Inatel
Instituto Nacional de Telecomunicações

BANCOS DE DADOS

Cap.4 - SQL (Pt.3)

Prof. MSc. Renzo P. Mesquita
renzo@inatel.br

Objetivos

- Começarmos a compreender recursos fundamentais para realização de consultas nos bancos SQL;
- Explorarmos detalhes de utilização do comando SELECT, pertencente à subfamília DQL (Data Query Language);

Capítulo 4
SQL (Pt.3)

4.10. A Subfamília DQL;
4.11. SELECT simples;
4.12. SELECT com duas tabelas;
4.13. SELECT com autorrelacionamento;
4.14. SELECT com produto cartesiano;
4.15. SELECT em valores distintos (DISTINCT);
4.16. SELECT com textos (LIKE);
4.17. SELECT com operadores aritméticos;
4.18. SELECT com ordenação (ORDER BY);
4.19. SELECT com funções prontas do SQL;

4.10. A Subfamília DQL

Os comandos SQL que lidam com a consulta de dados presentes dentro das tabelas dos Bancos de Dados pertencem à subfamília DQL (Data Query Language).

A Sintaxe básica para realização de consultas é a seguinte:

SELECT <lista de atributos>

FROM <lista de tabelas>

WHERE <condição>

<lista de atributos> - nome dos atributos, separados por vírgula, cujos valores serão recuperados pela consulta;

<lista de tabelas> - nome da(s) tabela(s) necessária(s) para a consulta;

<condição> - expressão condicional que filtra os registros que serão recuperados pela consulta.

4.11. SELECT Simples

Geralmente utilizado para quando queremos recuperar todos os campos ou apenas campos específicos de um registro de uma única tabela.

FORMA 1 - RECUPERANDO TODOS OS CAMPOS DE UM REGISTRO

```
SELECT *
  FROM Empregado
WHERE SSN = 514800938;
```

Observe o * atuando como um coringa para equivaler a todos os campos.

FORMA 2 - RECUPERANDO CAMPOS ESPECÍFICOS DE UM REGISTRO

```
SELECT PNome, DataNasc, Salario
  FROM Empregado
WHERE SSN = 514800938;
```

Observe a vírgula (,) separando os campos que se deseja buscar.

4.12. SELECT com duas tabelas

É mais que comum realizarmos buscas que possam trazer dados de mais de uma tabela ao mesmo tempo. Porém, é importante ressaltar que as tabelas envolvidas devem se relacionar de alguma forma.

FORMA - RECUPERANDO CAMPOS ESPECÍFICOS DE DUAS TABELAS

```
SELECT PNome, DNome
  FROM Empregado,Departamento
WHERE DNumero_Departamento = DNumero;
```

- A parte em destaque é a protagonista desta consulta;
- Veja que os campos que estão sendo igualados são aqueles que indicam o relacionamento entre as tabelas envolvidas;
- Isso obrigatoriamente deve ser feito para que os resultados sejam retornados de forma adequada (Faça um teste sem o que está no WHERE para ver o que acontece);
- No exemplo, mostraremos o nome de cada empregado e seu respectivo departamento.

4.13. SELECT com autorelacionamento

Como já vimos, tabelas podem se relacionar consigo mesmas. Quando realizamos uma consulta em um autorrelacionamento, a mesma tabela deve ser tratada como duas tabelas diferentes.

FORMA - RECUPERANDO DADOS DE UM AUTORRELACIONAMENTO

```
SELECT  emp.PNome, sup.PNome
  FROM  Empregado AS emp, Empregado AS sup
 WHERE  emp.SSN_Supervisor = sup.SSN;
```

- Observe a tabela Empregado sendo tratada de duas formas diferentes por meio da palavra AS;
- AS é chamado de ALIAS, ou apelido. Ela é largamente utilizada quando queremos renomear momentaneamente uma tabela;
- Observe que, com esta estratégia, podemos tratar uma mesma tabela como se fosse duas tabelas diferentes, e relacioná-las normalmente;
- No exemplo, mostraremos o nome de cada empregado e seu respectivo supervisor;

4.14. SELECT com produto cartesiano

Usado em situações em que queremos simular as possiveis combinações de registros de duas ou mais tabelas.

FORMA - REALIZANDO O PRODUTO CARTESIANO ENTRE REGISTROS DE TABELAS

```
SELECT *
  FROM Empregado, Projeto;
```

- Observe a falta do WHERE na consulta;
- O produto cartesiano fará a combinação de cada linha de uma tabela, com todas as linhas da outra tabela;
- Veja que toda consulta que envolva duas ou mais tabelas, naturalmente já realiza um produto cartesiano (igual vimos no tópico 4.12). O que muda é o filtro na cláusula WHERE, que retira combinações desnecessárias;
- No exemplo estamos buscando a combinação do registro de cada empregado com cada um dos projetos cadastrados. Aqui usamos o * para pegar todos os campos de cada registro, mas poderíamos também buscar apenas os campos necessários para combinarmos.

4.15. SELECT em valores distintos (DISTINCT)

Dentro de uma tabela, podemos ter colunas que geralmente possuem valores repetidos. E às vezes, em uma consulta, queremos listar apenas os diferentes valores e não mostrar repetições.

FORMA - MOSTRANDO APENAS DIFERENTES VALORES DE UMA PESQUISA

```
SELECT DISTINCT SSN_Supervisor
FROM Empregado;
```

- Um SELECT sem DISTINCT pode trazer resultados repetidos;
- No exemplo não usamos o WHERE, mas o mesmo também poderia ser usado para uma consulta mais precisa;
- Sabemos que uma empresa pode ter vários supervisores, mas no exemplo, estamos trazendo apenas os diferentes SSN dos supervisores dos Empregados, sem repetí-los;
- Veremos em breve outras aplicações interessantes deste operador.

