MINISTÉRIO DAS TELECOMUNICAÇÕES, TECNOLOGIAS
DE INFORMAÇÃO E COMUNICAÇÃO SOCIAL
MINISTÉRIO DA EDUCAÇÃO

UNIDADE X:PROGRAMABILIDADE - CONSULTAS AVANÇADAS E SUBCONSULTAS

Prof. Paulo Tumba

INSTITUTO DE TELECOMUNICAÇÕES

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE

Consultas SQL

- Em uma base de dados a operação de consulta/visualização de dados é feita através de um único comando, SELECT.
- À sintaxe básica poderão ser adicionadas diferentes cláusulas que permitem acrescer mecanismos que permitem:

– Restrição

– Comparação

– Ordenação

– Agrupamento de dados

– Junção de dados presentes em várias tabelas

- **Sintaxe básica:**

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, Tabelan

Restricão- Where

A operação de restrição permite restringir o número de registos (linhas) a apresentar como resultado de uma consulta à base de dados. Onde apenas se mostra algumas linhas que contêm os dados pretendidos.

É necessário associar uma condição a cláusula SELECT.

- **Sintaxe:**

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, ..., Tabelan

[WHERE condição]

Onde a condição é uma expressão lógica que pode conter operadores ógicos e relacionais.

Operações Relacionais

- **Operadores relacionais**

Operador	Descrição
=	Igual a
<	Menor que
>	Maior que
<=	Menor ou igual que
>=	Maior ou igual que
!= ou <>	Diferente

Operadores lógicos

Os operadores lógicos funcionam com expressões que devolvam um valor lógico.

– AND – As duas condições têm que ser satisfeitas.

– OR – Apenas uma das condições tem que ser satisfeita.

– NOT – Negação. É um operador unário.

Operador	Sintaxe
AND	Condição1 **AND** Condição2
OR	Condição1 **OR** Condição2
NOT	**NOT** Condição

Restrição – WHERE

Exemplos:

- Seleccionar todos os clientes maiores de 25 anos.

  SELECT * FROM clientes WHERE idade > 25

- Seleccionar todos os clientes maiores de 25 anos com ID igual a 1.

  SELECT * FROM clientes WHERE idade > 25 AND id = 1

- Problema: Seleccionar todos os clientes com ID entre 10 e 15?

SELECT * FROM clientes WHERE

Comparação valores nulos

- O valor NULL não é zero. É um indicador de inexistência de valor.
- A comparação com NULL tem que ser feita usando o operador IS.

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, ...,

Tabelan WHERE valor IS

[NOT] NULL

Exemplo:

```
SELECT * FROM clientes WHERE email IS NULL;
SELECT * FROM clientes WHERE email IS NOT NULL;
```

Comparação de strings

- O operador LIKE é usada na comparação de strings .
- A comparação de strings com operadores relacionais utiliza sempre a totalidade da string .
- Enquanto que o operador LIKE permite fazer comparações de partes da string .

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, ..., Tabelan

WHERE valor LIKE ‘valor’

Exemplo:

SELECT * FROM clientes WHERE pnome LIKE ‘Ana’

Wildcards

- Wildcards são caracteres especiais que podem ser usados em comparações de strings .

Para formar padrões de pesquisa.

Wildcard	Significado
%	Qualquer conjunto de zero ou mais caracteres
_	Um caracter qualquer

Problema:

Selecionar apenas os clientes cujo nome começa com A.

SELECT * FROM clientes WHERE pnome LIKE ...

Ordenacão – ORDER BY

Os dados presentes numa base de dados não são apresentados numa ordem particular, normalmente são apresentadas pela ordem na qual foram inseridos.

A ordenação dos resultados é feita pela cláusula ORDER BY.

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, ..., Tabelan

[WHERE condição]

ORDER BY Campo [ASC|DESC]

Onde:

– ASC indica que a ordenação será ascendente e DESC descendente.

