Por: Alice Dantas

Modelo Físico

O modelo físico traduz o modelo lógico em uma implementação real, com foco em como os dados vão estar organizados no disco.

Data Definition Language (DDL): comandos DDL são usados para definir e gerenciar a estrutura dos objetos de banco de dados, como tabelas, índices e esquemas.

Comando	Função
`CREATE`	Cria objetos no banco, como tabelas, índices, visões, esquemas etc.
`ALTER`	Altera a estrutura de um objeto existente (ex: adicionar colunas com ADD ou modificar tipo de uma coluna com MODIFY).
`DROP`	Exclui objetos existentes no banco, como tabelas ou visões.
`TRUNCATE`	Remove **todos os dados** de uma tabela, mas mantém sua estrutura.
`RENAME`	Renomeia um objeto de banco de dados, como uma tabela.
`COMMENT`	Adiciona comentários de descrição a objetos do banco.

```sql
-- Criar tabela
CREATE TABLE Aluno (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100)
);

-- Alterar tabela
ALTER TABLE Aluno ADD Email VARCHAR(150);

-- Apagar tabela
DROP TABLE Aluno;

-- Apagar todos os dados da tabela (sem deletar a estrutura)
TRUNCATE TABLE Aluno;

-- Renomear uma tabela
ALTER TABLE Aluno RENAME TO Estudante;
```

Data Manipulation Language (DML): comandos DML são usados para manipular os dados dentro dos objetos definidos pelo DDL, ou seja, para inserir, atualizar, excluir e consultar dados.

Comando	Função
SELECT	Consulta dados das tabelas.
INSERT	Insere novos registros nas tabelas.
UPDATE	Atualiza dados existentes nas tabelas.
DELETE	Exclui registros de uma tabela.

```sql
-- Inserir dados
INSERT INTO Aluno (ID, Nome) VALUES (1, 'Carlos Silva');

-- Consultar dados
SELECT * FROM Aluno WHERE Nome = 'Carlos Silva';

-- Atualizar dados
UPDATE Aluno SET Nome = 'Carlos S.' WHERE ID = 1;

-- Deletar dados
DELETE FROM Aluno WHERE ID = 1;
```

Por: Alice Dantas

Principais tipos de dados

Tipos Numéricos	Descrição	Exemplo
`INT`	Inteiro padrão (32 bits)	`122`
`DECIMAL(p,s)`	Precisão exata – útil para dinheiro	`10.99`
`FLOAT`	Ponto flutuante (menos preciso)	`3.14`
`DOUBLE`	Ponto flutuante (dupla precisão)	`3.1415926`

Tipo de Texto	Descrição	Exemplo
`CHAR(n)`	Tamanho fixo de n caracteres	`'ABC '`
`VARCHAR(n)`	Tamanho variável até n caracteres	`'Nome completo'`
`TEXT`	Texto longo (sem limite fixo)	`'Lorem ipsum...'`

Tipo de Data e Hora	Descrição	Exemplo
`DATE`	Data (ano-mês-dia)	`'2025-05-10'`
`TIME`	Hora (hora:minuto:segundo)	`'14:30:00'`
`DATETIME`	Data e hora (sem fuso horário)	`'2025-05-10 14:30:00'`
`YEAR`	Apenas o ano	`2025`

Por: Alice Dantas

Constraints

NOT NULL: A coluna não pode aceitar valores nulos (vazios).
`nome VARCHAR(100) NOT NULL`

AUTO_INCREMENT: O valor da coluna será gerado automaticamente, geralmente incrementando 1 a cada nova linha.
`id INT AUTO_INCREMENT PRIMARY KEY`

PRIMARY KEY: Define a coluna (ou conjunto de colunas) que identifica unicamente cada registro na tabela. Só pode haver uma por tabela e sempre é NOT NULL automaticamente.
`id INT PRIMARY KEY`

FOREIGN KEY: A coluna faz referência a outra tabela. Usado para criar relacionamentos entre tabelas.

```
cliente_id INT,
FOREIGN KEY (cliente_id) REFERENCES clientes(id)
```

DEFAULT: Define um valor padrão para a coluna caso nenhum valor seja informado.
`status VARCHAR(20) DEFAULT 'pendente'`

UNIQUE: Garante que os valores da coluna sejam únicos (não se repitam).
`email VARCHAR(100) UNIQUE`

CHECK: Impõe uma condição lógica que os valores da coluna devem obedecer.
`idade INT CHECK (idade >= 18)`

`ON DELETE` / `ON UPDATE`: Usado com `FOREIGN KEY` para definir o que acontece quando um registro da tabela referenciada for deletado ou atualizado. Exemplo:

```
FOREIGN KEY (cliente_id) REFERENCES clientes(id) ON DELETE
CASCADE
```

- `CASCADE`: deleta também os registros relacionados.
- `SET NULL`: define como `NULL`.
- `RESTRICT`: impede a exclusão/alteração se houver dependentes.

