VOLTAR

Criação de relatórios utilizando mais de uma tabela

Criar relatórios a partir de dados existentes no Banco de dados utilizando mais de uma tabela a partir dos relacionamentos.

NESTE TÓPICO

Marcar tópico

Em determinadas situações, apenas exibir dados existentes em uma tabela não é satisfatório para a leitura de um relatório, visto que os funcionários ou clientes de uma empresa não têm obrigação de conhecer os dados em detalhes. Portanto, o programador de banco de dados deve elaborar rotinas e instruções para as estruturas, para que seja possível gerar relatórios simples e de fácil entendimento a todos envolvidos na empresa.

Para criar tais rotinas e instruções, é necessário complementar ou buscar estes dados em tabelas diferentes, criar uma instrução que consiga acesso em outras tabelas e isso ocorre através do relacionamento que é fundamental para a consistência dos dados. Por exemplo, se quisesse saber o cargo de cada funcionário ou simplesmente o que um determinado funcionário possui, se não usar o conhecimento de junção, ou na prática "JOIN" das tabelas, teria um trabalho enorme. Dessa forma, vamos criar um exemplo de acordo com a seguinte estrutura:

Tabela Cargo:

```
1.  CREATE TABLE cargo(  CodCargo char(2) primary key, NomeCargo varchar(10) not null, V
    alorCargo number(6,2) not null);
```

Dados para a tabela cargo:

```
1.  INSERT INTO cargo VALUES('C1','CAIXA',  800.00);
2.  INSERT INTO cargo VALUES ('C2','VENDEDOR', 1200.00);
3.  INSERT INTO cargo VALUES ('C3','GERENTE' , 2400.00);
```

Tabela Funcionário:

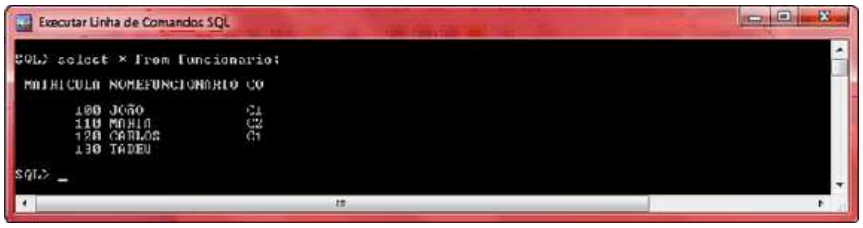
```
1.  CREATE TABLE funcionario (Matricula number(3) primary key, NomeFuncionario varchar(1
    5) NOT NULL, CodCargo char(2), FOREIGN KEY (CodCargo) REFERENCES CARGO)
```

Dados para a tabela Funcionário:

```
1.  INSERT INTO funcionario VALUES (100, 'JOÃO'   , 'C1');
2.  INSERT INTO funcionario VALUES (110, 'MARIA'  , 'C2');
3.  INSERT INTO funcionario VALUES (120, 'CARLOS' , 'C1');
4.  INSERT INTO funcionario VALUES (130, 'TADEU'  , NULL);
```

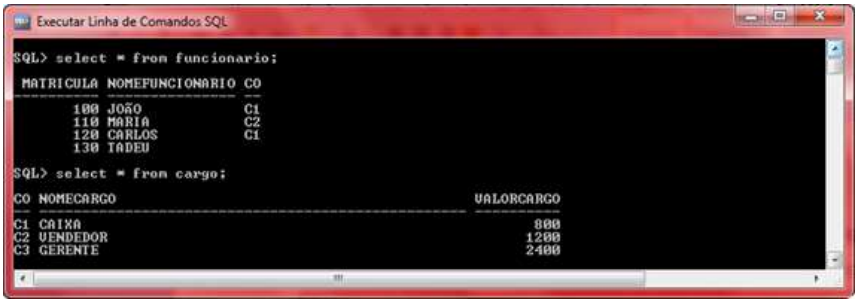
A instrução que resolveria o problema de mostrar o cargo ocupado por um ou por todos os funcionários sem o uso do recurso de junção *join* ficaria assim:

```
1.  SELECT * from funcionario;
```



Então, mostre este relatório ao seu superior ou deixe que ele seja exibido por um sistema, o que achou? Conseguiu saber o cargo de cada funcionário ou de um específico? Então, podemos realizar mais uma consulta:

```
1.  SELECT * from cargo;
```

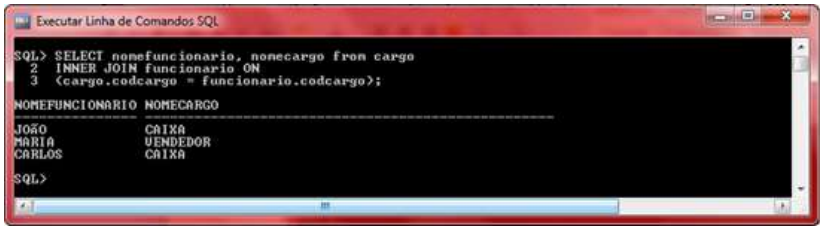


Agora sim, você conseguiu. Imagine se a tabela de funcionário tivesse uns 250 registros e a de cargo uns 15 cargos: seria interessante ficar movimentando sua tela com setas de direção ou Page *up* e *down*? Não seria nada bom, não é mesmo? Para elaborar um relatório que atenda as necessidades das tabelas, podemos usar uma opção de junção que é chamada

de INNER JOIN ou junção por equivalência. Neste caso, uma pesquisa seria executada em ambas as tabelas e através da verificação dos dados das chaves primária e estrangeira poderíamos mostrar na mesma instrução dados de tabelas diferentes, veja como ficaria:

```
1. SELECT nomefuncionario, nomecargo from cargo
2. INNER JOIN funcionario ON (cargo.codcargo = funcionario.codcargo);
```

Agora, veja este relatório:



O que achou? Melhor, não é?

