Comandos Avançados em SQL

1 Instrução Merge

A instrução MERGE foi introduzida inicialmente na versão Oracle 9i como parte da tecnologia de ETL (Extract Transform Load). São ferramentas de software cuja função é a extração de dados de diversos sistemas e transformação segundo determinadas regras de negócio. É uma das fases cruciais do Data Warehouse.

Suas características:

Possibilita estabelecer condicionalmente um insert ou update de dados em uma tabela, num único comando.

É mais flexível e eficiente do que soluções programadas a mão.

É possível fazer UPDATEs, INSERTs e DELETEs com condições separadas para cada.

1.1 Formato Geral

MERGE <hint> INTO <table_name>
USING <table_view_or_query>
ON (<condition>)
WHEN MATCHED THEN <update_clause>
DELETE <where_clause>
WHEN NOT MATCHED THEN <insert_clause>
[LOG ERRORS <log_errors_clause> <reject limit <integer | unlimited>];

ou

MERGE [hints] INTO table-name-1 | view-name-1 [alias-1]
USING table-name-2 | view-name-2 | subquery [alias-2]
ON (condition)
[merge-update-clause] [merge-insert-clause] [error-logging-clause];

1.2 Exemplo

```
CREATE TABLE Funcionario
(matricula     NUMBER(5),
 nome          VARCHAR2(30),
 depto         NUMBER(2),
 salario       NUMBER(10));

INSERT INTO funcionario VALUES (1, 'Daniel', 10, 100000);
INSERT INTO funcionario VALUES (2, 'Helena', 20, 100000);
INSERT INTO funcionario VALUES (3, 'Akito', 20, 50000);
INSERT INTO funcionario VALUES (4, 'Jaqueline', 20, 40000);
INSERT INTO funcionario VALUES (5, 'Ricardo', 20, 70000);
INSERT INTO funcionario VALUES (6, 'Joao', 20, 30000);
INSERT INTO funcionario VALUES (7, 'Clark', 20, 90000);
COMMIT;
```

1.2 Exemplo

```
CREATE TABLE bonus
(matricula     NUMBER(5),
 bonus         NUMBER(6) DEFAULT 100);

INSERT INTO bonus (matricula) VALUES (1);
INSERT INTO bonus (matricula) VALUES (2);
INSERT INTO bonus (matricula) VALUES (4);
INSERT INTO bonus (matricula) VALUES (6);
INSERT INTO bonus (matricula) VALUES (7);
COMMIT;
```

1.2 Exemplo

```
SELECT * FROM funcionario;

SELECT * FROM bonus;

MERGE INTO bonus b
USING
        (SELECT matricula, salario, depto
         FROM funcionario
         WHERE depto =20) e
ON
        (b.matricula = e.matricula)
WHEN MATCHED THEN
        UPDATE SET b.bonus = e.salario * 0.1
        DELETE WHERE (e.salario < 40000)
WHEN NOT MATCHED THEN
        INSERT (b.matricula, b.bonus)
        VALUES (e.matricula, e.salario * 0.05)
        WHERE (e.salario > 40000);

SELECT * FROM bonus;
```

1.3 Explicação

Para os funcionários que tiverem sua matrícula na tabela BONUS (MATCHED):

1) 10% de bônus será concedido (UPDATE);
2) Quem tiver o salário menor que 40000 não ganhará nada (DELETE).

Para os funcionários que não possuem matrícula na tabela de BONUS (NOT MATCHED):

1) 5% de bônus, contanto que o salário seja maior que 40000 (INSERT).

1.4 Exercício

```
create table table_dest
(  id  number primary key,
   txt varchar2(20)
);

insert into table_dest values (1,'one');
insert into table_dest values (3,'three');
insert into table_dest values (5,'five');

commit;
```

1.4 Exercício

```
create table table_source
(  id  number primary key,
   txt varchar2(20)
);

insert into table_source values (2,'TWO');
insert into table_source values (3,'THREE');

commit;
```

1.4 Exercício

```
select * from table_dest;

select * from table_source;

MERGE into
	table_dest d
USING
	table_source s
ON
	(s.id = d.id)
WHEN MATCHED THEN
	update set d.txt = s.txt
WHEN NOT MATCHED THEN
	insert (id, txt) values (s.id, s.txt);


select * from table_dest;
```

2 Insert em Várias Tabelas

A instrução INSERT ALL foi introduzida para facilitar a inserção de dados em tabelas diferentes, baseado no resultado de uma determinada tabela ou consulta.

