Tema 7

Elaboración de consultas básicas de selección

Lenguajes de bases de datos

En esta unidad se abordan cuestiones que, aunque están definidas por el estándar ANSI/ISO SQL, no están asumidas al 100% por todos los fabricantes. Por tanto, pueden existir ligeras diferencias de algunos productos con algunas de las especificaciones que se aquí se exponen.

Un lenguaje de bases de datos deberá pertenecer al menos a uno de los siguientes grupos

- **Lenguaje de definición de datos (DDL).** Se usa para la definición de dominios, tablas, vistas y restricciones de integridad.
- **Lenguaje de manipulación de datos (DML).** Se utiliza para realizar consultas, insertar, borrar o modificar registros o tuplas.
- **Lenguaje de control de datos (DCL).** Gestiona privilegios, autorizaciones, usuarios y permisos.

SQL básico

SQL (Structured Query Language) está compuesto por:

- **Comandos** (verbos) que indican la acción a realizar.SELECT, UPDATE, INSERT, CREATE,…
- **Cláusulas** que especifican condiciones para definir los datos que queremos seleccionar.FROM WHERE ORDER BY, GROUP BY,…
- **Operadores** de comparación y lógicos.AND, OR, < =,…
- **Funciones de agregado**: se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros (AVG, COUNT, SUM, MAX, MIN).

En una sentencia SQL se usan conjuntamente todos estos componentes

CONSULTAS EN SQL.

El formato genérico de una consulta sencilla es:

```
SELECT [ALL / DISTINCT]  [expre_colum1, expre_colum2, .., expre_column | * ]
FROM  [nombre_tabla1, nombre_tabla2, ....., nombre_tablan]
[WHERE condición ]
[ORDER BY expre_colum [DESC I ASC] [, expre_colum [DESC I ASC]...];
```

Donde ***expre_colum*** puede ser una columna de una tabla, una constante, una expresión aritmética, una función o varias funciones anidadas.

EJEMPLO

Por ejemplo:

```
SQL>SELECT apellido, fecha_alta,
    salario, salario + 100000
    FROM empleados;
```

En la condición se especifican operadores lógicos y/o de comparación:

FROM

- FROM *[nombre_tabla1, nombre_tabla2, nombre_tablan]*
- Especifica la tabla o lista de tablas de las que se recuperarán los datos.
- Ejemplo: consultamos los nombres de alumnos y su nota de la tabla ALUMNOS:

```
SQL> SELECT apenom, telef
     FROM alumnos;
```

WHERE

- [WHERE *condición]*
- Obtiene las filas que cumplen la condición expresada. La complejidad de la condición es prácticamente ilimitada. El formato de la condición es: **expresión operador expresión**.
- Las expresiones pueden ser: una constante, una expresión aritmética, un valor nulo o un **nombre de columna**. Los operadores de comparación se pueden observar en el Cuadro siguiente.

Operadores de comparación

=, >, <, >=, <=, i=, <>, !<, !>
IN, NOT IN, BETWEEN, NOT BETWEEN,
ANY, SOME, ALL, EXISTS, NOT EXISTS
LIKE, NOT LIKE, IS NOT, IS NOT NULL

Se pueden construir condiciones múltiples usando los operadores lógicos booleanos estándares: AND, OR y NOT. Se pueden emplear paréntesis para forzar el orden de evaluación

Ejemplo

- WHERE nota = 5
- WHERE (nota>=10) AND (curso=1)
- WHERE (nota IS NULL) OR (UPPER (nom_alum) = 'PEDRO')

ORDER BY

- [ORDER BY *expre_columna* [DESC I ASC] [, expre_columna [DESC I ASC] .. .]
- Especifica el criterio de clasificación del resultado de la consulta. ASC especifica ordenación ascendente, y DESC, descendente.
- Puede contener expresiones con valores de columnas; por ejemplo:

SQL> SELECT * FROM alumnos
ORDER BY nota/5;

ORDER BY

- Así mismo, es posible anidar los criterios. El situado más a la izquierda será el principal.
- **Ejemplo:**

SQL> SELECT * FROM alumnos **ORDER BY** curso desc, nom_alum;

(ordena por CURSO descendente y por NOM_ALUM ascendente).

