



## 3.1 Introducción

El lenguaje SQL (*Structured Query Language*) es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos relacional, por tanto, permite la comunicación con el sistema de gestión de la base de datos. Es tan conocido en bases de datos relacionales que muchos lenguajes de programación incorporan sentencias SQL como parte de su repertorio; tal es el caso de Visual Basic. Entre las principales características del SQL podemos destacar las siguientes:

- Es un lenguaje para todo tipo de usuarios (administradores, desarrolladores y usuarios normales).
- El usuario que emplea SQL especifica qué quiere, no dónde ni cómo.
- Permite hacer cualquier consulta de datos.
- Es posible manejarlo para consultas, actualizaciones, definición de datos y control.

Se puede usar de forma interactiva (el usuario escribe las órdenes desde el teclado de un terminal y al instante obtiene los resultados en la pantalla) y de forma embebida (mezclando las instrucciones SQL con las instrucciones propias del lenguaje, tal es el caso del lenguaje PL/SQL).

La Figura 3.1 muestra cómo funciona SQL en una arquitectura cliente/servidor. Por un lado, se dispone de una máquina servidora con una base de datos que contiene datos importantes para el negocio. Por otro lado, tenemos una máquina cliente con un usuario que está ejecutando una aplicación que necesita acceder a los datos allí almacenados. La aplicación realiza una petición al sistema gestor de base de datos, este la procesa y envía los datos a la aplicación que los solicitó.

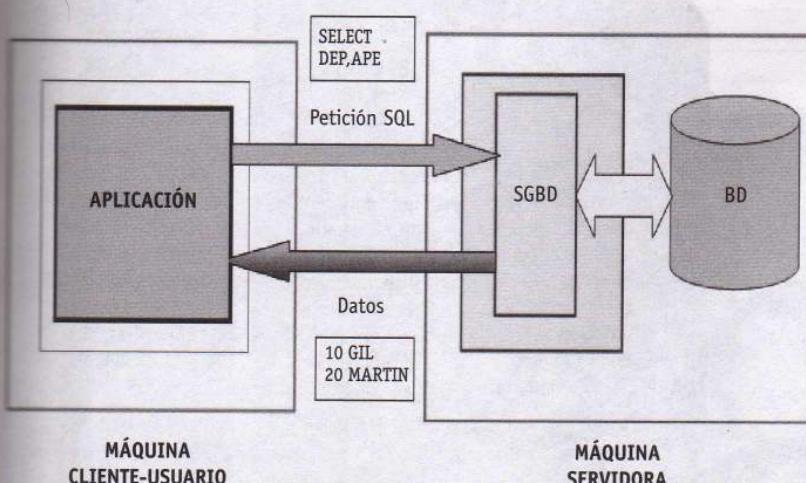


Figura 3.1. Cómo funciona SQL.



## 3.2 Historia y estandarización

El lenguaje SQL fue desarrollado sobre un prototipo de gestor de bases de datos relacionales denominado SYSTEM R y diseñado por IBM a mediados de los años setenta. Incluía lenguajes de consultas, entre ellos **SEQUEL** (*Structured English Query Language*). Más tarde se renombró como SQL.

En 1979 Oracle Corporation presentó la primera implementación comercial de SQL, que estuvo disponible antes que otros productos de IBM. Por su parte, IBM desarrolló productos herederos del prototipo SYSTEM R, como DB2 y SQL/DS.

El instituto **ANSI** (*American National Standard Institute*) adoptó el lenguaje SQL como estándar para la gestión de bases de datos relacionales en octubre de 1986. En 1987 lo adopta **ISO** (*International Standardization Organization*).

En 1989 el estándar **ANSI/ISO**, revisado y ampliado, se llamó **SQL-89** o estándar **SQL1**. Tres años más tarde se aprueba el estándar **ANSI SQL2** o **SQL-92**. En 1999 se aprueba el estándar **SQL:1999** que introduce mejoras con respecto al anterior. **SQL:2003** es el actual estándar. Revisa todos los apartados de SQL:1999 y añade uno nuevo, el apartado 14: **SQL/XML**.

La Figura 3.2 muestra un esquema con la evolución de los estándares y las novedades que va incorporando cada uno con respecto al anterior.

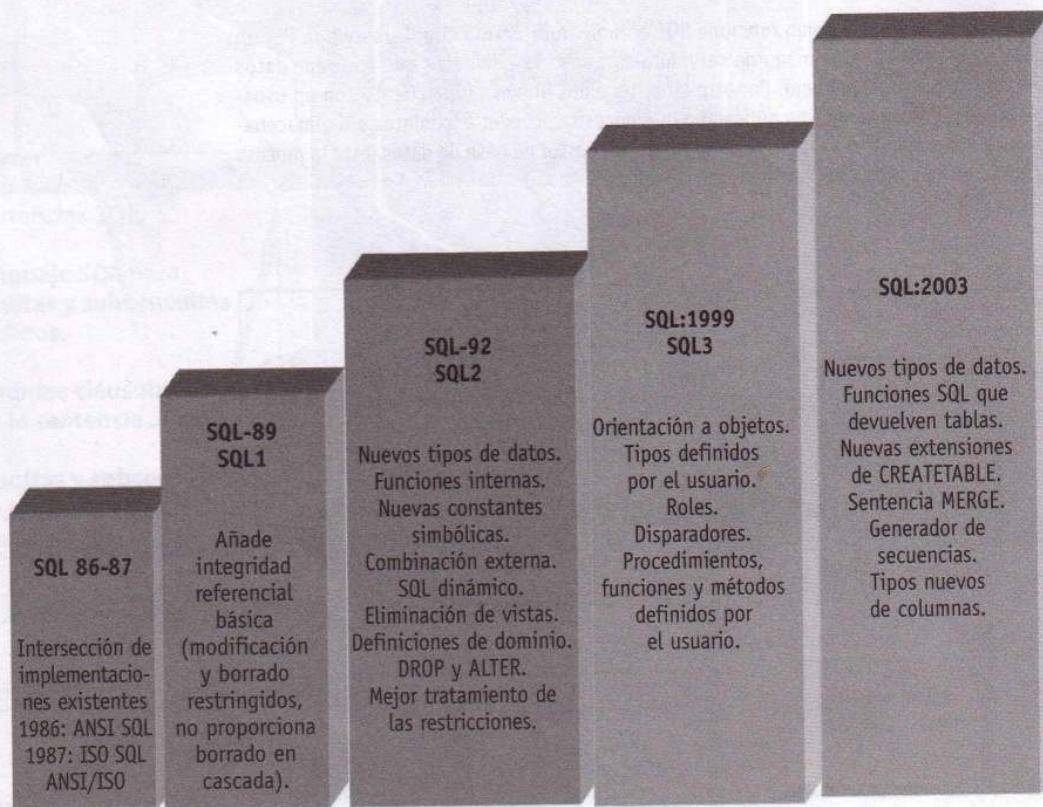


Figura 3.2.  
Estándares SQL.

### 3. Introducción a SQL

#### 3.3 Tipos de sentencias SQL



## 3.3 Tipos de sentencias SQL

El lenguaje SQL proporciona un gran repertorio de sentencias que se utilizan en variadas tareas, como consultar datos de la base de datos, crear, actualizar y eliminar objetos de la base de datos, crear, actualizar y eliminar datos de los objetos, controlar el acceso a la base de datos y a los objetos. Dependiendo de las tareas, podemos clasificar las sentencias SQL en varios tipos. En la Tabla 3.1 se han expuesto las sentencias más importantes y las usadas con mayor frecuencia.

SENTENCIA	DESCRIPCIÓN
<b>Manipulación de datos</b>	
SELECT	Recupera datos de la base de datos.
INSERT	Añade nuevas filas de datos a la base de datos.
DELETE	Suprime filas de datos de la base de datos.
UPDATE	Modifica datos existentes en la base de datos.
<b>Definición de datos</b>	
CREATE TABLE	Añade una nueva tabla a la base de datos.
DROP TABLE*	Suprime una tabla de la base de datos.
ALTER TABLE*	Modifica la estructura de una tabla existente.
CREATE VIEW*	Añade una nueva vista a la base de datos.
DROP VIEW*	Suprime una vista de la base de datos.
CREATE INDEX*	Construye un índice para una columna.
DROP INDEX*	Suprime el índice para una columna.
CREATE SYNONYM*	Define un alias para un nombre de tabla.
DROP SYNONYM*	Suprime un alias para un nombre de tabla.
<b>Control de acceso</b>	
GRANT	Concede privilegios de acceso a usuarios.
REVOKE	Suprime privilegios de acceso a usuarios.
<b>Control de transacciones</b>	
COMMIT	Finaliza la transacción actual.
ROLLBACK	Aborta la transacción actual.
<b>SQL Programático</b>	
DECLARE	Define un cursor para una consulta.
OPEN	Abre un cursor para recuperar resultados de consulta.
FETCH	Recupera una fila de resultados de consulta.
CLOSE	Cierra un cursor.

\*No existían en el estándar SQL1

Tabla 3.1. Tipos de sentencias SQL.

## 4. Componentes sintácticos de una sentencia

Las sentencias SQL tienen una forma básica. Véase la Figura 3.3. Todas comienzan con un verbo, que es una palabra clave que describe lo que hace la sentencia (por ejemplo SELECT, INSERT, UPDATE, CREATE). A continuación, le siguen una o más cláusulas que especifican los datos con los que opera la sentencia. Estas también comienzan con una palabra clave como WHERE o FROM. Algunas son opcionales y otras obligatorias. Muchas contienen nombres de tablas o de columnas, palabras reservadas, constantes o expresiones adicionales.

