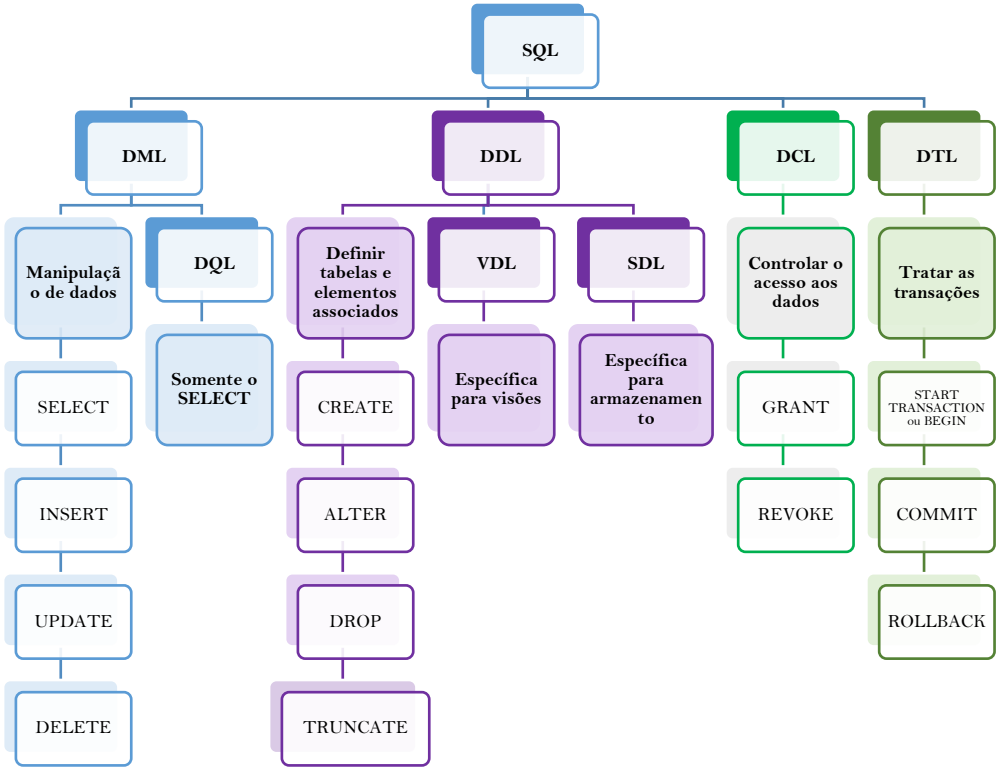
Tecnologia da Informação

RESUMO

SQL (DML)

LINGUAGEM SQL E SUBDIVISÕES



SINTAXE BÁSICA DO SELECT

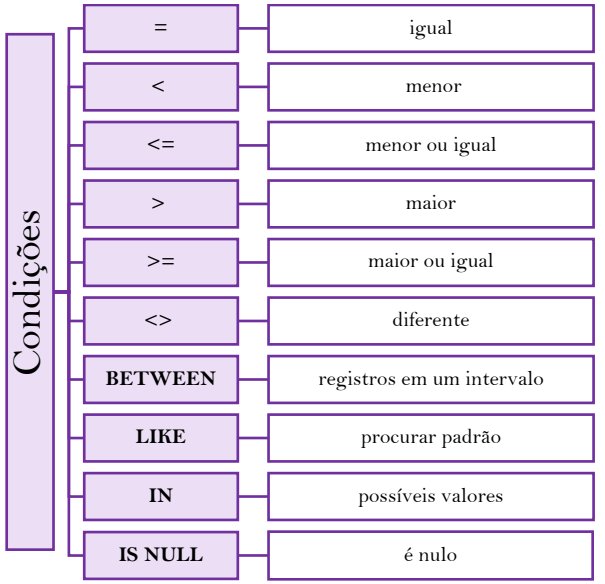
SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição;

SELECT * **FROM** nome_da_tabela **WHERE** condição;

