10

Consolidando o seu conhecimento

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) O tipo auto-incremento cria uma sequência numérica de números inteiros em um campo. Para definir este campo, é preciso configurá-lo na criação da tabela. Logo, digite o comando abaixo e execute-o:

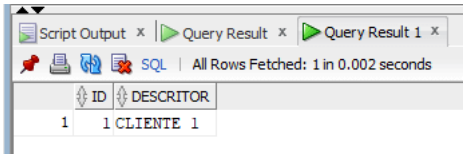
```
CREATE TABLE TAB_IDENTITY (
    ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    DESCRITOR VARCHAR(20),
    PRIMARY KEY (ID)
);
```

2) Para inserir um registro, não é necessário referenciar o campo auto-incremento no comando `INSERT`. Digite e execute:

```
INSERT INTO TAB_IDENTITY (DESCRITOR) VALUES ('CLIENTE 1');
```

3) Verifique o conteúdo da tabela. Digite e execute:

```
SELECT * FROM TAB_IDENTITY;
```

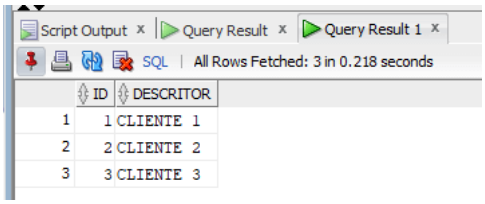


4) Agora, inclua mais dois registros:

```
INSERT INTO TAB_IDENTITY (DESCRITOR) VALUES ('CLIENTE 2');
INSERT INTO TAB_IDENTITY (DESCRITOR) VALUES ('CLIENTE 3');
```

5) Verifique o conteúdo da tabela. Digite e execute:

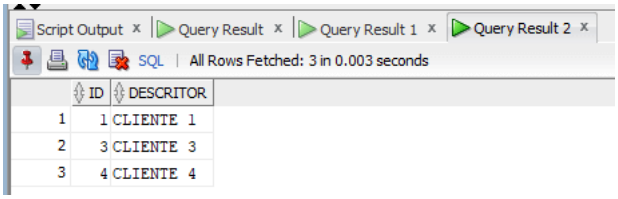
```
SELECT * FROM TAB_IDENTITY;
```



6) Ao apagar um registro, a sequência do contador não é interrompida. Digite e execute:

```
DELETE FROM TAB_IDENTITY WHERE ID = 2;
INSERT INTO TAB_IDENTITY (DESCRITOR) VALUES ('CLIENTE 4');
```

```
SELECT * FROM TAB_IDENTITY;
```

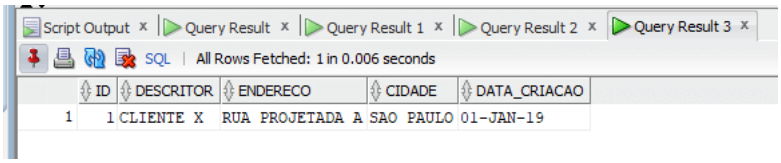


7) Você pode definir padrões para os campos. Com isto, um campo pode ter um valor *default* caso não seja referenciado no comando `INSERT`. Digite e execute o comando abaixo. Assim, padrões serão criados para os campos `CIDADE` e `DATA_CRIACAO`:

```
CREATE TABLE TAB_PADRAO (
    ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    DESCRITOR VARCHAR(10) NOT NULL,
    ENDERECO VARCHAR(100) NULL,
    CIDADE VARCHAR(50) DEFAULT ON NULL 'Rio de Janeiro',
    DATA_CRIACAO DATE DEFAULT ON NULL SYSDATE,
    PRIMARY KEY (ID)
);
```

8) Agora, insira um registro e selecione todos:

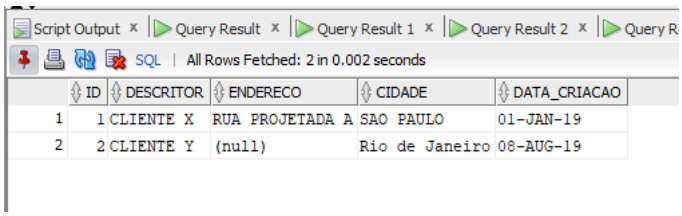
```
INSERT INTO TAB_PADRAO (DESCRITOR, ENDERECO, CIDADE, DATA_CRIACAO)
VALUES ('CLIENTE X','RUA PROJETADA A','SAO PAULO',TO_DATE('2019-01-01','YYYY-MM-DD'));

SELECT * FROM TAB_PADRAO;
```



Aqui, o comando `INSERT` funciona normalmente, porque todos os campos foram referenciados.

9) Repita o comando `INSERT`, mas agora usando somente os campos que não possuem padrões. Digite e execute:

```
INSERT INTO TAB_PADRAO(DESCRITOR) VALUES ('CLIENTE Y');

SELECT * FROM TAB_PADRAO;
```



Note que, para os campos que não foram referenciados no comando `INSERT`, os seus valores padrões foram incluídos na tabela.