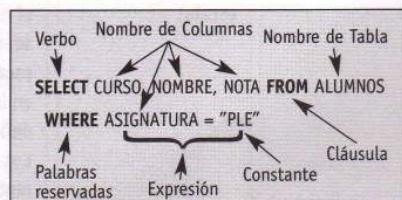


Figura 3.3. Componentes de una sentencia SQL.



## B. Cómo procesa el SGBD una sentencia

Para procesar una sentencia SQL el sistema gestor de base de datos recorre una serie de pasos:

1. *Analiza la sentencia.* Comprueba que está sintácticamente bien escrita.
2. *Valida la sentencia.* Comprueba la sentencia semánticamente. Es decir, comprueba si las tablas y las columnas referenciadas en la sentencia existen, si el usuario tiene privilegios para realizar esa operación sobre la tabla, etcétera.
3. *Optimiza la sentencia.* El sistema gestor de base de datos explora la forma de llevar a cabo la ejecución de la sentencia.
4. *Genera un plan de aplicación para la sentencia.* Se genera código ejecutable.
5. *Ejecuta el plan de aplicación.*

El análisis de la sentencia no requiere acceso a la base de datos y, por tanto, suele realizarse rápidamente. Sin embargo, la optimización es un proceso muy intensivo de CPU y precisa acceder a la base de datos.

## 3.4 Tipos de datos

Cuando se crea una tabla instrucción CREATE TABLE (ver Unidad 7) se debe especificar el tipo de dato para cada una de sus columnas, estos tipos de datos definen el dominio de valores que cada columna puede contener. La Tabla 3.2 muestra los distintos tipos de datos de Oracle SQL. Los códigos de los tipos de datos se emplean internamente por Oracle. El código del tipo de dato de una columna es devuelto cuando se usa la función DUMP (ver Unidad 4).

CÓDIGO	Tipo de dato	Características
1	VARCHAR2(tamaño)	Almacena cadenas de caracteres de longitud variable, la longitud se expresa en <i>tamaño</i> . La longitud máxima es de 4.000 caracteres (bytes), y la mínima es 1. Ejemplo: DIRECCION VARCHAR2(40)
96	CHAR(tamaño)	Almacena caracteres con una de longitud fija especificada en <i>tamaño</i> . La longitud máxima es de 2.000 bytes. El mínimo tamaño y tamaño por defecto es 1 byte. Ejemplo: CODIGO CHAR(6)
2	NUMBER(precisión, escala)	Este tipo almacena datos numéricos, tanto enteros como decimales, con o sin signo. <i>Precisión</i> representa el número total de dígitos que va a tener el dato que se define; el rango va de 1 a 38. <i>Escala</i> representa el número de dígitos a la derecha del punto decimal. El rango va de -84 a 127. Si se especifica una escala negativa, el número es redondeado tantos dígitos a la izquierda del punto decimal como se indicó en la escala. Si el número no tiene decimales, se puede omitir la escala. Ejemplo: SALARIO NUMBER(7,2) define la columna SALARIO con 7 dígitos, 2 de ellos decimales. Los ejemplos mostrados en la Tabla 3.3 muestran el modo en que Oracle almacena los datos con diferentes precisiones y escalas.

Tabla 3.2. Tipos de datos en Oracle 10g.

### 3. Introducción a SQL

#### 3.4 Tipos de datos



os:

ba  
eba  
rio

lle-

arse  
cisa

5

tipo  
ores  
s de  
digo  
14).

ud

00

nta  
re-  
na  
icó  
,2)  
fe-

CÓDIGO	Tipo de dato	Características
8	LONG	Almacena cadenas de caracteres de longitud variable que contengan hasta 2 gigabytes de información. Se puede usar este tipo para almacenar textos muy grandes. Ejemplo: TEXT LONG.
12	DATE	Este tipo de dato está sujeto a algunas restricciones: sólo se puede definir una columna LONG por tabla, no pueden aparecer en restricciones de integridad ( <i>constraints</i> ), no sirve para indexar, una función almacenada no puede devolver un valor LONG, no se puede utilizar como argumento de funciones, no se puede referenciar como subconsulta en la creación de tablas ni inserción de filas, las variables o argumentos de bloques PL/SQL no se pueden declarar como tipo LONG, no es posible su uso en cláusulas WHERE, GROUP BY, ORDER BY, CONNECT BY o DISTINCT, ni con operaciones de UNION, INTERSECT y MINUS.
23	RAW(tamaño)	Almacena información de fechas y horas. Para cada tipo DATE se almacena la siguiente información: Siglo/Año/Mes/Día/Hora/Minutos/Segundos. Por omisión, el valor para el formato de la fecha se especifica con el parámetro <b>NLS_DATE_FORMAT</b> , y es una cadena de caracteres, como ésta 'DD/MM/YY', que representa día del mes/dos dígitos del mes/dos últimos dígitos del año. El formato de la fecha se puede cambiar mediante la orden ALTER SESSION y variando el parámetro <b>NLS_DATE_FORMAT</b> . Ejemplo: FECHA DATE.
24	LONG RAW	Almacena datos binarios. Es similar al tipo VARCHAR2, con la diferencia de que maneja cadenas de bytes en lugar de cadenas de caracteres. El tamaño máximo es de 2.000 bytes.
69	ROWID	Cadena hexadecimal que representa la dirección de una fila en su tabla.
1	NVARCHAR2(tamaño)	Tipo similar al VARCHAR2 solo que el tamaño de un carácter depende de la elección del juego de caracteres. El máximo tamaño es de 4.000 bytes.
96	NCHAR(tamaño)	Tipo similar al CHAR solo que el tamaño de un carácter depende de la elección del juego de caracteres. El máximo tamaño es de 2.000 bytes.
112	CLOB	Similar a LONG, se usa para objetos carácter. El tamaño máximo es de 4 gigabytes.
112	NCLOB	Similar al anterior solo que el tamaño del carácter depende del juego de caracteres.
113	BLOB	Similar a LONG RAW, se usa para objetos binarios. El tamaño máximo es de 4 gigabytes.

Tabla 3.2 (Continuación). Tipos de datos en Oracle 10g.

Dato actual	Formato	Almacenamiento
7456123.89	NUMBER	7456123.89
7456123.89	NUMBER(9)	7456124
7456123.89	NUMBER(9,2)	7456123.89
7456123.89	NUMBER(9,1)	7456123.9
7456123.8	NUMBER(6)	ERROR-Valor mayor que el que permite la precisión especificada para esta columna.
.		
7456123.8	NUMBER(15,1)	7456123.8
7456123.89	NUMBER(7,-2)	7456100
7456123.89	NUMBER(-7,2)	ERROR-Especificador de precisión numérica está fuera de rango (1 a 38).

Tabla 3.3. Datos numéricos, precisiones y escalas.



## 3.5 SQL\*Plus

ahamemni en la página 5 para más información.

Para aprender el lenguaje SQL nos serviremos del intérprete de sentencias SQL llamado **SQL\*Plus**. Si deseamos empezar una sesión con SQL\*Plus tenemos que conectarnos como usuarios Oracle; por tanto, el administrador de la base de datos deberá proporcionarnos un identificador de usuario con su *password* asociado. Este usuario dispondrá de una serie de derechos sobre los objetos de la base de datos. También deberá proporcionarnos la *Cadena de conexión* o *Cadena de Host* para conectarnos a la base de datos. Ésta se crea con la utilidad *Net Configuration Assistant* de Oracle. No hace falta escribir nada en este campo si la base de datos es local.

En una instalación típica podemos entrar haciendo clic en *Inicio/Programas/Oracle Developer Suite/Application Development/SQL\*Plus*. O bien buscamos el archivo *sqlplusw.exe* y hacemos doble clic en él. Se abre una ventanita similar a la mostrada en la Figura 3.4, desde donde escribiremos el nombre de usuario, la contraseña y la cadena de conexión.

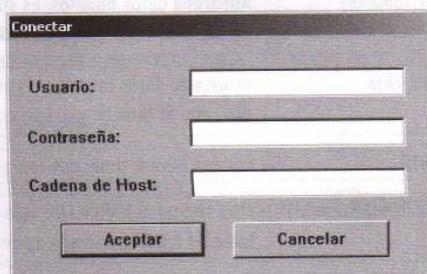


Figura 3.4. Nos conectamos a Oracle.

Una vez autenticados se visualiza una ventana similar a la mostrada en la Figura 3.5, desde la que veremos el *prompt SQL>* que nos indica que estamos conectados al gestor Oracle. Desde este momento podemos enviar sentencias SQL al gestor.

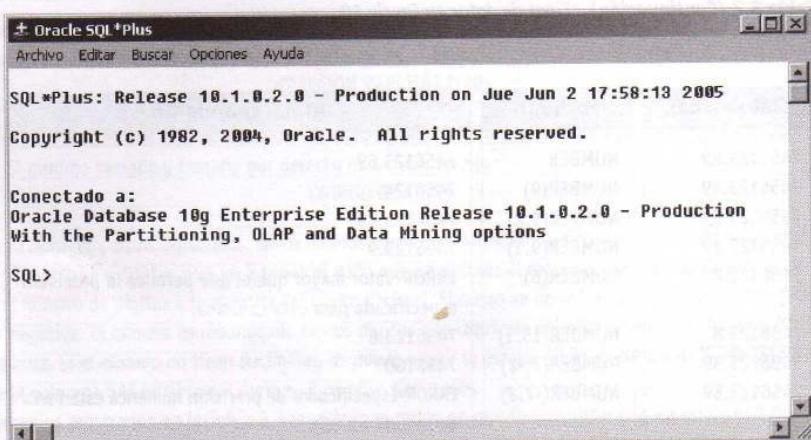
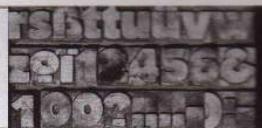


Figura 3.5. Ventana Oracle SQL\*Plus.

