8

DML
Insert, Update, Delete

Objetivos

- Descrever os 4 tipos de DMLs:
  - INSERT
  - UPDATE
  - DELETE
  - MERGE

Objetivos

- Descrever os 4 tipos de DMLs:
  - INSERT
    - Com subquery, select, create table
  - UPDATE
    - 1 ou múltiplos regs, colunas, subquery
  - DELETE
    - 1 ou múltiplos reg, fórmulas, subquery
  - MERGE

DMLs

- IMPORTANTE:
  - Por padrão DMLs (insert,update,delete)
    - necessitam de um commit para persistir
  - Entretanto algumas ferramentas (ex livesql)
    - fazem commit “implicito” a cada comando
  - Tenha **CERTEZA** como é o funcionamento
    - da ferramenta que você pretende usar!

INSERT

- INSERT INTO <tabela>
  (coluna1, coluna2, … coluna_n)

- VALUES
  (valor1, valor2, ... valor_n)

INSERT

```
create table fabricante (
        id      number          NOT NULL,
        nome    varchar2(100)   NOT NULL,
        obs     varchar2(50)
);
--------------------
Table created.
```

```
insert into fabricante (id, nome, obs) values (1000, 'Green LTDA', 'Pgto antrecipado');
-------------------
1 row(s) inserted.
```

```
insert into fabricante (id, nome, obs)
values (2000, 'Yellow LTDA', 'Pgto posterior');
-------------------
1 row(s) inserted.
```

INSERT

```
insert into fabricante (id, nome) values (3000, 'Blue LTDA');
--------------------
1 row(s) inserted.

insert into fabricante (nome, id) values ('Brown LTDA', 4000);
--------------------
1 row(s) inserted.

insert into fabricante values (5000, 'Black LTDA', null);
--------------------
1 row(s) inserted.
```

INSERT - organização

```
insert into fabricante (id, nome, obs) values (1000, 'Green LTDA', 'n/a');
```

```
insert into fabricante (id, nome, obs)
values (1000, 'Green LTDA', 'Pgto antrecipado');
```

```
insert into fabricante (id, nome, obs)
values (
         1000,
         'Green LTDA',
         'Pgto antrecipado'
         );
```

INSERT com subquery

```
insert into fabricante (id, nome, obs)
values (
          (select max(id)+1 from fabricante) ,
          'Green LTDA',
          'Pgto antrecipado'
          );

--------------------
1 row(s) inserted.
```

INSERT INTO … SELECT

```
create table fabricante_bkp (
        id      number          NOT NULL,
        nome    varchar2(100)   NOT NULL,
        obs     varchar2(50)
);


insert  into fabricante_bkp (id, nome, obs)
select  id, nome, obs
from    fabricante
where   id < 4500;
```

CREATE TABLE ... SELECT

```
create table fabricante_bkp (
        id      number          NOT NULL,
        nome    varchar2(100)   NOT NULL,
        obs     varchar2(50)
);
```

```
insert  into fabricante_bkp (id, nome, obs)
select  id, nome, obs
from    fabricante
where   id < 4500;
```

Ou fazer assim → alguma diferença?

```
create table fabricante_bkp as
select  id, nome, obs
from    fabricante
where   id < 4500;
```

CREATE TABLE ... SELECT

```
create table fabricante_bkp (
        id      number          NOT NULL,
        nome    varchar2(100)   NOT NULL,
        obs     varchar2(50)
);


insert  into fabricante_bkp (id, nome, obs)
select  id, nome, obs
from    fabricante
where   id < 4500;
```

Ou fazer assim → alguma diferença?
* DDL faz commit

```
create table fabricante_bkp as
select  id, nome, obs
from    fabricante
where   id < 4500;
```

CREATE TABLE … SELECT sem linhas

- Duplicar a estrutura sem dados

```
create table fabricante_bkp as
select  id, nome, obs
from    fabricante
where   1=0;
```

  - Ao invés de:

```
create table fabricante_bkp as
select  id, nome, obs
from    fabricante
where   id < 4500;
```

UPDATE

- UPDATE <tabela>
- SET col1 = val1,
      col2 = val2,
      ...
      coln = valn
- WHERE
    <condições>

UPDATE

```
update   fabricante
set      nome = 'Yellow S.A.',
         obs = 'Boletos 30d'
where    id = 2000;
----------------------
1 row(s) updated.
```

UPDATE múltiplas linhas

```
update   fabricante
set      obs = 'apenas pgto A.V.'
where    id <= 2000;
--------------------
2 row(s) updated.
```

UPDATE: múltiplas linhas, referenciando colunas

- Múltiplas linhas

```
update   hr.employees
set      salary = salary*1.1
where    employee_id <= 120;
```

