SQL Complexos

CONSULTA EM MAIS DE UMA TABELAS - JOINS

Comando Select – Mostrando dados de mais de uma tabela

- Através do comando *Select* podemos mostrar dados de uma ou mais tabelas, ou seja, podemos "juntar" dados de várias tabelas diferentes e mostrar como se fossem uma só.

- A este processo se dá o nome de "join (junção)" , sendo que a tabelas que fazem parte desta junção tem que estar relacionadas entre si

Comando Select – Mostrando dados de mais de uma tabela

- Síntaxe:

  SELECT *Tabela1.Campo1, Tabela2.Campo2*

  FROM *Tabela1, Tabela2*

  WHERE *(Tabela1.CampoRelacionado1 = Tabela2.CampoRelacionado2)*

Comando Select – Mostrando dados de mais de uma tabela

- Exemplo: mostrar a data da venda, número venda, código cliente e nome cliente (duas tabelas: Clientes e Vendas)

SELECT *Vendas.numvenda, Clientes.codcliente, vendas.datavenda,*
Clientes.nomecliente
FROM *Vendas, Clientes*
WHERE *(Vendas.codcliente = Clientes.codcliente)*

Mostrando dados de mais de uma tabela - ordenando

- Exemplo: mostrar a data da venda, número venda, código cliente e nome cliente (duas tabelas: Clientes e Vendas), ordenados pela Data da Venda

```
SELECT Vendas.numvenda,
vendas.codcliente, vendas.datavenda,
      Clientes.nomecliente
FROM Vendas, Clientes
WHERE (Vendas.codcliente =
Clientes.codcliente)
ORDER BY Vendas.datavenda
```

Operadores para filtrar informações – cláusula *Where*

- Podemos filtrar informações através de mais de uma condição, usando a cláusula Where vista anteriormente.
- Para isto usamos os operadores de condição listados a seguir:
- ***AND (E):*** filtra quando duas ou mais condições forem verdadeiras
- ***OR (OU):*** filtra quando uma as condições forem verdadeiras
- ***NOT (NÃO):*** nega, ou seja, inverte o valor da condição.

Mostrando dados de mais de uma tabela - filtrando

- Exemplo: mostrar a data da venda, número venda, código cliente e nome cliente (duas tabelas: Clientes e Vendas), mas somente dos clientes de São Paulo

SELECT *Vendas.numvenda, vendas.codcliente, vendas.datavenda, Clientes.nomecliente*
FROM *Vendas, Clientes*
WHERE *(Vendas.codcliente = Clientes.codcliente)*
AND (Clientes.Estado = 'SP')

Mostrando dados de mais de uma tabela – filtrando e ordenando

- Exemplo: mostrar a data da venda, número venda, código cliente e nome cliente (duas tabelas: Clientes e Vendas), mas somente dos clientes de São Paulo ordenados pelo nome do cliente

SELECT *Vendas.numvenda, vendas.codcliente, vendas.datavenda, Clientes.nomecliente*
FROM *Vendas, Clientes*
WHERE *(Vendas.codcliente = Clientes.codcliente)*
AND (Clientes.Estado = 'SP')
ORDER BY *Clientes.NomeCliente*

Atividades – bdlista 1

- Exiba o nome e salário dos vendedores e o prazo de entrega referente aos pedidos feitos pelos mesmos.
- Exiba o nome, endereço completo dos clientes e o prazo de entrega cuja seus pedidos têm prazo de entrega urgente (10 ou menos dias).
- Quais produtos fazem parte do pedido 138. Exiba suas respectivas descrições, valores unitários, a quantidade e o código do pedido.
- Qual outra situação (consulta) poderia se utilizar uma junção?

SQL Complexos

SUBSELECTS

SubSelects

- Este tipo de SQL proporciona um mecanismo para o aninhamento de subconsultas.
- Uma subconsulta é uma expressão select-from-where aninhada dentro de outra consulta.
- O conectivo ***IN*** testa se uma tupla é membro ou não de uma relação. Já o *NOT IN* verifica a ausência de membros em uma relação.

