MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

Secretaría General de Educación y Formación Profesional
Dirección General de Educación, Formación Profesional e Innovación Educativa
Subdirección General de Acción Educativa

BREVE INTRODUCCIÓN AL SQL
Aplicación al Programa Escuela

Última actualización: 17/10/2000

http:/atenea.pntic.mec.es/escuela/
escuela@atenea.pntic.mec.es

ÍNDICE:

BREVE INTRODUCCION AL "S Q L"

Nociones del lenguaje

Para el manejo de los datos que existen en las diferentes tablas que componen una **Base de Datos Relacional**, existe un lenguaje muy cómodo y útil denominado SQL, que responde a las palabras inglesas de ***"Lenguaje de Preguntas Estructuradas"***. Este lenguaje tiene capacidad para muy diversas operaciones a realizar sobre una base de datos, las tablas que la componen y los datos que almacenan dichas tablas. No obstante, vamos a centrarnos solamente en la parte correspondiente de SQL que permite la realización y visualización de consultas sobre los datos contenidos en la Base de Datos del Programa Escuela.

Para comprender bien el modo de realización de estas consultas, se considera necesario conocer los **conceptos básicos** a utilizar más adelante:

- Una **Base de Datos Relacional** almacena información en estructuras básicas denominadas TABLAS.
- Una **tabla** es una estructura sobre la que se almacena información. La tabla está compuesta por columnas y filas.
- Una **columna** (campo) es la representación de un atributo, propiedad o característica de una tabla.
- Una **fila** (registro) es una ocurrencia en una tabla.

Los nombres de las columnas no pueden estar repetidos en una tabla.

Los nombres de las tablas no pueden estar repetidos en una base de datos.

Las tablas que vamos a utilizar para los ejemplos son:

Tabla *ALUMNOS*

Contenido: Alumnos en alta en el centro

nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her_*
Juan	Pérez	Martín	P1A	28	23/08/1992	03
José	García	Gómez	P1A	34	12/02/1991	02
Pedro	San	Sánchez	P1C	37	30/04/1993	03
Luis	Ponce	Primo	S2A	28	05/11/1994	01
Tomas	López	Baruque	P2F	37	25/05/1993	04

Cada columna de la tabla *ALUMNOS* contiene los siguientes datos:

nomb : Nombre del alumno. Tipo CHAR (carácter).
ape1: Primer apellido del alumno. Tipo CHAR (carácter).
ape2: Segundo apellido del alumno. Tipo CHAR (carácter).
grupo: Compuesto por Nivel educativo, curso académico y aula. Tipo CHAR.
prv_nac: Código de la provincia de nacimiento del alumno. Tipo CHAR.
f_naci: Fecha de nacimiento, compuesta de día mes y año. Tipo DATE (fecha).
n_her: Es el número de hermanos que son (contando el alumno). Tipo NUMERICO.

Tabla *PROFEDG*

Contenido: **Profesores** en alta en el centro

t_nbt	*f_naci*	*ape1*	*ape2*	*nomb*
01	22/12/1974	Pérez	Díaz	Pedro
02	13/03/1962	López	López	Angel
03	28/09/1941	García	Nuñez	Tomas

t_nbt : Tipo de nombramiento. Tipo CHAR

El resto de las columnas tienen las mismas características que en la tabla *ALUMNOS.*

Tabla *PERSONAL*

Contenido: Otros personal en alta en el centro

dni	*f_naci*	*ape1*	*ape2*	*nomb*
11111111	22/12/1974	García	García	Pedro
33333333	01/12/1963	García	Rivera	José
44444444		Anelca	Cuentista	Nicolás
55555555	16/06/1966	Rouco	Varela	Eduardo

Las características de las columnas son las mismas que en la tabla *ALUMNOS*.

Tabla *LPROFTNO*

Contenido: Tipos de nombramiento del personal (profesores y personal no docente)

clave	*literal*
01	Definitivo
02	Provisional
03	Interino(t.indefinido)
04	Interino(sustitución)

clave: Código del tipo de nombramiento del profesor. Coincide con la columna *t_nbt* de la taba *PROFEDG* (Profesores).

Veamos un ejemplo:

El conjunto de todos los datos es la Tabla que llamamos *ALUMNOS* , está compuesta por las columnas *nomb, ape1, ape2, grupo, prv_nac, f_nac y n_her* . Son siete características de la tabla, y esta tabla está compuesta por cinco filas u ocurrencias, cada una de las cuales define las características de un alumno en particular.

Cada tabla de datos puede tener un número determinado de columnas que se definen en su creación y pueden ser aumentadas posteriormente.

El número de filas no se determina en la creación; es el resultado del número de registros (alumnos en éste ejemplo) que se hayan introducido en la tabla.

Todos los valores que contiene cada columna tiene las mismas características (son del mismo tipo): son alfanuméricos, numéricos, binarios, date (tipo fecha) etc.

La Base de Datos Relacional puede contener un número indeterminado de tablas y algunas de ellas guardarán relación con una o más tablas de la Base.

La parte del lenguaje SQL que vamos a utilizar, actuará sobre las tablas definidas en nuestra Base de Datos con unas características similares a las definidas anteriormente. Es decir, tendrá filas y columnas.

Para realizar consultas sobre los datos que almacenan las tablas, utilizaremos SENTENCIAS SQL y se realizan por asociación de valores, columnas, no por la localización física de los datos. Para ver ejemplos de consultas SQL pueden observarse las definidas en el Programa **ESCUELA**.

Una **SENTENCIA** está compuesta de **CLÁUSULAS.**

Una de las SENTENCIAS más útiles en SQL es la llamada **SENTENCIA SELECT** y comprende las cláusulas siguientes:

Cláusula **SELECT:** indica las columnas de deseamos obtener.

Cláusula **FROM:** indica la tabla desde donde se obtienen los datos de las columnas.

Cláusula **WHERE:** indica las condiciones que deseamos aplicar para la selección.

Cláusula **GROUP BY:** indica los agrupamientos que se quieren realizar.

Cláusula **HAVING:** indica las condiciones del agrupamiento.

Cláusula **ORDER BY:** indica los campos por donde deseamos ver ordenada la salida de la **SELECT**.

Cláusula **UNION:** permite concadenar el resultado de dos o más sentencias **SELECT**.

Hay que tener en cuenta en adelante, que unas veces tratamos de **SENTENCIA SELECT** y otras veces tratamos de **CLÁUSULA SELECT**. La diferencia es muy clara, **SENTENCIA SELECT** es el conjunto de todas las **CLÁUSULAS** y una de esas **CLÁUSULAS** también se denomina **SELECT**.

Más adelante desarrollaremos cada uno de estos apartados denominados cláusulas.

Las **PALABRAS RESERVADAS** son aquellas, en lengua inglesa, que no podemos utilizar para asignar nombres, ya que su uso puede originar contratiempos en la ejecución de las sentencias SQL.

Observará que en las cuatro cláusulas hay unas palabras en inglés, que es obligatorio utilizar correctamente y que comentamos a continuación tomando como ejemplo la tabla Alumnos anteriormente definida.

SELECT *nomb,ape1,ape2,grupo*

Cláusula SELECT:

Toda sentencia **SELECT** tiene que comenzar obligatoriamente con la palabra **SELECT.** A continuación, y separado por un blanco, hay que indicar las columnas que deseamos seleccionar, separadas éstas por una coma.

Podría quedar de la siguiente forma:

SELECT *nomb, ape1, ape2 ,GruPo*

Observará que las palabras tienen mezcla de letras mayúsculas y minúsculas, eso no tiene ninguna importancia para la realización de la operación de la **SELECT**; también se observa que la separación entre las palabras tienen más de un espacio, tampoco afecta nada, lo que sí es importante es dejar como mínimo un espacio en blanco separando la palabra **SELECT** de las columnas a seleccionar; los nombres de las columnas tienen que llevar una coma de separación aunque pueden dejarse espacios en blanco entre ellos.

En adelante, para mejor comprender cuales son las palabras de las cláusulas **SELECT**, pondremos en los ejemplos con **Mayúsculas negrita** las palabras claves obligatorias, los nombres de las columnas en ***minúsculas cursiva***, operadores y valores con letra **minúscula negrita** y los nombres de las tablas con ***Mayúsculas cursiva.***

Cláusula FROM:

Es la parte de la sentencia **SELECT** donde se indica desde qué tabla o tablas se van a obtener los datos a seleccionar.

FROM *ALUMNOS*

Cuando se utilizan más de una tabla como fuente de datos, los nombres de las tablas se separan con una coma.

FROM *ALUMNOS, PROFEDG*

Estas dos cláusulas (partes) de la **SELECT** son las mínimas necesarias para que se ejecute la sentencia.

Con éste ejemplo obtendríamos como resultado parte de las columnas de la tabla alumnos, que corresponde al ejemplo que vimos al principio.

SELECT *nomb,ape1,ape2,grupo*
FROM *ALUMNOS*

nomb	***ape1***	***ape2***	***grupo***
Juan	Pérez	Martín	P1A
José	García	Gómez	P1A
Pedro	San	Sánchez	P1C
Luis	Ponce	Primo	S2A
Tomas	López	Baruque	P2F

Cláusula WHERE:

Cuando queremos obtener una parte específica de la tabla, utilizaremos la cláusula **WHERE,** compuesta por la palabra **WHERE** continuada con un blanco y seguida por la condición que se desee aplicar. Por ejemplo, si solamente queremos seleccionar los registros correspondientes al grupo de alumnos de Primaria, primer curso, letra A, la cláusula **WHERE** seria de esta forma:

WHERE *grupo* = 'P1A'

La sentencia completa seria la siguiente:

SELECT *nomb, ape1, ape2, grupo*
FROM *ALUMNOS*
WHERE *grupo* = 'P1A'

La cláusula **WHERE** no es obligatoria, solamente la utilizaremos cuando deseemos seleccionar un subconjunto de filas de una tabla.

Toda la sentencia puede ir en una línea, teniendo en cuenta las separaciones con espacio en blanco entre las palabras claves de cada cláusula, quedando:

SELECT *nomb,ape1,ape2,grupo* **FROM** *ALUMNOS* **WHERE** *grupo* = 'P1A'

Pero es más cómodo de identificar la sentencia **SELECT** si escribimos las diferentes cláusulas empezando en una nueva línea. Cualquier cláusula puede utilizar mas de una línea.

Todo esto no afecta al funcionamiento de la **SELECT**, pero resulta más estético y cómodo para identificar todas las cláusulas al primer golpe de vista y detectar más fácilmente algún posible error en la escritura de la sentencia.

Cláusula GROUP BY:

Si deseamos que los datos obtenidos se presenten con un determinado orden, utilizaremos la cláusula **ORDER BY** indicando la columna o columnas por donde se quiera ordenar. Supongamos que en el ejemplo que llevamos queremos que el orden sea por grupo y dentro de grupo, alfabéticamente por el primer apellido. La sentencia **SELECT** con la cláusula **ORDER BY** quedaría así:

```
SELECT     nomb, ape1, ape2, grupo
FROM       ALUMNOS
WHERE      grupo='P1A'
ORDER BY   grupo, ape1
```

El resultado es el siguiente:

nomb	***ape1***	***ape2***	***grupo***
José	García	Gómez	P1A
Juan	Pérez	Martín	P1A
Pedro	San	Sánchez	P1C
Tomas	López	Baruque	P2F
Luis	Ponce	Primo	S2A

Hasta aquí hemos visto un ejemplo sencillo para ver cómo funciona la sentencia **SELECT.** En adelante vamos a completar todas las cláusulas que hemos visto y añadiremos las que faltan por ver.

CLÁUSULA SELECT

Anteriormente se dijo que la cláusula **SELECT** se utiliza para indicar las columnas que deseamos seleccionar de una o varias tablas. Veamos por ejemplo cómo seleccionar todas las columnas de la tabla *ALUMNOS* :

El modo más literal de hacerlo sería:

SELECT *nomb, ape1, ape2, grupo, prv_nac, f_naci, n_her*

Ahora bien, al coincidir que las columnas seleccionadas de la tabla son **todas,** se puede indicar mediante el uso de un asterisco que, de esta manera, sustituye a la lista de todas las columnas:

SELECT *

Suponiendo que la selección se hiciera desde la tabla *ALUMNOS*, el resultado seria el mismo que en el ejemplo anterior.