- CUIDADO !!! Update **SEM** where ….. TODA TABELA

```
update   hr.employees
set      salary = salary*1.1;
```

UPDATE: múltiplas linhas, referenciando colunas

- Múltiplas linhas, referenciando colunas

```
update   hr.employees
set      email = FIRST_NAME || '.' || LAST_NAME || '@it_company.com'
where    job_id = 'IT_PROG';
```

- Toda tabela, CUIDADO

```
update   hr.employees
set      email = FIRST_NAME || '.' || LAST_NAME || '@company.com';
```

UPDATE com subquery

```
update   hr.employees
set      salary = (
                   select avg(salary)
                   from   hr.employees
                   where  department_id=100
                   )
where    department_id = 50;
```

DELETE

- DELETE FROM <tabela>
- WHERE
  <condições>

```
delete   hr.employees
where    employee_id = 136;
```

DELETE múltiplos registros

```
delete   fabricante
where    id <= 3000;
--------------------
3 row(s) deleted.
```

```
delete   fabricante
where    id between 2000 and 4000;
--------------------
3 row(s) deleted.
```

DELETE com base em fórmula

```
delete   hr.employees
where    department_id = 50
and      salary > (
                  select avg(salary)
                  from   hr.employees
                  where  department_id = 100
                  );
```

DELETE com subquery

```
delete   NEW_EMPLOYEES
where    employee_id NOT IN
                    (
                     select employee_id
                     from   hr.employees
                    );
```

DELETE sem where

- CUIDADO !!!
  - Toda a tabela !!!

  Depende do tamanho pode:
  - Demorar
  - Gerar muito archivelog
  - Gera muito UNDO (rollback)

```
delete   FROM new_employees;
----------------------
107 row(s) deleted.
```

- Alternativa: TRUNCATE

  → mais CUIDADO ainda !!!
  - Não tem rollback !!!

```
truncate table new_employees;
---------------------
Table truncated.
```

Boa Prática …

- UPDATE e DELETE `tabela`
  - Copiar o <WHERE>
  - testar antes com

```
select count(*) from tabela <WHERE>
select *        from tabela <WHERE>
```

```
update  hr.employees
set     salary = 1000
where   department_id = 50;
```

```
select  *
from    hr.employees
where   department_id = 50;
```

```
delete  hr.employees
where   department_id = 50;
```

```
select  count(*)  from hr.employees
where   department_id = 50;
```

DMLs “avançados”

Extremamente úteis em DTLs, VLDBs, DSS, análise dados:

- Cargas Paralelas
- Insert Multitabela
- Insert Condicional
- Insert FIRST (condicional)
- MERGE

Instruções DML

- Cargas podem ser paralelizadas (PDML)

```
INSERT  /*+ PARALLEL(costs,2) */
INTO    costs
SELECT  * FROM old_costs;
```

Instruções DML

- Insert Multitabela
  - Os dados são inseridos em várias tabelas através de uma mesma instrução Insert

```
INSERT  ALL
   INTO sal_history VALUES(EMPID,HIREDATE,SAL)
   INTO mgr_history VALUES(EMPID,MGR,SAL)
SELECT empid, hiredate, sal, mgr
FROM   employees;
```

- Insert Condicional

```
INSERT ALL
   WHEN sal > 10000 THEN
   INTO sal_history VALUES(EMPID,HIREDATE,SAL)
   ...
```

Instruções DML

- FIRST Insert
  - Os dados são inseridos na tabela que satisfazer a primeira condição

```
INSERT FIRST
   WHEN sal > 25000 THEN
        INTO special_sal VALUES(DEPTID, SAL)
   WHEN hiredate >= '01/01/2000' THEN
        INTO history_00 VALUES(DEPTID,HIREDATE)
   ELSE
        INTO history VALUES(DEPTID, HIREDATE)
SELECT deptid, sal, hiredate
FROM   employees;
```

Instruções DML

Instrução MERGE

- Permite atualizar ou inserir dados de forma condicional
  - UPDATE se as linhas existirem
  - INSERT se for uma nova linha

✓ Uma só instrução

✓ Fácil de usar

✓ Aumenta o desempenho

- Útil para data warehouse

Instruções DML

- Exemplo de instrução Merge:

```
MERGE INTO copy_emp c
   USING employees e
   ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
   UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      ...
      c.department_id   = e.department_id
WHEN NOT MATCHED THEN
  INSERT VALUES(e.employee_id, e.first_name, e.last_name,
           e.email, e.phone_number, e.hire_date, e.job_id,
           e.salary, e.commission_pct, e.manager_id,
           e.department_id);
```

Resumo e Dúvidas

- Dúvidas ou comentários ... ?