INTRODUCCION AL LENGUAJE SQL
(PARA USUARIOS DE ORACLE)

INDICE

Prólogo

En este manual se describen las sentencias básicas del lenguaje SQL necesarias para consultar y actualizar los datos almacenados en Bases de Datos Relacionales. El resto de las tareas que se pueden realizar con el lenguaje, que corresponden a los Administradores de bases de datos no se tratarán en este manual.

Además en los dos últimos apartados, se inicia la descripción de las sentencias que permiten el tratamiento de tablas y vistas, la concesión y retirada de privilegios sobre ellas y el tratamiento de índices.

Se ha tomado como referencia el gestor de base de datos Oracle, por lo tanto, algunas de las sentencias, funciones y cláusulas no se utilizan igual o no existen en las implementaciones de SQL de otros fabricantes. Pero se han incluido al considerar que su funcionalidad es suficientemente significativa.

1 Introducción

El lenguaje SQL permite la comunicación con los Sistemas de Bases de Datos Relacionales(SGBD). La palabra SQL está formada por las iniciales de STRUCTURED QUERY LANGUAGE (Lenguaje de Consulta Estructurado). Esta definición inicial no describe completamente las posibilidades del lenguaje, ya que con él se hacen todas las operaciones necesarias para la gestión del SGBD (actualizaciones de datos, definición de objetos, operaciones de control y consultas de datos).

Una de las características más importantes de SQL es que sus sentencias pueden manejar conjuntos de registros. Esto confiere al lenguaje una gran potencia, consiguiéndose una alta productividad.

1.1 Sistemas de Bases de Datos relacionales

Un Sistema de Gestión de Base de Datos relacional (SGBD) es un conjunto de programas que se encarga de gestionar los datos almacenados en la Base de Datos.

Estos sistemas se caracterizan por:

Los datos se presentan a los usuarios en forma de tablas, formadas por filas y columnas.

Todas las operaciones se realizan con sentencias SQL.

Existe una independencia entre el almacenamiento físico de los datos y la estructura lógica de la base de datos (tablas).

El SGBD coordina todas las peticiones realizadas por los usuarios, asegurando en todo momento la integridad de los datos.

Además el SGBD tiene un conjunto de tablas predefinidas que forman el Diccionario de Datos o Catálogo. En estas tablas se guardan las definiciones de los objetos/estructuras (tablas, vistas, índices, etc.) que componen la base de datos. Estas tablas son mantenidas automáticamente por el SGBD y pueden ser consultadas por los usuarios

1.2 Utilización del lenguaje

Las sentencias SQL pueden escribirse directamente en un terminal interactivo en el cual se reciben los resultados de las operaciones pedidas. También se utilizan sentencias SQL dentro de programas de tercera y cuarta generación (COBOL, FORTRAN, C, PL/SQL, etc.) que tratan datos almacenados en bases de datos relacionales.

1.3 Estandarización del SQL

Los institutos American National Standarts Institure (ANSI) y International Standards Organitation (ISO) han aceptado SQL como el lenguaje de las bases de datos relacionales.

Las últimas normas sobre SQL publicadas llamadas SQL92 o SQL2 son:

ANSI X3.135-1992, “Database Language SQL”.
ISO/IEC 9075:1992, “Database Language SQL”.

Aunque los sistemas de bases de datos de la industria no cumplen con todas estas especificaciones, todos están realizados basándose en ellas, esto hace que las diferencias más importantes aparezcan en opciones y en funciones de las sentencias del lenguaje.

2 Características del Lenguaje SQL.

En este capítulo se introducen los conceptos de sentencia SQL y sus tipos, tablas y tipos de datos.

2.1 Propiedades

El lenguaje SQL se caracteriza por:

- - Lo utilizan todos los usuarios (administradores y usuarios finales).
- - El usuario indica que quiere hacer, no donde ni como hacerlo.
- - Permite realizar cualquier consulta o actualización de datos.
- - Se pueden manejar conjuntos de filas.

2.2 Sentencias SQL

Una sentencia SQL es una cadena de caracteres que se envía al SGBD para su ejecución. Contiene palabras del lenguaje, nombres de tablas y columnas, constantes y signos delimitadores.

Esta cadena de caracteres se compila automáticamente, generándose un procedimiento de ejecución que realiza la operación deseada. Si la sentencia es errónea o incompleta el SGBD genera un mensaje de error.

2.3 Tablas

Todos los datos almacenados en las Bases de Datos relacionales están dispuestos en estructuras llamadas tablas. Cada tabla se identifica con un nombre y está compuesta por un conjunto de columnas.

A su vez cada columna se define con un nombre, un tipo de dato(alfanumérico, numérico, fecha, etc.) y una longitud.

Las columnas se identifican por su nombre, no por su posición dentro de la tabla. Por tanto el orden de las columnas dentro de la tabla no tiene importancia.

La información dentro de la tabla se guarda en filas, conteniendo datos de las columnas de la tabla. No es significativo el orden en el que se almacenan las filas.

En cada intersección de una fila y una columna se almacena un solo valor. Aceptándose el valor NULL (ausencia de valor) que no ocupa espacio en la base de datos.

Los usuarios que crean las tablas son los únicos que pueden trabajar inicialmente con los datos almacenadas en ellas. Por tanto, según los criterios de confidencialidad fijados, es necesario autorizar el acceso al resto de los usuarios.

2.4 Tipos de datos

En la definición de columnas de las tablas se utilizan fundamentalmente tres tipos de datos:

- Columnas numéricas. Contienen números enteros, decimales y en coma flotante.

  La norma ANSI define tipos diferentes para cada uno de estos números: SMALLINT para enteros pequeños, INTEGER para enteros grandes, NUMERIC y DECIMAL para números con parte entera y parte fraccionaria y FLOAT, DOUBLE PRECISION Y REAL para números en formato en coma flotante.

  En el manual se utiliza el tipo **NUMBER(p,s)** de Oracle con el que se pueden implementan los tipos anteriores.

  En la definición **NUMBER(p,s)**, el entero ‘p’ se llama precisión y es el número de total de dígitos (parte entera y parte fraccionaria). El entero ‘s’, es el número de dígitos que puede tener la parte fraccionaria.

- Columnas alfanuméricas. Contienen cadenas de caracteres de longitud fija y de longitud variable.

  La norma ANSI define tipos de datos de longitud fija con CHARACTER, CHAR, NATIONAL CHARACTER, NCHAR, etc. y tipos de longitud variable con CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER VARYING, NCHAR VARYING, etc..

  En el manual se utiliza en la definición de este tipo de columnas el tipo **VARCHAR(n)** de Oracle, que permite cadenas de caracteres de longitud variable con valor máximo igual al entero ‘n’.

  Las columnas alfanuméricas de longitud variable se caracterizan principalmente porque el SGBD requiere un espacio de almacenamiento igual a la longitud real del dato no a la longitud máxima de definición de la columna.

- Columnas de fechas. Estas columnas contienen información de datos temporales (año, mes, día, hora, minuto y segundo).

  Este tipo de datos no está definido en la norma ANSI, por lo cual cada fabricante lo implementa de forma diferente. Se utilizan los tipos: TIME para contener horas, TIMESTAMP para contener instantes en el tiempo y DATE para contener días.

  En el manual se utiliza el tipo **DATE** de Oracle que contiene información de año, mes, día, hora, minuto y segundo.

Existen otros tipos de datos que permiten guardar información de gran tamaño(gigabytes) tanto de tipo binario como de tipo alfanumérico. Estos tipos tampoco se definen en la norma ANSI y por lo cual los fabricantes los implementan de forma diferente.

En el apartado de anexos se indica una equivalencia entre los tipos de datos ANSI y DB2 de IBM y los tipos de Oracle.

2.5 Tipos de sentencias

Las sentencias SQL se dividen en las siguientes categorías o grupos:

Sentencias de manipulación de datos (Data Manipulation Language DML).

Sentencias de definición de datos (Data Definition Language DDL).

Sentencias de control de transacción.

Sentencias de control de sesión.

Sentencias de control de sistema.

Sentencias SQL embebidas en programas.

2.5.1 DML (Data Manipulation Language)

Estas sentencias permiten consultar y actualizar datos.

Por ejemplo:

Consultar datos de una o más tablas y vistas (SELECT).

Añadir filas de datos en tablas (INSERT).

Actualizar valores de columnas en filas de tablas (UPDATE).