4.16. SELECT com textos (LIKE)

Muitas vezes é necessário criarmos pesquisas que obedeçam a condições com textos específicos ou padrões de caracteres.

FORMA 1 - PESQUISA COM TEXTO ESPECÍFICO

```
SELECT *
  FROM Projeto
WHERE PNome = "Projeto X";
```

FORMA 2 - PESQUISA SE BASEANDO EM UM PADRÃO DE CARACTERES

```
SELECT PNome
  FROM Empregado
WHERE Endereco LIKE "%Santa Rita%";
```

- Observe o uso do LIKE juntamente com o % para busca de padrões de caracteres;
- O % significa qualquer número de caracteres;
- Poderíamos também usar o __ (undescore) para representar um único caracter qualquer;
- No exemplo estamos buscando qualquer endereço que possui "Santa Rita" no meio.

4.17. SELECT com operadores aritméticos

Uso de operadores aritméticos podem ser aplicados com atributos de domínios numéricos para simularmos resultados.

FORMA - UTILIZANDO OPERADORES ARITMÉTICOS NAS PESQUISAS

```
SELECT PNome, Salario*1.25 AS 'Novo Salario'
  FROM Empregado
WHERE Sexo = 0 AND Salario <= 3000;
```

- É importante ressaltar que, como se trata de uma consulta, o novo valor do Salário não será modificado, mas apenas simulado;
- Veja que o AS também pode ser utilizado para darmos um novo nome para a coluna que terá seu valor modificado (o uso do AS neste caso não é obrigatório);
- Observe o uso também de operadores lógicos (AND, OR, NOT, etc..) para se criar pesquisas mais precisas;
- No exemplo, está sendo simulado como seria um aumento de 25% no salário das mulheres que recebem 3000 ou menos.

4.18. SELECT com ordenação (ORDER BY)

A fim de melhor visualizarmos alguns resultados, às vezes precisamos mostrar ao SQL o campo pelo qual ele precisará utilizar como referência para ordená-los, seja em ordem crescente ou decrescente.

FORMA - ORDENANDO OS RESULTADOS DE UMA PESQUISA EM ORDEM DECRESCENTE

```
SELECT   SSN_Empregado, Nome_Dependente
FROM     Dependente
WHERE    Parentesco = "Filho"
ORDER BY SSN_Empregado DESC;
```

- Veja que no exemplo, além de selecionarmos o campo pelo qual queremos ordenar a apresentação dos resultados (SSN_Empregado), também indicamos que queremos vê-los em ordem decrescente (DESC);
- Por padrão, quando não colocamos o DESC, os resultados são ordenados em ordem crescente do campo (ASC);
- A ordenação também funciona com textos e datas;

4.19. SELECT com funções prontas do SQL

A maioria dos bancos SQL já oferecem a possibilidade de utilizarmos de uma coletânea de funções pré-prontas para realização de consultas específicas.

FORMA - USANDO DE FUNÇÕES PRÉ-PRONTAS

```
SELECT  MIN(horas), MAX(horas)
  FROM  Trabalha_Em
 WHERE  PNumero_Projeto = 5;
```

FORMA - CONTANDO REGISTROS

```
SELECT  COUNT (DISTINCT PLocalizacao)
  FROM  Projeto
```

- Outras funções pré-prontas importantes são AVG (média), SUM (soma), entre outras;
- IMPORTANTE: as funções trabalham com um conjunto de valores, e retornam apenas um único resultado cada;
- Nos exemplos, a primeira consulta retorna o menor e maior número de horas que trabalham no Projeto 5 e a segunda conta em quantas localizações diferentes os projetos estão acontecendo;
- Também podemos criar nossas próprias funções por meio de FUNCTIONS.

4.19. SELECT com funções prontas do SQL

Exercício Proposto

Crie as seguintes consultas propostas sobre o Esquema abaixo:

EMPREGADO

PNome	MInicial	UNome	SSN	DataNasc
Endereco	Sexo	Salario	SSN_Supervisor	DNumero_Departamento

DEPARTAMENTO

DNome	DNumero	SSN_Empregado	DataInicio

DEPTO_LOCALIZACOES

LNumero	DLocalizacao	DNumero_Departamento

PROJETO

PNome	PNumero	PLocalizacao	DNumero_Departamento

TRABALHA_EM

SSN_Empregado	PNumero_Projeto	Horas

DEPENDENTE

SSN_Empregado	Nome_Dependente	Sexo	DataNasc	Parentesco

1) Busque os diferentes tipos de parentescos cadastrados na tabela Dependente;

2) Busque o último nome e data de nascimento de todos os empregados que são do sexo masculino e ordene os resultados pela data de nascimento dos mais velhos para os mais novos;

3) Busque o nome de cada dependente juntamente com o primeiro nome do seu responsável;

4) Busque a média salarial de todos os empregados da empresa que residem na cidade de São Paulo e trabalham no departamento de Engenharia;

5) Para cada supervisor, busque seu primeiro nome, o primeiro nome dos empregados que ele gerencia e a diferença salarial do salario dele para cada um de seus empregados;

6) Para cada empregado, busque seu primeiro nome, o nome dos projetos que ele trabalha e o número de horas em cada projeto.

FIM
DO
CAPÍTULO 4
(Parte 3)