Cuando se seleccionan todos los campos de una tabla utilizando el * no se pueden seleccionar mas columnas.

También podemos incluir en la selección varias veces la misma columna p.e.:

SELECT *nomb, ape1, ape2, grupo, prv_nac, f_naci, n_her, nomb, nomb, ape2*

En esta **SELECT** tendremos repetida tres veces la columna *nomb* y dos veces la columna *ape2.* Cuando veamos el concepto de **alias,** se indicará cómo obviar los nombres repetidos.

Cada base de datos tiene un propietario, que es quien puede utilizarla. Cada propietario puede autorizar a otras personas para que puedan acceder a su base de datos. Para indicar que a la tabla que se accede tiene otro propietario se pone el nombre del propietario delante del nombre de la tabla separado por un punto. El ejemplo siguiente indica que el propietario es **escuela** y el nombre de la tabla es *ALUMNOS.*

FROM escuela.*ALUMNOS*

Esta situación es posible que no se le presente nunca, pues lo general es que solamente trabaje con una base de datos, que será la suya.

Muchas veces necesitaremos obtener datos de mas de una tabla y puede darse que algún nombre de columna se repita en más de una tabla. Para que no exista confusión sobre la tabla de la que queremos sacar la columna, lo indicamos anteponiendo el nombre de la tabla al de la columna:

SELECT *ALUMNOS.nomb, ALUMNOS.ape1, ALUMNOS.ape2,*
PROFEDG.nomb, PROFEDG.ape1, PROFEDG,.ape2
FROM *ALUMNOS, PROFEDG*

Concepto de *Alias:*

Un **alias** de una tabla o de una columna es un nuevo nombre que se le da a la tabla o a la columna para poder nombrarla en lugar de su auténtico nombre. Principalmente se utiliza para abreviar el nombre, para denominar una concatenación de columnas, para denominar una substracción de columna o para dar nombres más significativos a tablas o columnas. El dominio del **alias** es sólo para la sentencia **SELECT** y su resultado.

El nombre de un alias no puede contener blancos intermedios.

Para evitar repetir el nombre de la tabla delante de cada nombre de columna, podemos utilizar el **alias** de cada tabla en lugar de su nombre. El alias de cada tabla se pone detrás del nombre de la tabla en la cláusula **FROM** separado por un blanco, como mínimo, y delante de las columnas separadas por un punto en la cláusula **SELECT**.

SELECT a.*nomb*,**a**.*ape1*, **a**.*ape2*,

***b**..nomb*, **b**.*ape1*, **b**.*ape2*
FROM *ALUMNOS* **a**, *PROFEDG* **b**

En este ejemplo el alias de la tabla *ALUMNOS* es **a** y el alias de la tabla *PROFEDG* es **b**.

También podemos denominar con **alias** la columna o expresión que obtengamos con la sentencia **SELECT** poniendo a continuación del nombre de la columna o expresión ***as*** y el nuevo nombre:

SELECT *nomb, ape1, ape2, n_her-1* as **hermanos_de**
FROM *ALUMNOS*

De esta forma a los resultados obtenidos, n_her-1, les asignará el nombre de **hermanos_de**, que podemos utilizar en la **SELECT** y en procesos posteriores vinculados a ésta **SELECT**. El resultado es:

nomb	***ape1***	***ape2***	***hermanos_de***
Juan	Pérez	Martín	2
José	García	Gómez	1
Pedro	San	Sánchez	2
Luis	Ponce	Primo	0
Tomas	López	Baruque	3

La utilización de la partícula **as** no es necesaria; basta con el nombre del alias, pero ante cualquier duda o error nos indicará que hemos asignado un alias.

En ocasiones nos interesará aplicar alguna función a alguna o algunas de las columnas seleccionadas.

En algunas circunstancias necesitaremos conocer los distintos valores que existan en alguna determinada columna, para ello utilizaremos la cláusula **DISTINCT** de la siguiente manera:

SELECT distinct(*prv_nac*)
FROM *ALUMNOS*

Con esto obtendremos los diferentes valores que contiene la columna *prv_nac* de la tabla Alumnos.

En éste caso, el resultado es:

Prv_nac
28
34
37

La **cláusula DISTINCT**, si se incluye, debe preceder a la expresión o columna a que se refiere, que irá encerrada entre paréntesis.

En otras situaciones podemos necesitar saber cuantos registros hay de un determinado valor o la suma de los valores contenidos en una columna, o el valor máximo o mínimo que encontramos en dicha columna.

Para ello utilizamos las **funciones.**

Por ejemplo, si queremos saber cuantos alumnos hay de una determinada provincia (*prv_nac*), lo obtendríamos de la siguiente manera:

SELECT **count**(prv_nac)
FROM *ALUMNOS*
WHERE *prv_nac*='28'

El resultado es:

Count of prv_nac
2

La función **count** en este ejemplo está asociada a la cláusula **WHERE** pero puede servirnos también para contar todos los registros que tiene una tabla:

SELECT count(*) **FROM** *ALUMNOS*

El resultado es:

count
5

También para contar cuantos valores no nulos (rellenos) tiene una columna de la forma siguiente:

SELECT count(*f_naci*) **FROM** *PERSONAL*

El resultado es:

count
3

(Hay un registro que tiene un valor NULO (no relleno) en la columna *f_naci* y la función **count** no cuenta los valores nulos).

Otras funciones comunes:

MAX: nos permite obtener el valor máximo contenido en una columna.

MIN: para el valor mínimo

Por ejemplo:

SELECT max(*n_her*) **FROM** *ALUMNOS*

El resultado es:

Max of n_her
4

En éste caso nos daría como resultado 4, porque es el número de hermanos máximo que existe en la tabla ALUMNOS.

AVG: se obtiene el valor medio de los datos numéricos de una columna.

SUM: se obtiene la suma de todos los valores de una columna numérica

Las funciones **AVG** y **SUM** solamente podemos utilizarlas con columnas de tipo numérico.

A las funciones **COUNT, SUM, AVG, MAX y MIN** se las denomina **funciones de grupo**.

Las funciones **MAX** y **MIN** sobre columnas de contenido alfanumérico devuelven el valor mayor o menor en el orden alfabético, sin diferenciar entre mayúsculas o minúsculas.

CLÁUSULA FROM

Anteriormente hemos visto para qué sirve esta cláusula y cómo se completa. Conviene recordar que **es obligatoria** para cualquier consulta; sin ella no puede ejecutarse la SELECT.

El formato de esta cláusula es:

FROM propietario.*nombretabla* [alias_*TABLA*] ...

Propietario corresponde al nombre del propietario de la tabla. Se omite cuando todas las tablas que se utilizan pertenecen al mismo propietario, que es nuestra situación. En adelante no lo utilizaremos.

Nombre de la tabla corresponde al nombre por el que se conoce a la tabla dentro de la Base de Datos. Este dato **sí es obligatorio**.

Alias.*TABLA* es un nombre que se usa para referirse a la tabla en el resto de la sentencia **SELECT,** para abreviar el nombre original y hacerlo más manejable, en el caso de existir más de una tabla en la consulta.

Anteriormente hemos visto un ejemplo de cómo se maneja el alias de la tabla.

CLÁUSULA WHERE

La cláusula **WHERE** dice a SQL qué condiciones tienen que cumplir las filas o registros de datos de una tabla para ser seleccionados por la cláusula **SELECT**.

El formato de esta cláusula es:

WHERE *expresión1* *operador* ***expresion2***

expresión1 y ***expresion2*** pueden ser nombres de campos, valores constantes o expresiones.

operador es un operador relacional que une las dos expresiones.

En esta cláusula **WHERE** no se pueden utilizar alias dados a columnas y expresiones en la cláusula **SELECT;** sí pueden en cambio utilizarse los dados a las tablas en la cláusula **FROM**.

Las columnas, constantes o expresiones que se utilicen en la cláusula **WHERE** no es necesario que figuren en la cláusula **SELECT.**

Más adelante veremos los distintos operadores que se puede utilizar.

Esta cláusula tiene que utilizarse siempre que se necesiten realizar algunas restricciones.

En el ejemplo siguiente vamos a utilizar como condición la columna *prv_nac* que no figura en la cláusula **SELECT:**

SELECT *nomb, ape1, ape2, grupo*
FROM *ALUMNOS*
WHERE *prv_nac* = '28' ***and*** *grupo* = '**S2A**'

El resultado es:

nomb	***ape1***	***ape2***	***grupo***
Luis	Ponce	Primo	S2A

puesto que es el único registro que cumple las dos condiciones.

Veamos un ejemplo en el que utilicemos dos tablas

SELECT a.*, b.*literal*
FROM *PROFEDG* **a,** *LPROFTNO* **b**

A la tabla *PROFEDG* se le asigna el alias **a** y a la tabla *LPROFTNO* el alias **b** en la cláusula **FROM**. Para seleccionar todas las columnas de la tabla *PROFEDG* lo indicamos en la cláusula **SELECT** con el alias.* .

De la otra tabla *LPROFTNO* solamente seleccionamos la columna *literal*; aunque utilizamos el alias para esta tabla, en este caso podríamos prescindir de él.

El resultado es:

t_nbt	***f_naci***	***ape1***	***ape2***	***nomb***	***literal***
01	22/12/1974	Pérez	Díaz	Pedro	Definitivo
01	22/12/1974	Pérez	Díaz	Pedro	Provisional
01	22/12/1974	Pérez	Díaz	Pedro	Interino (t.indefinido)
01	22/12/1974	Pérez	Díaz	Pedro	Interino (sustitución)
02	13/03/1962	López	López	Angel	Definitivo
02	13/03/1962	López	López	Angel	Provisional
02	13/03/1962	López	López	Angel	Interino (t.indefinido)
02	13/03/1962	López	López	Angel	Interino (sustitución)
03	28/09/1941	García	Nuñez	Tomás	Definitivo
03	28/09/1941	García	Nuñez	Tomás	Provisional
03	28/09/1941	García	Nuñez	Tomás	Interino (t.indfinido)
03	28/09/1941	García	Nuñez	Tomás	Interino (sustitución)

Vemos que con cada registro de la tabla *PROFEDG* se selecciona uno por cada registro de la tabla *LPROVTNO* con la columna literal diferente. Es debido a que se realiza el producto cartesiano entre las dos tablas al no tener definida ninguna relación de proceso entre las dos tablas.

Posiblemente no es este el resultado apetecido. Es posible que queramos que a cada profesor corresponda un único registro y en el mismo se indique el tipo de destino del mismo.

Normalmente utilizamos dos o más tablas en una consulta cuando la información recogida en una de ellas se completa con la contenida en la segunda. Por ejemplo, la tabla *PROFEDG* tiene los datos de los profesores y entre ellos una columna que contiene el tipo de nombramiento en forma de código, llamada *t_nbt* y la tabla *LPROFTNO* contiene una columna llamada *literal*, con los literales correspondientes a los tipos de nombramiento y otra llamada *clave* que contiene el código del tipo de nombramiento.

En este caso debemos establecer la **relación** entre las dos tablas, utilizando para ello la columna de contenido común a ambas tablas; *t_nbt*, en el caso de la tabla *PROFEDG*, y *clave* en la tabla *LPROFTNO*. Esta relación se lleva a cabo en cláusula **WHERE**

Así, si queremos obtener, entre otros datos, el literal del tipo de nombramiento de cada profesor tendremos que enlazar las dos tablas por las columnas *t_nbt y clave* , de la manera que se indica:

SELECT a.*,**b**.*literal*
FROM *PROFEDG* **a,***LPROFTNO* **b**
WHERE *a.t_nbt=b.clave*

El resultado ahora , más acorde con lo que se pretende, es:

t_nbt	***f_naci***	***ape1***	***ape2***	***nomb***	***literal***
01	22/12/1974	Pérez	Díaz	Pedro	Definitivo
02	13/03/1962	López	López	Angel	Provisional
03	28/09/1941	García	Nuñez	Tomas	Interino (t.indfinido)

Ahora el literal de la tabla *LPROFTNO* se corresponde a la columna *t_nbt* de la tabla *PROFEDG* según el valor de la columna *clave* de la tabla *LTROFTNO*.

A la **relación** o **enlace** de dos tablas para la realización de una consulta se le denomina **JOIN**.