10) Crie uma tabela auxiliar, que irá sempre ter o faturamento consolidado por data da venda. Execute o comando:

```
CREATE TABLE TAB_FATURAMENTO (
    DATA_VENDA DATE NULL,
    TOTAL_VENDA FLOAT
);
```

11) Apague o conteúdo das tabelas `ITEMS_NOTAS` e `NOTAS` :

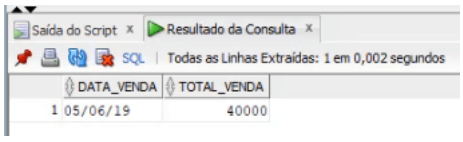
```
DELETE FROM ITEMS_NOTAS;
DELETE FROM NOTAS;
```

12) O objetivo é que, a cada inclusão de dados nas tabela `NOTAS` e `ITENS_NOTAS` , o valor de `TAB_FATURAMENTO` seja atualizado. Para isso, digite e execute:

```
INSERT INTO NOTAS (NUMERO, DATA_VENDA, CPF, MATRICULA, IMPOSTO)
VALUES ('0100', TO_DATE('2019-06-05','YYYY-MM-DD'), '1471156710', '235', 0.10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0100', '1040107', 1000, 10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0100', '1000889', 1000, 10);

INSERT INTO NOTAS (NUMERO, DATA_VENDA, CPF, MATRICULA, IMPOSTO)
VALUES ('0101', TO_DATE('2019-06-05','YYYY-MM-DD'), '3623344710', '235', 0.10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0101', '1040107', 1000, 10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0101', '1000889', 1000, 10);
```

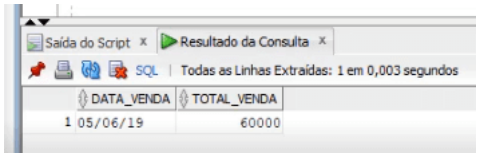
13) Para atualizar a tabela `TAB_FATURAMENTO` , você tem que executar o comando abaixo:

```
INSERT INTO TAB_FATURAMENTO
SELECT NOTAS.DATA_VENDA, SUM(ITEMS_NOTAS.QUANTIDADE * ITEMS_NOTAS.PRECO) AS TOTAL_VENDA
FROM NOTAS INNER JOIN ITEMS_NOTAS
ON NOTAS.NUMERO = ITEMS_NOTAS.NUMERO
GROUP BY NOTAS.DATA_VENDA;

SELECT * FROM TAB_FATURAMENTO;
```



14) Para manter a tabela `TAB_FATURAMENTO` atualizada, você tem que repetir o cálculo atual do valor total das vendas sempre que novos registros forem incluídos. Para isso, digite e execute:

```
INSERT INTO NOTAS (NUMERO, DATA_VENDA, CPF, MATRICULA, IMPOSTO)
VALUES ('0102', TO_DATE('2019-06-05','YYYY-MM-DD'), '3623344710', '235', 0.10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0102', '1040107', 1000, 10);

INSERT INTO ITEMS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0102', '1000889', 1000, 10);

DELETE FROM TAB_FATURAMENTO;

INSERT INTO TAB_FATURAMENTO
SELECT NOTAS.DATA_VENDA, SUM(ITEMS_NOTAS.QUANTIDADE * ITEMS_NOTAS.PRECO) AS TOTAL_VENDA
FROM NOTAS INNER JOIN ITEMS_NOTAS
ON NOTAS.NUMERO = ITEMS_NOTAS.NUMERO
GROUP BY NOTAS.DATA_VENDA;

SELECT * FROM TAB_FATURAMENTO;
```

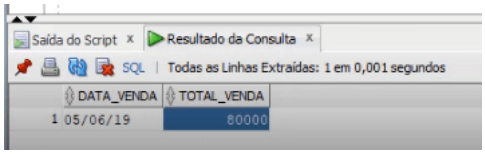


15) Você pode criar uma `TRIGGER`, para que a tabela `TAB_FATURAMENTO` seja recalculada sempre que um novo registro for incluído na tabela de `ITENS_NOTAS`. Para isso, digite e execute:

```
CREATE TRIGGER TG_CALCULO_FATURAMENTO
AFTER INSERT ON ITEMS_NOTAS
BEGIN
    DELETE FROM TAB_FATURAMENTO;
    INSERT INTO TAB_FATURAMENTO
        SELECT NOTAS.DATA_VENDA, SUM(ITEMS_NOTAS.QUANTIDADE * ITEMS_NOTAS.PRECO) AS TOTAL_VENDA
        FROM NOTAS INNER JOIN ITEMS_NOTAS
        ON NOTAS.NUMERO = ITEMS_NOTAS.NUMERO
        GROUP BY NOTAS.DATA_VENDA;
END;
```

16) Ao inserir novos registros, não é mais preciso executar o cálculo da tabela consolidada. Para isso, digite e execute:

```
INSERT INTO NOTAS (NUMERO, DATA_VENDA, CPF, MATRICULA, IMPOSTO)
VALUES ('0106', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0106', '1000889', 100, 10);

INSERT INTO ITENS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)
VALUES ('0106', '1002334', 100, 10);

SELECT * FROM TAB_FATURAMENTO;
```



17) Você criou uma `TRIGGER` somente para a inclusão de registros na tabela. Agora, crie uma `TRIGGER` para a atualização e uma para a exclusão. Para isso, digite e execute:

```
CREATE OR REPLACE TRIGGER TG_CALCULO_FATURAMENTO
AFTER INSERT OR UPDATE OR DELETE ON ITEMS_NOTAS
BEGIN
    DELETE FROM TAB_FATURAMENTO;
    INSERT INTO TAB_FATURAMENTO
        SELECT NOTAS.DATA_VENDA, SUM(ITEMS_NOTAS.QUANTIDADE * ITEMS_NOTAS.PRECO) AS TOTAL_VENDA
        FROM NOTAS INNER JOIN ITEMS_NOTAS
        ON NOTAS.NUMERO = ITEMS_NOTAS.NUMERO
        GROUP BY NOTAS.DATA_VENDA;
END;
```