Borrar filas de tablas (DELETE).

2.5.2 DDL (Data Definition Language)

Estas sentencia permiten definir, mantener y borrar objetos en la base de datos. Además incluye sentencias que permiten a un usuario autorizar el acceso a determinados objetos de su propiedad a otros usuarios.

Por ejemplo:

Crear, modificar y borrar objetos (CREATE, ALTER, DROP).

Cambiar el nombre de objetos (RENAME).

Borrar todos los datos de tablas sin borrar su estructura (TRUNCATE).

Realizar estadísticas sobre objetos (ANALYZE).

Autorizar y desautorizar privilegios y roles (grupos de privilegios) (GRANT, REVOKE).

Añadir comentarios al diccionario de datos (COMMENT).

2.5.3 Control de transacción.

Estas sentencias manejan los cambios realizados por sentencias DML dentro de una unidad lógica de trabajo de un usuario, llamada transacción.

Por ejemplo:

Hacer que los cambios realizados dentro de una transacción se guarden permanentemente (COMMIT).

Deshacer los cambios realizados dentro de una transacción (ROLLBACK).

Establecer propiedades para una transacción (SET TRANSACTION).

Una transacción comienza con el inicio de una sesión o mediante la última sentencia COMMIT o ROLLBACK ejecutada y termina con el fin de la sesión o con la ejecución de una sentencia COMMIT o ROOLBACK.

2.5.4 Control de sesión.

Estas sentencias permiten modificar las propiedades de la sesión actual del usuario.

Modificar alguna propiedad (ALTER SESSION).

Autorizar y desautorizar roles (grupos de privilegios) (SET ROLE).

2.5.5 Control de sistema.

Sentencia para cambiar las propiedades de la instancia de la base de datos (ALTER SYSTEM).

2.5.6 Sentencias SQL embebidas en programas

Estas sentencias permiten introducir sentencias DDL, DML, y sentencias de control de transacciones en programas realizados con lenguajes de tercera y cuarta generación.

3 Elementos de SQL

En este capítulo se van a ir describiendo las sentencias SQL con las que se realizan: consultas y actualizaciones de datos, tratamiento de tablas y vistas, y asignación y retirada de privilegios sobre tablas y vistas.

3.1 Consultas de datos

3.1.1 Objetivo

En este apartado se va a iniciar la descripción de la sentencia **SELECT**, que se utiliza para realizar las consultas de datos en las Bases de Datos relacionales.

3.1.2 Sentencia SELECT

Con esta sentencia se puede obtener cualquier combinación de filas y columnas de la tabla o las tablas consultadas.

La sentencia SELECT tienen cuatro palabras clave principales:

- SELECT

La palabra SELECT seguida por los nombres de las columnas que se quieren recuperar, separados por comas. **Es obligatoria**.

SELECT columna1[,columna2].

SELECT *

El carácter '*' indica que se quiere recuperar todas la columnas.

- FROM

La palabra FROM seguida de los nombres de las tablas de las que se quieren recuperar datos, separados por comas. **Es obligatoria**.

FROM tabla1 [,tabla2]

- WHERE

La palabra WHERE seguida de las condiciones de selección que deben cumplir cada una de las filas que se quieren recuperar. **Es opcional**.

WHERE condición1 [AND/OR condición2]

- ORDER BY

Las palabras ORDER BY seguida por las columnas por las que se quiere ordenar las filas seleccionadas. **Es opcional**.

ORDER BY columna1 [DESC][,columna2 [DESC]]

(**DESC** – indica ordenación en descendente).

La ordenación por defecto es ascendente y se puede indicar explícitamente con **ASC**.

Ejemplo: Recuperar los empleados que son economistas y ordenar el resultado por el nombre del empleado.

```
     SELECT numempl, nomempl, fchingr
       FROM empleado
      WHERE nomclab = 'ECONOMISTA'
   ORDER BY nomempl;
```

Resultado:

```
  NUMEMPL NOMEMPL              FCHINGR
--------- -------------------- --------
     2015 GARCÍA, PEDRO        21/06/89
     2190 RUIZ, MARÍA          08/12/90
```

Ejemplo: Recuperar todas las filas de la tabla de departamentos.

```
SELECT coddep, nomdept
  FROM departamento;
```

Resultado:

```
  CODDEPT NOMDEPT
--------- --------------
       10 DIRECCIÓN
       20 FINANZAS
       30 COMERCIAL
       40 ORGANIZACIÓN
       50 PRODUCCIÓN
```

Ejemplo: Recuperar todas las columnas y las filas de la tabla de departamentos.

```
SELECT *
  FROM departamento;
```

Ejemplo: Recuperar todas las filas de la tabla de empleados y ordenar el resultado por el código de departamento y por el nombre del empleado pero en orden descendente.

```
SELECT nomempl, coddept
  FROM empleado
 ORDER BY coddept, nomempl DESC;
```

Resultado:

```
NOMEMPL               CODDEPT
-------------------- ---------
MORENO, LUIS               10
GIL, ANA                   10
LLANOS, CRISTINA           20
LASA, FRANCISCO            20
GARCÍA, PEDRO              20
SANZ, PEDRO                30
MARTIN, DIANA              30
RUIZ, MARÍA                40
PEREZ, CARMEN              40
DURAN, ADRIAN              40
TORRES, SANCHO             50
MORENO, VICENTE            50
GONZALEZ, JUAN             50
FERNANDEZ, JOSÉ            50
DIEZ, CLARA                50
```

El mismo resultado se consigue con:

```
SELECT nomempl, coddept
  FROM empleado
 ORDER BY 2, 1 DESC;
```

Los dígitos de la cláusula ORDER BY hacen referencia al número de orden de las columnas indicadas en la cláusula SELECT.

3.1.3 Alias de columnas

Cuando se recupera los datos de una consulta, los nombres de las columnas se utilizan como cabeceras del informe. Si estos no resultan suficientemente significativos, pueden cambiarse en la misma sentencia SELECT, creando un alias de columna.

Ejemplo: Renombrado de las columnas de la tabla de departamentos.

```
SELECT coddept CÓDIGO,
       nomdept NOMBRE
  FROM departamento;
```

Resultado:

```
   CÓDIGO NOMBRE
--------- --------------
       10 DIRECCIÓN
       20 FINANZAS
       30 COMERCIAL
       40 ORGANIZACIÓN
       50 PRODUCCIÓN
```

3.1.4 Alías de tablas

Si en una consulta se recuperan datos de tablas que no pertenecen al usuario, es necesario indicar en la cláusula FROM, el nombre **completo** de la tabla (código del usuario que la creó y nombre de la tabla separados por un punto).

Ejemplo:

```
SELECT ORA001.coddept Código,
       ORA001.nomdept Nombre
  FROM ORA001.departamento;
```

Resultado:

```
   CÓDIGO NOMBRE
--------- --------------
       10 DIRECCIÓN
       20 FINANZAS
       30 COMERCIAL
       40 ORGANIZACIÓN
       50 PRODUCCIÓN
```

Para simplificar esta notación, se pueden definir en la misma sentencia SQL un alias para la tabla.

Ejemplo:

```
SELECT D.coddept Código,
       D.nomdept Nombre
  FROM ORA001.departamento D;
```

Este alias (D) se utiliza para hacer referencia a las columnas de la tabla.

3.1.5 Eliminación de filas repetidas.

Si se utiliza el parámetro **DISTINCT** delante de las columnas seleccionadas en la sentencia SELECT, se eliminan las filas repetidas.

Los valores nulos se consideran iguales.

Ejemplo: Recuperar los diferentes códigos de departamentos existentes en la tabla de empleados.

```
SELECT DISTINCT coddept
  FROM empleado;
```

Resultado:

```
  CODDEPT
---------
       10
       20
       30
       40
       50
```

Ejemplo: Recuperar las distintas comisiones existentes en la tabla de empleados.

```
SELECT DISTINCT comision
  FROM empleado;
```

Resultado: Esta consulta recupera cuatro filas ya que existen empleados sin comisión.

```
 COMISION
---------
       20
       21
       30
```

3.1.6 Operadores lógicos en WHERE

Los criterios de selección de filas de la cláusula WHERE pueden tener la complejidad que se necesite.

Pueden utilizarse condiciones de búsqueda múltiples usando los operadores lógicos: AND, OR y paréntesis para forzar el orden de la evaluación.