Si al utilizar dos tablas no indicamos la relación o **JOIN** que tienen las tablas entre sí, se producirá el producto cartesiano y se obtendrán por cada registro de la primera tabla indicada tantas selecciones como registros tenga la segunda tabla, tal y como se vio en un ejemplo anterior. Por eso es necesario no olvidarse de indicar el **JOIN** con las columnas correctas de relación.

Si utilizamos más de dos tablas, tendremos que relacionar cada tabla con una o varias de las otras.

El tipo de combinación más utilizado es el llamado **Simple JOIN o EquiJoin** que devuelve filas de dos o más tablas basándose en una condición de igualdad, siendo por tanto el criterio de coordinación , establecido por el operador igual **(=)**

Veamos un ejemplo:

SELECT **a**.*nomb*, **a**.*ape1*, **a**.*ape2*, **b**.*nomb*, **b**.*ape1*, **b**.*ape2*, **c**.*nomb*, **c**.*ape1*, **c**.*ape2*
FROM *ALUMNOS* **a,***PROFEDG* **b,** *PERSONAL* **c**
WHERE **a**.*nomb*=**b**.*nomb* **and a**.*nomb*=**c**.*nomb*

Obtenemos todos los nombres y los apellidos de las tablas *ALUMNOS***,** *PROFEDG* **y** *PERSONAL* cuando los nombres son idénticos en las tres tablas. El registro de cada tabla que cumpla con esta doble igualdad (relación), será seleccionado.

El resultado es:

a.*nomb*	**a.*ape1***	**a.*ape2***	**b.*nomb***	**b.*ape1***	**b.*ape2***	**c.*nomb***	**c.*ape,***	**c.*ape2***
Pedro	San	Sánchez	Pedro	Pérez	Díaz	Pedro	García	García

CLÁUSULA GROUP BY

Esta cláusula se utiliza para agrupar las filas seleccionadas y devolver una sola fila, resumiendo la información solicitada. Especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la tabla, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria (resultados) para cada grupo.

En ésta cláusula **GROUP BY** no se pueden utilizar alias.

Seguido de las palabras **GROUP BY** se especifican los nombres de una o más columnas cuyos resultados se desean agrupados. Tiene la forma:

GROUP BY ***expresión_****columna*

expresión_*columna* debe coincidir con la expresión o columna utilizada en la cláusula **SELECT**. Puede ser uno o más nombres de campo de una tabla, separados por coma o una o más expresiones separadas por comas. Si la sentencia **SELECT** contiene la cláusula **GROUP BY** , la lista de la cláusula **SELECT** solamente puede contener constantes, funciones de grupo y expresiones, idénticas a las que estarán en la cláusula **GROUP BY** . El siguiente ejemplo nos dice cuantos alumnos hay en cada grupo existente en el centro:

SELECT *grupo***, count(*)**
FROM *ALUMNOS*
GROUP BY *grupo*

Esta sentencia nos devolverá una fila por cada grupo diferente que exista en la columna grupo de la tabla *ALUMNOS* . Cada una de ellas contendrá el grupo y el número de alumnos en él, el resultado es el siguiente:

grupo	***número de alumnos***
P1A	2
P1C	1
P2F	1
S2A	1

El ejemplo anterior, pero contando el número de alumnos por grupo que tengan algún hermano en el centro, se indicaría de la siguiente manera:.

SELECT *grupo***, count(*)**

```
FROM      ALUMNOS
WHERE     n_her > 1
GROUP BY grupo
```

El resultado será:

grupo	número de alumnos
P1A	2
P1C	1
P2F	1

El grupo S2A no aparece por no existir ningún alumno de dicho grupo con algún hermano (*n_her*=1 en el centro.

CLÁUSULA HAVING

Esta cláusula especifica la condición que deben de cumplir los grupos. Sólo es válida si previamente se ha especificado la cláusula **GROUP BY**. Es decir, indica al SQL que incluya sólo ciertos grupos producidos por la cláusula **GROUP BY** en los resultados de la consulta. Al igual que la cláusula **WHERE**, utiliza una condición de búsqueda para especificar los grupos deseados.

La cláusula **HAVING** tiene la forma:

HAVING expresión1 *operador* **expresión2**

expresión1 y **expresión2** pueden ser nombres de campos, valores constantes o expresiones y estas no deben coincidir con una expresión de columna en la cláusula **SELECT**.

operador es un operador relacional que une las dos expresiones. Más adelante veremos los distintos operadores que se puede utilizar.

En esta cláusula **HAVING** sólo se pueden utilizar alias de tablas definidos en la cláusula **FROM**.

La sentencia siguiente nos mostrará el número de alumnos en cada grupo que tengan algún hermano (en *n_her* se cuenta al alumno) y que el número de integrantes de cada grupo seleccionado sea superior a 1:

```
SELECT    grupo, count(*)
FROM      ALUMNOS
WHERE     n_her > 1
GROUP BY grupo
HAVING    count(*) >1
```

El resultado es:

grupo	número de alumnos
P1A	2

Los grupos P1C y P2F no son seleccionados porque el número de registros existentes en cada grupo no es superior a 1 como indica la cláusula **HAVING**.

Si la sentencia **SELECT** tiene la cláusula **WHERE,** primero se filtran todas las filas que no satisfagan la condición, después se realiza el agrupamiento según la cláusula **GROUP** BY y finalmente se verifican los agrupamientos con la cláusula **HAVING** para rechazar los grupos que no la cumplan.

CLÁUSULA ORDER BY

Esta cláusula indica el orden en el que queremos ver los resultados de la consulta, en base a los datos de una o más columnas. Si se omite, los resultados saldrán ordenados por el orden en que los regisatros fueron introducidos en la tabla (orden natural).

Tiene la forma:

ORDER BY {*expresión_orden* [DESC | ASC], ...]

expresión_orden puede ser el nombre de un campo, expresión o el número de posición que ocupa la expresión de columna en la cláusula **SELECT**. Por defecto se ordenan ascendentemente (**ASC**, de menor a mayor). Si se deseara de mayor a menor se empleará **DESC** (descendente).

Por ejemplo, para mostrar los alumnos ordenados de mayor a menor según la provincia y, dentro de la provincia, por orden alfabético ascendente del nombre, se utilizaría:

SELECT *nomb, ape1, grupo, prv_nac pro, f_naci*
FROM *ALUMNOS*
ORDER BY *pro* ***desc,*** *nomb*

El resultado es:

nomb	***ape1***	***grupo***	***pro***	***f_naci***
Pedro	San	P1C	37	30/04/1993
Tomás	López	P2F	37	25/05/1993
José	García	P1A	34	12/02/1991
Juan	Pérez	P1A	28	23/08/1992
Luis	Ponce	S2A	28	05/11/1994

En la columna *prv_nac* de la cláusula **SELECT** hemos empleado el alias *pro*, que utilizamos también en la cláusula **ORDER BY** y aparece, en consecuencia, en la cabecera del listado.

Si la cláusula **ORDER BY** fuera

ORDER BY *pro, nomb*

El resultado sería:

nomb	***ape1***	***grupo***	***pro***	***f_naci***
Juan	Pérez	P1A	28	23/08/1992
Luis	Ponce	S2A	28	05/11/1994
José	García	P1A	34	12/02/1991
Pedro	San	P1C	37	30/04/1993
Tomas	López	P2F	37	25/05/1993

El orden también se puede indicar con el número de orden que tienen las columnas seleccionadas en la cláusula **SELECT.**

En el ejemplo anterior la cláusula **ORDER BY** tendría el mismo efecto si indicamos:

ORDER BY 4, 1

ya que la columna *prv_nac* (alias *pro*) ocupa el orden 4 en la **SELECT** y la columna *nomb* ocupa el orden 1.

Este método de indicar el **ORDER BY** es necesario cuando se utiliza la cláusula **UNION** , como veremos más adelante .

CLÁUSULA UNION

La cláusula **UNION** combina el resultado de dos o más sentencias **SELECT** en un único resultado. Este resultado se compone de todos los registros devueltos en ambas sentencias.

Si ponemos solamente **UNION**, los registros repetidos se omiten, seleccionado sólo uno de ellos.

Para que tenga en cuenta los registros repetidos y no repetidos tenemos que emplear **UNION ALL**.

Tiene la forma:

SELECT *sentencia* **UNION [ALL] SELECT** ***sentencia*** ...

Cuando se utilice el operador **UNION**, la lista de selección para cada sentencia **SELECT** debe tener el mismo número de expresiones de columnas con el mismo tipo de datos y en el mismo orden.

Por ejemplo,

SELECT *nomb*, *f_naci* **FROM** *ALUMNOS*
UNION
SELECT *nomb*, *f_naci* **FROM** *PROFEDG*
UNION
SELECT *nomb*, *f_naci* **FROM** *PERSONAL*

El resultado es el siguiente:

nomb	***f_naci***
Ángel	13/03/1962
Eduardo	16/06/1966
José	01/12/1963
José	12/02/1991
Juan	23/08/1992
Luis	05/11/1994
Nicolás	
Pedro	22/12/1974
Pedro	30/04/1993
Tomás	28/09/1941
Tomás	25/05/1993

En este ejemplo, cada SELECT tiene el mismo número de columnas y cada columna en el mismo orden y el mismo tipo de datos. Nos devuelve una lista única de alumnos, profesores y personal. Sin embargo, aún existiendo dos registros con nombre y fecha de nacimiento repetidos, solamente nos ha listado uno de ellos.

Si lo procesamos utilizado **UNION ALL:**

SELECT *nomb*, *f_naci* **FROM** *ALUMNOS*
UNION ALL
SELECT *nomb*, *f_naci* **FROM** *PROFEDG*
UNION ALL
SELECT *nomb*, *f_naci* **FROM** *PERSONAL*

se seleccionan todos los registros; es decir, tiene un registro más que en el ejemplo anterior, pues al poner **ALL** toma todos los registros de las tres tablas, los comunes y no comunes.

Nomb	***f_naci***
Juan	23/08/1992
José	12/02/1991
Pedro	30/04/1993
Luis	05/11/1994
Tomás	25/05/1993

Pedro	22/12/1974	
Ángel	13/03/1962	
Tomás	28/09/1941	
Pedro	22/12/1974	<<<<<<<<
José	01/12/1963	
Nicolás		
Eduardo	16/06/1966	

Si en las cláusulas **SELECT** solamente seleccionamos el nombre quedaría así:

SELECT *nomb* **FROM** *ALUMNOS*
UNION
SELECT *nomb* **FROM** *PROFEDG*
UNION
SELECT *nomb* **FROM** *PERSONAL*

El resultado elimina los nombres repetidos:

nomb
Angel
Eduardo
José
Juan
Luis
Nicolás
Pedro
Tomás

Las cláusulas **SELECT** no tienen por qué ser iguales, como es el caso en los ejemplos anteriores.

Igualmente las cláusulas **UNION** también pueden ser diferentes: unas pueden ser simplemente **UNION** y otras de la forma **UNION ALL**.

SELECT *ape1* **FROM** *ALUMNOS*
WHERE *grupo* **= 'PA1'**
UNION
SELECT *ape2* **FROM** *PROFEDG*
UNION ALL
SELECT *ape2* **FROM** *PERSONAL*
WHERE *nomb* <> **'José'**

En esta sentencia vemos que las diferentes **SELECT** de la **UNION** son diferentes; seleccionan diferentes columnas, en algunas con condiciones, y las uniones son igualmente diferentes: una es **UNION** y la otra **UNION ALL.**

Veamos cómo funciona por orden de ejecución:

1. **SELECT** *ape1* **FROM** *ALUMNOS*
   WHERE *grupo* **= 'PA1'**

   De la tabla *ALUMNOS* selecciona el primer apellido (*ape1*) de los registros cuya columna *grupo* contenga el valor **PA1**
   Resultado: **Pérez**
   García

2. **SELECT** *ape2* **FROM** *PROFEDG*

   De la tabla *PROFEDG* selecciona el segundo apellido *(ape2)*

Resultado: **Díaz**
López
Nuñez

3. Hace la **UNION** entre la **SELECT** del paso 1 y la del 2, eliminando los registros repetidos (ninguno en este caso):
Resultado: **Díaz**
García
López
Nuñez
Pérez

4. **SELECT** *ape2* **FROM** *PROFEDG*

De la tabla *PROFEDG* selecciona el segundo apellido *(ape2)*
Resultado: **Díaz**
López
Nuñez

5. **SELECT** *ape2* **FROM** *PERSONAL*
WHERE *nomb* <> **'José'**

De la tabla *PERSONAL* selecciona el segundo apellido *(ape2)* a excepción de aquellos registros que en la columna *nomb* tengan el valor José.
Resultado: **García**
Cuentista
Valera

6. **UNION ALL:** une todos los valores resultantes de 3 y 4 y proporciona el resultado final.
Resultado: **Cuentista**
Díaz
García
López
Nuñez
Pérez
Valera

Como ya se dijo al hablar de cláusula **ORDER BY,** si se llevan a cabo alguna unión de SELECT los nombres de las columnas que se seleccionen de las diferentes tablas pueden tener igual o distinto nombre.