### 3. Introducción a SQL

3.5 SQL\*Plus



Cuando trabajamos con SQL\*Plus disponemos de un *buffer* de edición que contiene la última sentencia SQL que se intentó ejecutar. Mientras una sentencia se encuentre en el *buffer*, será posible modificarla a través de un conjunto de comandos de edición: SQL> ED. Este comando invocará al editor del sistema (por ejemplo, NOTEPAD.EXE), que abrirá el fichero asociado al *buffer* de edición (afiedt.buf), véase la Figura 3.6. Se puede usar para su modificación o archivado.

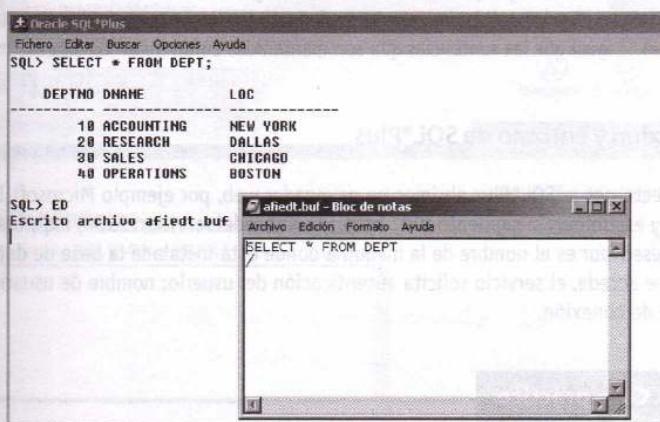


Figura 3.6. Buffer de edición afiedt.buf.

SQL> LIST	Visualiza el contenido del <i>buffer</i> . También podemos poner: SQL> L
SQL> LIST n	Se lista el número de línea n.
SQL> SAVE fichero	Almacena el contenido del <i>buffer</i> en <i>fichero.sql</i> .
SQL> GET fichero	Recupera en el <i>buffer</i> el contenido del archivo <i>fichero.sql</i> .
SQL> START fichero	Ejecuta el contenido almacenado en <i>fichero.sql</i> . Si el fichero no está en el directorio, tendríamos que poner todo el trayecto:
SQL> START c:\programas\fichero.sql	SQL> START c:\programas\fichero.sql
SQL> RUN	Repite la ejecución de la última sentencia o de lo que hay en el <i>buffer</i> . También podemos poner: SQL> R
SQL> INPUT	Añade una línea a continuación de la actual activa.
SQL> DEL	Borra la línea actual.
SQL> SPOOL fichero	Todas las salidas por pantalla se almacenarán en un <i>fichero.lst</i> .
SQL> SPOOL OFF	Libera el almacenamiento de salidas por pantalla.
SQL> CLEAR SCR	Borra la pantalla.

Tabla 3.4. Comandos que podemos usar desde el prompt.

#### Actividades propuestas

- 1 Conéctate como usuario Scott, contraseña Tiger y prueba las sentencias: SELECT \* FROM EMP; y SELECT \* FROM DEPT; y repasa todos los comandos vistos anteriormente. Anota lo que hace cada comando. Prueba las sentencias escribiendo cada palabra en una línea (es decir, escribimos SELECT, pulsamos la tecla Enter, escribimos \*, pulsamos la tecla Enter, y así sucesivamente). ¿Qué observas?



## 3.6 iSQL\*Plus

En Oracle 9i se incluyó la versión web de SQL\*Plus como componente adicional en la instalación de la base de datos. La versión 10g incluye este componente llamado **iSQL\*Plus** que se instala en el servidor durante la instalación de la base de datos. Permite a los usuarios la ejecución de sentencias SQL y PL/SQL a través de un navegador web en el que se indica la dirección URL del servidor. Las salidas se generan en formato HTML con la posibilidad de guardarlas en archivos al igual que las sentencias que son mantenidas en un histórico durante la sesión.

### Conexión y entorno de SQL\*Plus

Para conectarnos a iSQL\*Plus abrimos un navegador web, por ejemplo Microsoft Internet Explorer y escribimos la siguiente URL: <http://nombredelservidor:5560/isqlplus>, donde *nombredelservidor* es el nombre de la máquina donde está instalada la base de datos. Una vez que se accede, el servicio solicita autenticación del usuario: nombre de usuario, clave y cadena de conexión.

### Caso práctico



- 1 Veamos cómo nos conectamos desde una máquina cliente a una máquina servidora de nombre SERVIDOR que tiene instalada una base de datos *Oracle 10g*.

En primer lugar, abrimos el navegador y escribimos la siguiente URL: <http://servidor:5560/isqlplus>.

A continuación, escribimos el nombre de usuario: SCOTT, la contraseña TIGER y el identificador de conexión. Se muestra una imagen similar a la mostrada en la Figura 3.7. En este caso, el identificador de conexión coincide con el nombre de base de datos global o SID. Este dato lo debe suministrar el administrador de la base de datos.

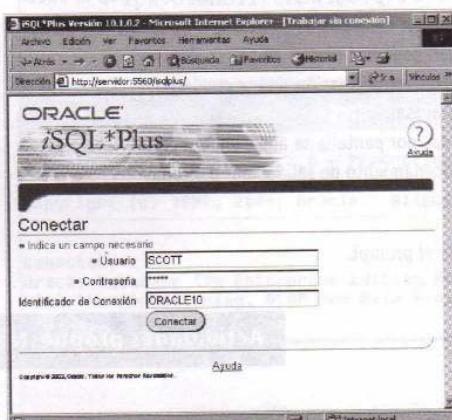


Figura 3.7. Conexión iSQL\*Plus.

Se hace clic en el botón *Conectar*. Tras validarse correctamente la conexión aparece la pantalla de trabajo de iSQL\*Plus.

### 3. Introducción a SQL

3.6 iSQL\*Plus



La pantalla de trabajo de iSQL\*Plus presenta un área donde se escribirán las sentencias SQL, una serie de botones en la parte inferior y varios hipervínculos en la parte superior. Véase la Figura 3.8. Estos elementos se resumen en la Tabla 3.5.

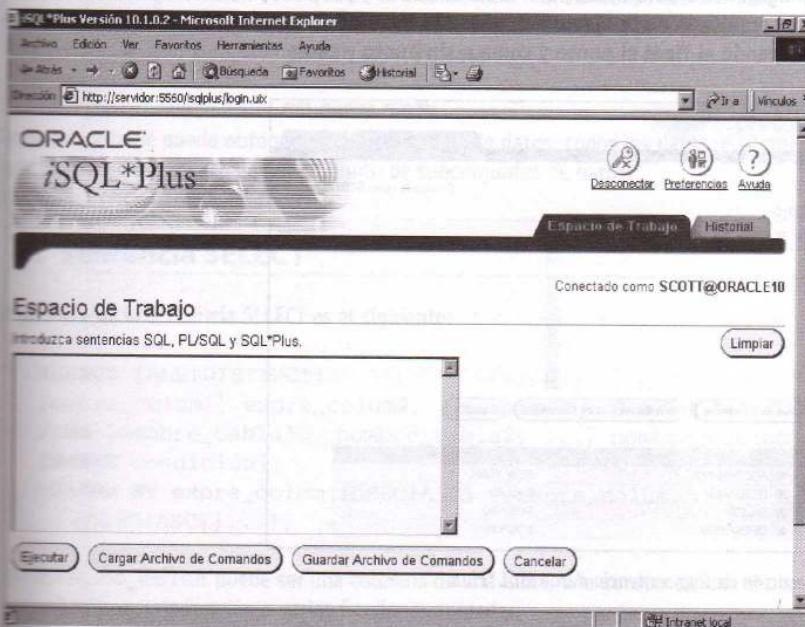


Figura 3.8. Espacio de trabajo iSQL\*Plus.

Elemento	Descripción
Ejecutar	Ejecuta la sentencia escrita en el espacio de trabajo, el resultado se muestra en la parte inferior debajo de los botones.
Cargar Archivo de Comandos	Muestra una nueva pantalla desde la que se puede cargar un archivo de comandos para ejecutarlo en el espacio de trabajo. Estos archivos tendrán la extensión <b>sql</b> .
Guardar Archivo de Comandos	Muestra un cuadro de diálogo donde hemos de dar un nombre al archivo que guardará lo escrito en el espacio de trabajo. Estos se guardan con extensión <b>sql</b> .
Cancelar	Cancela cualquier archivo de comandos que se esté ejecutando pero no limpia el espacio de trabajo.
Limpiar	Limpia el área de trabajo y todas las salidas mostradas.
Desconectar	Finaliza la sesión iSQL*Plus.
Preferencias	Abre una nueva pantalla desde la que se puede configurar los valores de la interfaz.
Ayuda	Abre la ayuda de iSQL*Plus en una ventana diferente.
Historial	Muestra una nueva pantalla desde la que se pueden ver y cargar las sentencias y archivos de comandos que se han ejecutado a lo largo de la sesión.

Tabla 3.5. Elementos del espacio de trabajo iSQL\*Plus.



#### Caso práctico

- 2 Desde el área de trabajo escribimos la siguiente orden: `SELECT * FROM DEPT`, y pulsamos el botón *Ejecutar*. En la parte inferior de la pantalla veremos el resultado de la ejecución, véase la Figura 3.9. Las órdenes SQL se pueden escribir en mayúscula o minúscula, colocando al final el punto y coma o sin punto y coma.

