26-8-2022

Database Programming with SQL 2 / 2

JUAN CARLOS HERRERA HERNANDEZ
UNIVERSIDAD POLITECNICA DE AGUASCALIENTES

Contenido

Section 11 – Ensuring Quality Queries Part I

11-1 Ensuring Quality Query Results

Solve a series of problems:

- Solve a series of problems Create a query to produce specified data
- Modify a query to produce specified data

Select * from user_tables	PURGE RECYCLEBIN;
select * from tab;	

Section 12 – DML

12-1 INSERT Statements

USER	Someone doing “real work” with the computer, using it as a means rather than an end
Transaction	Consists of a collection of DML statements that form a logical unit of work.
Explicit	Fully and clearly expressed; leaving nothing implied
INSERT INTO	Adds a new row to a table

The table copies will not inherit the associated primary-to-foreign-key integrity rules (relationship constraints) of the original tables.

Copy structure and data	Copy only structure

`CREATE TABLE copy_departments`
`as`
`SELECT * FROM departments;`
	`CREATE TABLE copy_departments`
	`   as (SELECT * FROM departments`
	`       where 1=2);`

Describe employees	SALARY NUMBER(6,2) Precision 6, Scale 2 maximum value allowed 9999.99
user	select user from dual;
sysdate	select sysdate from dual; -- default DD-Mon-YYYY
	select to_char(sysdate, 'Month fmdd, yyyy') from dual;

select columns	all columns

`INSERT INTO copy_departments`
`   (department_id, department_name, manager_id, location_id)`
`   VALUES (200,'Human Resources', 205, 1500);`
	`INSERT INTO copy_departments`
	`   VALUES (210,'Estate Management', 102, 1700);`

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, phone_number, hire_date, job_id, salary)
  VALUES
  (302,'Grigorz','Polanski', '8586667641', '15-Jun-2017', 'IT_PROG',4200);
```

```
INSERT INTO copy_employees
   (employee_id, first_name, last_name, email, hire_date, job_id)
    VALUES
   (303, 'Katie', 'Hernandez', ' ', TO_DATE('2017-07-20', 'yyyy-mm-dd'), 'MK_REP');
```

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
    SELECT employee_id, last_name, salary, commission_pct
    FROM employees
    WHERE job_id LIKE '%REP%';
```

12-2 Updating Column Values and Deleting Rows

UPDATE	Modifies existing rows in a table
Correlated subquery UPDATE	retrieves information from one table & uses the information to update another table
Integrity Constraint	Ensures that the data adheres to a predefined set of rules
Correlated subquery DELETE	deletes information on a linked table based on what was deleted on the other table
Delete	Removes existing rows from a table

```
UPDATE copy_employees
SET hire_date = sysdate
WHERE employee_id = 206;
```

```
UPDATE copy_employees
SET hire_date = sysdate,
    salary = (SELECT salary FROM copy_employees
              WHERE employee_id= 205),
    job_id = (SELECT job_id FROM copy_employees
              WHERE employee_id= 205)
WHERE employee_id = 206;
```

```
ALTER TABLE copy_employees
ADD (department_name varchar2(30));

UPDATE copy_employees e
SET e.department_name= (SELECT d.department_name
                        FROM departments d
                        WHERE e.department_id= d.department_id);
```

```
DELETE FROM departments
WHERE department_id = 50;

DELETE FROM copy_employees
WHERE department_id = 50;
```

```
DELETE FROM copy_employees
WHERE department_id =
     (SELECT department_id FROM departments
      WHERE department_name= 'Shipping');
```

```
SELECT * FROM copy_employees e
WHERE e.manager_id IN
     (SELECT d.manager_id
      FROM employees d
      GROUP BY d.manager_id
      HAVING count(d.department_id) < 2);
```

```
DELETE FROM copy_employees e
WHERE e.manager_id IN
     (SELECT d.manager_id
      FROM employees d
      GROUP BY d.manager_id
      HAVING count(d.department_id) < 2);
```

row-level locks, until you issue a COMMIT or ROLLBACK

```
SELECT e.employee_id, e.salary, d.department_name
FROM employees e JOIN departments d USING (department_id)
WHERE location_id = 1500 AND job_id= 'ST_CLERK'
FOR UPDATE
ORDER BY e.employee_id;

GRANT update, select ON employees TO schemas

User: SCHEMAS
update ESQUEMAS.employees e set salary = salary
where e.employee_id = 141;
```