Suas características:

Possibilita estabelecer mais de uma tabela como destino para os dados desejados.

É mais flexível e eficiente que soluções programadas a mão.

2.1 Formato Geral

```
INSERT ALL
INTO <table_name> VALUES (column_name_list)
INTO <table_name> VALUES (column_name_list)
...
<SELECT Statement>;
```

2.2 Exemplo

```
create table avaliacao
( matricula number(5),
  prova      varchar2(3),
  nota       number(4,2));


INSERT ALL
      INTO avaliacao VALUES (matricula,'AV1', 10)
      INTO avaliacao (matricula,prova, nota) VALUES (matricula,'AV2', 7)
      INTO avaliacao (matricula,prova) VALUES (matricula,'AV3')
SELECT * FROM FUNCIONARIO;


select * from funcionario;
```

2.3 Explicação

Para cada FUNCIONARIO serão feitas 3 inserções na tabela AVALIACAO:

1) Matricula, prova (AV1) e nota (10);
2) Matricula, prova (AV2) e nota (7) explicitamente;
3) Matricula e prova (AV3), sem a nota.

3 Usando a cláusula WHEN

```
create table bonus
(matricula     NUMBER(5),
 bonus         NUMBER(6) DEFAULT 100);

create table demitido
(matricula     Number(5)    PRIMARY KEY,
 nome          Varchar2(30));


INSERT ALL
WHEN salario IN (40000, 50000)
      THEN INTO bonus values (matricula, salario * 0.1)
WHEN salario IN (100000) AND  depto in (20)
      THEN INTO demitido values (matricula,nome)
SELECT * FROM FUNCIONARIO
```

3.1 Explicação

Para os dados da tabela FUNCIONARIO serão feitas inserções nas seguintes condições:

1) Se o salário for de 40000 ou 50000:
   Inserção de linha na tabela bônus com a sua matricula e bônus de 10% sobre o salário.

2) Se o salário for de 100000 e o departamento igual a 20:
   Inserção de linha na tabela demitido com a sua matricula e nome.

4 Insert com Cláusula Returning

A cláusula RETURNING pode ser utilizada para obter as informações sobre a linha ou linhas que foram processadas por um comando DML.

Por exemplo, é possível obter o ROWID da linha que acabou de ser incluída, sem a necessidade de submeter um novo comando SELECT para o banco de dados.

```
SET SERVEROUTPUT ON;
DECLARE    r      rowid;
BEGIN
      INSERT INTO funcionario (matricula, nome)
      VALUES (8, 'Morgana')
      RETURNING rowid INTO r;
      dbms_output.put_line(r);
END;
/
```

4.1 Exemplo

```
SET SERVEROUTPUT ON;
DECLARE
	v_NewRowid ROWID;
	v_FirstName students.first_name%TYPE;
	v_LastName students.last_name%TYPE;
	v_ID students.ID%TYPE;

BEGIN
	INSERT INTO students (ID, first_name, last_name, major, current_credits)
	VALUES (student_sequence.NEXTVAL, 'Xavier', 'Xemes', 'Nutrition', 0)
	RETURNING rowid INTO v_NewRowid;

	DBMS_OUTPUT.PUT_LINE('Newly inserted rowid is ' || v_NewRowid);

	UPDATE students SET current_credits = current_credits + 3
	WHERE rowid = v_NewRowid
	RETURNING first_name, last_name INTO v_FirstName, v_LastName;

	DBMS_OUTPUT.PUT_LINE('Name: ' || v_FirstName || ' ' || v_LastName);

	DELETE FROM students
	WHERE rowid = v_NewRowid
	RETURNING ID INTO v_ID;

	DBMS_OUTPUT.PUT_LINE('ID of new row was ' || v_ID);
END;
/
```