En el ejemplo siguiente la **UNION** de **SELECT** obtiene de cada tabla una columna diferente: *nomb*, *ape1* y *ape2* respectivamente. Al indicar el orden que deseamos obtener **no podemos utilizar un nombre de columna** ya que las tres tienen nombre diferente. No tenemos más remedio que utilizar el orden, que en este caso es 1 por seleccionar una columna.

SELECT *nomb* **FROM** *ALUMNOS*
UNION
SELECT *ape1* **FROM** *PROFEDG*
UNION
SELECT *ape2* **FROM** *PERSONAL*
ORDER BY 1

El resultado es:

nomb
Cuentista
García

José
Juan
López
Luis
Pedro
Pérez
Rivera
Tomás
Varela

Se observa que en el primer apellido de *PROFEDG* hay un García y en el segundo apellido de *PERSONAL* hay otro García; debido a la **UNION** solamente presenta uno, ordenando alfabéticamente los valores resultantes.

Si en éste último ejemplo utilizáramos el mismo alias en las columnas de las diferentes **SELECT**, podemos colocar ese alias en la cláusula **ORDER BY**, quedando así:

SELECT *nomb* **xxx FROM** *ALUMNOS*
UNION
SELECT *ape1* **xxx FROM** *PROFEDG*
UNION
SELECT *ape2* **xxx FROM** *PERSONAL*
ORDER BY xxx

Si los nombres de las columnas son iguales, sí podremos utilizarlo en la cláusula **ORDER BY**:

SELECT *ape1* **FROM** *ALUMNOS*
UNION
SELECT *ape1* **FROM** *PROFEDG*
UNION
SELECT *ape1* **FROM** *PERSONAL*
ORDER BY *ape1*

SELECT ANIDADAS O SUBCONSULTAS

Pueden existir consultas a la Base de Datos que requieran una restricción que sea el resultado de otra consulta.

Una subconsulta es una modalidad de **SELECT** que aparece dentro de otra sentencia **SELECT** y se las denomina también **SELECT ANIDADAS.** Se utiliza para responder una pregunta múltiple.

La **SELECT** que contiene una subconsulta se la denomina **SELECT PADRE.**

Las filas que devuelve la **SELECT ANIDADA** son utilizadas por la **SELECT PADRE.**

La **SELECT ANIDADA** se ejecuta una sola vez, independientemente de las filas que devuelva la consulta padre o **SELECT PADRE.**

Una **SUBCONSULTA** puede contener otra **SUBCONSULTA,** no existiendo límite en el nivel de anidación de las consultas.

Por ejemplo: deseamos saber el nombre y apellidos de los alumnos que tienen el mayor número de hermanos.

Mediante una **SELECT** se obtienen el nombre y apellidos, pero la condición de mayor número de hermanos sólo se puede obtenerse después de ejecutar una sentencia que contenga la función **max**(*n_her*).

Cuando esto se produce podemos actuar de dos formas:

1.- Efectuando la consulta que genera la restricción, ejecutando posteriormente la sentencia con el resultado de la restricción calculada:

SELECT max(*n_her***)**
FROM *ALUMNOS*

El valor máximo que nos devuelve es 4. Posterioremente realizaríamos la sentencia definitiva con:

SELECT *nomb, ape1, ape2, n_her*
FROM *ALUMNOS*
WHERE *n_her* **= 4**

Dándonos como resultado

nomb	***ape1***	***ape2***	***n_her***
Tomas	López	Baruque	4

2.- La segunda forma es más compleja, pero indudablemente más práctica. La sentencia que genera la restricción compleja se anida dentro de la principal encerrándola entre paréntesis:

SELECT *nomb, ape1, ape2,n_her*
FROM *ALUMNOS*
WHERE *n_her***=(SELECT max(***n_her***) FROM** *ALUMNOS* **)**

El resultado es el mismo, pero se realiza con una sola sentencia **SELECT** que contiene a su vez otra sentencia **SELECT** como **SUBCONSULTA**.

Expresiones SQL

Una expresión consiste normalmente en dos o más identificadores relacionados entre sí mediante un operador.

Cuando en una cláusula de la sentencia **SELECT** utilizamos un nombre de columna, o mezclamos nombres de columnas con operadores, funciones o constantes, estamos utilizando **EXPRESIONES SQL**.

Cada cláusula puede tener una o varias expresiones.

Las expresiones de una misma cláusula, se separan entre sí con una coma.

En los ejemplos que hemos visto anteriormente, se han utilizado expresiones.

Las expresiones se utilizan en las cláusulas **SELECT, WHERE, HAVING y ORDER BY** de las sentencias **SELECT**.

Las expresiones nos permiten realizar operaciones matemáticas, así como tratar cadenas de caracteres, operadores de manipulación de fechas, etc. para construir consultas complejas.

Los elementos que compones las expresiones son:

- Nombres de columnas
- Constantes
- Operadores numéricos
- Operadores de caracteres
- Operadores de fechas
- Operadores de relación
- Operadores lógicos
- Funciones

Nombres de columnas

Las expresiones más comunes son los propios nombres de las columnas, que o pueden ser combinados con otros elementos de las expresiones. Recordemos algún ejemplo de los vistos anteriormente:

SELECT *nomb, ape1, grupo, prv_nac, f_naci*
FROM *ALUMNOS*
ORDER BY *prv_nac* **desc,** *nomb*

Los nombres de columnas de la tabla *ALUMNOS, nomb, ape1, grupo, prv_nac y f_naci* son expresiones que nos indican las columnas que deseamos seleccionar. Solamente tienen un identificador, el nombre de la columna, y, por tanto, no necesitan ningún operador de relación.

En la cláusula ORDER BY tenemos la expresión *prv_nac* ***desc*** que indica el tipo de ordenación descendente elegido para la columna *prv_nac***.** La expresión *nomb* indica que dentro de la ordenación por provincia de nacimiento deseamos ordenar ascendentemente por la columna *nomb.*

Constantes

Las constantes son valores fijos que no cambian en ningún momento del proceso. Por ejemplo:

SELECT *nomb***,** *ape1, ape2, grupo, prv_nac, f_naci, n_her, n_her*-**1** *hermanos de*
FROM *ALUMNOS*

En la expresión *n_her* - 1, el valor 1 es una constante y *n_her* es el nombre de la columna. Además de seleccionar todas las columnas de la tabla *ALUMNOS,* calcula el valor *hermanos_de* que se obtiene al restar 1 al número de hermanos del alumno, sin contarse él mismo (*n_her* cuenta al alumno), quedando así:

Nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her***	***hermanos_de_***
Juan	Pérez	Martín	P1A	28	23/08/1992	3	2
José	García	Gómez	P1A	34	12/02/1991	2	1
Pedro	San	Sánchez	P1C	37	30/04/1993	3	2
Luis	Ponce	Primo	S2A	28	05/11/1994	1	0

Tomás	López	Baruque	P2F	37	25/05/1993	4	3

En éste ejemplo podíamos haber caído en la tentación de escribir:

SELECT **,n_her-1 hermanos_de* ,

ya que seleccionamos todas las columnas de la tabla *ALUMNOS* y vimos al principio que utilizando el * es suficiente. Sin embargo, en éste caso no podemos utilizarlo, porque al poner * para seleccionar todos las columnas de la tabla *ALUMNOS* ya no se puede seleccionar ninguna columna más.

Las constantes de caracteres se tienen que encerrar entre dos comillas simple (‘) o doble ("). Para cerrar un string hay que utilizar una pareja de comillas iguales.

Para que una comilla simple o doble aparezca como elemento de una constante, ha de ponerse doble. (Por ejemplo, si la constante deseada es O’Donnel, deberá figurar como ‘O’’Donnel’ u “O’Donnel”

Veamos un ejemplo con constantes de caracteres:

SELECT *nomb*, ' **”ALUMNO” ’ FROM** *ALUMNOS*
UNION ALL
SELECT *nomb* **, ” ’PROFESOR’ ” FROM** *PROFEDG*
UNION ALL
SELECT *nomb* **, ’ OTRO ‘’PERSONAL’’’ FROM** *PERSONAL*

nomb	***obtenido de***
Juan	**“ALUMNO”**
José	**“ALUMNO”**
Pedro	**“ALUMNO”**
Luis	**“ALUMNO”**
Tomás	**“ALUMNO”**
Pedro	**‘PROFESOR’**
Ángel	**‘PROFESOR‘**
Tomás	**‘PROFESOR’**
Pedro	**OTRO ‘PERSONAL’**
José	**OTRO’ PERSONAL’**
Nicolás	**OTRO’ PERSONAL’**
Eduardo	**OTRO’ PERSONAL’**

Este ejemplo ya le vimos al definir la **UNION**; aquí le añadimos una constante, diferente según la tabla de donde obtenemos el nombre. Los literales creados en las diferentes **SELECT** tienen que tener las mismas características de tipo de dato y longitud. Aquí son string de caracteres.

OPERADORES

Un operador se utiliza para manipular datos de forma individual y devuelve un valor. También se le denomina operando o argumento.

Operadores Numéricos (aritméticos)

Se utilizan para realizar operaciones con datos de tipo numérico:

Operador	**significado**
+	Suma
-	Resta
*	Multiplicación
/	División

Las operaciones aritméticas pueden figurar en las cláusulas **SELECT** y **WHERE**

SELECT *nomb, ape1, ape2, n_her * 4* as her_ por_4
FROM *ALUMNOS*
WHERE *n_her* > 2

El resultado es:

nomb	***ape1***	***ape2***	***her_por_4***
Juan	Pérez	Martín	12
Pedro	San	Sánchez	12
Tomás	López	Baruque	16

Ya hemos visto varios ejemplos con el operador suma; el resto de operadores indicados se utilizan de la misma forma.

En cualquier expresión que se utilicen, pueden utilizarse todos los operadores aritméticos que se necesiten.

El orden de ejecución de los operadores es el siguiente:

/
*
+ -

Ante la igualdad de orden de ejecución, se comienza a ejecutar por la izquierda.

Para obligar a realizar las operaciones en un determinado orden, se utilizan los paréntesis, ejecutándose en el orden desde el interior al exterior.

Ejemplo: a + b * c – d / e el orden de ejecución es d / e = h
b * c = j
a + j = k
k - h = x

Ejemplo: ((a + b) * c - d) / e el orden de ejecución es a + b = m
m* c = n
n - d = p
p / e = y

Operadores de caracteres

Las expresiones de caracteres pueden incluir los siguientes operadores:

Operador	**significado**
||	Concatenación de cadenas de caracteres.

ejemplo:

SELECT *nomb* || *ape1* || *ape2*
FROM *ALUMNOS*

nomb || ape1 || ape2
JuanPérezMartín
JoséGarcíaGómez
PedroSanSánchez
LuisPoncePrimo
TomásLópezBaruque

Para dejar separación entre el nombre y los apellidos y poner un alias al campo calculado haríamos así:

SELECT *nomb* || ‘ ‘ || *ape1* || ‘ ‘ || *ape2* *nombre*

FROM *ALUMNOS*

Resultado*:*

nombre
Juan Pérez Martín
José García Gómez
Pedro San Sánchez
Luis Ponce Primo
Tomas López Baruque

Podemos utilizar el signo + en lugar de ||, obteniendo el mismo resultado:

SELECT *nomb* **+** ‘ ‘ **+** *ape1* **+** ‘ ‘ **+** *ape2* *nombre*

Operadores de relación

Sirven para comparar dos valores, que pueden ser columnas, literales o expresiones calculadas. El resultado es un valor booleano que se utiliza en la cláusula **WHERE** para determinar si cumple o no la condición de selección. Pueden utilizarse varias relaciones siempre que se enlacen con operadores lógicos.