12-3 DEFAULT Values, MERGE, and Multi-Table Inserts

A **data warehouse** is a collection of data designed to support business-management decision making. Data warehouses contain a wide variety of data, such as sales data, customer data, payroll, accounting, and personnel data, which presents a coherent picture of business conditions at a single point in time.

```
CREATE TABLE my_employees (
hire_date DATE DEFAULT SYSDATE,
first_name VARCHAR2(15),
last_name VARCHAR2(15));
```

```
-- Explicit
INSERT INTO my_employees
(hire_date, first_name, last_name)
VALUES (DEFAULT, 'Angelina','Wright');
```

```
-- Implicit
INSERT INTO my_employees
(first_name, last_name)
VALUES('Angelina','Wright');
```

```
UPDATE my_employees
SET hire_date = DEFAULT
WHERE last_name = 'Wright';
```

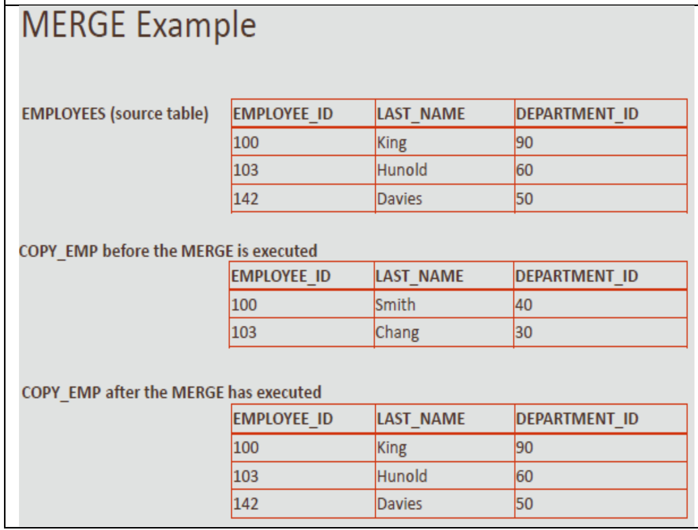
```
UPDATE my_employees
SET hire_date = '21-SEP-89'
WHERE last_name = 'Wright';
```

```
UPDATE copy_employees
SET hire_date = to_date('1989-09-21', 'yyyy-mm-dd')
WHERE employee_id = 100;
```

```
MERGE will INSERT and UPDATE
simultaneously.

MERGE INTO destination-table USING
source-table
ON matching-condition
WHEN MATCHED THEN UPDATE
SET ……
WHEN NOT MATCHED THEN INSERT
VALUES (……);
```

```
MERGE INTO copy_emp c USING employees e
ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN UPDATE
    SET
        c.last_name = e.last_name,
        c.department_id = e.department_id
WHEN NOT MATCHED THEN INSERT
    VALUES (e.employee_id, e.last_name,
e.department_id);
```



MERGE Example

EMPLOYEES (source table)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

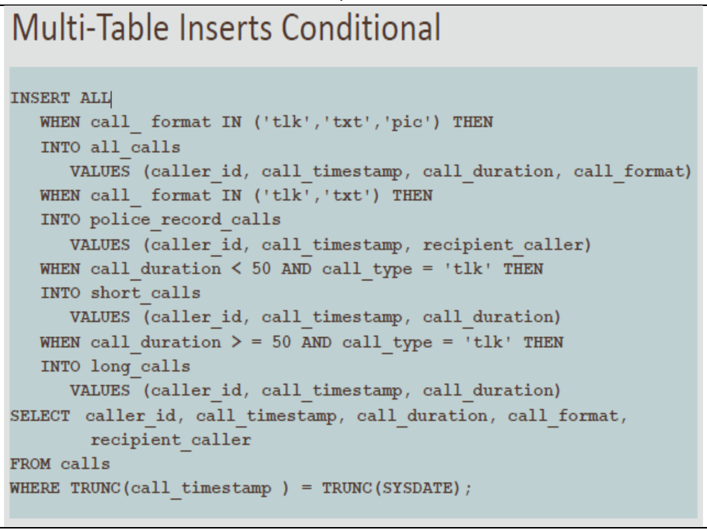
COPY_EMP before the MERGE is executed

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	Smith	40
103	Chang	30

COPY_EMP after the MERGE has executed

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

ALL , FIRST



Multi-Table Inserts Conditional

```
INSERT ALL
   WHEN call_ format IN ('tlk','txt','pic') THEN
   INTO all_calls
      VALUES (caller_id, call_timestamp, call_duration, call_format)
   WHEN call_ format IN ('tlk','txt') THEN
   INTO police_record_calls
      VALUES (caller_id, call_timestamp, recipient_caller)
   WHEN call_duration < 50 AND call_type = 'tlk' THEN
   INTO short_calls
      VALUES (caller_id, call_timestamp, call_duration)
   WHEN call_duration > = 50 AND call_type = 'tlk' THEN
   INTO long_calls
      VALUES (caller_id, call_timestamp, call_duration)
SELECT  caller_id, call_timestamp, call_duration, call_format,
        recipient_caller
FROM calls
WHERE TRUNC(call_timestamp ) = TRUNC(SYSDATE);
```