-cols=2 --number-char-cols=3 --auto-generate-sql

mysqlslap --user=root --password --delimiter=";" --create="CREATE TABLE a (b int);
INSERT INTO a VALUES (23)" --query="SELECT * FROM a" --concurrency=80 --
iterations=20

mysqlslap --delimiter=";" --create="CREATE TABLE a (b int); INSERT INTO a VALUES
(23)" --query="SELECT * FROM a" --concurrency=50 --iterations=200
```

BATENDO FORTE NO BANCO com mysqlslap



Boas práticas

✅ Use índices com sabedoria: evite excessos!
✅ Sempre analise o plano de execução para decisões de performance
✅ No MySQL, prefira InnoDB para integridade e transações
✅ No PostgreSQL, otimize com os tipos corretos de índices (BTREE, GIN, etc.)

Ferramentas recomendadas:
- [Generatedata.com](https://www.generatedata.com) para dados falsos
- pgAdmin (PostgreSQL) e DBeaver (multi-SGBD)

Exercícios

Ver arquivo **BD2-A07-Roteiro.sql** no Blackboard

-- "Índices são como atalhos: ajudam se bem pensados, atrapalham se usados sem necessidade."