Oracle SQL – Data Manipulation Language

Stephan Karrer

Datenmanipulation (DML)

- Zeilen hinzufügen, Werte verändern, Zeilen löschen
- Data Manipulation Language Statements:
  - INSERT,
  - UPDATE,
  - DELETE,
  - MERGE

INSERT: Neue Zeilen einfügen

```
INSERT  INTO departments (department_id, manager_id,
                          location_id, department_name)
        VALUES ( 70, 100, 1700, 'Public Relations' );


INSERT  INTO departments
        VALUES ( 70, 'Public Relations', NULL, NULL);


INSERT   INTO my_employees (employee_id, last_name, hire_date)
        VALUES ( 116, 'Miller', SYSDATE);
```

INSERT: Zeilen aus vorhandenen Tabellen kopieren

```
INSERT  INTO my_employees
        SELECT  *  FROM employees;

INSERT  INTO my_departments (id, name)
        SELECT department_id, department_name
                FROM departments
                WHERE manager_id = 100;
```

UPDATE: Vorhandene Zeilen ändern

```
UPDATE  employees
        SET department_id = 70,
            manager_id = 100
        WHERE employee_id = 137 ;

UPDATE  my_employees
        SET department_id = 70 ;
```

UPDATE: Spalten mit Unterabfragen aktualisieren

```
UPDATE  employees
        SET  job_id = (SELECT job_id FROM employees
                               WHERE employee_id = 207),
             department_id = (SELECT department_id
                               FROM departments
                               WHERE department_name = 'IT')
        WHERE employee_id = 67;
```

INSERT und UPDATE: Verwendung von DEFAULT-Werten

```
INSERT INTO departments
       VALUES (280, 'Recreation', DEFAULT, 1700);

UPDATE  departments
        SET manager_id = DEFAULT
        WHERE department_id = 20;
```

DELETE: Zeilen löschen

```
DELETE FROM employees  WHERE employee_id = 67;

DELETE FROM my_employee;

DELETE FROM employees
       WHERE department_id = (SELECT department_id
                              FROM departments
                              WHERE department_name =
                                   'Public Relations' );
```

INSERT: Zieltabelle mittels Unterabfrage definieren

```
INSERT INTO
         (SELECT  employee_id, last_name, hire_date,
                  job_id, salary, department_id

            FROM employees

            WHERE department_id = 50)

      VALUES  (1089, 'Maier',
               TO_DATE('07-JUN-99', 'DD-MON-RR'),
               'ST_CLERK', 5000, 50);
```

INSERT-Anweisungen für mehrere Tabellen (ab Version 9i)

```
INSERT ALL
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date, sales_sun)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+1, sales_mon)
  SELECT product_id, customer_id, weekly_start_date, sales_sun,
         sales_mon
    FROM sales_input_table;
```

```
INSERT ALL
   INTO  sal_history VALUES (empid, hired, sal)
   INTO  mgr_history VALUES (empid, mgr, sal)
   SELECT  employee_id empid, hire_date hired, salary sal,
           manager_id mgr
     FROM employees WHERE department_id = 50;
```

INSERT-Anweisungen für mehrere Tabellen mit Bedingung

```
INSERT ALL
  WHEN order_total < 1000000
    THEN INTO small_orders
  WHEN order_total > 1000000 AND order_total < 2000000
    THEN INTO medium_orders
  WHEN order_total > 2000000
    THEN INTO large_orders
  SELECT order_id, order_total, sales_rep_id, customer_id
    FROM orders;
```

```
INSERT ALL
  WHEN order_total < 1000000
    THEN INTO small_orders
  WHEN order_total > 1000000 AND order_total < 2000000
    THEN INTO medium_orders
  ELSE INTO large_orders
  SELECT order_id, order_total, sales_rep_id, customer_id
    FROM orders;
```

INSERT FIRST -Anweisung mit Bedingung

```
INSERT FIRST
    WHEN ottl < 100000 THEN
      INTO small_orders
      VALUES(oid, ottl, sid, cid)
  WHEN ottl > 100000 and ottl < 200000 THEN
     INTO medium_orders
     VALUES(oid, ottl, sid, cid)
  WHEN ottl > 290000 THEN
     INTO special_orders
     VALUES(oid, ottl, sid, cid)
  ELSE INTO large_orders
     VALUES(oid, ottl, sid, cid)
  SELECT o.order_id oid, o.customer_id cid,
         o.order_total ottl, o.sales_rep_id sid,
         c.credit_limit cl, c.cust_email cem
     FROM orders o, customers c
     WHERE o.customer_id = c.customer_id;
```

MERGE: Bedingungsabhängiges Aktualisieren bzw. Einfügen

Syntax:

```
MERGE INTO  target_table [table_alias]
  USING (table|view|subquery) [alias]
  ON  (condition)
  WHEN MATCHED THEN
     UPDATE SET
         column1 = col_value1,
         column2 = col_value2,
         ...
     [ DELETE WHERE (where_condition) ]
  WHEN NOT MATCHED THEN
     INSERT (column_list)
     VALUES (column_values) ;
```

MERGE: Beispiel

```
MERGE INTO bonuses D
  USING (SELECT employee_id, salary, department_id
           FROM employees
           WHERE department_id = 80)  S
  ON (D.employee_id = S.employee_id)
  WHEN MATCHED THEN
     UPDATE SET D.bonus = D.bonus + S.salary*.01
     DELETE WHERE (S.salary > 8000)
  WHEN NOT MATCHED THEN
     INSERT (D.employee_id, D.bonus)
     VALUES (S.employee_id, S.salary*0.1)
     WHERE  (S.salary <= 8000);
```