A dinâmica entre DDL, DML e DQL na evolução de bancos relacionais e sistemas NoSQL

Pedro Taiette Sato Librais (2000373)
Thiago Henrique do Rego (2002255)

UNIMAR – 2025

Agenda

1. Contexto
2. Conceitos
3. Exemplos
4. Ciclo de vida
5. SQL x NoSQL
6. Boas práticas
7. Conclusão

1. Contexto

- Bancos de dados são base crítica
- SQL (Structured Query Language) é padrão dominante
- Três subconjuntos centrais:
  - **DDL** — estrutura
  - **DML** — manipulação
  - **DQL** — consulta

Objetivo: explicar o papel de cada linguagem e mostrar a relação com NoSQL.

2. Subconjuntos da SQL

Subconjunto	Foco	Uso principal
DDL	Estrutura	Criação e mudanças
DML	Dados	Uso operacional
DQL	Consulta	Projeção e Busca

Eles se complementam em momentos diferentes.

DDL — Data Definition Language

Comandos: `CREATE`, `ALTER`, `DROP`

Responsável por:

- Criar tabelas e restrições
- Definir tipos e chaves
- Estabelecer desempenho e integridade do banco

DML — Data Manipulation Language

Comandos: `INSERT`, `UPDATE`, `DELETE`

Utilizado para:

- Inserir registros
- Atualizar informações
- Remover dados

Base das operações rotineiras em um sistema.

DQL — Data Query Language

Comando principal: `SELECT`

Usado em:

- Relatórios
- Auditorias
- Extração de informação

Transforma dados em conhecimento.

3. Exemplos em PostgreSQL

DDL

```
CREATE TABLE Animal (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(50) NOT NULL,
  especie VARCHAR(50),
  data_nascimento DATE
);
```

DML

```sql
INSERT INTO Animal (nome, especie, data_nascimento)
VALUES ('Estrela', 'Bovina', '2021-03-15');
```

DQL

```sql
SELECT nome, especie
FROM Animal
WHERE data_nascimento > '2020-01-01';
```

4. Ciclo de vida do banco

1. Construção (DDL)

Definição de tabelas, colunas e relacionamentos.

2. Manipulação (DML + ajustes DDL)

Inserção, atualização e mudanças estruturais.

3. Uso operacional (DQL)

Consultas e relatórios.

Interdependência

- **DDL** cria a base
- **DML** mantém os dados ativos
- **DQL** extrai informação útil

O ciclo se repete conforme o sistema evolui.

Pensando em Minecraft...

DDL - Construir o mapa/mundo
DML - Adicionar e modificar os objetos/jogadores
DQL - Procurar coisas no mapa ("Quantos mobs existem perto de mim?")

5. SQL e NoSQL

Compatibilidade parcial

- Cassandra (CQL)
- Apache Hive / Spark SQL
- MongoDB com conectores SQL-like

Facilita integração híbrida.

Limitações e diferenças

- Estrutura flexível (schema-less)
- JOINs limitados ou ausentes
- Consistência eventual
- Foco em escalabilidade horizontal

Menos dependência de DDL formal.

Quando usar NoSQL?

Cenários comuns:

- Alto volume de dados
- Alta disponibilidade
- Estrutura variável
- Dados distribuídos ou semiestruturados

Muitos sistemas combinam **SQL + NoSQL**.

6. Boas práticas

- Versionamento de esquema
- Ferramentas de migração
- Princípio do menor privilégio
- Ambientes separados (dev/teste/prod)

Evita falhas e inconsistências.

7. Conclusão

- DDL, DML e DQL são complementares
- Cada linguagem tem seu papel
- Dialetos mudam a sintaxe, não a função
- NoSQL complementa, não substitui
- Boas práticas garantem integridade e longevidade

Referências

- Date (2004)
- Elmasri & Navathe (2016)
- Sadalage & Fowler (2012)

- Obrigado Éttore por nos apresentar o Marp :)