Los operadores de relación que se pueden utilizar son los siguientes:

Operador	Significado	Ejemplo
=	**Igual a**	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* =’28’
<>	**Distinto de** (no igual a)	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* <>’28’
>	**Mayor que**	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* > ‘28’
>=	**Mayor o igual que**	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* >= ‘28’
<	**Menor que**	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* <’28’
<=	**Menor o igual que**	**SELECT** * **FROM** *ALUMNOS* **WHERE** *prv_nac* <=’28’

Like Compara la similitud del valor de una columna o parte de ella con un patrón dado.

SELECT * **FROM** *PERSONAL*
WHERE *ape1* **like** **‘**García’

(El resultado es obtener todos los registros con valor *ape1* que contienen ‘García’)

Cuando queramos que el string con el que se compara se localice al principio, final o cualquier punto de la columna, emplearemos el carácter % de la siguiente manera:

Para que la comparación se haga desde el principio, finalizamos el string con **%**

WHERE *nomb* **like** ‘Pe%’ selecciona los nombres que comiencen por **Pe**

Para que la comparación se haga en las últimas posiciones, comenzar el string con **%**

WHERE *nomb* **like** '%arc' selecciona todos los que finalicen con la secuencia **arc**.

Para hacer la comparación en cualquier posición, comienzo y final del string con **%**

WHERE *nomb* **like** '%arc%' selecciona todos los que tengan la secuencia **arc** en cualquier parte de la columna *nomb*.

Para que la comparación no tenga en cuenta algún carácter, pero sí su posición, utilizamos e l carácter _ Se pueden colocar cuantos _ se quieran.

Por ejemplo encontrar una cadena de caracteres que en la segunda posición tenga la letra **C**

WHERE *nomb* **like** '_C%' no importa cuál sea el primer carácter.

WHERE *nomb* **like** '%V_H__C%' buscará la cadena en cualquier lugar de la columna *nomb* que contenga la letra **V** otro carácter la letra **H** otras dos caracteres cualquiera y la **C**.

SELECT *
FROM *ALUMNOS*
WHERE *ape2* **like '%ar%'**

El resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Juan	Pérez	Martín	P1A	28	23/08/1992	3
Tomas	López	Baruque	P2F	37	25/05/1993	4

not **Negación**

SELECT * **FROM** *PERSONAL*
WHERE *ape1* **not like '**García'

(Se obtienen todos los registros con valor *ape2* que no sea 'García').

Is Null **Igual a nulo (vacío)**

En la definición de una tabla se especifica los posibles valores que pueden contener cada columna, uno de los posibles valores es el nulo. No hay que confundir el valor cero con el nulo, el cero es un valor mientras que el nulo es ausencia de valor; lo mismo ocurre con blancos y nulos. En algunas ocasiones interesa conocer o condicionar si el valor de una columna es **NULO** para ello utilizamos este operador.

SELECT * **FROM** *PERSONAL*
WHERE *f_naci* **is null**

Si miramos la tabla *PERSONAL* que hay al principio, veremos que hay un registro que tiene la columna *f_naci* sin valor , el resultado de la **SELECT** es:

dni	***f_naci***	***ape1***	***ape2***	***nomb***
44444444		Anelca	Cuentista	Nicolás

Is Not Null **No es nulo (**no está vacío, tiene algún valor**)**

Es lo contrario **is null** veamos un ejemplo:

SELECT * **FROM** *PERSONAL*
WHERE *f_naci* **is not null**

Se obtienen todos los registros con algún valor en la columna *f_nac*:

dni	***f_naci***	***ape1***	***ape2***	***nomb***
11111111	22/12/1974	García	García	Pedro

33333333	01/12/1963	García	Rivera	José
55555555	16/06/1966	Rouco	Varela	Eduardo

Between **Rango de valores entre una cota inferior y otra superior**

Cuando la comparación que deseamos realizar se encuentra en un intervalo de valores, podemos utilizar esta forma de expresarlo, poniendo la columna a comparar , el operador **Between** y los valores extremos del intervalo por ejemplo:

SELECT * **FROM** *ALUMNOS*
WHERE *prv_nac* **between** '28' and '35'

El resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Juan	Pérez	Martín	P1A	28	23/08/1992	3
José	García	Gómez	P1A	34	12/02/1991	2
Luis	Ponce	Primo	S2A	28	05/11/1994	1

También podemos utilizar **Not Between**

SELECT * **FROM** *ALUMNOS*
WHERE *prv_nac* **not between** '28' and '35'

El resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Tomás	López	Baruque	P2F	37	25/05/1993	4

In **Pertenencia a un conjunto de valores de una columna**

Otra forma de indicar los diferentes valores a tener en cuenta la condición de selección, es indicar todos los valores que deseamos que intervengan después de indicar el nombre de la columna la palabra **in.**

Por ejemplo: se condiciona la selección a que los valores de la columna *grupo* de la tabla *ALUMNOS* sean P1A, P1B o P1C

SELECT * **FROM** *ALUMNOS*
WHERE *grupo* **in** ('P1A','P1B','P1C')

El resultado sería:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Juan	Pérez	Martín	P1A	28	23/08/1992	3
José	García	Gómez	P1A	34	12/02/1991	2
Pedro	San	Sánchez	P1C	37	30/04/1993	3

Del *grupo* P1B no selecciona ninguno porque no existe ningún registro cuya columna *grupo* tenga dicho valor.

Se puede utilizar la fórmula **not in** que funciona seleccionando los registros que no cumplan la condición indicada. Siguiendo con el ejemplo tendremos:

SELECT * **FROM** *ALUMNOS*
WHERE *grupo* **not in** ('P1A','P1B','P1C')

El resultado sería:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***

Luis	Ponce	Primo	S2A	28	05/11/1994	1
Tomas	López	Baruque	P2F	37	25/05/1993	4

que son los registros que no había seleccionado antes.

Exists **Cierto, si una subconsulta devuelve como mínimo un registro.**
Este operador le utilizaremos cuando necesitemos condicionar la selección a que el valor de una o más columnas exista en otra tabla o en otra columna de la misma tabla.

Veamos algunos ejemplos:

Seleccionar de la tabla *ALUMNOS* todos los registros que el valor de la columna *ape1* exista en la columna *ape2* de la tabla *PROFEDG*.

SELECT *
FROM *ALUMNOS* a
WHERE exists(**SELECT** *ape2* **FROM** *PROFEDG* b **WHERE** a.*ape1*=b.*ape2*)

El resultado es:

nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her_*
Tomás	López	Baruque	P2F	37	25/05/1993	4

Que es el único registro que el valor de *ape1* figura en la columna *ape2* de la tabla *PROFEDG*

Obtengamos de la tabla *PERSONAL* los registros que el valor de *ape1* figure en algún registro de la columna *ape2* de la misma tabla:

SELECT *
FROM *PERSONAL* a
WHERE exists(SELECT *ape2* **FROM** *PERSONAL* b **WHERE** a.*ape1*=b.*ape2*)

El resultado es:

dni	*f_naci*	*ape1*	*ape2*	*nomb*
11111111	22/12/1974	García	García	Pedro
33333333	01/12/1963	García	Rivera	José

Se puede utilizar **no exists**, por ejemplo de la tabla *ALUMNOS* para obtener todos los registros cuyo valor de *ape1* no figure en la columna *ape2*, la sentencia es:

SELECT *
FROM *ALUMNOS* a
WHERE not exists(SELECT *ape2* **FROM** *PERSONAL* b **WHERE** a.*ape1*=b.*ape2*)

El resultado es:

nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her_*
Juan	Pérez	Martín	P1A	28	23/08/1992	3
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Luis	Ponce	Primo	S2A	28	05/11/1994	1
Tomás	López	Baruque	P2F	37	25/05/1993	4

Selecciona de la tabla *ALUMNOS* todos los registros cuyo valor de *ape1* no existe en todos los valores de la columna *ape2* de la tabla *PERSONAL*.

any Compara el valor de una columna con cada uno de los valores obtenidos en la subconsulta, retornando verdadero si uno (alguno) de ellos cumple la condición.

Any debe ir precedido de =, <, <=, >, >=, <> o =

Some Funciona igual que **any**.

All **C**ompara un valor con cada valor devuelto por una subconsulta retornando cierto si todos ellos cumplen la condición.

All debe ir precedido de =, <, <=, >, >= , <> o =.

Seleccionar los alumnos cuya fecha de nacimiento sea inferior a todas las fechas de nacimiento de los profesores.

SELECT *
FROM *ALUMNOS* x
WHERE (x.*f_naci* > **all** (**SELECT** y.*f_naci* **FROM** *PROFEDG y*))

Todas las fechas de nacimiento de la tabla *ALUMNOS* son superiores a todas las fechas de nacimiento le la tabla *PROFEDG*, por tanto selecciona todos; el resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Juan	Pérez	Martín	P1A	28	23/08/1992	3
José	García	Gómez	P1A	34	12/02/1991	2
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Luis	Ponce	Primo	S2A	28	05/11/1994	1
Tomás	López	Baruque	P2F	37	25/05/1993	4

Operadores Lógicos

Los operadores lógicos son **and** y **or** y nos permiten conectar dos predicados o condiciones en la cláusula **WHERE**.

Como vimos, al tratar los operadores, cada condición genera una variable boolena. Los operadores lógicos tienen la misión de conectar estas variables booleanas para determinar si la cláusula **WHERE** se cumple y el registro puede ser seleccionado.

Cuando existen dos o más condiciones deberán estar unidas por **and** o **or**.

Por ejemplo:

WHERE (*grupo* = 'P1A') **and** (*n_her* >= 3)

Si se cumple la condición *grupo* = 'P1A' genera la variable booleana **VERDADERO**, si no, genera **FALSO**.

Si se cumple la condición *n_her* >= 3 genera la variable booleana **VERDADERO**, si no, genera **FALSO**.

Para que la condición general se cumpla, tiene que devolver **VERDADERO y VEDADERO.**

Si el resultado fuera **FALSO** en alguna de las condiciones, la selección no se realiza.

Veamos lo que ocurre al procesar la sentencia siguiente:

SELECT *nomb*, *ape1, ape2, grupo, prv_nac, f_naci, n_her*
FROM *ALUMNOS*
WHERE (*grupo* = 'P1A') **and** (*n_her* >= 3)

Hay dos registros que cumplen la condición *grupo* = 'P1A' , pero sólo uno de ellos cumple la condición *n_her*>= '3'. Por tanto la selección quedará así:

Nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her*
Juan	Pérez	Martín	P1A	28	23/08/1992	3

Si utilizamos el operador **or** sería :

SELECT *
FROM *ALUMNOS*
WHERE (*grupo* = 'P1A') **or** (*n_her* >= 3)

Hay un registro que cumple las dos condiciones y otros que solamente cumple una; por tanto el resultado es:

nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her*
Juan	Pérez	Martín	P1A	28	23/08/1992	3
José	García	Gómez	P1A	34	12/02/1991	2
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Tomás	López	Baruque	P2F	37	25/05/1993	4

El operador lógico **not** es útil para negar la condición booleana: niega una condición.

Por ejemplo:

SELECT *
FROM *ALUMNOS*
WHERE **not** (*grupo* = 'P1A' **and** *n_her* >= 3)

En el ejemplo sin el **not** veíamos que se seleccionaban los registros que cumplían las dos condiciones; en este caso queremos seleccionar los que no cumplan esas dos condiciones, es decir,

not (**VERDADERO and VERDADERO**)

Cualquier otra combinación sería la correcta. Por tanto el resultado no es el resto de los registros que en el caso anterior no seleccionaba, sino que selecciona los registros en los que una o las dos condiciones sean **FALSAS**:

Nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her*
José	García	Gómez	P1A	34	12/02/1991	2
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Luis	Ponce	Primo	S2A	28	05/11/1994	1
Tomas	López	Baruque	P2F	37	25/05/1993	4

Se podría poner el **not** para uno de los elementos de la condición

SELECT *
FROM *ALUMNOS*
WHERE (**not** *grupo* = 'P1A') **and** (*n_her* >= 3)

En esta situación se seleccionarían los registros que no cumplen la condición *grupo* = 'P1A' y sí cumplen la condición *n_her* >= 3. El resultado es:

nomb	*ape1*	*ape2*	*grupo*	*prv_nac*	*f_naci_*	*n_her*
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Tomás	López	Baruque	P2F	37	25/05/1993	4

En estos ejemplos podíamos haber cambiado las condiciones de proceso en lugar de poner el operador lógico **not,** sustituyendo *grupo* <>' P1A' por *grupo* = 'P1A' y *n_her* > 2 por *n_her* >= 3 .

