DML

SELECT

INSERT

UPDATE

DELETE

MERGE

TRUNCATE ???

DML

INSERT

INSERT INTO table [(column [,column…])] VALUES (value [,value…]);

INSERT INTO table [(column [, column…])] subquery;

DML

INSERT

```
INSERT INTO emp_copy (employee_id, last_name, hire_date, email, job_id)
VALUES (1000,'WATSON','03-Nov-13', 'jwatson@hr.com', 'SA_REP');
```

```
INSERT INTO emp_copy (employee_id, last_name, hire_date, email, job_id)
VALUES (   1000,
           upper('Watson'),
           to_date('03-Nov-13','dd-mon-yy'),
           lower('JWatson@hr.com'),
           upper('sa_rep'));
```

DML

INSERT

```
INSERT INTO emp_copy (employee_id, last_name, hire_date, email, job_id)
VALUES (   1000 + 1,
           user,
           sysdate - 7,
           'jwatson@hr.com',
           'SA_REP');
```

DML

INSERT

```
INSERT ALL
    WHEN 1=1 THEN INTO
        emp_no_name (department_id,job_id,salary,commission_pct,hire_date)
        VALUES (department_id,job_id,salary,commission_pct,hire_date)
    WHEN department_id <> 80 THEN INTO
        emp_non_sales(employee_id,department_id,salary,hire_date)
        VALUES (employee_id,department_id,salary,hire_date)
    WHEN department_id = 80 THEN INTO
        emp_sales(employee_id,salary,commission_pct,hire_date)
        VALUES (employee_id,salary,commission_pct,hire_date)
SELECT employee_id,department_id,job_id,salary,commission_pct,hire_date
FROM hr.employees
WHERE hire_date > sysdate - 30;
```

DML

UPDATE

UPDATE table SET column=value [,column=value…] [WHERE condition];

DML

UPDATE

```
UPDATE employees
SET salary= (    SELECT salary
                 FROM employees
                 WHERE employee_id=206);
```

```
UPDATE employees
SET salary= (    SELECT salary
                 FROM employees
                 WHERE last_name='Abel');
```

DML

DELETE

DELETE FROM tables [WHERE condition];

DML

DELETE

```
DELETE FROM employees
WHERE employee_id=206;

DELETE FROM employees
WHERE last_name LIKE 'S%';

DELETE FROM employees
WHERE department_id=&Which_department;

DELETE FROM employees
WHERE department_id IS NULL;
```

DML

DELETE

```
DELETE FROM employees
WHERE department_id IN
    (   SELECT department_id FROM departments
        WHERE location_id IN
        (   SELECT location_id FROM locations
            WHERE country_id IN
            (   SELECT country_id FROM countries
                WHERE region_id IN
                (   SELECT region_id FROM regions
                    WHERE region_name='Europe'))));
```

DML

TRUNCATE

DML

MERGE

```
MERGE INTO employees e
USING new_employees n ON (e.employee_id = n.employee_id)

    WHEN MATCHED THEN
        UPDATE SET e.salary=n.salary

    WHEN NOT MATCHED
        THEN INSERT (employee_id, last_name, salary, email, job_id)
        VALUES (n.employee_id, n.last_name, n.salary, n.email, n.job_id);
```

DML

TRANSACCIONES

El concepto de transacción es una parte del paradigma de base de datos relacional. Una transacción consiste en una o más sentencias DML, seguidas por uno de los dos comandos ROLLBACK o COMMIT.

Es posible utilizar el comando SAVEPOINT para conseguir un grado de control en las transacciones.

DML

TRANSACCIONES - ACID

ATOMICIDAD	El principio de atomicidad establece que todas las partes de una transacción o ninguna de ellas deben completarse
CONSISTENCIA	Los resultados de una consulta deben ser consistentes con el estado de la base de datos en el momento en el que se ejecuta la consulta.
AISLAMIENTO	El principio de aislamiento establece que una transacción incompleta (es decir, una transacción de la que no se ha hecho COMMIT) debe ser “invisible” para el resto del mundo.
DURABILIDAD	El principio de durabilidad establece que una vez completada la transacción, debe ser imposible que la base de datos la pierda

DML

TRANSACCIONES

COMMIT;

ROLLBACK;

SAVEPOINT savepoint; ???

...

AUTOCOMMIT ON|OFF;