3.1.6.1 Comprobaciones con valores simples

Todos estos operadores funcionan con letras o números y con columnas o literales.

3.1.6.1.1 Igual que, mayor que, menor que y distinto de.

Piso	=	3	- Piso igual a 3.
Piso	>	3	- Piso mayor que 3.
Piso	>=	3	- Piso mayor o igual que 3.
Piso	<	3	- Piso menor que 3.
Piso	<>	3	- Piso distinto de 3.
Piso	^=	3	- Piso distinto de 3.
Piso	!=	3	- Piso distinto de 3.

Ejemplo: Recuperar los empleados con código de departamento menor que 45.

```
SELECT numempl, nomempl, fchingr, coddept
  FROM empleado
 WHERE coddept < 45;
```

Resultado:

```
  NUMEMPL NOMEMPL              FCHINGR    CODDEPT
--------- -------------------- -------- ---------
     1100 MORENO, LUIS         17/11/81        10
     1500 PEREZ, CARMEN        09/12/82        40
     2011 DURAN, ADRIAN        20/09/88        40
     2015 GARCÍA, PEDRO        21/06/89        20
     2134 LASA, FRANCISCO      22/03/90        20
     2167 SANZ, PEDRO          12/09/90        30
     2190 RUIZ, MARÍA          08/12/90        40
     2345 MARTIN, DIANA        28/10/95        30
     2565 GIL, ANA             03/02/99        10
```

3.1.6.1.2 LIKE, NOT LIKE

Este operador permite la comparación con patrones.

Utiliza dos caracteres especiales en las cadenas de comparación:

Subrayado (_), representa un posición.
Tanto por ciento (%), representa una serie de espacios o caracteres.

Tema **LIKE** '_M%'	Cualquier tema cuyo segundo carácter sea una M.
Tema **LIKE** '%MA%'	Cualquier tema que contenga MA.
Tema **LIKE** '__M'	Cualquier tema de tres posiciones que termine en M.
Tema **LIKE** 'M_'	Cualquier tema de dos posiciones que empiece por M.
Tema **LIKE** 'M%'	Cualquier tema que empiece por M.

Las mayúsculas y minúsculas son significativas.

Ejemplo: Recuperar los empleados que el nombre empiece por G.

Las constantes alfanuméricas deben ir entre comillas simples.

```
SELECT numempl, nomempl
  FROM empleado
 WHERE nomempl LIKE 'G%';
```

Resultado:

```
  NUMEMPL NOMEMPL
--------- ------------------
     2015 GARCÍA, PEDRO
     2120 GONZALEZ, JUAN
     2565 GIL, ANA
```

3.1.6.1.3 NULL, NOT NULL

Este operador se utiliza para comprobar si existe un valor en una columna de una fila.

Tema **IS NULL** - La columna Tema está vacía.
Tema **IS NOT NULL** - La columna Tema tiene información.

Ejemplo: Recuperar los empleados sin comisión.

```
SELECT numempl, nomempl, comision
  FROM empleado
 WHERE comision IS NULL;
```

Resultado:

```
  NUMEMPL NOMEMPL               COMISION
--------- -------------------- ---------
     1100 MORENO, LUIS
     1500 PEREZ, CARMEN
     1578 MORENO, VICENTE
     2001 DIEZ, CLARA
     2011 DURAN, ADRIAN
     2120 GONZALEZ, JUAN
     2121 TORRES, SANCHO
     2134 LASA, FRANCISCO
     2167 SANZ, PEDRO
     2456 FERNANDEZ, JOSÉ
     2565 GIL, ANA
```

3.1.6.2 Comprobaciones con listas de valores

Estos operadores realizan comparaciones con varios valores a la vez.

3.1.6.2.1 IN, NOT IN

Este operador realiza una comparación con una lista de valores separadas por comas. Se utiliza con valores numéricos y alfanuméricos.

Estado **IN** (‘A’,’B’,’C’) - Estado es igual a uno de los valores de la lista.

Estado **NOT IN** (‘A’,’B’,’C’) - Estado es distinto a los valores de la lista.

Piso **IN** (1,2,3) - Piso es igual a uno de los números de la lista.

Piso **NOT IN** (1,2,3) - Piso es distinto a los números de la lista.

Ejemplo: Recuperar los empleados de los departamentos 20 ó 40.

```
SELECT numempl, nomempl, coddept
  FROM empleado
 WHERE coddept IN (20,40);
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT
--------- -------------------- ---------
     1500 PEREZ, CARMEN                40
     2011 DURAN, ADRIAN                40
     2015 GARCÍA, PEDRO                20
     2134 LASA, FRANCISCO              20
     2190 RUIZ, MARÍA                  40
```

3.1.6.2.2 ALL, ANY / SOME

Estos operadores realizan una comparación con una lista de valores separadas por comas, deben ser precedidos por =, >, <, <=, >= <>. Se utiliza con valores numéricos y alfanuméricos.

Salario > = **ALL** (1200,2000).

- La comparación es verdadera si se cumple para todos los valores de la lista.

Salario = **ANY** (1200, 2000)

- La comparación es verdadera si se cumple para uno cualquiera de los valores de la lista.

El operador **'= ANY'** es equivalente a **IN** y **'<> ALL'** es equivalente a **NOT IN**.

Ejemplo: Recuperar los empleados de los departamentos 20 ó 40.

```
SELECT numempl, nomempl, coddept
  FROM empleado
 WHERE coddept = ANY (20,40);
```

Resultado:

```
  NUMEMPL NOMEMPL                  CODDEPT
--------- -------------------- ---------
     1500 PEREZ, CARMEN                40
     2011 DURAN, ADRIAN                40
     2015 GARCÍA, PEDRO                20
     2134 LASA, FRANCISCO              20
     2190 RUIZ, MARÍA                  40
```

3.1.6.2.3 BETWEEN, NOT BETWEEN

Este operador realiza una comparación con relación al intervalo fijado por dos valores.

Estado **BETWEEN** 'B' AND 'F'	Estado es igual a 'B' o 'F' o a cualquier carácter entre ellos (alfabéticamente).
Estado **NOT BETWEEN** 'B' AND 'F'	Estado es menor a 'B' o mayor a 'F' (alfabéticamente).
Pagina **BETWEEN** 3 AND 20	Pagina es igual a 3 o 20 o a cualquier valor entre ellos.
Pagina **NOT BETWEEN** 3 AND 20	Pagina es menor a 3 o mayor a 20.

Ejemplo: Recuperar los empleados de los departamentos comprendidos entre 20 ó 40.

```
SELECT numempl, nomempl, coddept
  FROM empleado
 WHERE coddept BETWEEN 20 AND 40;
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT
--------- -------------------- ---------
     1500 PEREZ, CARMEN                40
     2011 DURAN, ADRIAN                40
     2015 GARCÍA, PEDRO                20
     2134 LASA, FRANCISCO              20
     2167 SANZ, PEDRO                  30
     2190 RUIZ, MARÍA                  40
     2345 MARTIN, DIANA                30
```

3.1.6.3 Combinaciones lógicas

Cuando se necesita más de una condición para seleccionar las filas, se utilizan operadores lógicos AND y OR para relacionarlas, paréntesis para fijar el orden de evaluación y NOT para negarlas.

Hay que poner cuidado en la codificación de estas condiciones ya que al tener AND más precedencia (valor) que OR, se evaluarán primero las condiciones que aparecen a ambos lados de ella.

Ejemplo: Recuperar los empleados del departamento 40 con un salario superior a 2000.

```
SELECT numempl, nomempl, coddept, salario
  FROM empleado
 WHERE coddept = 40
   AND salario > 2000;
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT   SALARIO
--------- -------------------- --------- ---------
     1500 PEREZ, CARMEN               40      6000
     2011 DURAN, ADRIAN               40      3600
     2190 RUIZ, MARÍA                 40      3124
```

Ejemplo: Recuperar los empleados que sean TÉCNICO o VENDEDOR.

```
SELECT numempl, nomempl, coddept, nomclab
  FROM empleado
 WHERE nomclab = 'TÉCNICO'
    OR nomclab = 'VENDEDOR';
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT NOMCLAB
--------- -------------------- --------- ------------
     1578 MORENO, VICENTE             50 TÉCNICO
     2120 GONZALEZ, JUAN              50 TÉCNICO
     2345 MARTIN, DIANA               30 VENDEDOR
```

3.1.7 Subconsultas

Se puede incluir una sentencia SELECT como parte de la cláusula WHERE de una sentencia SELECT principal. Esta sentencia se denomina "subconsulta" y debe ir entre paréntesis.