Cuando en una cláusula **WHERE** se utiliza más de un operador lógico, para determinar correctamente el ámbito de actuación de cada operador, es bueno utilizar paréntesis.

El resultado de la **SELECT** procesada cuando vimos el operador **Between**

SELECT *
FROM *ALUMNOS*
WHERE *prv_nac* **between** ‘28’ and ‘35’

obtiene los mismos resultados que si la planteamos de esta otra forma:

SELECT *
FROM *ALUMNOS*
WHERE *prv_nac* >=‘28’ **and** *prv_nac* <= ‘35’

El resultado es:

Nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her***
Juan	Pérez	Martín	P1A	28	23/08/1992	3
José	García	Gómez	P1A	34	12/02/1991	2
Luis	Ponce	Primo	S2A	28	05/11/1994	1

Prioridad de los operadores

En expresiones con más de una condición, el orden en el que éstas se evalúan es muy importante. La siguiente tabla muestra el orden en el que son evaluados los operadores. Los operadores que figuran en la primera línea se evalúan los primeros, luego los de la segunda y así sucesivamente. Los operadores que figuren en la misma línea se evalúan de izquierda a derecha según aparezcan en la expresión.

prioridad operador

1 *, /
2 +, -
3 =, <, <= ,>, >=, =, <>,Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
4 NOT
5 AND
6 OR

El siguiente ejemplo muestra la importancia de la prioridad de los operadores:

SELECT *
FROM *ALUMNOS*
WHERE *grupo* = ‘S2A’ **or** *f_naci* < ‘01/01/1992’ **and** *pov_nac* <> ‘28’

El resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her***
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Luis	Ponce	Primo	S2A	28	05/11/1994	1
Tomas	López	Baruque	P2F	37	25/05/1993	4

Al evaluar AND en primer lugar, esta consulta nos devuelve aquellos alumnos con provincia de nacimiento diferente a 28 y que además hayan nacido con posterioridad de 01/01/1992 o que sean del *grupo* S2A.

Para forzar a que la cláusula se evalúe en orden distinto, por ejemplo, que muestre aquellos alumnos que sean del *grupo* S2A o que hayan nacido después de 01/01/1992 y que además de una de las dos condiciones anteriores, su provincia de nacimiento sea diferente a 28, será necesario usar paréntesis para encerrar las condiciones que deban evaluarse primero, es decir:

SELECT *

FROM *ALUMNOS*
WHERE (*grupo* = 'S2A' **or** *f_naci* > '01/01/1992') **and** *prv_nac* <> '28'

El resultado es:

nomb	***ape1***	***ape2***	***grupo***	***prv_nac***	***f_naci_***	***n_her_***
Pedro	San	Sánchez	P1C	37	30/04/1993	3
Tomás	López	Baruque	P2F	37	25/05/1993	4

Funciones

Las funciones permiten realizar con los datos o columnas operaciones adicionales a las ya vistas, pudiendo participar como operadores en las expresiones.

Una función representa un valor único que se obtiene aplicando unas determinadas operaciones a otros valores dados, que se llaman argumentos. Se especifica como una palabra predefinida seguida de los argumentos entre paréntesis y separados por alguna palabra específica o blancos.

El lenguaje SQL dispone de un conjunto de funciones que pueden usarse en las consultas y son las siguientes.

Función **Descripción**

SUBSTRING Extrae una subcadena de una columna o cadena de caracteres. Desde donde se obtiene la subcadena o substring tiene que ser CHAR(carácter) o convertirlo con la función CAST

Los parámetros son:
Cadena de caracteres o columna de tipo CHAR
from posición del primer carácter a extraer
for número de caracteres a extraer. Si se omite éste parámetro, tomará hasta el final.

SELECT *nomb, ape1, ape2,substring(grupo* **from 3 for 1**) as aula
FROM *ALUMNOS*

El resultado es:

nomb	***ape1***	***ape2***	***aula***
Juan	Pérez	Martín	A
José	García	Gómez	A
Pedro	San	Sánchez	C
Luis	Ponce	Primo	A
Tomás	López	Baruque	F

extrae desde la posición 3 una sola posición del grupo, que corresponde al aula.

UPPER Convierte todos los caracteres de una columna alfabética en mayúsculas.
Si la columna o el string no es CHAR (alfabética) hay que transformarlo mediante la función CAST
Ejemplo, seleccionar nombre y apellidos para que se editen todo en mayúsculas:

SELECT upper(*nomb)*, **upper** (*ape1)*, **upper** (*ape2)*
FROM *ALUMNOS*

El resultado es:

nomb	***ape1***	***ape2***
JUAN	PÉREZ	MARTÍN
JOSË	GARCÍA	GÓMEZ
PEDRO	SAN	SÁNCHEZ
LUIS	PONCE	PRIMO
TOMÁS	LÓPEZ	BARUQUE

LOWER Funciona igual que UPPER pero la conversión, en este caso, se realiza a minúsculas.
Ejemplo: seleccionar nombre y apellidos para que se editen todo en mayúsculas:

SELECT lower (*nomb)*, **lower** (*ape1)*, **lower** (*ape2)*
FROM *ALUMNOS*

El resultado es:

nomb	***ape1***	***ape2***
juan	pérez	martín
josé	garcía	gómez
pedro	san	sánchez
luis	ponce	primo
tomás	lópez	baruque

TRIM Borra los caracteres de la izquierda, la derecha o ambos lados de una columna de tipo string o CHAR que coincidan con un carácter indicado.

Si la columna es numérica, hay que realizar la transformación del tipo de dato con la función CAST.

LEARDING indica que elimina los caracteres de la parte izquierda.
TRAILING " " " " " " " " derecha.
BOTH " " " " " " las dos partes.

Ejemplos:

	resultado	
TRIM(LEADING "_" FROM " ___*ABC*_")	"ABC_"	elimina los _ de la izquierda
RIM(TRAILING "_" FROM " _*ABC*_")	"_ABC"	elimina los _ de la derecha
TRIM(BOTH "X" FROM " XXXX*ABCXXX*")	"ABC"	elimina las X de derecha e izquierda

Esta función tiene poca utilidad para utilizarla en una **SELECT** , pero en alguna ocasión puede ser útil.

CAST Convierte un valor específico a un tipo de dato específico.
El valor específico puede ser una columna, parte de una columna, concatenación u otra expresión. Los tipos de datos CHAR para los valores alfanuméricos.
INTEGER para los valores numéricos enteros.
DOUBLE " " " " con decimales
DATE " " " de fechas.

SELECT cast(f_*naci* as CHAR) convierte la fecha de nacimiento como cadena de caracteres y ser tratada como tal, en lugar de utilizarla como DATE

Si quisiéramos seleccionar de la tabla *ALUMNOS* los registros cuyo año de nacimiento sea mayor de 1992 y sumarle 5 unidades al año, haríamos lo siguiente:

SELECT nomb, ape1,ape2, f_naci,
cast (**substring**(cast(f_naci as char(10)) **from** 7 **for** 4) as integer)+5 as ww
FROM *ALUMNOS*
WHERE substring(**cast** (f_*naci* as char(10)) **from** 7 **for** 4) > '1992'

El año '1992' aquí lo ponemos entre comillas porque es un literal con el que se compara el literal obtenido con la función **substring** que actúa sobre el string en que convierte la función **cast** la columna *f_naci*, que es una fecha.

Cuando se toman columnas del tipo DATE (fechas) hay que tener cuidado si existen blancos en las fechas, pues la función **cast** no los considera como carácter.

El resultado es:

nomb	***ape1***	***ape2***	***f_naci_***	***www_***
Pedro	San	Sánchez	30/04/1993	1998
Luis	Ponce	Primo	05/11/1994	1999
Tomás	López	Baruque	25/05/1993	1998

En la cláusula **SELECT** una función **cast** interna en otra función **cast**, la función interna convierte la columna *f_naci* de tipo DATE a tipo CHAR la función externa convierte el string de las 4 últimas posiciones en un tipo INTEGER , que es el año y le suma 5 a ese valor.

EXTRACT Extrae de una columna de tipo DATE. o TIMESTAMP los valores del día mes y año.

El formato que tienen los campos DATE (que contienen una fecha) es de **DD/MM/YYYY**

DD: dos dígitos para indicar el día.
MM: dos dígitos para indicar el mes.
YYYY: cuatro dígitos para indicar el año.

En el día y el mes cuando es inferior a 10, tienen un cero a la izquierda.

Si queremos obtener la lista de los alumnos con la fecha de nacimiento desglosada en día, mes y año, cuando el año de nacimiento sea mayor de 1992, la **SELECT** sería la siguiente:

SELECT *nomb,ape1,ape2,f_naci,***extract** (DAY from *f_naci*) as día,
extract (MONTH from *f_naci*) as mes,
extract(YEAR from *f_naci*) as año
FROM *ALUMNOS*
WHERE **extract** (YEAR from *f_naci*) > 1992

Aquí utilizamos 1992 como un valor numérico porque la función **extract**, que obtiene el año correspondiente a la fecha *f_naci* nos da un valor numérico.

Y el resultado es el siguiente:

nomb	***ape1***	***ape2***	***f_naci***	***día***	***mes***	***año_***
Pedro	San	Sánchez	30/04/1993	30	4	1993
Luis	Ponce	Primo	05/11/1994	5	11	1994
Tomás	López	Baruque	25/05/1993	25	5	1993

Si comparamos esta **SELECT** con la del ejemplo de la función **cast** veremos que para seleccionar el Año de la columna *f_naci* podemos utilizar indistintamente la función **cast** o la **extract** pero para poder utilizar esa selección del año como campo de tipo CHAR o tipo INTEGER tenemos que utilizar **cast**. El ejemplo que hay con la función **cast**, podríamos plantearle de esta otra forma:

SELECT nomb,ape1,ape2,f_naci,
cast(**extract** (YEAR from f_naci) as integer) + 5 as ww
FROM *ALUMNOS*
WHERE **extract** (YEAR from *f_naci*) > 1992

El resultado sería el mismo que vimos en el ejemplo de la función CAST.

ANEXO I

Comentarios a las consultas incluidas como ÚTILES en el módulo de CONSULTAS LIBRES del "PROGRAMA ESCUELA"

A continuación se comentan las diferentes SELECT preparadas en el PROGRAMA ESCUELA para visualizar o editar determinados REPORT que, considerados como potencialmente útiles, se ofrecen al usuario ya preparados, no modificables y listas para su visualización y/o impresión.
Se pretende de este modo ofertar, además, unos modelos de consultas que puedan servir de guía al usuario para la creación de otros similares.
Existen algunos elementos y conceptos comunes a todas o varias de las sentencias SELECT que, a fin de no reiterarse en el comentario de cada una de ellas, se procede a comentar con carácter general.

- La cláusula SELECT recoge los nombres de las columnas que vamos a seleccionar.
- La cláusula FROM os indica la tabla o tablas de las que se seleccionan las columnas.
- La cláusula WHERE expresa las condiciones que tienen que cumplirse para poder seleccionar los registros . Si falta la cláusula WHERE, se seleccionan todos los registros.
- La cláusula ORDER BY indica la ordenación en que deben aparecer los registros seleccionados. Si no se incluye esta cláusula, el orden de salida es natural, de acuerdo cómo fueron introducidos los registros. La ordenación se realiza por la primera columna indicada, dentro de ese orden, continúa con la siguiente columna indicada y así sucesivamente con las restantes. En lugar del nombre de la columna puede indicarse el alias dado a la columna o expresión en la cláusula SELECT o también el orden numérico que ocupan en la cláusula SELECT.
- En la cabecera de los REPORT siempre figura el nombre de las columnas o expresiones que aparecen en la cláusula SELECT, las constantes(string) que figuren en esta cláusula no se editan en la cabecera.
- Cuando se utiliza un "alias", éste aparece en la cabecera del REPORT en lugar del nombre de la columna o expresión.