ALL | DISTINCT

- **ALL**
- Recupera todas las filas, aunque algunas estén repetidas. Es la opción por omisión.
- **DISTINCT**
- Sólo recupera las filas que son distintas. Por ejemplo, consultamos los departamentos (colum-na DEPT_NO) de la tabla EMPLE.

```
SQL> SELECT dept_no FROM emple;
20
30
30
20
30
30
10
20
10
30
20
30
20
10
.....
14 filas seleccionadas.
```

```
SQL> SELECT DISTINCT DEPT_NO FROM EMPLE;
DEPT_NO
---------
      10
      20
      30
```

Select

- El formato de SELECT que nos permite seleccionar las columnas de una tabla es éste:

SELECT [ALL I DISTINCT] [expre_colum1, expre_colum2, **. .,** expre_column | *]
FROM [nombre_ tabla1, nombre_ tabla2, …, nombre_tablan]

SELECCIÓN DE TODOS LOS CAMPOS DE UNA TABLA

- SELECT * FROM TABLA;

SQL> SELECT emp_no, apellido, oficio, dir, fecha_alt, salario, comision, dept_no
FROM emple;

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISIÓN	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/80	104000		20
7499	ARROYO	VENDEDOR	7698	20/02/80	208000	39000	30
7521	SALA	VENDEDOR	7698	22/02/81	162500	65000	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/81	386750		20
7654	MARTÍN	VENDEDOR	7698	29/09/81	162500	182000	30

2. Ponemos *, que representa a todas las columnas de la tabla:

SQL> SELECT * FROM EMPLE ;

Obteniendo así el mismo listado anterior.

Ejemplo

SELECT *
FROM coches
ORDER BY marca, potencia desc;

Ejemplo1

Selección de columnas o campos

Es posible seleccionar solo unos determinados campos:

SELECT campo1,campo2,…, campoX
FROM tabla;

Ejemplo

SQL> SELECT DNOMBRE, DEPT_NO FROM DEPART;

ALIAS DE COLUMNAS

- A veces el nombre de una columna resulta demasiado largo, si resulta demasiado largo, corto o críptico, podemos utilizar un alias de columna. Éste se pone entre comillas dobles a la derecha de la columna. (En Access se emplea la palabra reservada AS)
- SELECT campo1 “alias_campo1”,…
- Ejemplo3

Ejemplo

```
SQL> SELECT  dnombre "Departamento", dept_no "Número Departamento"
     FROM DEPART;
```

Departamento	Número Departamento
CONTABILIDAD	10
INVESTIGACIÓN	20
VENTAS	30
PRODUCCIÓN	40

Operadores aritméticos

- Sirven para formar expresiones con constantes, valores de columnas y funciones de valores de columnas. Devuelven un valor numérico como resultado de realizar los cálculos indicados.

Operador aritmético	Operación
+	Suma
-	Resta
*	Multiplicación
/	División

```
SELECT col1+col2, col1*col2, 45*col1
FROM tabla;
```

Operadores aritméticos. Ejemplo

- Se trata de obtener la nota media de cada alumno, visualizamos por cada uno de ellos su nombre y su nota media.

```
SQL> SELECT nombre_alumno "Nombre
  Alumno", (nota1+nota2+nota3)/3 "Nota Media"
  FROM notas_alumnos ;
```

Operadores aritméticos

Algunos de ellos se pueden utilizar **también con fechas**:

f1 - f2 Devuelve el número de días que hay entre las fechas *f1* y *f2*.

f + n Devuelve una fecha que es el resultado de sumar *n* días a la fecha *f*.

f – n Devuelve una fecha que es el resultado de restar *n* días a la fecha *f*.

Operadores de comparación

Igual =

Distinto != (En Access se utiliza <>)

Menor que <

Mayor que >

Menor o igual <=

Mayor o igual >=

Operadores lógicos: AND, OR y NOT.

AND. Se utiliza cuando queremos que se cumplan las dos condiciones a la vez. La condición resultante será cierta, solo si ambas son ciertas.