Primero se ejecuta la subconsulta, y el resultado obtenido se evalúa con la sentencia principal.

3.1.7.1 Subconsultas que generan valores simples.

Todos los operadores simples se pueden utilizar con subconsultas, siempre que el resultado de la esta sea un valor simple.

Ejemplo: Recuperar los empleados que trabajan en Toledo.

```
SELECT numempl, nomempl, coddept, codlugt
  FROM empleado
 WHERE codlugt = (SELECT codlugt
                    FROM lugartrabajo
                      WHERE localid = 'TOLEDO');
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT   CODLUGT
--------- -------------------- --------- ---------
     1500 PEREZ, CARMEN               40        92
     1578 MORENO, VICENTE             50        92
     2001 DIEZ, CLARA                 50        92
     2011 DURAN, ADRIAN               40        92
     2120 GONZALEZ, JUAN              50        92
     2121 TORRES, SANCHO              50        92
```

Si la subconsulta recupera más de un valor y en la condición se utiliza el igual se genera un error al no poder realizar la comparación.

Ejemplo: Recuperar los empleados que trabajan en los departamentos Comercial o Finanzas.

```
SELECT numempl, nomempl, coddept
  FROM empleado
 WHERE coddept = (SELECT coddept
                    FROM departamento
                   WHERE nomdept = 'COMERCIAL'
                      OR nomdept = 'FINANZAS');
```

Resultado: La subconsulta recupera dos valores, generándose un error.

```
           WHERE CODDEPT = (SELECT CODDEPT
                            *
ERROR AT LINE 3:
ORA-01427: SINGLE-ROW SUBQUERY RETURNS MORE THAN ONE ROW
```

3.1.7.2 Subconsultas que generan listas de valores.

Cuando las subconsultas recuperan más de un valor es necesario utilizar los operadores IN, NOT IN, ALL, ANY/SOME que manejan listas de valores.

Ejemplo: Recuperar los empleados que trabajan en los departamentos Comercial o Finanzas.

```
SELECT numempl, nomempl, coddept
  FROM empleado
 WHERE coddept IN (SELECT coddept
                     FROM departamento
                    WHERE nomdept = 'COMERCIAL'
                       OR nomdept = 'FINANZAS');
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT
--------- -------------------- ---------
     2015 GARCÍA, PEDRO                20
     2134 LASA, FRANCISCO              20
     2167 SANZ, PEDRO                  30
     2345 MARTIN, DIANA                30
```

Si se utiliza el operador **ALL** la comparación es verdadera: si la condición se cumple para todas las filas recuperadas en la subconsulta, o si en esta no se recupera ninguna fila.

Si se utiliza uno de los operadores **ANY** o **SOME** la comparación es verdadera si se cumple para una de las filas recuperadas en la subconsulta. La comparación es falsa si la subconsulta no recupera ninguna fila.

El operador **IN** es equivalente a **'=ANY'** y **NOT IN** es equivalente a **'<> ALL'**.

3.1.8 Combinaciones de tablas

Cuando para obtener toda la información de una consulta se necesita combinar más de una tabla, esto ocurre frecuentemente cuando los datos están normalizados, hay que realizar combinaciones de tablas.

Esto es posible si las tablas tienen columnas que contienen el mismo tipo de información.

Ejemplo: Recuperar los empleados con el nombre de su lugar de trabajo.

```
SELECT numempl, nomempl, nomlugt, localid
  FROM empleado e, lugartrabajo l
 WHERE e.codlugt = l.codlugt;
```

Resultado:

```
  NUMEMPL NOMEMPL              NOMLUGT    LOCALID
--------- -------------------- ---------- ---------------
     1100 MORENO, LUIS         CENTRAL    Madrid
     2015 GARCÍA, PEDRO        CENTRAL    Madrid
     2134 LASA, FRANCISCO      CENTRAL    Madrid
     2345 MARTIN, DIANA        CENTRAL    Madrid
     2565 GIL, ANA             CENTRAL    Madrid
     2167 SANZ, PEDRO          CENTRAL    Madrid
     1500 PEREZ, CARMEN        FABRICA    Toledo
     1578 MORENO, VICENTE      FABRICA    Toledo
     2121 TORRES, SANCHO       FABRICA    Toledo
     2120 GONZALEZ, JUAN       FABRICA    Toledo
     2001 DIEZ, CLARA          FABRICA    Toledo
     2011 DURAN, ADRIAN        FABRICA    Toledo
```

La combinación de las tablas se realiza por medio de la condición de la cláusula WHERE que iguala las columnas de las dos tablas a enfrentar.

3.1.9 Vistas

Una vista es una consulta almacenada en la base de datos con un nombre. Tiene la misma estructura que una tabla: filas y columnas. Pero no contiene datos.

La información se recupera de los datos de la tabla o las tablas que forman parte de la definición de la vista.

3.1.9.1 Creación de vistas

Ejemplo: Crear una consulta (vista) con el nombre “wtoledano” con todos los empleados que trabajan en Toledo.

```
CREATE VIEW wtoledano AS
   SELECT numempl,nomempl,jefe,fchnac,
          fchingr,salario,comision,coddept
     FROM empleado
    WHERE codlugt = (SELECT codlugt
                       FROM lugartrabajo
                      WHERE localid = 'Toledo');
```

La vista se utiliza igual que una tabla.

Ejemplo: Recuperar los empleados con salario < 4000 y que trabajan en Toledo.

```
  SELECT numempl,nomempl,jefe,fchingr,salario
    FROM wtoledano
   WHERE salario < 4000
ORDER BY salario;
```

Resultado:

```
  NUMEMPL NOMEMPL                    JEFE FCHINGR    SALARIO
--------- -------------------- --------- -------- ---------
     2120 GONZALEZ, JUAN             2001 09/03/89      3450
     2011 DURAN, ADRIAN              1500 20/09/88      3600
```

3.1.9.2 Borrado de vistas

Si se tiene que modificar la definición de la vista, hay que borrar la definición anterior y repetir la sentencia de creación. Esto se realiza con la sentencia DROP VIEW:

```
DROP VIEW wtoledano;
```

3.1.10 Resumen

Con la sentencia SELECT se obtiene cualquier combinación de filas y columnas de las tablas consultadas.

Esta sentencia tiene dos palabras clave obligatorias SELECT y FROM, y dos cláusulas no obligatorias, pero muy importantes WHERE y ORDER BY.

Con la palabra SELECT se indican las columnas seleccionadas y con la palabra FROM las tablas utilizadas.

Con la cláusula WHERE se seleccionan las filas recuperadas por la consulta y con la cláusula ORDER BY se ordena el resultado.

Las vistas son definiciones de consultas guardadas en la base de datos con un nombre. Se utilizan igual que una tabla y recuperan en cada momento los datos actuales de las tablas sobre las que están definidas.

Se denomina subconsulta a una sentencia SELECT que se utiliza dentro de la cláusula WHERE de una sentencia SELECT principal.

3.1.11 Ejercicios

Ejercicio1: Recuperar los empleados con las letras OR en la segunda y tercera posición del nombre.

Ejercicio2: Recuperar los empleados sin comisión de los departamentos 20 y 40.

Ejercicio3: Recuperar los empleados de los departamentos que tienen un código menor que el correspondiente al departamento Comercial.

Ejercicio4: Recuperar los empleados con el nombre de su lugar de trabajo, el nombre de su departamento y su situación laboral.

Ejercicio5: Obtener los empleados del departamento de Martín que tengan su misma comisión.

3.2 Manipulación de texto

3.2.1 Objetivo

En este apartado se van a utilizar una serie de funciones que sirven para modificar los datos alfanuméricos recuperados en las consultas.

Algunas funciones modifican los datos alfanuméricos, otras generan un resultado que indica algo sobre ellos.