En la mayor parte de las consultas nos interesa seleccionar el NOMBRE completo del alumno, pero como éste no existe en la tabla, si no en forma de APE1 (Primer Apellido), APE2 (2º Apellido) y NOMB (Nombre), utilizamos en todos los casos la expresión APE1+' '+APE2+', '+NOMB as NOMBRE.
O sea, la expresión anteriormente indicada se compone de la columna APE1 (primer apellido), un espacio en blanco, la columna APE2 (segundo apellido), una coma seguida de un espacio en blanco y la columna NOMB (nombre).
El signo + indica la concatenación de estos valores y la partícula as introduce el alias con el que queremos referirnos a la expresión, en este caso NOMBRE.

LAS SENTENCIAS SELECT PREPARADAS SON LAS SIGUIENTES

1. ALUMNOS (TODOS) F. NACI.: (AP)

```
SELECT APE1+' ' +APE2+', '+NOMB AS NOMBRE, F_NACI AS F_NACIMIENTO
FROM ALUMNOS
ORDER BY NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el NOMBRE completo de los alumnos (mediante la expresión explicada más arriba) y su fecha nacimiento (columna F_NACI, para la que se utiliza el alias F_NACIMIENTO).
El orden en que aparecen los registros es el ascendente por NOMBRE.

2. ALUMNOS (TODOS) F. NAC (FEC)

```
SELECT APE1+' ' +APE2+', '+NOMB AS NOMBRE, F_NACI AS NACIMIENTO
FROM ALUMNOS
ORDER BY F_NACI
```

Se trata de la misma sentencia anterior pero ordenada por fecha de nacimiento

3. ASISTENCIA ALUMNOS (TODOS)

```
 SELECT APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el NOMBRE completo de los alumnos y genera una tira con cuadros a la que se asigna el alias DIAS.

Al no incluirse la cláusula ORDER BY, los registros saldrán en el orden natural.

4. ASISTENCIA ALUMNOS (TODOS) POR GRUPO

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS
ORDER BY GRUPO,NOMBRE
```

Se trata de la misma SELECT anterior, pero en este caso los registros aparecen ordenados por GRUPO y, dentro del GRUPO, por NOMBRE..

5. ASISTENCIA ALUMNOS COMEDOR (GRUPOS)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘01’
ORDER BY GRUPO, NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el GRUPO y NOMBRE completo de los alumnos y la tira utilizada en los ejemplos anteriores, pero limitando el alcance de la consulta a aquellos alumnos que sean usuarios del servicio de comedor

Para ello, la cláusula SELECT incluye las columnas GRUPO, la expresión compuesta para el NOMBRE y la tira con el ya mencionado alias DIAS.

La cláusula WHERE establece que los registros de la tabla ALUMNOS a seleccionar deben ser aquellos que en su columna NUMERO tengan un valor que se encuentra también en la tabla USUASERV (de Usuarios de Servicios) y que además en la columna CLAVE de ésta última se encuentre el valor ‘01’, clave utilizada para servicio de COMEDOR.

Los registros aparecerán ordenados por la columna GRUPO y, dentro del GRUPO, por orden alfabético de NOMBRE.

6. ASISENCIA ALUMNOS DESAYUNO (GRUPOS)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘02’
ORDER BY GRUPO, NOMBRE
```

Se trata del mismo ejemplo anterior pero referido al servicio de desayuno, con clave '02' en la tabla USUASERV (de usuarios de servicios).

7. ASISTENCIA ALUMNOS TRANSPORTE (GRUPOS)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘03’
ORDER BY GRUPO, NOMBRE
```

Se trata del mismo ejemplo anterior pero referido al servicio de transporte, con clave '03' en la tabla USUASERV (de usuarios de servicios).

8. ASISTENCIA ALUMNOS RESIDENCIA (GRUPOS)

```
SELECT GRUPO, APE1+' ' ‘ +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘04’
ORDER BY GRUPO, NOMBRE
```

Se trata del mismo ejemplo anterior pero referido al servicio de residencia, con clave '04' en la tabla USUASERV (de usuarios de servicios).

9. ASISTENCIA ALUMNOS COMEDOR (NOMBRE)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘01’
ORDER BY NOMBRE, GRUPO
```

Se trata del mismo ejemplo que el nº 5, en el que se ha variado el orden: se procede a ordenar por NOMBRE y, por si hubiera alumnos con el mismo nombre, por GRUPO.

10. ASISTENCIA ALUMNOS DESAYUNO (NOMBRE)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘02’
ORDER BY NOMBRE, GRUPO
```

Se trata del mismo ejemplo que el nº 6, en el que se ha variado el orden: se procede a ordenar por NOMBRE y, por si hubiera alumnos con el mismo nombre, por GRUPO.

11. ASISTENCIA ALUMNOS TRANSPORTE (NOMBRE)

```
SELECT GRUPO, APE1+' '+APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘03’
ORDER BY NOMBRE, GRUPO
```

Se trata del mismo ejemplo que el nº 7, en el que se ha variado el orden: se procede a ordenar por NOMBRE y, por si hubiera alumnos con el mismo nombre, por GRUPO.

12. ASISTENCIA ALUMNOS RESIDENCIA (NOMBRE)

```
SELECT GRUPO, APE1+' ' +APE2+', '+NOMB AS NOMBRE,
‘|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|’ AS DIAS
FROM ALUMNOS, USUASERV
WHERE ALUMNOS.NUMERO = USUASERV.NUMERO
AND USUASERV.CLAVE = ‘04’
ORDER BY NOMBRE, GRUPO
```

Se trata del mismo ejemplo que el nº 8, en el que se ha variado el orden: se procede a ordenar por NOMBRE y, por si hubiera alumnos con el mismo nombre, por GRUPO.

13. ALUMNOS: RELIGION CATOLICA (G)

```
SELECT GRUPO, APE1+' '+APE2+', '+NOMB AS NOMBRE
FROM ALUMNOS
WHERE RELI = ‘01’
ORDER BY  GRUPO
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS las columnas GRUPO y NOMBRE completo.

Establece la condición de que la columna RELI de tabla ALUMNOS tenga el valor ‘01’, clave asignada a la religión CATOLICA.

Los registros aparecerán ordenados por la columna GRUPO.

14. ALUMNOS: RELIGION CATOLICA (N)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO
FROM ALUMNOS
WHERE RELI = ‘01’
ORDER BY NOMBRE
```

Se trata del mismo ejemplo anterior, pero los registros aparecerán ordenados por NOMBRE.

15. ALUMNOS: RELIGION ISLAMICA (G)

```
SELECT GRUPO, APE1+’ ‘+APE2+’, ‘+NOMB AS NOMBRE
FROM ALUMNOS
WHERE RELI = ‘04’
ORDER BY  GRUPO
```

Esta sentencia SELECT es la misma que la del ejemplo nº 13, pero los registros seleccionados de la tabla ALUMNOS serán aquellos que contengan en la columna RELI el valor '04', clave asignada a la religión ISLÄMICA

Los registros aparecerán ordenados por la columna GRUPO.

16. ALUMNOS: RELIGION ISLAMICA (N)

```
SELECT APE1+' '+APE2+', '‘+NOMB AS NOMBRE, GRUPO
FROM ALUMNOS
WHERE RELI = ‘04’
ORDER BY NOMBRE
```

Se trata del mismo ejemplo anterior, pero los registros aparecerán ordenados por NOMBRE.

17. ALUMNOS: RELIGION EVANGE. (G)

```
SELECT GRUPO, APE1+' '+APE2+', '+NOMB AS NOMBRE
FROM ALUMNOS
WHERE RELI = ‘02’
ORDER BY  GRUPO
```

Esta sentencia SELECT es la misma que la del ejemplo nº 13, pero los registros seleccionados de la tabla ALUMNOS serán aquellos que contengan en la columna RELI el valor '02', clave asignada a la religión EVANGELICA.

Los registros aparecerán ordenados por la columna GRUPO.

18. ALUMNOS: RELIGION EVANGE. (N)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO
FROM ALUMNOS
WHERE RELI = ‘02’
ORDER BY NOMBRE
```

Se trata del mismo ejemplo anterior, pero los registros aparecerán ordenados por NOMBRE.

19. ALUMNOS: NO RELIGION (GRUPOS)

```
SELECT GRUPO, APE1+' '+APE2+', '+NOMB AS NOMBRE
FROM ALUMNOS
WHERE RELI = ‘00’
ORDER BY  GRUPO
```

Esta sentencia SELECT es la misma que la del ejemplo nº 13, pero los registros seleccionados de la tabla ALUMNOS serán aquellos que contengan en la columna RELI el valor '00', clave asignada a la los alumnos que no optan por ninguna religión.

Los registros aparecerán ordenados por la columna GRUPO.

20. ALUMNOS: NO RELIGION (NOMBRE)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO
FROM ALUMNOS
WHERE RELI = ‘00’
ORDER BY NOMBRE
```

Se trata del mismo ejemplo anterior, pero los registros aparecerán ordenados por NOMBRE.

21. ALUMNOS: DOMIC. Y TFNOS.

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, DOMI AS DOMICILIO, TFNO
AS TFN1, CNTC1 AS PERSONA1, TFNO2 AS TFN2, CNTC2 AS PERSONA2
FROM ALUMNOS
ORDER BY NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el NOMBRE completo, las columnas DOMI (domicilio), TFNO (teléfono), CNTC1 (persona de contacto 1), TFNO2 (otro teléfono) y CNTC2 (otra persona de contacto).

Los registros aparecerán ordenados por NOMBRE.

22. ALUMNOS CON N.E.E. (TIPO)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO, LITERAL AS TIPO
FROM ALUMNOS, LTIPNEE
WHERE T_NEE = CLAVE
AND CLAVE <> '00'
ORDER BY NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas GRUPO, y de la tabla LTIPNEE (tipos de necesidades educativas especiales) la columna LITERAL. .

Se establecen como condiciones que la columna T_NEE de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LTIPNEE y que ésta columna tenga un valor diferente a '00' es decir, que tengan algún tipo de necesidad.

Los registros aparecerán ordenados por NOMBRE.

23. ALUMNOS CON N.C.E. (TIPO)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO, LITERAL AS TIPO
FROM ALUMNOS, LTIPNEE
WHERE A_CMP = CLAVE
AND CLAVE <> '08'
ORDER BY NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS las columnas APE1, APE2, NOMB y GRUPO, LITERAL de la tabla LTIPNEE.

Tiene como condiciones que la columna T_NEE de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LTIPNEE (Tipos de necesidades educativas especiales) y que ésta columna tenga un valor diferente a '08' éste valor representa "No necesitan compensación educativa".

Los registros aparecerán ordenados por el alias NOMBRE.

24. ALUMNOS EXTRANJEROS (NACIONAL)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO, LITERAL AS NACIONALIDAD
FROM ALUMNOS, LPAISES
WHERE NACION = CLAVE
AND NACION <> '724'
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS las columnas NOMBRE completo y GRUPO, y de la tabla LPAISES la columna LITERAL.

Incluye como condiciones que la columna NACION de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LPAISES (nombre de los diferentes países) y que esta columna tenga un valor diferente a ‘724’, valor que representa a España.

Los registros aparecen ordenados por el alias NOMBRE..

25. ALUMNOS EXTRANJEROS (PAIS NAC)

```
SELECT APE1+' '+APE2+', '+NOMB AS NOMBRE, GRUPO, LITERAL AS PAIS_NAC
FROM ALUMNOS, LPAISES
WHERE PAI_NAC = CLAVE
AND PAI_NAC <> ‘724’
ORDER BY NOMBRE
```

Esta sentencia SELECT es idéntica a la anterior, pero en lugar del país que indica la NACIONALIDAD del alumno (NACION), selecciona el PAIS DE NACIMIENTO (PAIS_NAC).

26. PROFESORES Y TUTORIAS

```
SELECT APE1 +'  '+ APE2 +', '+ NOMB AS NOMBRE,GRU_TUT TUTORIA
FROM PROFEDG
ORDER BY NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla PROFEDG las columnas NOMBRE completo del profesor y GRU_TUT (grupo de alumnos del que es tutor).

Los registros aparecerán ordenados por NOMBRE.

27. ALUMNOS USUARIOS DE COMEDOR

```
SELECT X.NUMERO, X.APE1+’ ‘+X.APE2+’, ‘+X.NOMB NOMBRE, X.GRUPO
FROM ALUMNOS X
WHERE EXISTS (SELECT * FROM USUASERV WHERE X.NUMERO=NUMERO AND CLAVE=’01’)
ORDER BY NOMBRE
```

Esta sentencia es similar a la nº 5, pero no se recoge la tira (alias DIAS) que aquella escribe.

