02

Juntando dados de várias tabelas

Compras sem um comprador? Toda compra tem um comprador!

Vamos agora guardar, além dos dados da compra, os dados da pessoa que fez a compra. Podemos criar campos como `NOME`, `ENDERECO`, `TELEFONE` e etc, na tabela `COMPRAS`. Mas essa abordagem nos trará problemas: e se a pessoa mudar de endereço? Precisaremos atualizar esses dados em todas as linhas da tabela daquela pessoa? Essa operação pode ser cara.

Uma outra abordagem seria criar então uma tabela para cada pessoa. Por exemplo, tabela `COMPRAS_DO_MAURICIO`, `COMPRAS_DO_ADRIANO`, `COMPRAS_DO_RICARDO`, e etc. Mas a solução também não é a mais elegante: a cada novo comprador, precisaríamos criar uma nova tabela.

Vamos primeiro criar uma tabela para representar todas as pessoas que já fizeram uma compra:

```
CREATE TABLE COMPRADORES (
    ID NUMBER PRIMARY KEY,
    NOME VARCHAR2(30) NOT NULL,
    ENDERECO VARCHAR2(30) NOT NULL,
    TELEFONE VARCHAR2(20) NOT NULL
);
```

E vamos criar a sequência para incrementar o `ID` de `COMPRADORES`:

```
CREATE SEQUENCE ID_COMPRADORES_SEQ;
```

Faça o teste. Insira 2 compradores na tabela:

```
INSERT INTO COMPRADORES (ID, NOME, ENDERECO, TELEFONE) VALUES (ID_COMPRADORES_SEQ.NEXTVAL, 'FLA
INSERT INTO COMPRADORES (ID, NOME, ENDERECO, TELEFONE) VALUES (ID_COMPRADORES_SEQ.NEXTVAL, 'NIC
```

Referenciando a compra ao seu comprador

O problema agora é referenciar uma compra para um comprador. Sabemos que para identificar um comprador em específico podemos usar seu `ID`. Se uma compra pertence a um comprador, basta que a tabela `COMPRAS` tenha uma coluna que guarda o `ID` do comprador. Vamos criar essa coluna:

```
ALTER TABLE COMPRAS ADD (COMPRADOR_ID NUMBER);
```

Veja que existe uma relação entre a tabela de compradores e a tabela de compras: cada compra tem um comprador, mas um comprador tem muitas compras. Chamamos essa relação de **um-pra-muitos**, ou **one-to-many**.

Apenas para testes, vamos colocar uma parte das compras para o comprador com id 1, e outra para o comprador de id 2:

```
UPDATE COMPRAS SET COMPRADOR_ID = 1 WHERE ID < 8;
UPDATE COMPRAS SET COMPRADOR_ID = 2 WHERE ID >= 8;
```

Veja que todas as compras com `ID` menor 8 agora apontam para o comprador com `ID` 1, e todas as compras com `ID` maior-ou-igual a 8 apontam para o comprador com `ID` 2.

Agora, com todos os campos `COMPRADOR_ID` preenchidos na tabela `COMPRAS`, vamos alterá-la, dizendo que `COMPRADOR_ID` não pode ser nulo, pois uma compra sempre tem um comprador:

```
ALTER TABLE COMPRAS MODIFY (COMPRADOR_ID NUMBER NOT NULL);
```

Vamos agora selecionar todas as compras e seus compradores. Podemos dizer na instrução SQL que queremos exibir mais de uma tabela. Basta separar as tabelas com vírgula:

```
SELECT * FROM COMPRAS, COMPRADORES;
```

Veja que ele nos trouxe um resultado gigante. Como são duas tabelas diferentes, e ele não sabe juntá-las, o Oracle nos devolveu toda as combinações possíveis entre os elementos das duas tabelas. Ou seja, se nossa tabela de `COMPRAS` tivesse apenas uma única compra de 100,00, e a tabela de COMPRADORES tivesse 2 pessoas, o Mauricio e o Adriano, então esse `SELECT` nos devolveria uma compra de 100,00 para o Mauricio e uma compra de 100,00 para o Adriano.

Juntando duas tabelas diferentes

Precisamos dizer ao Oracle que queremos juntar esses dados, e que a coluna que faz essa junção é justamente a coluna `COMPRADOR_ID`, da tabela `COMPRAS`, com a coluna `ID`, da tabela `COMPRADORES`. Basta dizermos que essa é a coluna que faz a junção ou, em inglês, o `JOIN` ou 'INNER JOIN':

```
SELECT * FROM COMPRAS JOIN COMPRADORES ON COMPRAS.COMPRADOR_ID = COMPRADORES.ID;
SELECT * FROM COMPRAS INNER JOIN COMPRADORES ON COMPRAS.COMPRADOR_ID = COMPRADORES.ID;
```

Agora sim, o Oracle tem informações suficientes para juntar os dados das duas tabelas! Ele sabe que o `COMPRADOR_ID` aponta para um `ID` de `COMPRADOR`. Portanto, ele nos trouxe todas as compras junto com os dados do comprador!

Chave estrangeira

Veja que a coluna `COMPRADOR_ID` é como se fosse um link para um comprador. Colunas que tem justamente essa finalidade são chamadas de **foreign key**, **chave estrangeira**, ou simplesmente **FK**. Podemos inclusive deixar isso claro no Oracle, para que ele não permita que seja inserida uma FK inválida (ou seja, uma FK que não aponta para um comprador válido).

Vamos fazer essa alteração na tabela, e dizer que a coluna COMPRADOR_ID é uma FK:

```
ALTER TABLE COMPRAS ADD FOREIGN KEY (COMPRADOR_ID) REFERENCES COMPRADORES(ID);
```

Veja que indicamos nessa consulta SQL o nome da coluna que é uma FK (no caso, `COMPRADOR_ID`) e dissemos que ela aponta para o campo `ID` da tabela `COMPRADORES`.

Os `JOINs` nos permitem juntar tabelas que se relacionam através das FKs. É um recurso poderoso que você deve praticar!