Sintaxe:

```
SELECT Campo1
FROM Tabela1
WHERE Campo1 IN (  SELECT Campo2
                   FROM Tabela2  );
```

Exemplificando

Para exemplificar será utilizado o seguinte Modelo:

Banco de Dados: BANCO

Tabelas:

agencia = {id_age, nome_age, cidade_age, fundos_age}
cliente = {id_cli, nome_cli, rua_cli, cidade_cli}
emprestimo = {numero_emp, valor_emp, id_age_emp}
devedor = {id_cli_dev, numero_emp_dev}
conta = {numero_con, id_age_con, saldo_con}
depositante = {id_cli_dep, numero_con_dep}

SubSelects

Considere a seguinte consulta:

- Encontre todos os clientes que tenham tanto conta quanto empréstimo no banco

Inicialmente encontramos todos os cliente que tenham conta:

```
SELECT id_cli_dep
FROM depositante;
```

SubSelect

- **Agora** precisamos encontrar todos os cliente que contraíram empréstimo do banco:

```
SELECT id_cli_dev
FROM devedor;
```

SubSelect

Para saber quem tem conta e ao mesmo tempo recorreu a empréstimo é só aninhar as consultas:

```
SELECT DISTINCT id_cli_dep
FROM depositante
WHERE id_cli_dep IN (SELECT id_cli_dev
                     FROM devedor);
```

SubSelect

Já Para saber:

- Quais clientes possuem empréstimo mas não tenham conta no banco

```
SELECT id_cli_dev
        FROM devedor
        WHERE id_cli_dev NOT IN (SELECT id_cli_dep
                                 FROM depositante);
```

Atividades

1. Encontre os nomes de todas as agências que tenham fundos maiores que ao menos uma agência localizada em Fernandópolis.

Respostas

```
SELECT nome_age
FROM agencia
WHERE fundos_age > (SELECT fundos_age
                    FROM agencia
                    WHERE cidade_age='Fernandópolis')
```

ou

```
SELECT t.nome_age
FROM agencia t, agencia s
WHERE t.fundos_age> s.fundos_age
        AND s.cidade_age = ‘Fernandópolis’
```

ANY e ALL

- ANY: Retorna as linhas que satisfaçam a comparação sobre **qualquer** valor do conjunto

- Exemplo: Encontre as agências que possuem fundos maiores que qualquer agência da cidade de São Carlos

```
SELECT nome_age
FROM agencia
WHERE fundos_age > ANY  (SELECT fundos_age
                         FROM agencia);
```

ALL

- ALL: Retorna as linhas que satisfaçam a comparação sobre todos valores do conjunto.

- Exemplo: Encontre os números e os valores dos empréstimos cujo valores são maiores do que todas as contas

```
SELECT numero_emp
FROM emprestimo
WHERE valor_emp > ALL (SELECT saldo_con
                       FROM conta);
```

SQL Complexos

INNER JOIN

OUTER JOIN

O QUE É UM JOIN?

- As junções SQL são utilizadas quando precisamos selecionar dados de duas ou mais tabelas.
- Existem as junções com estilo non-ANSI (junção com WHERE).
- E as junções ANSI join (com JOIN).
  - Estas podem ser de dois tipos, as INNER JOINS e as OUTER JOINS.
  - A padrão é a INNER JOIN. Ela pode ser escrito com apenas JOIN.

Tipos

1. **Inner Join:** É a junção simples entre duas ou mais tabelas e retorna as linhas que satisfazem a condição de junção.
2. **Outer Join:** Retorna as linhas satisfazem a condição de junção além de todas as linhas de uma tabela pertencente a junção. É dividido em 3tipos:
    1. Left Outer Join: Retorna as linhas satisfazem a condição de junção além de todas as linhas da tabela da esquerda.
    2. Right Outer Join: Retorna as linhas que satisfazem a condição de junção além de todas as linhas da tabela da direita.
    3. Full Outer Join: Retorna as linhas que satisfazem a condição de junção além de todas as linhas de ambas as tabelas.

INNER JOIN

- Encontre o nome do cliente e o número de suas respectivas contas

```
select c.nome_cli, d.numero_con_dep
from cliente c, depositante d
where c.id_cli = d.id_cli_dep;
```

INNER:

```
select c.nome_cli, d.numero_con_dep
from cliente c
~~INNER~~ JOIN depositante d
on c.id_cli = d.id_cli_dep;
```

OUTER JOIN: Left Outer Join

- O gerente Pedro pediu para você uma relação contendo o nome de todos os clientes, suas respectivas cidades e suas respectivas contas. Os clientes que fizeram empréstimo, ou seja, que não possuírem conta mas sim empréstimos também devem ser mostrados.

```
select c.nome_cli, c.cidade_cli, d.numero_con_dep
from cliente c
LEFT OUTER JOIN depositante d
on c.id_cli=d.id_cli_dep;
```

OUTER JOIN: Left Outer Join

- O vendedor José da Silva quer um relatório com a descrição de todos os produtos que custam mais de R$1.00 e o número de todos os pedidos que estes estão. (Lista1)

```
select p.descricao, it.num_pedido
from produto p
LEFT OUTER JOIN item_pedido it
on p.codigo_produto= it.codigo_produto
where p.val_unit> 1;
```

Mais...

- Manual PostgreSQL pt

http://www.postgresql.org.br/docs

- Manual PostgreSQL en

http://www.postgresql.org/docs/9.0/static/