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela;

SELECT DISTINCT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição;

CONDIÇÕES NA CLÁUSULA WHERE



SELECT * **FROM** Clientes **WHERE** Pais='Mexico';

BETWEEN, IN, LIKE, IS NULL E IS NOT NULL

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** coluna **BETWEEN** valor1 **AND** valor2;

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** coluna **IN** (valor1, valor2, ...);

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** coluna **LIKE** padrão;

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** coluna **IS NULL;**

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** coluna **IS NOT NULL;**

PADRÃO NO LIKE

Operador LIKE	Procurar padrão em uma coluna
%	Substitui um número qualquer de 0 ou mais caracteres.
_	Substitui um único caractere.
LIKE 'A%'	Qualquer string que inicie com A.
LIKE '%A'	Qualquer string que termine com A.
LIKE 'A_'	String de dois caracteres que tenha a primeira letra A e o segundo caractere seja qualquer outro.

COMPARAÇÃO COM NULL

A comparação com NULL não deve ser feita com os operadores lógicos = ou <>, mas sim com IS NULL e IS NOT NULL. Ao comparar qualquer coisa com NULL usando os operadores lógicos comuns, será retornado um resultado desconhecido na comparação (UNKNOWN) e, por isso, não serão retornadas linhas.

MAIS DE UMA CONDIÇÃO E NEGAÇÃO

AND

- Registros em que todas as condições são verdadeiras.
- **SELECT** coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição1 **AND** condição2 **AND** condição3 ...;

OR

- Registros em que pelo menos uma das condições é verdadeira.
- **SELECT** coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição1 **OR** condição2 **OR** condição3 ...;

NOT

- Registros que não satisfazem uma condição.
- **SELECT** coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE NOT** condição;

ALIAS OU APELIDO

Sobre os alias:

- São usados para fornecer um **nome temporário a uma tabela ou coluna** em uma tabela.
- Costumam ser usados para **tornar os nomes das colunas mais legíveis**.
- Existe **apenas para a duração da consulta**.

SELECT coluna1 **AS** nova, coluna2 **FROM** nome_da_tabela **WHERE** condição;

OU

SELECT coluna1 nova, coluna2 **FROM** nome_da_tabela **WHERE** condição;

SELECT coluna1, coluna2... **FROM** nome_da_tabela **AS** nova **WHERE** condição;

OU

SELECT coluna1, coluna2... **FROM** nome_da_tabela nova **WHERE** condição;

ORDENAÇÃO COM ORDER BY

A linguagem SQL permite que o usuário **ordene as tuplas no resultado de uma consulta** pelos valores de um ou mais atributos que aparecem, usando a cláusula **ORDER BY**.

A ordem padrão está em ordem crescente de valores. A palavra-chave **DESC** pode ser usada para ordenar os resultados em **ordem decrescente** de valores. A palavra-chave **ASC** pode ser usada para especificar a **ordem crescente** explicitamente.

A sintaxe básica para esse comando é:

SELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição **ORDER BY** coluna **ASC**;

ELECT coluna1, coluna2, ... **FROM** nome_da_tabela **WHERE** condição **ORDER BY** coluna **DESC**;

Uma sintaxe possível para a cláusula ORDER BY é a que indica o número da coluna ao invés de seu nome. O número indica qual coluna da cláusula SELECT será usada para a ordenação. Assim, se for 1, será usada a primeira coluna, se for 2, a segunda, e, assim, sucessivamente.

SELECT cpf, nome **FROM** funcionario **WHERE** salario > 1000 **ORDER BY** 1 **ASC**;

(ORDENAÇÃO PELO CPF)

FUNÇÃO DE AGREGAÇÃO

As **funções de agregação** são usadas para **resumir informações de várias tuplas em uma síntese de tupla única**. Existem diversas funções de agregação embutidas no SQL: **COUNT, SUM, MAX, MIN e AVG**.