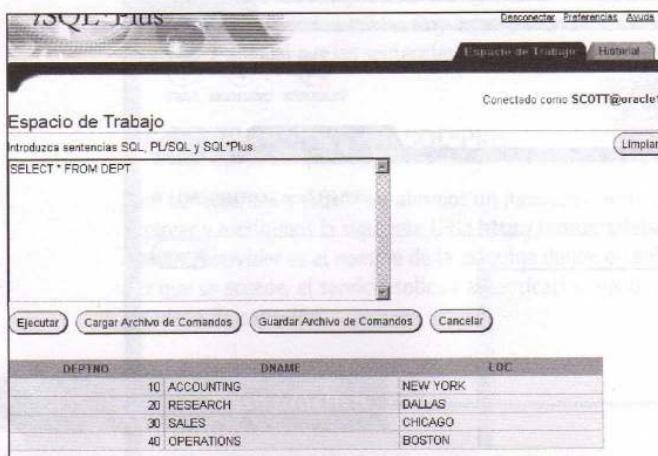


Figura 3.9. Ejecución de una sentencia desde iSQL\*Plus.

- Utiliza el botón *Guardar Archivo de Comandos* para guardar lo escrito en un archivo, por ejemplo en *myscript.sql*.
- Limpia el área de trabajo haciendo clic en el botón *Limpiar*.
- Haz clic en el botón *Cargar Archivo de Comandos*, y después clic en el botón *Examinar* para localizar el archivo guardado anteriormente y cargarlo. Hacer clic en el botón *Cargar*. Se visualiza de nuevo el área de trabajo con el archivo de comandos.
- Haz clic en el botón *Ejecutar* para ejecutar la sentencia SQL. Se visualizan una serie de mensajes de error (ya que no son sentencias SQL ni PL/SQL) y, a continuación, el resultado de la sentencia SELECT.
- Limpia el área de trabajo y el área de salida haciendo clic en el botón *Limpiar* que está en la parte inferior derecha de la pantalla.
- Haz clic en *Historial*, se visualiza una nueva pantalla. Hacer clic en la sentencia SELECT, vemos que se carga en el área de trabajo. Ejecutar la sentencia de nuevo.
- Cierra la sesión haciendo clic en el botón *Desconectar*.



#### Actividades propuestas

- 2 Prueba la sentencias: `SELECT * FROM EMP` y `SELECT * FROM DEPT` y repasa todos los conceptos vistos en el caso práctico anterior.

### 3. Introducción a SQL

#### 3.7 Consulta de los datos



## 3.7 Consulta de los datos

Para recuperar información o, lo que es lo mismo, para realizar consultas a la base de datos, utilizaremos una única sentencia SELECT. El usuario emplea esta sentencia con el nivel de complejidad apropiado para él: especifica qué es lo que quiere obtener, no dónde

ni cómo. De la consulta se puede obtener: cualquier unidad de datos, todos los datos, cualquier subconjunto de datos, cualquier conjunto de subconjuntos de datos.

### A. Sentencia SELECT

El formato de la sentencia SELECT es el siguiente:

```
SELECT [ALL|DISTINCT]
[expresión_colum1, expresión_colum2, ..., expresión_colum | * ]
FROM [nombre_tabla1, nombre_tabla2, ..., nombre_tablaN]
[WHERE condición]
[ORDER BY expresión_colum [DESC|ASC] [,expresión_colum
[DESC|ASC]]...];
```

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

### B. Cláusulas de SELECT

La única cláusula de la sentencia SELECT que es obligatoria es la cláusula FROM, el resto son opcionales.

- **FROM:** FROM [nombre\_tabla1, nombre\_tabla2, ..., nombre\_tablaN]

Especifica la tabla o lista de tablas de las que se recuperarán los datos. Por ejemplo, consultamos los nombres de alumnos y su nota en la tabla ALUMNOS:

```
SQL> SELECT NOM_ALUM, NOTA FROM ALUMNOS;
```

Si el usuario que hace la consulta no es el propietario de la tabla, es necesario especificar el nombre de usuario delante de ella: *nombre\_usuario.nombre\_tabla*. Por ejemplo, si el usuario propietario de la tabla se llama PROFESOR emplearemos:

```
SQL> SELECT NOM_ALUM, NOTA FROM PROFESOR.ALUMNOS;
```

Es posible asociar un nuevo nombre a las tablas mediante *alias*. Por ejemplo, si la tabla ALUMNOS le damos el nombre de A, las columnas de la tabla irán acompañadas de A.

```
SQL> SELECT A.NOM_ALUM, A.NOTAS FROM ALUMNOS A;
```

No importa si se usan mayúsculas o minúsculas.



### 3. Introducción a SQL

#### 3.7 Consulta de los datos



=, >, <, >=, <=, !=, <>
IN, NOT IN, BETWEEN, NOT BETWEEN
LIKE

Tabla 3.6. Operadores de comparación.

- **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 Tabla 3.6.

Se pueden construir condiciones múltiples usando los operadores lógicos booleanos estándares: AND, OR y NOT. Está permitido emplear paréntesis para forzar el orden de evaluación. Veamos unos ejemplos de condiciones en la cláusula WHERE:

```
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|ASC] [,expre\_columna [DESC|ASC]] ...]

Esta cláusula especifica el criterio de clasificación del resultado de la consulta. ASC especifica una ordenación ascendente, y DESC descendente.

Puede contener expresiones con valores de columnas, por ejemplo:

```
SQL> SELECT * FROM ALUMNOS ORDER BY NOTA/5;
```

Asimismo, es posible anidar los criterios. El situado más a la izquierda será el principal. Por ejemplo: SQL> SELECT \* FROM ALUMNOS ORDER BY NOM\_ALUM, CURSO DESC; (ordena por NOM\_ALUM ascendente y por CURSO descendente).

También se puede indicar mediante un número, que indica la posición de la columna a la derecha de SELECT, el criterio de clasificación. Por ejemplo: SQL>SELECT DEPT\_NO, DNOMBRE, LOC FROM DEPART ORDER BY 2; ordena la salida por la segunda columna que es DNOMBRE.

- **ALL:** Con la cláusula ALL recuperamos 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 (columna DEPT\_NO) de la tabla EMPLE. El resultado lo podemos observar en la Tabla 3.7: SQL> SELECT DEPT\_NO FROM EMPLE;

Aparecen todas las filas de la tabla EMPLE donde la columna DEPT\_NO no sea nula; también aparecen números de departamento repetidos. Con DISTINCT se eliminan las filas repetidas, como podemos observar en la Tabla 3.8: SQL> SELECT DISTINCT DEPT\_NO FROM EMPLE;

DEPT_NO
20
30
30
20
30
30
10
20
10
30
20
30
20
10

14 filas seleccionadas.

Tabla 3.7. Salida sin DISTINCT.

DEPT_NO
10
20
30

Tabla 3.8. Salida con DISTINCT.

### 3. Introducción a SQL

#### 3.7 Consulta de los datos



## C. Selección de columnas

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

```
SELECT [ALL|DISTINCT]
       [expresion_colum1, expresion_colum2, ..., expresion_colum | * ]
    FROM [nombre_tabla1, nombre_tabla2, ..., nombre_tablaN]
```

A modo de ejemplo, vamos a realizar consultas sobre las tablas EMPLE y DEPART.

Veamos antes la descripción de estas tablas con la orden DESCRIBE, que nos da un resumen de la tabla y de sus columnas:

```
SQL> DESC EMPLE
      Nombre          ¿Nulo?  Tipo
-----  -----
EMP_NO           NOT NULL  NUMBER(4)
APELIDO          VARCHAR2(10)
OFICIO           VARCHAR2(10)
DIR              NUMBER(4)
FECHA_ALT        DATE
SALARIO          NUMBER(7)
COMISION         NUMBER(7)
DEPT_NO          NOT NULL  NUMBER(2)
```

```
SQL> DESC DEPART
      Nombre          ¿Nulo?  Tipo
-----  -----
DEPT_NO          NOT NULL  NUMBER(2)
DNOMBRE          VARCHAR2(14)
LOC               VARCHAR2(14)
```

Los scripts de creación de las tablas para esta unidad se encuentran en el CD-ROM.

NOT NULL	Significa que la columna no puede tener valores nulos.
VARCHAR2	Tipo de datos cadena de longitud variable. Puede contener cualquier carácter. La longitud máxima de la cadena se define entre paréntesis.
DATE	Tipo de datos fecha.
NUMBER	Tipo de datos numérico. El número de dígitos se expresa entre paréntesis.

Tabla 3.9. Descripción de los tipos.

En cuanto a la selección de todas las columnas de la tabla EMPLE, podemos recuperar las filas de dos formas:

1. Ponemos los nombres de todas las columnas, uno tras otro, separados por comas: SQL>  
SELECT EMP\_NO, APELLIDO, OFICIO, DIR, FECHA\_ALT, SALARIO,  
COMISION, DEPT\_NO FROM EMPLE;
2. Ponemos \*, que representa a todas las columnas de la tabla: SQL> SELECT \* FROM  
EMPLE;



### 3. Introducción a SQL

#### 3.7 Consulta de los datos

Para seleccionar determinadas columnas, sólo se ponen los nombres de las columnas que nos interesan. Por ejemplo, para escoger el nombre (DNOMBRE) y el número de departamento (DEPT\_NO) de la tabla DEPART emplearemos:

```
SQL> SELECT DNOMBRE, DEPT_NO FROM DEPART;
```

	DNOMBRE	DEPT_NO
	CONTABILIDAD	10
	INVESTIGACION	20
	VENTAS	30
	PRODUCCION	40

#### D. Selección por fila

Para seleccionar determinadas filas necesitamos emplear la cláusula WHERE en la sentencia SELECT, [WHERE condición].

#### Caso práctico

- 3 Seleccionamos de la tabla EMPLE a todos los empleados del departamento 20 (DEPT\_NO =20). Además, la consulta debe aparecer ordenada por la columna APELLIDO. Los campos que hay que consultar son: número de empleado, apellido, oficio y número de departamento:

```
SQL> SELECT EMP_NO, APELLIDO, OFICIO, DEPT_NO FROM EMPLE WHERE DEPT_NO = 20 ORDER BY APELLIDO;
```

EMP_NO	APELLIDO	OFICIO	DEPT_NO
7876	ALONSO	EMPLEADO	20
7902	FERNANDEZ	ANALISTA	20
7788	GIL	ANALISTA	20
7566	JIMENEZ	DIRECTOR	20
7369	SANCHEZ	EMPLEADO	20

Consulta de los empleados cuyo oficio sea 'ANALISTA' ordenado por número de empleado (columna EMP\_NO):

```
SQL> SELECT * FROM EMPLE WHERE OFICIO = 'ANALISTA' ORDER BY EMP_NO;
```

Seleccionar de la tabla EMPLE aquellas filas del departamento 10 y cuyo oficio sea 'ANALISTA'. La consulta se ha de ordenar de modo descendente por APELLIDO y también de manera descendente por número de empleado (columna EMP\_NO):

```
SQL> SELECT * FROM EMPLE WHERE DEPT_NO=10 AND OFICIO = 'ANALISTA' ORDER BY APELLIDO DESC, EMP_NO DESC;
```

### 3. Introducción a SQL

#### 3.7 Consulta de los datos



#### Actividades propuestas



- 3 A partir de la tabla ALUMO405 (véase Tabla 3.10) que contiene los datos de alumnos matriculados en el curso 2004/2005 para un centro de enseñanza:

Columna	Tipo de dato	Descripción
DNI	VARCHAR2(10)	DNI del alumno
NOMBRE	VARCHAR2(15)	Nombre del alumno
APELLIDOS	VARCHAR2(20)	Apellidos del alumno
FECHA_NAC	DATE	Fecha de nacimiento
DIRECCION	VARCHAR2(20)	Dirección del alumno
POBLACION	VARCHAR2(20)	Población del alumno
PROVINCIA	VARCHAR2(20)	Provincia del alumno
CURSO	NUMBER(1)	Curso del alumno (1, 2, 3, 4)
NIVEL	VARCHAR2(3)	Nivel (ESO, BAC, DAI, ASI, ADM, COM)
CLASE	CHAR(2)	Aula en la que está el alumno
FALTAS1	NUMBER(2)	Faltas primer trimestre
FALTAS2	NUMBER(2)	Faltas segundo trimestre
FALTAS3	NUMBER(2)	Faltas tercer trimestre

Tabla 3.10. Tabla ALUMO405.

- Obtén todos los datos de los alumnos.
- Obtén los siguientes datos de alumnos: DNI, NOMBRE, APELLIDOS, CURSO, NIVEL y CLASE.
- Obtén todos los datos de alumnos cuya población sea 'GUADALAJARA'.
- Obtén el NOMBRE y APELLIDOS de todos los alumnos cuya población sea 'GUADALAJARA'.
- Consulta el DNI, NOMBRE, APELLIDOS, CURSO, NIVEL y CLASE de todos los alumnos ordenado por APELLIDOS y NOMBRE ascendenteamente.

#### E. Crear y utilizar alias de columnas

Cuando se consulta la base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si el nombre resulta demasiado largo, corto o críptico, existe la posibilidad de cambiarlo con la misma sentencia SQL de consulta creando un ALIAS. El ALIAS se pone entre comillas dobles, a la derecha de la columna.



### 3. Introducción a SQL

#### 3.8 Operadores aritméticos

##### 4 Utilizamos alias para las columnas DEPT\_NO y DNOMBRE:

```
SQL> SELECT DNOMBRE "Departamento", DEPT_NO "Número Departamento" FROM DEPART;
```

Departamento	Número Departamento
CONTABILIDAD	10
INVESTIGACION	20
VENTAS	30
PRODUCCION	40

## 3.8 Operadores aritméticos

Operador aritmético	Operación
+	Suma
-	Resta
*	Multiplicación
/	División

Tabla 3.11. Operadores aritméticos.

Los **operadores aritméticos** sirven para formar expresiones con constantes, valores de columnas y funciones de valores de columnas. Son los que podemos observar en el Tabla 3.11.

Una forma posible de utilizar los operadores puede ser:

```
SELECT col1*col2, col1-col2 FROM tablal WHERE col1+col2 = 34;
```

##### 5 Disponemos de la tabla NOTAS\_ALUMNOS, que contiene las notas de los alumnos de primer curso de ciclo DAI obtenidas en los tres módulos. La descripción de la tabla es la siguiente:

```
SQL> DESC NOTAS_ALUMNOS
```

Nombre

NOMBRE\_ALUMNO

NOTA1

NOTA2

NOTA3

¿Nulo? [ ]

Tipo

NOT NULL

VARCHAR2 (25)

NUMBER (2)

NUMBER (2)

NUMBER (2)

### 3. Introducción a SQL

#### 3.9 Operadores de comparación y lógicos



(Continuación)

```
SQL> SELECT * FROM NOTAS_ALUMNOS;
```

NOMBRE_ALUMNO	NOTA1	NOTA2	NOTA3
Alcalde García, M. Luisa	5	5	5
Benito Martín, Luis	7	6	8
Casas Martínez, Manuel	7	5	5
Corregidor Sánchez, Ana	6	9	8

Se trata de obtener la nota media de cada alumno. Visualizamos por cada uno de ellos su nombre y su nota media (suma de las tres notas dividida entre tres):

```
SQL> SELECT NOMBRE_ALUMNO "Nombre Alumno", (NOTA1+NOTA2+NOTA3)/3 "Nota Media" FROM NOTAS_ALUMNOS;
```

Nombre Alumno	Nota Media
Alcalde García, M. Luisa	5
Benito Martín, Luis	7
Casas Martínez, Manuel	5,66666667
Corregidor Sánchez, Ana	7,66666667

## 3.9 Operadores de comparación y lógicos

Los operadores de comparación y los operadores lógicos, se resumen en las Tablas 3.12 y 3.13.

Operador	Función	Operador	Función
=	Igual a	AND	Devuelve el valor TRUE cuando las dos condiciones son verdaderas.
>	Mayor que	OR	Devuelve el valor TRUE cuando una de las dos condiciones es verdadera.
>=	Mayor o igual que		
<	Menor que		
<=	Menor o igual que		
!= <>	Distinto de	NOT	Devuelve el valor TRUE si la condición es falsa.

Tabla 3.12. Operadores de comparación.

Tabla 3.13. Operadores lógicos.



### 3. Introducción a SQL

#### 3.10 Operadores de comparación de cadenas de caracteres



#### Caso práctico

- 6 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
```

## 3.10 Operadores de comparación de cadenas de caracteres

Para comparar cadenas de caracteres, hasta ahora hemos utilizado el operador de comparación Igual a (=). Así, a partir de la tabla EMPLE, obtenemos el apellido de los ANALISTAS del departamento 10:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO = 'ANALISTA' AND DEPT_NO=10;
```

Pero este operador no nos sirve si queremos hacer consultas de este tipo:

Obtener los datos de los empleados cuyo apellido empiece por una «P» u «obtener los nombres de alumnos que incluyan la palabra Pérez».

Para especificar este tipo de consultas, en SQL usamos el operador LIKE que permite utilizar los siguientes caracteres especiales en las cadenas de comparación:

% Comodín: representa cualquier cadena de 0 o más caracteres.

'\_ ' Marcador de posición: representa un carácter cualquiera.

En la cláusula WHERE este operador se utilizará de la siguiente manera:

```
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'.
```

### 3. Introducción a SQL

#### 3.10 Operadores de comparación de cadenas de caracteres



#### Caso práctico

##### A continuación se muestran ejemplos de uso de la cláusula LIKE:

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'.

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.

- A partir de la tabla EMPLE, obtén aquellos apellidos que empiecen por una 'J':

```
SQL> SELECT APELLIDO FROM EMPLE WHERE APELLIDO LIKE 'J%';
APELIDO
-----
JIMENEZ
JIMENO
```

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

```
SQL> SELECT APELLIDO FROM EMPLE WHERE APELLIDO LIKE '_R%';
APELIDO
-----
ARROYO
```

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

```
SQL> SELECT APELLIDO FROM EMPLE WHERE APELLIDO LIKE 'A%O%';
APELIDO
-----
ARROYO
ALONSO
```

A continuación, consideramos la tabla LIBRERÍA, cuya descripción es la siguiente:

Nombre	¿Nulo?	Tipo
TEMA	NOT NULL	CHAR(15)
ESTANTE		CHAR(1)
EJEMPLARES		NUMBER(2)

(Continúa)



ESTUDIANTES  
1234567  
100%

### 3. Introducción a SQL

#### 3.11 NULL y NOT NULL

(Continuación)

- Consultamos las filas de la tabla LIBRERIA cuyo tema sea 'LABORES'; usamos el operador '=':

```
SQL> SELECT * FROM LIBRERIA WHERE TEMA='LABORES';
      TEMA      E   EJEMPLARES
-----  -----

```

- Hacemos lo mismo, pero ahora manejando el operador LIKE:

