SQL para Principiantes (Parte 4): La Cláusula ORDER BY

by admin - lunes, octubre 05, 2020

https://dbandtech.com/sql-para-principiantes-parte-4-la-clausula-order-by/

- **Preparar**
- **Introducción**
- **Identificar columnas: expresión, posición y alias de columna**
- **Orden ascendente (ASC) y descendente (DESC)**
- **Manejo de NULL: NULLS FIRST y NULLS LAST**

Preparar

Puede realizar todas estas consultas en línea de forma gratuita utilizando SQL Fiddle.

Los ejemplos de este artículo requieren que estén presentes las siguientes tablas.

```
--DROP TABLE employees PURGE;
--DROP TABLE departments PURGE;

CREATE TABLE departments (
  department_id   NUMBER(2) CONSTRAINT departments_pk PRIMARY KEY,
  department_name VARCHAR2(14),
  location        VARCHAR2(13)
);

INSERT INTO departments VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO departments VALUES (20,'RESEARCH','DALLAS');
INSERT INTO departments VALUES (30,'SALES','CHICAGO');
INSERT INTO departments VALUES (40,'OPERATIONS','BOSTON');
COMMIT;
CREATE TABLE employees (
  employee_id   NUMBER(4) CONSTRAINT employees_pk PRIMARY KEY,
  employee_name VARCHAR2(10),
  job           VARCHAR2(9),
  manager_id    NUMBER(4),
  hiredate      DATE,
  salary        NUMBER(7,2),
  commission    NUMBER(7,2),
  department_id NUMBER(2) CONSTRAINT emp_department_id_fk REFERENCES d
epartments(department_id)
);
INSERT INTO employees VALUES (7369,'SMITH','CLERK',7902,to_date('17-12
-1980','dd-mm-yyyy'),800,NULL,20);
```

```
INSERT INTO employees VALUES (7499,'ALLEN','SALESMAN',7698,to_date('20
-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO employees VALUES (7521,'WARD','SALESMAN',7698,to_date('22-
2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO employees VALUES (7566,'JONES','MANAGER',7839,to_date('2-4
-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO employees VALUES (7654,'MARTIN','SALESMAN',7698,to_date('2
8-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO employees VALUES (7698,'BLAKE','MANAGER',7839,to_date('1-5
-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO employees VALUES (7782,'CLARK','MANAGER',7839,to_date('9-6
-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO employees VALUES (7788,'SCOTT','ANALYST',7566,to_date('13-
JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO employees VALUES (7839,'KING','PRESIDENT',NULL,to_date('17
-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO employees VALUES (7844,'TURNER','SALESMAN',7698,to_date('8
-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO employees VALUES (7876,'ADAMS','CLERK',7788,to_date('13-JU
L-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO employees VALUES (7900,'JAMES','CLERK',7698,to_date('3-12-
1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO employees VALUES (7902,'FORD','ANALYST',7566,to_date('3-12
-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO employees VALUES (7934,'MILLER','CLERK',7782,to_date('23-1
-1982','dd-mm-yyyy'),1300,NULL,10);
COMMIT;
```

Estas tablas son una variante de las tablas **EMP y DEPT** del esquema **SCOTT**. Verá muchos ejemplos de Oracle en Internet utilizando las tablas del esquema de **SCOTT**. Puede encontrar las definiciones de la tabla original en el script "$ORACLE_HOME/rdbms/admin/utlsampl.sql".

Introducción

El mejor lugar para comenzar es con esta cita de la documentación oficial de ORACLE (aquí).

"Utilice la cláusula **ORDER BY** para ordenar las filas devueltas por la declaración. Sin un **order_by_clause**, no existe garantía de que la misma consulta ejecutada más de una vez recupere filas en el mismo orden".
Cuando comienza a aprender **SQL**, es muy fácil olvidar esto y comenzar a creer que existe un patrón predecible para la salida de datos. Hay un par de razones por las que esto puede ocurrir.

Si está utilizando pequeñas cantidades de datos de prueba, todas las filas pueden cargarse en un solo bloque, por lo que las filas pueden devolverse de una manera predecible. Una vez que se gradúe con datos

en vivo, su suposición puede resultar incorrecta.
Puede haber una clasificación implícita realizada por una o más de las operaciones en su consulta, lo que hace que el resultado sea predecible, pero eso puede ser solo para la versión específica y posiblemente la versión del parche que está ejecutando. Oracle puede cambiar el algoritmo en cualquier momento en el futuro, lo que podría "romper" su aplicación. Oracle le dijo cómo protegerse en la declaración anterior.
La documentación utiliza la siguiente descripción de texto para la **cláusula ORDER BY**.

```
ORDER [ SIBLINGS ] BY
{ expr | position | c_alias }
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
  [, { expr | position | c_alias }
     [ ASC | DESC ]
     [ NULLS FIRST | NULLS LAST ]
  ]...
```

Las siguientes secciones demostrarán algunas de estas opciones.