A sintaxe básica para essas funções é:

SELECT FUNCAO(coluna1) **FROM** nome_da_tabela **WHERE** condição;
em que **FUNCAO** é qualquer uma das funções de agregação.

O quadro a seguir apresenta as definições dessas funções:

FUNÇÃO	RETORNO
MIN	Menor valor de uma coluna.
MAX	Maior valor de uma coluna.
COUNT	Número de linhas que atendem a um critério.
AVG	Média dos valores de uma coluna numérica.
SUM	Soma dos valores de uma coluna numérica.

A cláusula **COUNT** pode ser usada com o nome da coluna, * ou com 1:

- **COUNT(nome_da_coluna)**: retorna o número de linhas excluindo-se da contagem as linhas que possuem nulo para a coluna desejada.
- **COUNT(*) ou COUNT(1)**: retorna o número total de linhas, independentemente de valores nulos registrados para qualquer campo.

A cláusula **SUM** pode ser usada com o nome da coluna ou com um número indicativo da quantidade a ser somada:

- **SUM(nome_da_coluna)**: retorna o somatório dos valores presentes em nome_da_coluna.
- **SUM(1)**: retorna um somatório, sendo somado 1 para cada registro encontrado. Resultado similar a COUNT(*) ou COUNT(1).
- **SUM(2)**: retorna um somatório, sendo somado 2 para cada registro encontrado.
- **SUM(N)**: retorna um somatório, sendo somado N para cada registro encontrado.

AGRUPAMENTOS

A linguagem SQL tem uma cláusula **GROUP BY** para **aplicar agrupamentos**.

SELECT colunas **FROM** nome_da_tabela **WHERE** condição **GROUP BY** coluna;

A cláusula **HAVING** pode ser usada para **definir uma condição para um agrupamento** com GROUP BY.

SELECT colunas **FROM** nome_da_tabela **WHERE** condição **GROUP BY** coluna **HAVING** condição;

PRODUTO CARTESIANO

O **Produto Cartesiano** seleciona **todos os pares de linhas das duas relações de entrada** (independentemente de ter ou não os mesmos valores em atributos comuns). A nova relação possui todos os atributos que compõem cada uma das relações que fazem parte da operação.

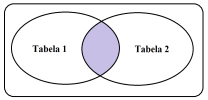
Em SQL, o produto cartesiano é indicado com o uso de vírgulas entre as tabelas desejadas.

SELECT tabela1.coluna1, tabela2.coluna2., ... **FROM** tabela1, tabela2 **WHERE** condição;

A quantidade de linhas do resultado do produto cartesiano é dada pela multiplicação da quantidade de linhas das tabelas de entrada. Logo, se A possui 10 linhas e B possui 100 linhas, então SELECT * FROM A, B irá possuir 1000 linhas.

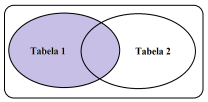
Porém, tome muito cuidado, pois pode haver alguma outra condição após o WHERE ou mesmo nas tabelas de entrada que altere essa quantidade de linhas do resultado.

JUNÇÕES (JOINS)



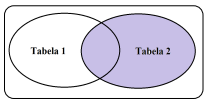
INNER JOIN (ou simplemente JOIN)

- Retorna somente os registros que possuem valores relacionados em ambas as tabelas, isto é, as intersecções.



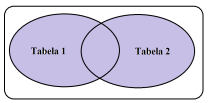
LEFT JOIN (ou LEFT OUTER JOIN)

- Retorna todos os registros da tabela da esquerda, e os registros relacionados da tabela da direita.
- Preenche campos não relacionados na tabela da direita com NULL.



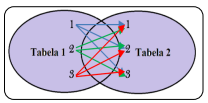
RIGHT JOIN (ou RIGHT OUTER JOIN)

- Retorna todos os registros da tabela da direita, e os registros relacionados da tabela da esquerda.
- Preenche campos não relacionados na tabela da esquerda com NULL



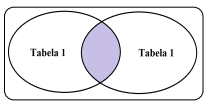
FULL OUTER JOIN

- Retorna todos os registros, independente de relação.
- Preenche campos não relacionados em qualquer das tabelas com NULL.