28. ALUMNOS USUARIOS DE TRANSPORTE

```
SELECT X.NUMERO, X.APE1+’ ‘+X.APE2+’, ‘+X.NOMB NOMBRE,
X.GRUPO
FROM ALUMNOS X
WHERE EXISTS (SELECT * FROM USUASERV WHERE X.NUMERO=NUMERO AND CLAVE=’03’)
ORDER BY NOMBRE
```

Esta sentencia es similar a la nº 7, pero no se recoge la tira (alias DIAS) que aquella escribe.

29. TRANSPORTE: RUTAS Y PARADAS

```
SELECT N_RUTA Nº_RUTA, N_PARA Nº_PARADA , L_PARA LITERAL_PARADA
FROM RUTAS
```

ORDER BY N_RUTA,N_PARA

Esta sentencia SELECT selecciona de la tabla RUTAS las columnas N_RUTA (nº de ruta de transporte), N_PARA (nº de parada) y L_PARA (literal de la parada). Utiliza un alias para cada una de las tres columnas para la cabecera del listado.

Los registros aparecerán ordenados por la columna N_RUTA y dentro de ésta por N_PARA.

30. ALUMNOS: F. NACIMIENTO Y ALTA

```
SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,F_NACI NACIMIENTO,F_ALTA ALTA
FROM ALUMNOS
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas GRUPO, F_NACI y F_ALTA.

Los registros aparecerán ordenados por el alias NOMBRE.

31. ALUMNOS: OTRAS RELIGIONES (N)

```
SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,LITERAL RELIGION
FROM ALUMNOS,RELIGION
WHERE RELI=CLAVE AND RELI>'01'
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columna GRUPO, y de la tabla RELIGION la columna LITERAL.

Condiciona que la columna RELI de tabla ALUMNOS sea mayor que '01', es decir, que opte por la enseñanza de alguna de las religiones, y que coincida con el valor de la columna CLAVE de la tabla RELIGION. El código '01' corresponde a la " RELIGION CATOLICA" y el código '00' corresponde a "SIN RELIGION".

El listado sale ordenado por el alias NOMBRE.

32. ALUMNOS: IDIOMAS ELEGIDOS

```
SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,X.LITERAL IDIOMA1,Y.LITERAL IDIOMA2
FROM ALUMNOS,IDIOMAS X,IDIOMAS Y
WHERE (IDIO1=X.CLAVE OR IDIO2=Y.CLAVE) AND IDIO1||IDIO2>'00'
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y la columna GRUPO y, de la tabla IDIOMAS, la columna LITERAL.

La tabla IDIOMAS se utiliza como alias "X " y como alias "Y" ya que IDIO1 e IDIO2 de la tabla ALUMNOS obtienen el nombre del idioma de la misma tabla (IDIOMAS).

Se establece la condición de que la columna IDIO1 o la IDIO2 de tabla ALUMNOS sea igual a la columna CLAVE de la tabla IDIOMAS y que la columna IDIO1 o la IDIO2 de la tabla ALUMNOS sea mayor que "00".

El código "00" en la columna CLAVE de la tabla IDIOMAS, indica que no se ha elegido idioma alguno.
La expresión IDIO1||IDIO2>'00' indica que la columna IDIO1 o la IDIO2 deben contener un valor mayor que '00'.

El listado sale ordenado por el alias NOMBRE.

33. ALUMNOS: OPTATIVAS ELEGIDAS

```
SELECT APE1+' '+APE2+', '‘+NOMB NOMBRE,GRUPO,X.LITERAL OPTATIVA1,Y.LITERAL OPTATIVA2

FROM ALUMNOS,OPTATIVA X,OPTATIVA Y
WHERE (OPTA1=X.CLAVE OR OPTA2=Y.CLAVE)
AND OPTA1||OPTA2>’00’
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NONMMBRE completo del alumno y la columna GRUPO y, de la tabla OPTATIVA , la columna LITERAL.

La tabla OPTATIVA utiliza como alias “X “ e “Y” ya que OPTA1 e OPTA2 de la tabla ALUMNOS obtienen el LITERAL de la optativa de la misma tabla (OPTATIVA).
Condiciona que la columna OPTA1 o la OPTA2 de tabla ALUMNOS sea igual a la columna CLAVE de la tabla OPTATIVA y que la columna OPTA1 o la OPTA2 de la tabla ALUMNOS tengan un valor mayor que “00”.

El código “00” de la tabla de optativas, corresponde a SIN OPTATIVA.

Estas condiciones permiten selecciona el registro del alumno que tenga escogida una o dos optativas,.

El listado sale ordenado por el alias NOMBRE.

34. ALUMNOS: DOMIC. Y TFNOS. (GRU)

```
SELECT GRUPO,APE1+' '+APE2+', '+NOMB NOMBRE,DOMI DOMICILIO,TFNO TELEFONO1,CNTC1 CONTACTO1,TFNO2 TELEFONO2,CNTC2 CONTACTO2
FROM ALUMNOS
ORDER BY GRUPO,NOMBRE
```

Esta sentencia SELECT, selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas DOMI, TFNO,CNTC1,TFNO2 y CNTC2.

El listado sale ordenado por el alias NOMBRE.

35. ALUMNOS EXTRANJEROS (AÑO NACI)

```
SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,F_NACI NACIMIENTO,LITERAL PAIS
FROM ALUMNOS,LPAISES
WHERE PAI_NAC=CLAVE AND PAI_NAC<>’724’
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas GRUPO, F_NACI y la columna LITERAL de la tabla LPAISES.

Se recogen como condiciones que la columna PAI_NAC de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LPAISES (relación de países) y que la columna PAI_NAC tenga un valor diferente a ‘724’, valor que representa a España.

El listado sale ordenado por el alias NOMBRE.

36. ALUMNOS CON N.E.E. (AÑO NACIM)

SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,F_NACI NACIMIENTO,LITERAL NECESIDAD
FROM ALUMNOS,LTIPNEE
WHERE T_NEE=CLAVE AND T_NEE<>'00'
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas GRUPO y F_NACI y, de la tabla LTIPNEE, la columna LITERAL

Se establecen como condiciones que la columna T_NEE de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LTIPNEE (tipos de necesidades educativas especiales) y que esta columna tenga un valor diferente a '00', valor que indica que el alumno no tiene necesidades educativas especiales.

Los registros aparecerán ordenados por NOMBRE.

37. ALUMNOS DE 6º DE PRIMARIA

SELECT APE1+' '+APE2+', '+NOMB NOMBRE,F_NACI NACIMIENTO,DOMI DOMICILIO,LOCA LOCALIDAD,TFNO TELEFONO,NOM_PAD PADRE,NOM_MAD MADRE
FROM ALUMNOS
WHERE CURSO='P6'
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas F_NACI, DOMI, LOCA, TFNO, NOM_PAD y NOM_MAD.

Recoge como condiciones que la columna CURSO de la tabla ALUMNOS tenga el valor "P6" , es decir, corresponda a sexto curso de primaria.

El listado sale ordenado por el alias NOMBRE.

38. ALUMNOS CON N.C.E. (AÑO NACIM)

SELECT APE1+' '+APE2+', '+NOMB NOMBRE,GRUPO,F_NACI NACIMIENTO,LITERAL COMPENSACION
FROM ALUMNOS,LTIPNEE
WHERE A_CMP=CLAVE AND A_CMP<>'08'
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas GRUPO y F_NACI, y la columna LITERAL de la tabla LTIPNEE.

Recoge como condiciones que la columna A_CMP de la tabla ALUMNOS sea igual a la columna CLAVE de la tabla LTIPNEE (tipos de necesidades educativas especiales) y que la columna A_CMP tenga un valor diferente a '08', valor que indica que existe necesidad de enseñanza compensatoria.

Los registros aparecerán ordenados por el alias NOMBRE .

39. ALUMNOS DE 2º DE E.S.O.

SELECT APE1+' '+APE2+', '+NOMB NOMBRE,F_NACI NACIMIENTO,DOMI DOMICILIO,LOCA LOCALIDAD,TFNO TELEFONO,NOM_PAD PADRE,NOM_MAD MADRE
FROM ALUMNOS
WHERE CURSO='S2'

ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas F_NACI, DOMI, LOCA, TFNO, NOM_PAD y NOM_MAD.

Recoge como condiciones que la columna CURSO de la tabla ALUMNOS sea igual a “S2”, es decir, a segundo curso de secundaria.

Los registros aparecerán ordenados por el alias NOMBRE .

40. ALUMNOS DE 1º DE PRIMARIA

SELECT APE1+' '+APE2+', '+NOMB NOMBRE,F_NACI NACIMIENTO,DOMI DOMICILIO,LOCA LOCALIDAD,TFNO TELEFONO,NOM_PAD PADRE,NOM_MAD MADRE
FROM ALUMNOS
WHERE CURSO=’P1’
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y las columnas F_NACI, DOMI, LOCA, TFNO, NOM_PAD y NOM_MAD.

Recoge como condiciones que la columna CURSO de la tabla ALUMNOS sea igual a “P1”, es decir, a primer curso de primaria.

Los registros aparecerán ordenados por el alias NOMBRE .

41. ALUMNOS: LIBRE 4 ACTIVADO

SELECT APE1+' '+APE2+', '+NOMB NOMBRE, GRUPO
FROM ALUMNOS
WHERE LIBRE4=TRUE
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y la columna GRUPO.

Recoge como condición que la columna LIBRE4 de la tabla ALUMNOS esté activada (TRUE). Esta columna es de tipo Booleano y puede estar activada (TRUE) o desactivada (FALSE) y representa una situación del registro.

Los registros aparecerán ordenados por el alias NOMBRE.

42. ALUMNOS: LIBRE 4 DESACTIVADO

SELECT APE1+ ' '+APE2+', '+NOMB NOMBRE, GRUPO
FROM ALUMNOS
WHERE LIBRE4=FALSE
ORDER BY NOMBRE

Esta sentencia SELECT selecciona de la tabla ALUMNOS el NOMBRE completo y la columna GRUPO.

Recoge como condición que la columna LIBRE4 de la tabla ALUMNOS esté desactivada (FALSE). Esta columna es de tipo Booleano, puede estar activada (TRUE) o desactivada (FALSE) y representa una situación del registro.

Los registros aparecerán ordenados por el alias NOMBRE .

43. PROFES F. NACIM. (APELLIDO)

```
SELECT APE1 +'  '+ APE2 + ' , ' + NOMB AS NOMBRE, F_NACI AS F_NACIMIENTO
FROM PROFEDG
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla PROFEDG el NOMBRE completo y la columna F_NACI.

Los registros aparecerán ordenados por el alias NOMBRE .

44. PROFES F. NACIM. (F.NA)

```
SELECT APE1 + ' ' + APE2 +', ' + NOMB AS NOMBRE, F_NACI AS F_NACIMIENTO
FROM PROFEDG
ORDER BY F_NACI
```

Se trata de la misma consulta anterior, pero ordenada por fecha de nacimiento.

45. PROFESORES (LISTA SIMPLE)

```
SELECT APE1 + ' ' + APE2 + ', ' + NOMB AS NOMBRE
FROM PROFEDG
ORDER BY NOMBRE
```

Se trata de una simple lista de profesores ordenada alfabéticamente.

46. PROFESORES (LISTA PARA FIRMAS)

```
SELECT APE1 + ' ' + APE2 + ', ' + NOMB AS NOMBRE,
‘ _______________________’ AS FIRMA
FROM PROFEDG
ORDER BY NOMBRE
```

Esta sentencia SELECT selecciona de la tabla PROFEDG el NOMBRE completo de los profesores; crea un string, consistente en una línea.

Los registros aparecerán ordenados por el alias NOMBRE.

47. PROFESORES (FIRMAS CLAUSTRO)

```
SELECT APE1 + ' ' + APE2 + ', ' + NOMB AS NOMBRE,
‘ _______________________’ AS FIRMA
FROM PROFEDG
ORDER BY NOMBRE
```

Se trata del mismo ejemplo anterior, pero en la cabecera del listado aparecerá el título 'FIIRMAS CLAUSTRO' en lugar de 'LISTA PARA FIRMAS' del anterior caso.