– Por defeito, a ordenação é ascendente

Ordenação – ORDER BY

Exemplo: Seleccionar todos os clientes, ordenando o resultado por idade.

```
SELECT * FROM clientes ORDER BY idade
```

ID	Nome	Morada	Idade	Gen
69	António	Viana	NULL	M
45	João	Viana	18	M
142	Ana	Prenda	22	F
18	Manuel	Cazenga	23	M

Ordenacão – ORDER BY

Problema: Seleccionar todos os clientes, ordenando o resultado por morada e idade.

SELECT * FROM clientes ORDER BY ...

ID	Nome	Morada	Idade	Gen
18	Manuel	Cazenga	23	M
142	Ana	Prenda	22	F
69	António	Viana	NULL	M
45	João	Viana	18	M

Predicados

Para resolver algumas situações como: eliminação de dados duplicados, limite de registos apresentados, a linguagem SQL dispõe de alguns predicados que são associados a cláusula SELECT.

Exemplo: Mostrar as 3 encomendas com maior valor.

```
SELECT * FROM encomendas ORDER BY valor DESC LIMIT 3
```

Exemplo: Mostrar as 5 ultimas encomendas que foram canceladas.

```
SELECT * FROM encomendas where estado = 0 ORDER BY id desc limit 5;
```

Funções de agregação

As funções de agregação retornam um único valor, calculado com base nos valores presentes num campo.

Função	Descrição
COUNT	Devolve o número de linhas
MAX	Devolve o maior valor de uma coluna
MIN	Devolve o menor valor de uma coluna
SUM	Devolve a soma de todos os valores da coluna
AVG	Devolve a média de todos os valores da coluna

Funções de agregação

Sintaxe:

SELECT FUNÇÂO(Campo) FROM Tabela

Exemplo1: Qual o preço mais alto existente na tabela de produtos?

SELECT MAX(preco) FROM produtos

Exemplo2: Qual é o preço médio dos produtos?

SELECT AVG(preco) FROM produtos

ALIAS - Pseudônimo

- É possível associar um nome temporário a um determinado campo (coluna) ou tabela representada no resultado do comando SELECT.
- Isto é feito através da cláusula AS.

Exemplo3: Qual o preço mais baixo existente na tabela de produtos?

```
SELECT MIN(preco) AS MenorPreço FROM produtos
```

Problema1: Quantos clientes tem a tabela cliente?

```
SELECT ...
```

Problema2: Qual o total da encomenda 1?

```
SELECT ...
```

Agrupamento – GROUP BY

- Para o tratamento de informação agrupada, SQL disponibiliza o comando GROUP BY.
- A cláusula GROUP BY divide o resultado de um SELECT em grupos de resultados.

Características:

– Todos os atributos presentes no SELECT têm que estar presentes na cláusula GROUP BY;

– Os registos são processados em grupos de características semelhantes;

– Usa-se em conjunto com as funções de agregação.

Sintaxe: SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2,...,Tabelan

[WHERE condição][GROUP BY Campo1, Campo2,...,Campon]

Agrupamento – GROUP BY

Exemplo: Contar o número de Encomendas por estado.

SELECT estado, COUNT(id) FROM encomendas GROUP BY estado

Problema: Seleccionar todas as Encomendas agrupando por ID do cliente. SELECT ...

Consultas em várias tabelas

Em base de dados a ligação de tabelas permite extrair através de um único comando SELECT dados contidos em várias tabelas distintas.

A ligação entre tabelas é feita por meio de chaves estrangeiras e as respectivas chaves primárias. Colocando-se na cláusula FROM o conjunto de tabelas que se pretende ligar.

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2, ..., Tabelan

É necessário ainda fazer a ligação por meio de algum campo pelo qual as diferentes tabelas estão relacionadas, i.e. Igualdade entre dois campos. Esta ligação é feita na cláusula WHERE e chama-se equi-junção (equi-join).

Consultas em várias tabelas

SELECT Campo1, Campo2, ..., Campon FROM Tabela1, Tabela2

WHERE Tabela1.chave_estrangeira = Tabela2.chave_primária

- Caso não se coloque a cláusula WHERE obtém-se o produto cartesiano das tabelas, i.e, associa cada linha presente numa tabela com o conjunto de linhas presentes nas outras tabelas indicadas na cláusula FROM. Se executarmos a instrução:

SELECT * FROM clientes, encomendas

Consultas em várias tabelas

O resultado será o produto cartesiano das duas tabelas. Onde o número total de registos apresentados será Número de clientes x Número de encomendas. Pois, que para cada registo presente na tabela cliente será associado a cada registo presente na tabela encomendas.