Por: Alice Dantas

Comando SELECT

Sintaxe Básica:

```
SELECT *
FROM <nome_tabela>;
```

Filtrando Registros com a Cláusula WHERE

```
SELECT column1, column2
FROM <nome_tabela>
WHERE <condição>;
```

WHERE com LIKE

Exemplos: selecionar clientes cujo nome começa com "A“ e tenha “Ana” como parte do nome, respectivamente.

```
SELECT nome
FROM Clientes
WHERE nome LIKE 'A%’;
```

```
SELECT nome
FROM Clientes
WHERE nome LIKE ‘%Ana%';
```

WHERE com BETWEEN

Exemplo: selecionar clientes entre 20 e 30 anos.

```
SELECT nome, idade
FROM Clientes
WHERE idade BETWEEN 20 AND 30;
```

Ordenando Resultados com ORDER BY

```
SELECT column1, column2
FROM <nome_tabela>
ORDER BY column1 ASC/DESC;
```

Limitando Resultados com LIMIT

```
SELECT column1, column2
FROM <nome_tabela>
LIMIT 10;
```

Eliminando Resultados Duplicados com a Cláusula DISTINCT

```
SELECT DISTINCT column1
FROM <nome_tabela>;
```

Por: Alice Dantas

Agrupando Registros com a Cláusula GROUP BY

```
SELECT column1, f_agregada(cln2)
FROM <nome_tabela>
GROUP BY column1;
```

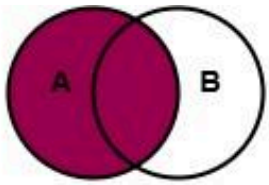
* As funções agregadas realizam cálculos em um conjunto de valores. COUNT(): Conta o número de registros. SUM(): Soma valores. AVG(): Calcula a média. MIN(), MAX(): Retorna o menor e o maior valor.

Filtrando Grupos com a Cláusula HAVING

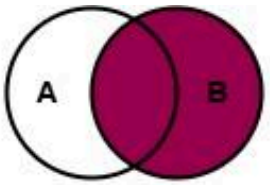
```
SELECT column1, f_agregada(cln2)
FROM <nome_tabela>
GROUP BY column1
HAVING f_agregada(cln2) > <valor>;
```

* Filtra os **grupos** gerados pelo GROUP BY com base em condições aplicadas às funções agregadas.

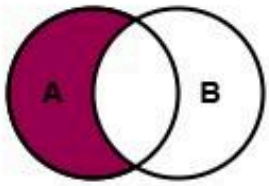
Combinando registros de duas ou mais tabelas com JOIN



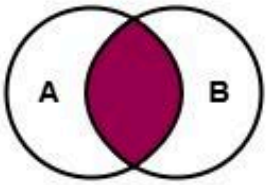

```
SELECT <coluna>
FROM tabelaA A
LEFT JOIN tabelaB B
ON A.key = B.key
```



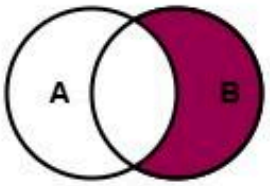

```
SELECT <coluna>
FROM tabelaA A
RIGHT JOIN tabelaB B
ON A.key = B.key
```



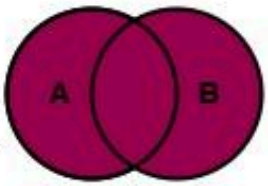

```
SELECT <coluna>
FROM tabelaA A
LEFT JOIN tabelaB B
ON A.key = B.key
WHERE B.key IS NULL
```



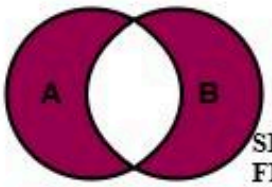

```
SELECT <coluna>
FROM tabelaA A
INNER JOIN tabelaB B
ON A.key = B.key
```




```
SELECT <coluna>
FROM tabelaA A
RIGHT JOIN tabelaB B
ON A.key = B.key
WHERE A.key IS NULL
```




```
SELECT <coluna>
FROM tabelaA A
FULL OUTER JOIN tabelaB B
ON A.key = B.key
```




```
SELECT <coluna>
FROM tabelaA A
FULL OUTER JOIN tabelaB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

Por: Alice Dantas

Views

Uma *VIEW* é como uma consulta salva. Você usa uma *VIEW* quando quer reutilizar um SELECT complexo, organizar melhor o banco de dados ou dar acesso controlado aos dados.

Sintaxe:

```
CREATE VIEW nome_da_view AS
SELECT colunas
FROM tabela
WHERE condição;
```

Observe a estrutura abaixo:

```
CREATE TABLE clientes (
    id INT PRIMARY KEY,
    nome VARCHAR(50),
    cidade VARCHAR(50),
    saldo DECIMAL(10, 2)
);
```

Criando a view:

```
CREATE VIEW clientes_ricos AS
SELECT nome, cidade, saldo
FROM clientes
WHERE saldo > 1000;
```

Usando uma view

```
SELECT * FROM clientes_ricos;
```