SQL> SELECT * FROM EMPLEADOS WHERE **OFICIO = 'VENDEDOR' AND SALARIO > 1500**;

OR. Se utiliza cuando queramos que se cumpla la primera condición, o la segunda o ambas). La condición resultante será cierta si alguna de las dos lo es

SQL> SELECT * FROM EMPLEADOS WHERE **OFICIO = 'VENDEDOR' OR SALARIO > 1500;**

NOT se utiliza cuando no queremos que se cumpla una condición:

SQL> SELECT * FROM EMPLEADOS WHERE NOT OFICIO = 'VENDEDOR';

Combinación de operadores AND y OR

- Estos operadores se pueden combinar de forma ilimitada pero se deben utilizar paréntesis para agrupar aquellas expresiones que deseamos que se evalúen juntas.
- **Select * from emple where salario>2000 and (depart=10 or depart=20);** NO ES IGUAL A
- **Select * from emple where salario>2000 and depart=10 or depart=20;**
- La primera obtendrá empleados de los departamentos 10 y 20 que tengan sueldos mayores a 2000.
- La segunda obtendrá empleados del departamento 10 con sueldo mauor a 2000 y empleados del departamento 20 con cualquier sueldo.

Reglas de precedencia

- Operadores de comparación
- NOT
- AND
- OR
-

Ejemplo

- A partir de la tabla NOTAS_ALUMNOS, deseamos obtener aquellos nombres de alumnos que tengan un 7 en NOTA1 y cuya media sea mayor que 6

```
SQL> SELECT nombre_alumno FROM notas_alumnos
       WHERE nota1=7 AND (nota1+nota2+nota3)/3 >6;

NOMBRE_ALUMNO
--------------------------
Benito Martín, Luis
```

Operadores de comparación de cadenas de caracteres

LIKE: este operador se utiliza para comparar cadenas de caracteres. Permite utilizar caracteres especiales en las cadenas de comparación:

% representa cualquier cadena de 0 o más caracteres

_ representa un único carácter cualquiera

WHERE columna **LIKE** 'caracteres_especiales'

En una sentencia WHERE se pueden usar varias cláusulas LIKE anidadas por operadores AND/OR:

WHERE col1 LIKE 'car_especiales ' AND | OR col2 LIKE 'car_especiales;

Ejemplo

- LIKE 'DIRECTOR' la cadena 'Director'.
- LIKE 'M%' cualquier cadena que empiece por 'M'
- LIKE '%X%' cualquier cadena que contenga una 'X'.
- LIKE '__M' cualquier cadena de 3 caracteres terminada en 'M'.
- LIKE 'N_' una cadena de 2 caracteres que empiece por 'N'
- LIKE '_R%' cualquier cadena cuyo segundo carácter sea una R

Nota

- Hemos de tener en cuenta que las mayúsculas y minúsculas son significativas: 'm' no es lo mismo que 'M' y que las constantes alfanuméricas deben encerrarse siempre entre comillas simples.

Ejemplo

Obtener aquellos apellidos que tengan una 'R' en la segunda posición:

```
SQL> SELECT apellido FROM emple
        WHERE apellido LIKE '_R%';

APELLIDO
----------
ARROYO
```

Ejemplo

Obtener aquellos apellidos que empiecen por 'A' y tengan una 'O' en su interior:

```
SQL> SELECT apellido FROM emple
       WHERE apellido LIKE 'A%O%';

APELLIDO
----------
ARROYO
ALONSO
```

NULL y NOT NULL

- Se dice que una columna de una fila es NULL si está completamente vacía. Para comprobar si el valor de una columna es nulo empleamos la expresión: *columna* IS NULL.
- Si queremos saber si el valor de una columna no es nulo, utilizamos: *columna* IS NOT NULL.
- Cuando comparamos con valores nulos o no nulos no podemos utilizar los operadores de igualdad, mayor o menor.

Cualquier expresión aritmética que contenga algún valor nulo retornará un valor nulo.

Salario*comision retornará valor NULL si alguno de los dos operandos es NULL

Ejemplo

- Si queremos consultar los apellidos de los empleados cuya comisión no sea nula teclea-remos esto:
- SQL> SELECT apellido FROM emple
  WHERE comision IS NOT NULL;

Comprobaciones con conjuntos de valores

- Hasta ahora, todas las comprobaciones lógicas que hemos visto comparan una columna o expresión con un valor. Ejemplo: OFICIO = 'ANALISTA' AND DEPT_NO=10;
- Pero también podemos comparar una columna o una expresión con una lista de valores utilizando los operadores IN y BETWEEN.