- É o que se pretende e saber as encomendas associadas a cada cliente. Nota- se que as duas tabelas estão relacionadas através do campo ID do Cliente, então, deve-se colocar na cláusula WHERE a condição de ligação.

```
SELECT * FROM clientes, encomendas
    WHERE id = id
```

Ambiguidade de campos

É comum existir em tabelas diferentes campos com o mesmo nome. E quando se junta dados de várias tabelas, o SGBD não tem como saber a que campo a instrução se refere, criando assim um erro de ambiguidade.

```
SELECT * FROM clientes, encomendas
WHERE id = id
```

Para tratar a questão da ambiguidade deve-se: Distinguir os campos informando o nome da tabela.

```
SELECT * FROM clientes, encomendas
WHERE clientes.id = encomendas.id_cliente
```

Junções -

Uma outra forma de se obter a ligação de tabelas é através das junções. Que podem ser internas ou externas.

- Junção interna – são apresentados registos em que exista ligação entre as tabelas . I.e, faz a ligação entre duas tabelas considerando apenas dados em comum.

– INNER JOIN

- Junção externa – estende o conjunto de registos obtidos. I.e, faz a ligação entre tabelas considerando dados em comum e exclusivos também.

– RIGHT OUTER JOIN

– LEFT OUTER JOIN

Junção interna – INNER JOIN

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1 INNER JOIN Tabela2 ON

Tabela1.chave_estrangeira = Tabela2.chave_primária

Exemplo:

SELECT * FROM clientes INNER JOIN encomendas ON clientes.id = encomendas.id_cliente.

Problema: Seleccionar todas as encomendas feitas e seus respectivos detalhes.

SELECT...

Junção externa– OUTER JOIN

RIGHT JOIN (junção à direita) – são considerados todos os registos da tabela à direita (da cláusula) e apenas os registos correspondentes da tabela à esquerda.

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1 RIGHT JOIN Tabela2 ON Tabela1.chave_estangeira = Tabela2.chave_primária

LEFT JOIN (junção à esquerda) - são considerados todos os registos da tabela à esquerda (da cláusula) e apenas os registos correspondentes da tabela à direita.

Sintaxe:

SELECT Campo1, Campo2, ..., Campon FROM Tabela1 LEFT JOIN Tabela2 ON Tabela1.chave_estangeira = Tabela2.chave_primária

Junção externa– OUTER JOIN

RIGHT JOIN

SELECT * FROM clientes RIGHT JOIN encomendas ON clientes.id = encomendas.id_cliente

Considera-se todos os registos da tabela encomenda (tabela à direita) e as respectivas correspondências na tabela cliente (tabela à esquerda). O resultado será igual ao INNER JOIN pois, para cada ocorrência de encomendas existe uma correspondência de um cliente, porque não existe

Junção externa– OUTER JOIN

LEFT JOIN

SELECT * FROM Cliente LEFT JOIN

Encomenda ON clientes.id = encomendas.id_cliente

Considera-se todos os registos databela cliente (tabela à esquerda) e as respectivas correspondências na tabela encomendas (tabela à direita). Se não existir correspondências na tabela à direita os valores são preenchidos a NULL.

Subconsultas

Uma subconsulta ou subquery consiste nada mais do que uma instrução SQL dentro de outra instrução SQL.

– Também designadas por subinterrogações ou consultas aninhadas.

A subconsulta pode ser colocada:

– No comando SELECT associadas as cláusulas SELECT, FROM, WHERE;

– Nos comandos INSERT, UPDATE e DELETE;

– Dentro de outra subconsulta.

Características

- **Características**

1-Existe uma consulta interior e outra exterior;

2-A consulta interior deverá ser colocada entre parêntesis curvos ();

3- As subconsultas podem devolver, como resultado, uma ou mais linhas e uma ou mais colunas. Se devolver mais do que uma coluna estas têm que estar na mesma ordem das colunas que aparecem na condição;

4-Múltiplos níveis de consultas.