```
SQL> SELECT * FROM LIBRERIA WHERE TEMA LIKE 'LABORES';
```

ninguna fila seleccionada

La consulta no devuelve nada, debido a que la columna TEMA está definida con el tipo CHAR(15). El tipo CHAR rellena blancos a la derecha hasta formar la longitud de 15 caracteres. Para que funcione la consulta tendremos que utilizar el comodín %:

```
SQL> SELECT * FROM LIBRERIA WHERE TEMA LIKE 'LABORES%';
      TEMA      E   EJEMPLARES
-----  -----
      LABORES    B       20

```

Si la columna TEMA fuese de tipo VARCHAR2, no sería necesario usar el comodín % con el operador LIKE.

#### 3.11 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, usamos la expresión: *columna IS NOT NULL*. Cuando comparamos con valores nulos o no nulos no podemos utilizar los operadores de igualdad, mayor o menor.

Por ejemplo, a partir de la tabla EMPLE, consultamos los apellidos de los empleados cuya comisión es nula:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE COMISION IS NULL;
```

Si queremos consultar los apellidos de los empleados cuya comisión no sea nula teclearemos esto:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE COMISION IS NOT NULL;
```



## 3.12 Comprobaciones con conjuntos de valores

Hasta ahora, todas las comprobaciones lógicas que hemos visto comparan una columna o expresión con un valor. Por ejemplo:

```
OFICIO = 'ANALISTA' AND DEPT_NO=10.
```

Permiten también comparar una columna o una expresión con una lista de valores utilizando los operadores **IN** y **BETWEEN**.

### A Operador 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)
```

#### Caso práctico

##### ■ La lista de valores está formada por números:

- Consulta los apellidos de la tabla EMPLE cuyo número de departamento sea 10 o 30:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE DEPT_NO IN(10,30);
```

- Consulta 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);
```

##### La lista de valores está formada por cadenas:

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

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO IN ('VENDEDOR', 'ANALISTA', 'EMPLEADO');
```

- Consulta 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');
```



### 3. Introducción a SQL

#### 3.12 Comprobaciones con conjuntos de valores

A continuación:

## B. Operador BETWEEN

El operador **BETWEEN** comprueba si un valor está comprendido o no (NOT) dentro de un rango de valores, desde un valor inicial a un valor final. Su formato es:

<expresión> [NOT] BETWEEN valor\_inicial AND valor\_final



### Caso práctico

- 9 A partir de la tabla EMPLE, obtén el apellido y el salario de los empleados cuyo salario esté comprendido entre 1500 y 2000:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE WHERE SALARIO BETWEEN 1500 AND 2000;
```

APELLIDO	SALARIO
ARROYO	1500
SALA	1625
MARTIN	1600
MUÑOZ	1690

En esta sentencia se visualizan los empleados cuyo salario es 1500, 2000 o cualquier valor comprendido entre ambos.

- A partir de la tabla EMPLE, obtener el apellido y el salario de los empleados cuyo SALARIO no esté comprendido entre 1500 y 2000:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE WHERE SALARIO NOT BETWEEN 1500 AND 2000;
```

APELLIDO	SALARIO
SANCHEZ	1040
JIMENEZ	2900
NEGRO	3005
CEREZO	2885
GIL	3000
REY	4100
TOVAR	1350
ALONSO	1430
JIMENO	1335
FERNANDEZ	2900

10 filas seleccionadas.

Con esta sentencia tenemos los empleados cuyo SALARIO es menor que 1500 o mayor que 2000.

### 3. Introducción a SQL

#### 3.13 Combinación de operadores AND y OR



## 3.13 Combinación de operadores AND y OR

Los operadores AND y OR se pueden combinar de forma ilimitada, pero hay que tener cuidado al usarlos y utilizar paréntesis para agrupar aquellas expresiones que se deseen evaluar juntas; si no nos servimos de los paréntesis, es posible que los resultados no sean los deseados.

El orden de prioridad de los operadores lógicos es el siguiente: el primero NOT, después AND y, por último, OR. Para alterar el orden se utilizan paréntesis.

### Caso práctico

A partir de la tabla EMPLE, obtén el APELLIDO, SALARIO y DEPT\_NO de los empleados cuyo salario sea mayor de 2000 en los departamentos 10 o 20:

```
SQL> SELECT APELLIDO, SALARIO, DEPT_NO FROM EMPLE WHERE  
      SALARIO >2000 AND (DEPT_NO=10 OR DEPT_NO=20);
```

APELLIDO	SALARIO	DEPT_NO
JIMENEZ	2900	20
CEREZO	2885	10
GIL	3000	20
REY	4100	10
FERNANDEZ	2900	20

Sin utilizar paréntesis, la consulta anterior dará el siguiente resultado:

```
SQL> SELECT APELLIDO, SALARIO, DEPT_NO FROM EMPLE WHERE SALARIO >2000 AND  
      DEPT_NO=10 OR DEPT_NO=20;
```

APELLIDO	SALARIO	DEPT_NO
SANCHEZ	1040	20
JIMENEZ	2900	20
CEREZO	2885	10
GIL	3000	20
REY	4100	10
ALONSO	1430	20
FERNANDEZ	2900	20

7 filas seleccionadas.

En el ejemplo, al no usar paréntesis se obtienen los empleados cuyo SALARIO sea > 2000 y el DEPT\_NO = 10; o bien, aquellos cuyo DEPT\_NO sea 20.

- La consulta inicial también se puede hacer recurriendo al operador IN:

```
SQL> SELECT APELLIDO, SALARIO, DEPT_NO FROM EMPLE WHERE SALARIO >2000 AND DEPT_NO  
      IN(10,20);
```



## 3.14 Subconsultas

Algunas veces necesitamos obtener los datos de una tabla en función de los datos que se obtienen de otra tabla. Por ejemplo, si queremos saber cuántos empleados tienen el mismo oficio que 'PEPE', podemos hacer una consulta que devuelva el nombre del oficio de 'PEPE' y otra que devuelva la cantidad de empleados con ese mismo oficio.

A partir de la tabla EMPLE, podemos obtener el número de empleados con el mismo oficio que 'PEPE'.

A veces, para realizar alguna operación de consulta, necesitamos los datos devueltos por otra consulta; así, si queremos obtener los datos de los empleados que tengan el mismo oficio que 'PEPE', hemos de averiguar el oficio de 'PEPE' (primera consulta). Una vez conocido este dato, podemos averiguar qué empleados tienen el mismo oficio que 'PEPE' (segunda consulta). Este problema se puede resolver usando una subconsulta, que no es más que una sentencia SELECT dentro de otra SELECT. Las **subconsultas** son aquellas sentencias SELECT que forman parte de una cláusula WHERE de una sentencia SELECT anterior. Una subconsulta consistirá en incluir una declaración SELECT como parte de una cláusula WHERE. El formato de una subconsulta es similar a éste:

```
SELECT ...  
FROM ...  
WHERE columna operador_comparativo (SELECT ...  
FROM ...  
WHERE ...);
```

La subconsulta (el comando SELECT entre paréntesis) se ejecutará primero y, posteriormente, el valor extraído es «introducido» en la consulta principal.

### Caso práctico

- 11 Con la tabla EMPLE, obtén el APELLIDO de los empleados con el mismo OFICIO que 'GIL'. Para ello, descomponemos el enunciado en dos consultas. Primero averiguamos el OFICIO de 'GIL':

```
SQL> SELECT OFICIO FROM EMPLE WHERE APELLIDO = 'GIL';
```

```
OFICIO  
-----  
ANALISTA
```

- A continuación, visualizamos el APELLIDO de los empleados con el mismo OFICIO que 'GIL':

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO='ANALISTA';
```

```
APELLIDO  
-----  
GIL  
FERNANDEZ
```

- Es posible resumir estas dos consultas con una sentencia SELECT que forma parte de la cláusula WHERE:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO = (SELECT OFICIO FROM EMPLE WHERE APELLIDO = 'GIL');
```

```
APELLIDO  
-----  
GIL  
FERNANDEZ
```

### 3. Introducción a SQL

#### 3.14 Subconsultas



#### Actividades propuestas

- 4 Muestra los datos (apellido, oficio, salario y fecha de alta) de aquellos empleados que desempeñen el mismo oficio que "JIMENEZ" o que tengan un salario mayor o igual que "FERNANDEZ".

#### A. Condiciones de búsquedas en subconsultas

Las subconsultas normalmente aparecen como parte de la condición de búsqueda de una cláusula WHERE o HAVING. Las condiciones de búsqueda que nos podemos encontrar en una subconsulta son:

- **Test de comparación en subconsultas ( $>$ ,  $<$ ,  $\neq$ ,  $\leq$ ,  $\geq$ ,  $=$ ).** Compara el valor de una expresión con un valor único producido por una subconsulta. Ejemplo: Obtener aquellos apellidos de empleados cuyo oficio es igual al oficio de 'GIL':

```
SELECT APELLIDO FROM EMPLE WHERE OFICIO = (SELECT OFICIO  
FROM EMPLE WHERE APELLIDO = 'GIL');
```

- **Test de pertenencia a un conjunto devuelto por una subconsulta (IN).** Comprueba si el valor de una expresión coincide con uno del conjunto de valores producido por una subconsulta. Ejemplo: Obtener aquellos apellidos de empleados cuyo oficio sea alguno de los oficios que hay en el departamento 20:

```
SELECT APELLIDO FROM EMPLE WHERE OFICIO IN (SELECT OFICIO  
FROM EMPLE WHERE DEPT_NO=20);
```

- **Test de existencia (EXISTS , NOT EXISTS).** Examina si una subconsulta produce alguna fila de resultados. El test es TRUE si devuelve filas, si no es FALSE. Ejemplo: Listar los departamentos que tengan empleados:

```
SELECT DNOMBRE, DEPT_NO FROM DEPART WHERE EXISTS (SELECT  
* FROM EMPLE WHERE EMPLE.DEPT_NO= DEPART.DEPT_NO);
```

Para calcular los que no tengan empleados se usa el operador NOT EXISTS.