3.2.2 Las funciones de cadenas

Notación : FUNCIÓN (Columna alfanumérica o Cadena de caracteres [,opción])

Nombre	**Utilización**
|| (Concatenación)	Une dos cadenas de caracteres.
INITCAP	Cambia las primeras letras de las palabras de una cadena de caracteres.
LOWER	Convierte todas las letras de la cadena de caracteres a minúsculas.
UPPER	Convierte todas las letras de la cadena a mayúsculas.
RPAD	Aumenta el tamaño de una cadena, añadiendo un conjunto de caracteres en la parte derecha.
LPAD	Aumenta el tamaño de una cadena, añadiendo un conjunto de caracteres en la parte izquierda.
RTRIM	Suprime un conjunto de caracteres de la parte derecha de la cadena.
LTRIM	Suprime un conjunto de caracteres de la parte izquierda de la cadena.

LENGTH Indica la longitud de la cadena.

INSTR Busca la localización de un carácter en una cadena.

SUBSTR Extrae una parte de una cadena.

Las funciones de datos alfanuméricos también se pueden utilizar en las cláusulas ORDER BY y WHERE.

3.2.2.1 || Concatenación

Esta función enlaza columnas y literales.

3.2.2.2 RPAD y LPAD

RPAD aumenta la longitud de una columna por la derecha con un conjunto de caracteres. LPAD hace lo mismo pero por la izquierda.

Notación: RPAD (cadena, longitud, 'caracteres')
Notación: LPAD (cadena, longitud, 'caracteres')

Ejemplo: Recuperar los empleados del departamento 50 y rellenando hasta 20 caracteres por la derecha con '*.-'.

```
SELECT RPAD(nomempl,20,'*.-')
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
RPAD(NOMEMPL,20,'*.-
--------------------
MORENO, VICENTE*.-*.
DIEZ, CLARA*.-*.-*.-
GONZALEZ, JUAN*.-*.-
TORRES, SANCHO*.-*.-
FERNANDEZ, JOSÉ*.-*.
```

3.2.2.3 RTRIM y LTRIM

RTRIM y LTRIM suprimen caracteres por la derecha o por la izquierda.

Notación RTRIM (cadena, 'caracteres')

Ejemplo: Suprimir los caracteres E, N, A, T, R por la derecha en el nombre de los empleados del departamento 50.

```
SELECT nomempl,RTRIM(nomempl,'ENATR')
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
NOMEMPL              RTRIM(NOMEMPL,'ENATR
-------------------- --------------------
MORENO, VICENTE      MORENO, VIC
DIEZ, CLARA          DIEZ, CL
GONZALEZ, JUAN       GONZALEZ, JU
TORRES, SANCHO       TORRES, SANCHO
FERNANDEZ, JOSÉ      FERNANDEZ, JOSÉ
```

Notación LTRIM (cadena, 'caracteres')

Ejemplo: Suprimir los caracteres E, P, U, R por la izquierda en el nombre de los empleados del departamento 40.

```
SELECT nomempl,LTRIM(nomempl,'EPUR')
  FROM empleado
 WHERE coddept = 40;
```

Resultado:

```
NOMEMPL              LTRIM(NOMEMPL,'EPUR'
-------------------- --------------------
PEREZ, CARMEN        Z, CARMEN
DURAN, ADRIAN        DURAN, ADRIAN
RUIZ, MARÍA          IZ, MARÍA
```

3.2.2.4 LOWER, UPPER y INITCAP

LOWER - Convierte todas las letras de la cadena de caracteres a minúsculas.

Notación LOWER (cadena)

UPPER - Convierte todas las letras de la cadena de caracteres a mayúsculas.

Notación UPPER (cadena)

INITCAP - Cambia las primeras letras de las palabras de una cadena de caracteres.

Notación INITCAP (cadena)

Ejemplo: Recuperar el nombre de los empleados del departamento 20.

```
SELECT LOWER(nomempl),UPPER(nomempl),INITCAP(nomempl)
  FROM empleado
 WHERE coddept = 20;
```

Resultado:

```
LOWER(NOMEMPL)       UPPER(NOMEMPL)       INITCAP(NOMEMPL)
-------------------- -------------------- --------------------
garcía, pedro        GARCÍA, PEDRO        García, Pedro
lasa, francisco      LASA, FRANCISCO      Lasa, Francisco
llanos, cristina     LLANOS, CRISTINA     Llanos, Cristina
```

3.2.2.5 LENGTH

Indica el número de caracteres de una columna o literal, incluyendo cualquier carácter y los espacios en blanco.

Notación LENGTH (cadena)

Ejemplo: Recuperar el número de caracteres de los nombres de los empleados del departamento 40.

```
SELECT nomempl, LENGTH(nomempl)
  FROM empleado
 WHERE coddept = 40;
```

Resultado:

```
NOMEMPL               LENGTH(NOMEMPL)
--------------------- ---------------
PEREZ, CARMEN                      13
DURAN, ADRIAN                      13
RUIZ, MARÍA                        11
```

3.2.2.6 INSTR

Permite la búsqueda de un conjunto de caracteres en una cadena, indicando la posición donde se encuentra.

Tiene dos opciones: en la primera se indica la posición desde la que se inicia la búsqueda, la segunda indica el número de ocurrencia que se quiere verificar.

Notación INSTR (cadena, 'caracteres'[,comienzo[,ocurrencia]])

Ejemplo: Recuperar el lugar del primer carácter ',' que aparece en el nombre de los empleados del departamento 50, comenzando la búsqueda en la primera posición.

```
SELECT nomempl, INSTR(nomempl,',')
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
NOMEMPL              INSTR(NOMEMPL,',')
-------------------- ------------------
MORENO, VICENTE                       7
DIEZ, CLARA                           5
GONZALEZ, JUAN                        9
TORRES, SANCHO                        7
FERNANDEZ, JOSÉ                      10
```

3.2.2.7 SUBSTR

Recupera una subcadena del contenido de una columna o literal.

Notación SUBSTR (cadena, comienzo[,longitud]])

Ejemplo: Recuperar la subcadena de cinco caracteres desde la columna nombre de empleado empezando por la tercera posición.

```
SELECT nomempl, SUBSTR(nomempl,3,5)
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
NOMEMPL              SUBSTR(NOMEMPL,1,5)
-------------------- --------------------
MORENO, VICENTE      RENO
DIEZ, CLARA          EZ
GONZALEZ, JUAN       NZALE
TORRES, SANCHO       RRES
FERNANDEZ, JOSÉ      RNAND
```

3.2.3 Funciones de cadenas en cláusulas WHERE y ORDER BY.

Ejemplo: Recuperar los empleados de los departamentos con código menor de 40 y ordenar según la longitud del nombre.

```
SELECT nomempl, length(nomempl)
  FROM empleado
 WHERE coddept < 40
 ORDER BY LENGTH(nomempl);
```

Resultado:

```
NOMEMPL              LENGTH(NOMEMPL)
-------------------- ---------------
GIL, ANA                           8
SANZ, PEDRO                       11
MORENO, LUIS                      12
GARCÍA, PEDRO                     13
MARTIN, DIANA                     13
LASA, FRANCISCO                   15
LLANOS, CRISTINA                  16
```

Ejemplo: Recuperar los nombres de los empleados con longitudes del nombre menores de 13 y ordenar según la longitud del nombre.

```
SELECT nomempl, length(nomempl)
  FROM empleado
 WHERE LENGTH(nomempl) < 13
 ORDER BY 2;
```

Resultado:

```
NOMEMPL              LENGTH(NOMEMPL)
-------------------- ---------------
GIL, ANA                           8
DIEZ, CLARA                       11
SANZ, PEDRO                       11
RUIZ, MARÍA                       11
MORENO, LUIS                      12
```

3.2.4 Resumen

Por medio de funciones se puede modificar o describir alguna característica de las cadenas de caracteres recuperadas en las consultas.

Las funciones LOWER, UPPER, INITCAP, LPAD, RPAD, LTRIM, RTRIM, y SUBSTR, modifican el contenido de columnas alfanuméricas.

Las funciones LENGTH y INSTR indican la longitud y el lugar donde se encuentra ciertos caracteres.

3.2.5 Ejercicios

Ejercicio1: Recuperar los empleados del departamento 40, rellenando hasta 20 caracteres por la izquierda con '_*'.

Ejercicio2: Recuperar el apellido de los empleados del departamento 50 desde la columna nomempl.

Ejercicio3: Recuperar el nombre de los empleados del departamento 50 desde la columna nomempl.

Ejercicio4: Obtener las tres últimos caracteres de los nombres de los departamentos ordenados alfabéticamente.