Exemplo

Tendo em conta a tabela seguinte. Seleccionar o ID da encomenda de valor mais baixo.

ID_Encomenda	ID_Cliente	Estado	Valor
100	69	Pendente	15000
101	18	Em processamento	30000
102	142	Cancelada	12500
103	18	Processada	5000
104	18	Processada	18500
105	142	Processada	20000
106	142	Em processamento	55000

Subconsultas

O problema com subconsultas resolve-se em dois passos:

1. Encontrar o valor mais baixo;

2. Encontrar a encomenda com valor mais baixo.

Para a resolução do primeiro problema seria:

```
SELECT MIN(Valor) FROM encomendas
```

Para a resolução do segundo seria:

```
SELECT id FROM encomendas WHERE valor = 5000
```

Subconsultas

Com o uso de subconsultas, pode-se resolver o mesmo problema com apenas uma única instrução SQL.

```
SELECT id FROM encomendas WHERE Valor = (SELECT MIN(valor)
FROM encomendas)
```

Resultado:

ID_Encomenda
103

Cláusulas IN e EXISTS

IN - Permite verificar se um determinado valor existe no conjunto de resultados de uma subconsulta.

– Existe uma relação: Elemento pertence ao conjunto.

– Pode-se usar em conjunto com a cláusula NOT.

Sintaxe:

Expressão [NOT] IN (Subconsulta)

EXISTS - tem como objectivo verificar se a execução da subconsulta resultou ou não alguma linha.

– Pode ser usada em conjunto com a cláusula NOT.

– É um operador Unário.

Sintaxe:

Consulta [NOT] EXISTS (Subconsulta)

Exemplo - IN

Tendo em conta as tabelas, Seleccionar os clientes (id e pnome) que têm as suas encomendas em processamento.

ID_Encomenda	ID_Cliente	Estado	Valor
100	69	Pendente	15000
101	18	Em processamento	30000
102	142	Cancelada	12500
103	18	Processada	5000
104	18	Processada	18500
105	142	Processada	20000
106	142	Em processamento	55000

ID	Nome	Morada	Idade	Gen
142	Ana	Prenda	22	F
45	João	Viana	18	M
18	Manuel	Cazenga	23	M
69	António	Viana	NULL	M

Exemplo - IN

Resolução:

SELECT id, pnome FROM clientes WHERE id IN (SELECT id_cliente FROM encomendas WHERE estado LIKE 'Em processamento')

Resultado:

ID	Nome
142	Ana
18	Manuel

Exemplo - EXISTS

Tendo em conta as tabelas, seleccionar os clientes (ID e Nome) que não fizeram nenhuma encomenda.

ID_Encomenda	ID_Cliente	Estado	Valor
100	69	Pendente	15000
101	18	Em processamento	30000
102	142	Cancelada	12500
103	18	Processada	5000
104	18	Processada	18500
105	142	Processada	20000
106	142	Em processamento	55000

ID	Nome	Morada	Idade	Gen
142	Ana	Prenda	22	F
45	João	Viana	18	M
18	Manuel	Cazenga	23	M
69	António	Viana	NULL	M

Exemplo - EXISTS

Resolução:

```
SELECT ID, Nome FROM Cliente WHERE NOT EXISTS (SELECT * FROM Encomendas
WHERE ID_Cliente = ID)
```

Resultado:

ID	Nome
45	João

Exercícios

1- Selecionar o ID_Encomenda das encomendas feitas ao produto mais caro.

– Use o operador IN para realizar a consulta.

2- Selecionar o Nome do funcionário que processou a encomenda de valor mais elevado.

3- Selecionar todos os fornecedores que não estão associados a nenhuma compra.

– Use o operador EXISTS para realizar a consulta.

4- Selecionar o subtotal das encomendas feitas a produtos pertencentes a mesma categoria que

o produto de ID_Produto = 7.

INSTITUTO DE TELECOMUNICAÇÕES

MUITO OBRIGADO!