CROSS JOIN

- Retorna todos os registros da primeira relacionados com todos os registros da segunda.
- É o produto cartesiano.



SELF JOIN

- União de uma tabela com ela mesma.

SELECT colunas **FROM** tabela1 **JOIN** tabela2 **ON** tabela1.coluna = tabela2.coluna;

SELECT colunas **FROM** tabela1 **INNER JOIN** tabela2 **USING** (coluna);

OPERADORES DE CONJUNTOS

OPERADOR	RETORNO
UNION	Todas as linhas pertencentes as consultas envolvidas, sem as repetições.
UNION ALL	Todas as linhas pertencentes as consultas envolvidas, incluindo as repetições.
INTERSECT	Linhas que estão tanto na primeira quanto na segunda consulta. Intersecção, sem repetições.
EXCEPT	Linhas que estão na primeira, mas não estão na segunda, sem repetições.

SELECT colunas **FROM** tabela1

UNION OU UNION ALL OU INTERSECT OU EXCEPT

SELECT colunas **FROM** tabela2;

CONSULTA ANINHADA

Uma subconsulta, consulta interna ou seleção interna é uma consulta que está aninhada dentro de uma instrução SELECT, INSERT, UPDATE ou DELETE ou em outra subconsulta.

As subconsultas podem ser comparadas com a consulta externa com o uso de operadores IN (ou NOT IN), ANY, ALL e EXISTS (ou NOT EXISTS), além dos operadores básicos =, <, <=, >, >=, <>.

ALL E ANY

Os operadores **ANY** e **ALL** permitem realizar uma comparação entre o valor de uma única coluna e um conjunto de outros valores.

- **ANY**: TRUE se **QUALQUER um dos valores** da subconsulta **atender a condição**.

  SELECT colunas **FROM** tabela WHERE coluna operador **ANY** (subconsulta);

- **ALL**: TRUE se **TODOS os valores** da subconsulta **atenderem a condição**.

  SELECT colunas **FROM** tabela WHERE coluna operador **ALL** (subconsulta);

EXISTS E NOT EXISTS

A cláusula **EXISTS** faz uma **verificação se existe algum resultado para a subconsulta informada**. **Caso haja**, o **resultado da consulta principal é exibido**. É muito comum sua utilização quando se deseja trazer resultados onde um valor específico existe dentro de outra tabela. A sintaxe básica é:

SELECT colunas **FROM** tabela **WHERE EXISTS (SELECT** colunas **FROM** tabela WHERE condicão**)**;

Da mesma forma, também há a cláusula **NOT EXISTS**, somente **retorna o resultado da consulta principal, se não houver nenhum resultado para a subconsulta**.

CLÁUSULAS ESPECIAIS

OPERADOR	RETORNO
CASE	Percorre condições e retorna um valor para a 1ª condição atendida.
TOP, LIMIT ou FETCH FIRST	Especifica o número de registros a serem retornados.
OFFSET	Pula um número de registros antes de começar a exibir.
SELECT INTO	Copia dados de uma tabela para uma nova tabela

INSTRUÇÃO DELETE

A instrução básica para **deletar registros existentes de uma tabela** é a instrução **DELETE**.

DELETE FROM nome_da_tabela **WHERE** condição;

INSTRUÇÃO UPDATE

A instrução básica para **atualizar os registros de uma tabela** é a instrução **UPDATE**.

UPDATE nome_da_tabela **SET** coluna1 = valor1, coluna2 = valor2 ... **WHERE** condição;

INSTRUÇÃO INSERT INTO

A instrução básica para **inserir novos registros em uma tabela** é a instrução **INSERT INTO**.

INSERT INTO nome_da_tabela (coluna1, coluna2, coluna3, ...) **VALUES** (valor1, valor2, valor3, ...);

INSERT INTO nome_da_tabela **VALUES** (valor1, valor2, valor3, ...);