Na instrução acima, a primeira mudança acontece após o nome da tabela em que informamos o tipo de junção que será realizada, INNER JOIN é o processo de equivalência, ou seja, só haverá exibição quando a comparação de igualdade for verdadeira, isso acontece após a ligação ON. A tabela também poderia ser desenvolvida da seguinte forma:

```
1. SELECT nomefuncionario, nomecargo from cargo, funcionario
2. WHERE cargo.codcargo = funcionario.codcargo;
```

Note que nas duas condições, antes do nome da coluna, foi colocado o nome da tabela, chamamos este processo de qualificador, mas por que usamos isso? Simplesmente porque a instrução identifica que uma mesma coluna está nas duas tabelas que serão usadas, logo, o mecanismo de banco de dados não sabe onde procurar. Neste momento devemos indicar a busca através do nome da tabela, assim nada não ficará perdido. E, também, para não interferirmos redigitando o nome de uma tabela, podemos colocar um apelido nela, por exemplo:

```
1. SELECT nomefuncionario, nomecargo from cargo c, funcionario f
2. WHERE c.codcargo = f.codcargo;
```

O uso do JOIN é a ligação que fazemos entre duas tabelas na pesquisa de dados e, necessariamente, deve existir em um JOIN a chave primária que estabelecerá a relação com uma chave estrangeira, esta é a condição e ligação.

Vamos imaginar outras situações, agora usaremos comandos mais complexos.

```
SELECT nomefuncionario, nomecargo from cargo
INNER JOIN funcionario ON (cargo.codcargo = funcionario.codcargo)
WHERE valorcargo > 800;
```

A instrução deve retornar apenas uma linha, pois o cargo de gerente que tem salário de R$ 2400,00 não está vinculado a nenhum funcionário, para que ele apareça é necessário outro JOIN.

Outro exemplo:

```
SELECT e.sobrenome,e.numdepto,d.nome FROM empregado e, departamento d
WHERE e.numdepto = d.numdepto;
```

A instrução procura pelo sobrenome, número do departamento e nome do departamento de todos os funcionários, realizando a busca nas tabelas "empregado" e "departamento", ou ainda:

```
SELECT e.sobrenome,e.numdepto,d.nome FROM empregado e
INNER JOIN departamento d ON ( e.numdepto = d.numdepto);
```

Exemplo:

```
SELECT num_pedido, nome_ven, nome_clie from pedido p
INNER JOIN vendedor v ON (p.cod_ven = v.cod_ven)
INNER JOIN cliente c ON (p.cod_clie – c.cod_clie);
```

Esta instrução mostra o número, cliente e vendedor que estão relacionados ao pedido.Por exemplo: exibir os pedidos com seus produtos, ordenados pelo número do pedido.

```
SELECT p.num_pedido, descricao, val_unit FROM item_pedido i
INNER JOIN pedido p ON (p.num_pedido = i.num_pedido)
INNER JOIN produto pr ON (pr.cod_prod = i.cod_prod)
ORDER BY p.num_pedido;
```

Mostrar produtos vendidos pelo João:

```
SELECT nome_ven, descricao FROM pedido p
INNER JOIN vendedor v ON (p.cod_ven = v.cod_ven)
INNER JOIN item_pedido I ON (p.num_pedido = i.num_pedido)
INNER JOIN produto pr ON (pr.cod_prod = i.cod_prod)
WHERE upper(nome_ven) = 'JOÃO'
```

Referências

BEIGHLEY, Lynn. *Use a Cabeça SQL*. Rio de Janeiro: Alta Books, 2008.

FANDERUFF, Damaris. *Dominando o Oracle 9i: Modelagem e Desenvolvimento,* São Paulo: Makron, 2003.

GRAVES, Mark. *Projeto de banco de dados com XML*. São Paulo: Pearson, 2003.

MORELLI, Eduardo Terra. *Oracle 9i Fundamental: SQL, PL/SQL e Administração,* São Paulo, Editora Érica, 2002.

PRICE, Jason. *Oracle Database 11g SQL*. (tradução: João Eduardo Nóbrega Tortello). Porto Alegre: Bookman, 2009.

SILVA, Robson. *Oracle Database 10g Express Edition*. São Paulo: Editora Érica, 2007.

Avalie este tópico

ANTERIOR
Criação de relatórios utilizando filtros, operadores e funções em banco de dados

Índice

Criação de relatórios utilizando mais de uma tabela

PRÓXIMO