IN

- El operador IN nos permite comprobar si una expresión pertenece o no (NOT) a un conjunto de valores, haciendo posible la realización de comparaciones múltiples; su formato es:
- <expresión> [NOT] **IN** (lista de valores separados por comas).

Ejemplo

- Consultar los apellidos de la tabla EMPLE cuyo número de departamento sea 10 ó 30:

```
SQL> SELECT apellido FROM emple
        WHERE dept_no IN (10,30);
```

Ejemplo

- Consultar los apellidos de la tabla EMPLE cuyo número de departamento no sea ni 10 ni 30:

```
SQL> SELECT apellido FROM emple
        WHERE dept_no NOT IN(10,30);
```

Ejemplo

- Consultar los apellidos de la tabla EMPLE cuyo oficio sea 'VENDEDOR', 'ANALISTA' o 'EMPLEADO':

```
SQL> SELECT apellido FROM emple
 WHERE oficio IN
   ('VENDEDOR','ANALISTA','EMPLEADO');
```

Ejemplo

- Consultar los apellidos de la tabla EMPLE cuyo oficio no sea ni 'VENDEDOR' ni 'ANALISTA' ni 'EMPLEADO':

```
SQL> SELECT apellido FROM emple
        WHERE oficio NOT IN
('VENDEDOR','ANALISTA','EMPLEADO');
```

BETWEEN

Este operador comprueba si un valor está comprendido o no dentro de un rango de valores, desde un valor inicial a un valor final.

[NOT] BETWEEN valor_ini AND valor_fin

Ejemplo

- A partir de la tabla EMPLE, obtener el apellido y el salario de los empleados cuyo SALARIO esté comprendido entre 150000 y 200000:
- SQL> SELECT apellido, salario FROM emple
  WHERE salario BETWEEN 150000 AND 200000;

Ejemplo

- A partir de la tabla EMPLE, obtener el apellido y el salario de los empleados cuyo SALARIO no esté comprendido entre 150000 y 200000
- SQL> SELECT apellido, salario FROM emple

WHERE salario NOT BETWEEN 150000 AND 200000;

Nota

Un error relativamente frecuente consiste en utilizar expresiones del tipo:

1000 <= SALARIO <= 2000

Este tipo de expresiones es ilegal y provocará un error ya que al evaluar la primera parte de la expresión se sustituirá por un valor lógico de tipo *true/false* y este resultado no puede compararse con un valor numérico. La expresión correcta sería:

SALARIO BETWEEN 1000 AND 2000

O bien:

SALARIO >= 1000 AND SALARIO <= 2000

Operadores de concatenación

Para unir dos o más cadenas se utiliza el operador de concatenación ||

Ej.: 'buenos' || 'días' daría como resultado 'buenosdias'

En Access para concatenar cadenas se utiliza el signo +, no reconoce ||

Constantes

Constantes numéricas.

Construidas mediante una cadena de dígitos que puede llevar un punto decimal, y que pueden ir precedidos por un signo + ó -.

También se pueden expresar constantes numéricas empleado el formato de coma flotante. Ej.: 34.345E-8).

Constantes de cadena.

Consisten en una cadena de caracteres encerrada entre comillas simples. (Ej.: 'Hola Mundo').

Constantes de fecha.

En realidad las constantes de fecha se escriben como constantes de cadena sobre las cuales se aplicarán las correspondientes funciones de conversión (TO_DATE) o bien, el gestor de la base de datos realizará una conversión automática de tipo. (Ej.: '27-SEP-1997').

Precedencia o prioridad en los operadores.

Prioridad	Operador	Operacion
1º	**, NOT	exponenciación, negación
2º	*, /	multiplicación, división
3º	+, -, ||	suma, resta, concatenación. Para concatenar Access se utiliza el signo +.
4º	= , != , < , > , <= , >= , IS NULL , LIKE , BETWEEN , IN	Comparación. En Access la comparación != es <>
5º	AND	conjunción
6º	OR	inclusión