3.3 Manipulación de números

3.3.1 Objetivo

En este apartado se va a utilizar una serie de funciones que sirven para operar con los datos numéricos recuperados en las consultas. Se agrupan en tres tipos: las que trabajan con valores simples, las que trabajan con grupos de valores y las que trabajan con listas de valores.

Algunas funciones modifican los valores originales, otras generan un resultado que indica algo sobre los ellos.

3.3.2 Las funciones de valores simples

Nombre	**Utilización**
Valor1 + valor2	Suma
Valor1 – valor2	Resta.
Valor1 * valor2	Multiplicación.
Valor1 / valor2	División.
ABS(valor)	Valor absoluto.
CEIL(valor)	Entero más pequeño igual o mayor que valor.
FLOOR(valor)	Entero mayor más pequeño o igual que valor.
EXP(valor)	e (2,71828183...) elevado al exponente valor.
MOD(valor, divisor)	Resto de dividir valor por divisor.
NVL(valor, sustituto)	Si valor es nulo lo reemplaza por sustituto.
POWER(valor, exponente)	Eleva valor al exponente.
ROUND(valor, precisión)	Redondea el valor con la precisión dada.
SIGN(valor)	1 si valor es positivo, -1 si valor es negativo.
SQRT(valor)	Raíz cuadrada de valor.

TRUNC(valor, precisión) Trunca el valor con la precisión.

VSIZE(valor) Tamaño de almacenamiento de valor.

LN(valor) Logaritmo en base e de valor.

LOG(valor) Logaritmo en base 10 de valor.

3.3.2.1 Suma, resta, multiplicación y división.

Ejemplo: Recuperar la fila con nombre igual a 'DECIMAL ALTO' y realizar operaciones con las columnas positivo y negativo.

```
SELECT nombre, positivo, negativo, vacio,
        positivo + negativo suma,
        positivo - negativo resta,
        positivo * negativo producto,
        positivo / negativo división
   FROM numeros
  WHERE nombre = 'DECIMAL ALTO';
```

Resultado:

```
NOMBRE         POSITIVO  NEGATIVO     VACIO      SUMA     RESTA  PRODUCTO  DIVISIÓN
------------- --------- --------- --------- --------- --------- --------- ---------
DECIMAL ALTO     77,777   -88,888            -11,111   166,665 -6913,442     -,875
```

Ejemplo: Recuperar la fila con nombre igual a ‘DECIMAL ALTO’ y realizar operaciones con las columnas positivo y vacio.

```
SELECT nombre, positivo, negativo, vacio,
       positivo + vacio suma,
       positivo - vacio resta,
       positivo * vacio producto,
       positivo / vacio división
  FROM numeros
 WHERE nombre = 'DECIMAL ALTO';
```

Resultado:

```
NOMBRE         POSITIVO  NEGATIVO     VACIO      SUMA     RESTA  PRODUCTO  DIVISIÓN
-------------- --------- --------- --------- --------- --------- --------- ---------
DECIMAL ALTO      77,777   -88,888
```

El contenido de las columnas suma, resta, producto y división es NULL al haber operado con una columna que no contiene valor.

Para evitar estos resultados existe una función **NVL** que permite sustituir los valores nulos por un valor.

Ejemplo: Recuperar la fila con nombre igual a 'DECIMAL ALTO' y realizar operaciones con las columnas positivo y vacio, pero utilizando la función NVL para sustituir el NULL por cero o uno.

```
SELECT nombre, positivo, negativo, vacio,
        positivo + NVL(vacio,0) suma,
        positivo - NVL(vacio,0) resta,
        positivo * NVL(vacio,1) producto,
        positivo / NVL(vacio,1) división
   FROM numeros
  WHERE nombre = 'DECIMAL ALTO';
```

Resultado:

NOMBRE	POSITIVO	NEGATIVO	VACIO	SUMA	RESTA	PRODUCTO	DIVISIÓN
DECIMAL ALTO	77,777	-88,888		77,777	77,777	77,777	77,777

3.3.2.2 TRUNC y ROUND.

TRUNC - Corta los números para que tengan un número de dígitos de precisión.

Notación TRUNC(valor, precisión).

Ejemplo: Recuperar las columnas positivo y negativo truncando con un dígito de precisión.

```
SELECT nombre, positivo, negativo,
       TRUNC(positivo,1),
       TRUNC(negativo,1)
  FROM numeros;
```

Resultado:

NOMBRE	POSITIVO	NEGATIVO	TRUNC(POSITIVO,1)	TRUNC(NEGATIVO,1)
PARTE ENTERA	11	-33	11	-33
DECIMAL BAJO	22,22	-44,44	22,2	-44,4
DECIMAL MEDIO	55,5	-55,5	55,5	-55,5
DECIMAL ALTO	77,777	-88,888	77,7	-88,8

ROUND - Redondea los números con el número de dígitos de precisión.

Notación ROUND(valor, precisión).

Ejemplo: Recuperar las columnas positivo y negativo redondeando con un dígito de precisión.

```
SELECT nombre, positivo, negativo,
       ROUND(positivo,1),
       ROUND(negativo,1)
  FROM numeros;
```

Resultado:

NOMBRE	POSITIVO	NEGATIVO	ROUND(POSITIVO,1)	ROUND(NEGATIVO,1)
PARTE ENTERA	11	-33	11	-33
DECIMAL BAJO	22,22	-44,44	22,2	-44,4
DECIMAL MEDIO	55,5	-55,5	55,5	-55,5
DECIMAL ALTO	77,777	-88,888	77,8	-88,9

3.3.2.2.1 Precisión negativa

Estas funciones también trabajan con precisión negativa.

Ejemplo: Recuperar las columnas positivo y negativo redondeando con un dígito de precisión negativo.

```
SELECT nombre, positivo, negativo,
       ROUND(positivo,-1),
       ROUND(negativo,-1)
  FROM numeros;
```

Resultado:

```
NOMBRE         POSITIVO  NEGATIVO ROUND(POSITIVO,-1) ROUND(NEGATIVO,-1)
-------------- --------- --------- ------------------ ------------------
PARTE ENTERA         11       -33                 10                -30
DECIMAL BAJO      22,22    -44,44                 20                -40
DECIMAL MEDIO      55,5     -55,5                 60                -60
DECIMAL ALTO     77,777   -88,888                 80                -90
```

Ejemplo: Recuperar las columnas positivo y negativo truncando con un dígito de precisión negativo.

```
SELECT nombre, positivo, negativo,
       TRUNC(positivo,-1),
       TRUNC(negativo,-1)
  FROM numeros;
```

Resultado:

```
NOMBRE         POSITIVO  NEGATIVO TRUNC(POSITIVO,-1) TRUNC(NEGATIVO,-1)
-------------- --------- --------- ------------------ ------------------
PARTE ENTERA         11       -33                 10                -30
DECIMAL BAJO      22,22    -44,44                 20                -40
DECIMAL MEDIO      55,5     -55,5                 50                -50
DECIMAL ALTO     77,777   -88,888                 70                -80
```

3.3.3 Funciones trigonométricas.

Nombre	**Utilización**
COS(valor)	Coseno de valor.
COSH(valor)	Coseno hiperbólico de valor.
SIN(valor)	Seno de valor.
SINH(valor)	Seno hiperbólico de valor.
TAN(valor)	Tangente de valor.
TANH(valor)	Tangente hiperbólica de valor.

Valor se indica en radianes (grados multiplicados por pi dividido por 180).

Ejemplo: Calcular el coseno de pi radianes.

```
SELECT COS(3.141516) FROM DUAL;
```

Resultado:

```
COS(3.141516)
-------------
           -1
```

Ejemplo: Calcular el coseno de cero radianes.

```
SELECT COS(0) FROM DUAL;
```

Resultado:

```
COS(0)
---------
        1
```

3.3.4 Las funciones de grupos de valores

Estas funciones dan información sobre un grupo de filas

Nombre	**Utilización**
AVG(valor)	Media de valor para un grupo de filas.
COUNT(valor)	Cuenta las filas.
MAX(valor)	Máximo de valor para un grupo de filas.
MIN(valor)	Mínimo de valor para un grupo de filas.
STDDEV(valor)	Desviación estándar de valor para un grupo de filas.
SUM(valor)	Suma de valor para un grupo de filas.
VARIANCE(valor)	Varianza de valor para un grupo de filas.

Los valores nulos son ignorados en los cálculos realizados por estas funciones.