- **Test de comparación cuantificada (ANY y ALL).** Se usan en conjunción con los operadores de comparación ( $>$ ,  $<$ ,  $\neq$ ,  $\leq$ ,  $\geq$ ,  $=$ ).

ANY compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta, si alguna de las comparaciones individuales da como resultado TRUE, ANY devuelve TRUE, si la subconsulta no devuelve nada devolverá FALSE. Ejemplo: obtener los datos de los empleados cuyo salario sea igual a algún salario de los empleados del departamento 30:

```
SELECT * FROM EMPLE WHERE SALARIO = ANY (SELECT SALARIO  
FROM EMPLE WHERE DEPT_NO=30);
```





### 3. Introducción a SQL

#### 3.14 Subconsultas

**ALL** compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta, si todas las comparaciones individuales da como resultado TRUE, ALL devuelve TRUE, en caso contrario devuelve FALSE. Ejemplo: Obtener los datos de los empleados cuyo salario sea menor a cualquier salario de los empleados del departamento 30:

```
SELECT * FROM EMPLE WHERE SALARIO < ALL (SELECT SALARIO  
FROM EMPLE WHERE DEPT_NO=30);
```

#### B. Subconsultas que generan valores simples

Se trata de aquellas subconsultas que devuelven una fila o un valor simple; en éstas se utilizan los operadores de comparación ( $>$ ,  $<$ ,  $\neq$ ,  $\leq$ ,  $\geq$ ,  $=$ ). Si la subconsulta obtiene más de una fila, se produce un mensaje de error. Por ejemplo, con la siguiente consulta se pretende obtener los apellidos de los empleados cuyo oficio coincide con algún oficio del departamento 20:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO =  
2 (SELECT OFICIO FROM EMPLE where DEPT_NO=20);  
(SELECT OFICIO FROM EMPLE where DEPT_NO=20)  
*  
ERROR en línea 2:  
ORA-01427: la subconsulta de una sola fila devuelve más  
de una fila
```

En el departamento 20 hay varios oficios por tanto la subconsulta devuelve varias filas. Por ello, cuando una subconsulta devuelve más de una fila no se puede recurrir a los operadores de comparación.

#### C. Subconsultas que generan listas de valores

Son aquellas subconsultas que devuelven más de una fila o más de un valor. Cuando una subconsulta devuelva más de un valor, usaremos el operador IN en la cláusula WHERE. La solución al ejemplo anterior sería la siguiente:

```
SQL> SELECT APELLIDO FROM EMPLE WHERE OFICIO IN  
(SELECT OFICIO FROM EMPLE where DEPT_NO=20);
```

El tipo de dato de la expresión situada después de WHERE debe coincidir con el tipo de dato devuelto por la subconsulta.

### 3. Introducción a SQL

#### 3.14 Subconsultas

rs6tttuvw  
20114556  
10025102



#### Caso práctico

- 12 Usamos las tablas **EMPLE** y **DEPART**. Queremos consultar los datos de los empleados que trabajen en 'MADRID' o 'BARCELONA'. La localidad de los departamentos se obtiene de la tabla **DEPART**. Hemos de relacionar las tablas **EMPLE** y **DEPART** por el número de departamento. Para ello, descomponemos esa consulta en varias: primero tenemos que localizar los números de departamento que estén en la localidad de 'MADRID' o 'BARCELONA':

```
SELECT DEPT_NO FROM DEPART WHERE LOC IN ('MADRID', 'BARCELONA');
```

DEPT_NO
-----
20
30

A continuación, seleccionamos los datos de los empleados que estén en esos departamentos:

```
SELECT EMP_NO, APELLIDO, OFICIO, DIR, FECHA_ALT, SALARIO, COMISION, DEPT_NO FROM  
EMPLE WHERE DEPT_NO IN(20, 30);
```

Por último, reunimos estas dos sentencias SELECT utilizando una subconsulta:

```
SELECT EMP_NO, APELLIDO, OFICIO, DIR, FECHA_ALT, SALARIO, COMISION, DEPT_NO  
FROM EMPLE WHERE DEPT_NO IN (SELECT DEPT_NO FROM DEPART WHERE LOC IN  
( 'MADRID', 'BARCELONA' ));
```

- Consulta los apellidos y oficios de todos los empleados del departamento 20 cuyo trabajo sea idéntico al de cualquiera de los empleados del departamento 'VENTAS':

```
SQL> SELECT APELLIDO, OFICIO FROM EMPLE WHERE DEPT_NO=20 AND  
2 OFICIO IN (SELECT OFICIO FROM EMPLE WHERE DEPT_NO =  
3 (SELECT DEPT_NO FROM DEPART WHERE DNOMBRE='VENTAS'));
```

APELLIDO	OFICIO
-----	-----
SANCHEZ	EMPLEADO
JIMENEZ	DIRECTOR
ALONSO	EMPLEADO

- Obtén el apellido de los empleados con el mismo oficio y salario que 'GIL'. En esta consulta se introduce una variante: hasta ahora las subconsultas nos devolvían una columna, aunque pueden devolver más de una. En este caso, la subconsulta devuelve dos columnas, el oficio y el salario:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE WHERE (OFICIO, SALARIO) =  
2 (SELECT OFICIO, SALARIO FROM EMPLE WHERE APELLIDO ='GIL');
```

APELLIDO	SALARIO
-----	-----
GIL	3000
FERNANDEZ	3000

(Continúa)



(Continuación)

Si se detallan varios campos en la cláusula WHERE, deben encerrarse entre paréntesis, y han de coincidir en número y tipo de datos con los de la sentencia SELECT de la consulta interior. Cada columna de la cláusula WHERE se referirá a la correspondiente columna de la subconsulta. También podríamos haber puesto esto:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE  
2 WHERE OFICIO= (SELECT OFICIO FROM EMPLE WHERE APELLIDO = 'GIL')  
3 AND SALARIO= (SELECT SALARIO FROM EMPLE WHERE APELLIDO = 'GIL');
```



#### Actividades propuestas

5 Presenta los apellidos y oficios de los empleados que tienen el mismo trabajo que "JIMENEZ".

Muestra en pantalla el APELLIDO, OFICIO y SALARIO de los empleados del departamento de "FERNANDEZ" que tengan su mismo salario.

#### D. Subconsultas correlacionadas

Una **subconsulta correlacionada** es aquella que hace referencia a una columna o varias de la consulta más externa. A veces la subconsulta hace uso de columnas que tienen el mismo nombre que las columnas de las tablas usadas en la consulta más externa. Si la subconsulta necesita acceder a esas columnas deberá definirse un alias en la tabla más externa. Por ejemplo, deseamos obtener los datos de los empleados cuyo salario sea el máximo salario de su departamento:

```
SELECT * FROM EMPLE E WHERE SALARIO = (SELECT  
MAX(SALARIO) FROM EMPLE WHERE DEPT_NO = E.DEPT_NO);
```

La subconsulta devuelve para cada fila que se recupere de la consulta más externa el máximo salario del departamento que se está recuperando en la consulta externa; para referenciar a dicho departamento se necesita el alias **E** usado en la tabla de la consulta externa.

#### 3.15 Combinación de tablas

Hasta ahora, en las consultas que hemos realizado sólo se ha utilizado una tabla, indicada a la derecha de la palabra FROM; pero hay veces que una consulta necesita columnas de varias tablas. En este caso, las tablas se expresarán a la derecha de la palabra FROM.



Sintaxis general:

```
SELECT    columnas de las tablas citadas en la cláusula
          "from"
  FROM     tabla1, tabla2, ...
 WHERE    tabla1.columna = tabla2.columna;
```

Cuando combinamos varias tablas, hemos de tener en cuenta una serie de reglas:

- Es posible unir tantas tablas como deseemos.
- En la cláusula SELECT se pueden citar columnas de todas las tablas.
- Si hay columnas con el mismo nombre en las distintas tablas de la cláusula FROM, se deben identificar, especificando NombreTabla.NombreColumna.
- Si el nombre de una columna existe sólo en una tabla, no será necesario especificarla como NombreTabla.NombreColumna. Sin embargo, hacerlo mejoraría la legibilidad de la sentencia SELECT.
- El criterio que se siga para combinar las tablas ha de especificarse en la cláusula WHERE. Si se omite esta cláusula, que especifica la condición de combinación, el resultado será un PRODUCTO CARTESIANO, que emparejará todas las filas de una tabla con cada fila de la otra.

**Caso práctico**

- ③ A partir de las tablas EMPLE y DEPART obtenemos los siguientes datos de los empleados: APELLIDO, OFICIO, número de empleado (EMP\_NO), nombre de departamento (DNOMBRE) y localidad (LOC). Estas tablas tienen en común el campo DEPT\_NO, por el que se combinan las tablas:

```
SQL> SELECT APELLIDO, OFICIO, EMP_NO, DNOMBRE, LOC FROM EMPLE, DEPART
  2  WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO;
```

APELLIDO	OFICIO	EMP_NO	DNOMBRE	LOC
SANCHEZ	EMPLEADO	7369	INVESTIGACION	MADRID
ARROYO	VENDEDOR	7499	VENTAS	BARCELONA
SALA	VENDEDOR	7521	VENTAS	BARCELONA
JIMENEZ	DIRECTOR	7566	INVESTIGACION	MADRID
MARTIN	VENDEDOR	7654	VENTAS	BARCELONA
NEGRO	DIRECTOR	7698	VENTAS	BARCELONA
CEREZO	DIRECTOR	7782	CONTABILIDAD	SEVILLA
GIL	ANALISTA	7788	INVESTIGACION	MADRID
REY	PRESIDENTE	7839	CONTABILIDAD	SEVILLA
TOVAR	VENDEDOR	7844	VENTAS	BARCELONA
ALONSO	EMPLEADO	7876	INVESTIGACION	MADRID

(Continúa)



(Continuación)

APELLIDO	OFICIO	EMP_NO	DNOMBRE	LOC
JIMENO	EMPLEADO	7900	VENTAS	BARCELONA
FERNANDEZ	ANALISTA	7902	INVESTIGACION	MADRID
MUÑOZ	EMPLEADO	7934	CONTABILIDAD	SEVILLA

14 filas seleccionadas.

Todos los empleados con un número de departamento 10 serán emparejados con los datos del departamento 10; los empleados con un número de departamento 20 se emparejarán con los datos del departamento 20 y, así, sucesivamente. Si se omite la cláusula WHERE, resultará un PRODUCTO CARTESIANO donde se emparejaría cada fila de una tabla con todas las filas de la otra tabla. En este caso: 14 filas de la tabla EMPLE por 4 filas de la tabla DEPART = 56 filas.

- Veamos un ejemplo en el que se combinan las siguientes 3 tablas.

**ALUMNOS:** Contiene los datos de los alumnos. La descripción es ésta:

SQL> DESC ALUMNOS	Nombre	¿Nulo?	Tipo
	DNI	NOT NULL	VARCHAR2 (10)
	APENOM		VARCHAR2 (30)
	DIREC		VARCHAR2 (30)
	POBLA		VARCHAR2 (15)
	TELEF		VARCHAR2 (10)

**ASIGNATURAS:** Contiene los nombres de asignaturas con sus códigos correspondientes. La descripción es la siguiente:

SQL> DESC ASIGNATURAS	Nombre	¿Nulo?	Tipo
	COD	NOT NULL	NUMBER (2)
	NOMBRE		VARCHAR2 (25)

**NOTAS:** Contiene las notas de cada alumno en cada asignatura. Se relaciona con la tabla ALUMNOS por la columna DNI y con la tabla ASIGNATURAS por la columna COD. La descripción es:

SQL> DESC NOTAS	Nombre	¿Nulo?	Tipo
	DNI	NOT NULL	VARCHAR2 (10)
	COD	NOT NULL	NUMBER (2)
	NOTA		NUMBER (2)

- Realiza una consulta para obtener el nombre de alumno, su asignatura y su nota:

```
SQL> SELECT APENOM, NOMBRE, NOTA FROM ALUMNOS, ASIGNATURAS, NOTAS  
2 WHERE ALUMNOS.DNI=NOTAS.DNI AND NOTAS.COD=ASIGNATURAS.COD;
```

(Continúa)

### 3. Introducción a SQL

#### 3.15 Combinación de tablas



(Continuación)

APENOM	NOMBRE	NOTA
Alcalde García, Elena	Prog. Leng. Estr.	6
Alcalde García, Elena	Sist. Informáticos	5
Alcalde García, Elena	Ánalysis	6
Díaz Fernández, María	FOL	8
Cerrato Vela, Luis	FOL	6
Díaz Fernández, María	RET	7
Cerrato Vela, Luis	RET	8
Díaz Fernández, María	Entornos Gráficos	8
Cerrato Vela, Luis	Entornos Gráficos	4
Díaz Fernández, María	Aplic. Entornos 4 <sup>a</sup> Gen	9
Cerrato Vela, Luis	Aplic. Entornos 4 <sup>a</sup> Gen	5

11 filas seleccionadas.

- Obtén los nombres de alumnos matriculados en 'FOL':

```
SQL> SELECT APENOM FROM ALUMNOS, ASIGNATURAS, NOTAS WHERE
2  ALUMNOS.DNI=NOTAS.DNI AND NOTAS.COD=ASIGNATURAS.COD
3  AND NOMBRE='FOL';
```

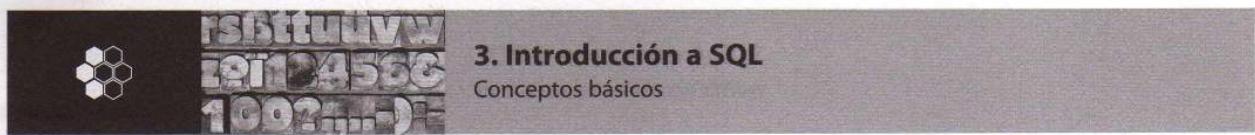
APENOM
Cerrato Vela, Luis
Díaz Fernández, María

#### Actividades propuestas



● Visualiza los nombres de alumnos que tengan una nota entre 7 y 8 e la asignatura de "FOL".

Visualiza los nombres de asignaturas que no tengan suspensos.



## Conceptos básicos



Las Figuras 3.10 y 3.11 muestran un resumen de la sentencia SELECT vista hasta el momento.

**CONSULTAS SIMPLES CON SELECT**

**SELECCIÓN DE COLUMNAS**

```
SELECT * FROM EMPLE;          SELECT APELLIDO, SALARIO FROM EMPLE;
Selección: σ(EMPLE)           Proyección: πAPELLIDO, SALARIO(EMPLE)
```

**SELECCIÓN DE FILAS: WHERE**

```
SELECT * FROM EMPLE WHERE DEPT_NO=20 OR SALARIO >1000;
Selección: σDEPT_NO=20 OR SALARIO>1000(EMPLE)
```

**Condiciones básicas de búsqueda:**

```
SELECT APELLIDO FROM EMPLE WHERE SALARIO BETWEEN 1000 AND 2000;
SELECT APELLIDO FROM EMPLE WHERE DEPT_NO IN(10,30);
SELECT APELLIDO FROM EMPLE WHERE APELLIDO LIKE '_R%';
SELECT APELLIDO FROM EMPLE WHERE COMISION IS NULL;
```

**Condiciones de búsqueda complejas: OR, AND, NOT**

```
Filas duplicadas- DISTINCT: SELECT DISTINCT DEPT_NO FROM EMPLE;
```

Figura 3.10. Consultas sencillas con SELECT.

**Ordenación de los resultados de una consulta: ORDER BY**

Ascendente: **ASC**, opción por defecto. Descendente: **DESC**.

```
SELECT APELLIDO, SALARIO FROM EMPLE ORDER BY DEPT_NO DESC, APELLIDO;
SELECT APELLIDO, SALARIO+SALARIO*0.2 FROM EMPLE ORDER BY 2 DESC;
```

**CONSULTAS MULTITABLA**

**DEPART:** (DEPT\_NO, DNAME, LOC)  
**EMPLE:** (EMP\_NO, APELLIDO, SALARIO, COMISION, DEPT\_NO, DIR)

**PRODUCTO CARTESIANO**

```
SELECT * FROM DEPART, EMPLE;
Producto cartesiano: DEPART X EMPLE
```

**COMBINACIÓN INTERNA**

```
SELECT * FROM DEPART, EMPLE WHERE DEPART.DEPT_NO = EMPLE.DEPT_NO;
```

**Combinación:** DEPART \* EMPLE <sub>(DEPART.DEPT\_NO=EMPLE.DEPT\_NO)</sub>

```
SELECT * FROM DEPART D, EMPLE E WHERE D.DEPT_NO = E.DEPT_NO
```

← →  
Alias de tablas

Figura 3.11. Combinación de tablas.



## Actividades complementarias



### Tablas EMPLE y DEPART

- ⑥ Selecciona el apellido, el oficio y la localidad de los departamentos de aquellos empleados cuyo oficio sea "ANALISTA".
- ⑦ Obtén los datos de los empleados cuyo director (columna DIR de la tabla EMPLE) sea "CEREZO".
- ⑧ Obtén los datos de los empleados del departamento de "VENTAS".
- ⑨ Obtén los datos de los departamentos que NO tengan empleados.
- ⑩ Obtén los datos de los departamentos que tengan empleados.
- ⑪ Obtén el apellido y el salario de los empleados que superen todos los salarios de los empleados del departamento 20.

### Tabla LIBRERIA

- ⑫ Visualiza el tema, estante y ejemplares de las filas de LIBRERIA con ejemplares comprendidos entre 8 y 15.
- ⑬ Visualiza las columnas TEMA, ESTANTE y EJEMPLARES de las filas cuyo ESTANTE no esté comprendido entre la "B" y la "D".

- ⑨ Visualiza con una sola orden SELECT todos los temas de LIBRERIA cuyo número de ejemplares sea inferior a los que hay en "MEDICINA".
- ⑩ Visualiza los temas de LIBRERIA cuyo número de ejemplares no esté entre 15 y 20, ambos incluidos.

### Tablas ALUMNOS, ASIGNATURAS y NOTAS

- ⑪ Visualiza todas las asignaturas que contengan tres letras "o" en su interior y tengan alumnos matriculados de "Madrid".
- ⑫ Visualiza los nombres de alumnos de "Madrid" que tengan alguna asignatura suspendida.
- ⑬ Muestra los nombres de alumnos que tengan la misma nota que tiene "Díaz Fernández, María" en "FOL" en alguna asignatura.
- ⑭ Obtén los datos de las asignaturas que no tengan alumnos.
- ⑮ Obtén el nombre y apellido de los alumnos que tengan nota en la asignatura con código 1.
- ⑯ Obtén el nombre y apellido de los alumnos que no tengan nota en la asignatura con código 1.