Identificar columnas: expresión, posición y alias de columna
Hay tres formas principales de identificar las columnas incluidas en la operación de clasificación. Probablemente, la expresión más común es especificar una o más columnas en una lista separada por comas. En el siguiente ejemplo, los resultados están ordenados por las columnas **SALARIO y COMISIÓN.**

```
SELECT e.salary, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary, e.commission;

    SALARY COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250        500 WARD
      1250       1400 MARTIN
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

Entre otras cosas, las expresiones pueden incluir varias columnas combinadas. En el siguiente ejemplo, los resultados están ordenados por la suma de las columnas **SALARIO y COMISIÓN**. La función NVL convierte cualquier valor **NULO** en la columna **COMISIÓN** a cero, para que el resultado de la adición sea más obvio.

```
SELECT e.salary, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary + NVL(e.commission,0);

    SALARY COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1500          0 TURNER
      1250        500 WARD
      1600        300 ALLEN
      1250       1400 MARTIN
      2850            BLAKE

6 rows selected.

SQL>
```

Se puede hacer referencia a las columnas por su posición de columna. Recuerde, si modifica la lista **SELECT**, también tendrá que modificar la cláusula **ORDER BY**.

```
SELECT e.salary, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY 1;

    SALARY COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250        500 WARD
      1250       1400 MARTIN
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

También se puede hacer referencia a las columnas por su alias de columna. En el siguiente ejemplo, la columna **SALARY** tiene el alias de SAL, que se utiliza en la cláusula **ORDER BY**.

```
SELECT e.salary AS sal, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY sal;

       SAL COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250        500 WARD
      1250       1400 MARTIN
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

Orden ascendente (ASC) y descendente (DESC)

El orden predeterminado es ascendente, por lo que las siguientes declaraciones son funcionalmente equivalentes.

```
    -- Orden ascendente (ASC) por defecto.


SELECT e.salary AS sal, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary;

       SAL COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250        500 WARD
      1250       1400 MARTIN
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.
```

```
SQL>
-- Explicitly setting ASC.
SELECT e.salary AS sal, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary ASC;

       SAL COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250        500 WARD
      1250       1400 MARTIN
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

Para cambiar a descendente, use la palabra clave **DESC.**

```
SELECT e.salary AS sal, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary DESC;

       SAL COMMISSION EMPLOYEE_N
---------- ---------- ----------
      2850            BLAKE
      1600        300 ALLEN
      1500          0 TURNER
      1250        500 WARD
      1250       1400 MARTIN
       950            JAMES

6 rows selected.

SQL>
```

Cada columna de la cláusula **ORDER BY** puede tener un orden diferente.

```
SELECT e.salary, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary ASC, e.commission DESC;

    SALARY COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250       1400 MARTIN
      1250        500 WARD
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

Manejo de NULL: NULLS FIRST y NULLS LAST

Un orden ascendente asume **NULLS LAST**. Puede especificarlo explícitamente si lo desea.

```
SELECT e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.commission ASC;

COMMISSION EMPLOYEE_N
---------- ----------
         0 TURNER
       300 ALLEN
       500 WARD
      1400 MARTIN
           JAMES
           BLAKE

6 rows selected.

SQL>
```

El valor predeterminado se puede modificar especificando **NULLS FIRST**.

```
SELECT e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.commission ASC NULLS FIRST;

COMMISSION EMPLOYEE_N
---------- ----------
           JAMES
           BLAKE
         0 TURNER
       300 ALLEN
       500 WARD
      1400 MARTIN

6 rows selected.

SQL>
```

Un orden descendente asume **NULLS FIRST**. Puede especificarlo explícitamente si lo desea.

```
SELECT e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.commission DESC;

COMMISSION EMPLOYEE_N
---------- ----------
           BLAKE
           JAMES
      1400 MARTIN
       500 WARD
       300 ALLEN
         0 TURNER

6 rows selected.

SQL>
```

El valor predeterminado se puede modificar especificando **NULLS LAST.**

```
SELECT e.commission, e.employee_name
FROM   employees e
```

```
WHERE  department_id = 30
ORDER BY e.commission DESC NULLS LAST;

COMMISSION EMPLOYEE_N
---------- ----------
      1400 MARTIN
       500 WARD
       300 ALLEN
         0 TURNER
           JAMES
           BLAKE

6 rows selected.

SQL>
```

Cada columna de la cláusula **ORDER BY** puede tener un manejo **NULL** diferente.

```
SELECT e.salary, e.commission, e.employee_name
FROM   employees e
WHERE  department_id = 30
ORDER BY e.salary ASC NULLS FIRST, e.commission DESC NULLS LAST;

    SALARY COMMISSION EMPLOYEE_N
---------- ---------- ----------
       950            JAMES
      1250       1400 MARTIN
      1250        500 WARD
      1500          0 TURNER
      1600        300 ALLEN
      2850            BLAKE

6 rows selected.

SQL>
```

PDF generated by Kalin's PDF Creation Station