3.3.4.1 SUM, MAX, MIN, AVG.

Ejemplo: Calcular la suma de los salarios, el salario máximo , el salario mínimo y el salario medio de los empleados.

```
SELECT SUM(salario),
       MAX(salario),
       MIN(salario),
       AVG(salario
  FROM empleado;
```

Resultado:

```
SUM(SALARIO) MAX(SALARIO) MIN(SALARIO) AVG(SALARIO)
------------ ------------ ------------ ------------
       52149         9000          800    3724,9286
```

3.3.4.2 COUNT.

Esta función cuenta filas.

Notación: **COUNT([DISTINCT|ALL] valor)**

COUNT(DISTINCT salario)	Cuenta el número de salarios diferentes.
COUNT(salario)	Cuenta el número de filas con valores no nulos en salario
COUNT(*)	Cuenta todas las filas.

Ejemplo: Calcular el número de salarios distintos, el número de empleados con salario y el número de empleados.

```
SELECT COUNT(DISTINCT salario),
         COUNT(salario),
         COUNT(*)
    FROM empleado;
```

Resultado:

```
COUNT(DISTINCTSALARIO) COUNT(SALARIO)  COUNT(*)
---------------------- -------------- ---------
                    13             14        15
```

3.3.5 Las funciones de listas

Estas funciones trabajan sobre un grupo de valores o de columnas.

Nombre	**Utilización**
GREATEST(valor1,valor2....)	Mayor valor de la lista.
LEAST(valor1,valor2....)	Menor valor de la lista.

Ejemplo: Seleccionar el número mayor y menor entre las columnas numempl y jefe.

```
SELECT numempl, jefe, GREATEST(numempl,jefe),
                      LEAST(numempl,jefe)
  FROM empleado;
```

Resultado:

```
  NUMEMPL      JEFE GREATEST(NUMEMPL,JEFE) LEAST(NUMEMPL,JEFE)
--------- --------- -------------------- ------------------
     1100
     1500      2190                 2190               1500
     1578      2001                 2001               1578
     2001      1100                 2001               1100
     2011      1500                 2011               1500
     2015      1100                 2015               1100
     2120      2001                 2120               2001
     2121      1578                 2121               1578
     2134      2015                 2134               2015
     2167      2345                 2345               2167
     2190      1100                 2190               1100
     2345      1100                 2345               1100
     2456      1578                 2456               1578
     2565      1100                 2565               1100
     1015
```

3.3.6 Resumen

Por medio de funciones se puede modificar los valores numéricos recuperados en las consultas. Las funciones de valores simples (ABS, MOD, TRUNC, ROUND, etc.) se aplican fila a fila. Las funciones de listas (GREASTEST, LEAST) comparan columnas y seleccionan un valor.

Tanto las funciones de valores simples como las de listas no generan resultado si encuentran un valor NULL.

Las funciones de grupos de valores (AVG, COUNT, MAX, etc.) dan información sobre un grupo de filas. Estas funciones ignoran los valores NULL.

3.3.7 Ejercicios

Ejercicio1: Hallar el número de empleados del departamento 50, el número de comisiones distintas y la suma y la media de las comisiones.

Ejercicio2: Calcular la media de las comisiones.

Ejercicio3: Hallar la masa salarial anual (salario + comisiones) de la empresa.

3.4 Manipulación de fechas

3.4.1 Objetivo

En este apartado se va a utilizar una serie de funciones que permiten operar con datos de fechas y horas recuperados en las consultas.

Algunas funciones modifican los valores originales. Otras funciones generan un resultado que indica algo sobre estos valores.

3.4.2 SYSDATE

La variable del sistema o pseudo-columna SYSDATE, permite recuperar la fecha y hora del sistema. Se utiliza como una columna especial.

```
SELECT SYSDATE FROM DUAL;
```

Resultado:

```
SYSDATE
-------------
10-JUN-99
```

La tabla DUAL no tiene datos se utiliza para probar funciones y para hacer cálculos rápidos.

3.4.3 Funciones de fechas

Nombre	**Utilización**
ADD_MONTHS(fecha, n)	Suma n meses a fecha.
GREATEST(fecha1,fecha2,....)	Mayor fecha de la lista.
LEAST(fecha1,fecha2,....)	Menor fecha de la lista.

LAST_DAY(fecha)	Último día del mes de la fecha.
MONTHS_BETWEEN(fecha1,fecha2)	Meses entre dos fechas
NEXT_DAY(fecha, 'ddd')	Fecha del día de la siguiente semana identificado por ddd.
NEW_TIME(fecha1, 'eee', 'fff')	Fecha de la zona horaria fff ,cuando en la zona horaria eee la fecha es igual fecha1.
TO_CHAR(fecha, 'máscara')	Cambia una fecha a un valor tipo carácter con el formato indicado en la máscara.
TO_DATE(cadena, 'máscara')	Cambia una cadena a una fecha con el formato indicado en la máscara.

Ejemplo: Añadir 6 meses a la fecha de ingreso de los empleados dados de baja (tabla bajas).

```
SELECT nomempl, fchingr, ADD_MONTHS(fchingr,6)
  FROM bajas;
```

Resultado:

```
NOMEMPL              FCHINGR  ADD_MONT
-------------------- -------- --------
GARCÍA , FERNANDO    17/11/65 17/05/66
HERNANDEZ,CARMEN     09/12/60 09/06/61
SAN JUAN, VICENTE    03/03/63 03/09/63
DIEZ, CARLOS         02/04/60 02/10/60
LOPEZ, DIEGO         20/09/55 20/03/56
LLANOS, CRISTINA     21/06/66 21/12/66
```

Ejemplo: Calcular el último día del mes de la fecha de ingreso de los empleados dados de baja (tabla bajas)..

```
SELECT nomempl, fchingr, LAST_DAY(fchingr)
  FROM bajas;
```

Resultado:

```
NOMEMPL              FCHINGR  LAST_DAY
-------------------- -------- --------
GARCÍA , FERNANDO    17/11/65 30/11/65
HERNANDEZ,CARMEN     09/12/60 31/12/60
SAN JUAN, VICENTE    03/03/63 31/03/63
DIEZ, CARLOS         02/04/60 30/04/60
LOPEZ, DIEGO         20/09/55 30/09/55
LLANOS, CRISTINA     21/06/66 30/06/66
```

3.4.4 Funciones de conversión.

TO_CHAR(fecha, 'máscara')

Convierte una fecha a un valor tipo carácter, con el formato indicado en la máscara.

Ejemplo: Recuperar el día, el mes y el año del Sistema.

```
SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY')
  FROM DUAL;
```

Resultado:

```
TO_CHAR(SY
----------
21/06/2000
```

TO_DATE(cadena, 'máscara')

Convierte una cadena de caracteres con un formato de fecha según se indica en la máscara en un dato de tipo DATE.

```
SELECT TO_DATE('15-06-00','DD-MM-YY')
  FROM DUAL;
```

Resultado:

```
TO_DATE(
--------
15/06/00
```

3.4.5 Formatos de máscaras de fechas

Estos formatos se utilizan con TO_CHAR y con TO_DATE;

MM	Número de mes.
RM	Mes en números romanos.
MON	Abreviatura de tres letras del mes.
MONTH	Mes con todas las letras.
DDD	Número del día del año.
DD	Número del día del mes.
D	Número del día de la semana.
DY	Abreviatura de tres letras del día de la semana.
DAY	Nombre del día de la semana.
YYYY	Año con cuatro dígitos.
SYYYY	Año con signo, 1000 a. C. = -1000.
YYY	Los tres últimos dígitos del año.
YY	Los dos últimos dígitos del año.
Y	El último dígito del año.
RR	Los dos últimos dígitos relativos a la fecha actual.
YEAR	Nombre del año.
Q	Número del trimestre.
WW	Número de semana del año.
IW	Semana en el año según el estandar ISO*.
W	Número de semana en el mes.
J	Días Julianos desde el 31 de diciembre del 4713 a.C..
HH	Hora del día, de 1-12.
HH12	Hora del día, de 1-12.
HH24	Hora del día, de 1-24.
MI	Minuto de la hora.
SS	Segundo del minuto.
SSSS	Segundos desde la medianoche, desde 0 a 86399.
A.M. o P.M.	Muestra A.M. o P.M. dependiendo del momento del día.
AM o PM	Lo mismo que A.M.

3.4.5.1 Prefijos y sufijos de máscaras de formato.

Estos formatos se utilizan con TO_CHAR.

fm (Prefijo de MONTH o DAY)	Suprime los espacios en blanco de los meses y de los días
TH (Sufijo de un número)	Indica el cardinal.
SP (Sufijo de un número)	Escribe el texto del número.
SPTH o THSP (Sufijos de números)	Combinación de TH y SP.

Ejemplo: Recuperar el nombre del mes, el ordinal del día en inglés y en minúscula (dd), y el año de la fecha de ingreso de los empleados de los departamentos con código menor de 30.

```
SELECT nomempl, fchingr,
       TO_CHAR(fchingr,'Month, ddth, YYYY')
  FROM empleado
 WHERE coddept < 30;
```

Resultado:

```
NOMEMPL              FCHINGR  TO_CHAR(FCHINGR,'MONTH
-------------------- -------- ----------------------
MORENO, LUIS         17/11/81 Noviembre , 17th, 1981
GARCÍA, PEDRO        21/06/89 Junio     , 21st, 1989
LASA, FRANCISCO      22/03/90 Marzo     , 22nd, 1990
GIL, ANA             03/02/99 Febrero   , 03rd, 1999
LLANOS, CRISTINA     21/06/66 Junio     , 21st, 1966
```

3.4.6 ROUND Y TRUNC con fechas.

Cuando se inserta o actualiza una fecha por medio de un literal se le asigna por omisión a la hora el valor de 12 A.M.(Medianoche).

La función ROUND sobre una fecha, pasa la hora a 12 A.M del día si la hora es anterior al mediodía, y a las 12 A.M. del día siguiente si la hora es posterior al mediodía.

La función TRUNC se comporta de forma similar, pero con la diferencia que siempre pasa la hora a las 12 A.M.

Ejemplo: Calcular la diferencia entre una fecha y SYSDATE.

```
SELECT TO_DATE('15-06-00','DD-MM-YY') - SYSDATE,
       SYSDATE
  FROM DUAL;
```

Resultado: Resulta un valor con decimales al incluir SYSDATE la fecha y la hora.

```
TO_DATE('15-06-00','DD-MM-YY')-SYSDATE SYSDATE
-------------------------------------- --------
                            -6,954479 21/06/00
```

Ejemplo: Calcular la diferencia entre una fecha y SYSDATE utilizando TRUNC.

```
SELECT TO_DATE('15-06-00','DD-MM-YY') - TRUNC(SYSDATE),
       SYSDATE
 FROM DUAL;
```

Resultado: Un valor entero.

```
TO_DATE('15-06-00','DD-MM-YY')-TRUNC(SYSDATE) SYSDATE
--------------------------------------------- --------
                                           -6 21/06/00
```

3.4.7 Resumen

Existe una variable del sistema o pseudo-columna SYSDATE con la que se recupera del sistema la fecha y hora actual.

Con las funciones ADD_MONTHS, LAST_DAY, GREATEST, LEAST, etc. se modifican las fechas de las columnas de tipo DATE.

Las funciones TRUNC y ROUND redondea la hora a las 12 A.M. en las fechas de las columnas de tipo DATE.

La función TO_CHAR transforma una cadena de caracteres en una fecha (DATE).

La función TO_DATE transforma una fecha (DATE) en una cadena de caracteres.

3.4.8 Ejercicios

Ejercicio1: Calcular los años de trabajo de los empleados en la compañía.

Ejercicio2: Calcular los Días correspondientes al viernes y al lunes siguiente a la fecha contenida en SYSDATE.

Ejercicio3: Recuperar el nombre del mes sin espacios en blanco, el ordinal del día en inglés y en minúscula (dd), y el año de la fecha de ingreso de los empleados de los departamentos con código menor de 30.

Ejercicio4: Recuperar los empleados con código de departamento menor de 30 y su fecha de ingreso con el formato 'Ingresado el 17 de Noviembre de 1979, a las 12:30'.

3.5 Funciones de transformación.

3.5.1 Objetivo

En este apartado se van a utilizar dos funciones con las que se puede controlar la salida basándose en la entrada. Estas son TRANSLATE Y DECODE.

3.5.2 TRANSLATE

Es una función que hace una sustitución carácter a carácter en una cadena.

Notación: TRANSLATE (cadena, existe, cambia)

Recorre cada carácter de '**cadena**' y lo busca en **'existe',** y si lo encuentra lo sustituye por el carácter que ocupa la misma posición en '**cambia**'.

Ejemplo: Sustituir en 43789: los 2 por C, los 4 por H, los 5 por K, los 7 por P y los 8 por L.

```
SELECT TRANSLATE(43789,24578,'CHKPL')
  FROM DUAL;
```

Resultado:

```
TRANS
-----
H3PL9
```

3.5.3 DECODE

Se puede considerar como una sustitución valor a valor. Para cada '**valor**' DECODE comprueba si coincide con '**si_valx**' y en este caso lo cambia por '**entoncesx**', si no coincide con ninguno lo cambia por si_no_val_def.

Notación: **DECODE**(valor, si_val1,entonces1,
si_val2, entonces2,
si_val3, entonces3,
si_no_val_def)

Esta función es equivalente a IF, ELSIF y ELSE.

Ejemplo: Sustituir los valores de la columna dgsitlab por literales.

```
SELECT nomempl,
       DECODE(dgsitlab,'A','Activo',
                       'B','Baja',
                       'J','Jubilado',
                       'Sin situación')
  FROM empleado
 WHERE coddept > 10
   AND coddept < 50;
```

Resultado:

```
NOMEMPL              DECODE(DGSITL
-------------------- ------------
PEREZ, CARMEN        Activo
DURAN, ADRIAN        Activo
GARCÍA, PEDRO        Activo
LASA, FRANCISCO      Activo
SANZ, PEDRO          Activo
RUIZ, MARÍA          Sin situación
MARTIN, DIANA        Activo
LLANOS, CRISTINA     Jubilado
```

Ejercicios

Ejercicio1: Recuperar los nombres de los empleados eliminando las vocales y cambiando las comas por guiones.

3.6 Agrupación de filas.

3.6.1 Objetivo

En este apartado se van a utilizar dos cláusulas no obligatorias de la sentencia SELECT que actúan sobre grupos de filas. Estas son GROUP BY y HAVING.

3.6.2 GROUP BY

Agrupa filas según los valores de las columnas de agrupación indicadas.

Notación: GROUP BY column1[,columna2]

Por cada valor de las columnas de agrupamiento se genera una fila en el resultado de la consulta. Cada una de ellas puede contener: columnas generadas con funciones de grupo y columnas utilizadas para agrupar filas.

Ejemplo: Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando las filas (empleados) con salarios y las filas (empleados) de cada departamento.

```
SELECT coddept,SUM(salario),COUNT(salario),COUNT(coddept)
  FROM empleado
 WHERE codlugt in (SELECT codlugt
                     FROM lugartrabajo
                    WHERE localid = 'Madrid')
 GROUP BY coddept;
```

Resultado:

```
  CODDEPT SUM(SALARIO) COUNT(SALARIO) COUNT(CODDEPT)
--------- ------------ -------------- --------------
       10        10950              2              2
       20         7850              3              3
       30         4950              2              2
```

Si en la consulta anterior se selecciona una columna de la tabla que no se ha utilizado en la agrupación de filas, se genera un error.

```
SELECT nomdept,MIN(salario)
  FROM empleado
 WHERE codlugt in (SELECT codlugt
                     FROM lugartrabajo
                    WHERE localid = 'Madrid')
 GROUP BY coddept;
```

Resultado:

```
SELECT NOMEMPL,MIN(SALARIO)
       *
ERROR AT LINE 1:
ORA-00937: NOT A SINGLE-GROUP GROUP FUNCTION
```

3.6.3 HAVING

Selecciona las filas generadas con la cláusula GROUP BY que cumplen una condición.

Notación: HAVING condición

Ejemplo: Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando los empleados con salarios y los empleados de cada departamento, seleccionando sólo los grupos con código de departamento > 12.

```
SELECT coddept,SUM(salario),COUNT(salario),COUNT(coddept)
    FROM empleado
   WHERE codlugt in (SELECT codlugt
                       FROM lugartrabajo
                      WHERE localid = 'Madrid')
   GROUP BY coddept
  HAVING coddept > 12;
```

Resultado:

```
  CODDEPT SUM(SALARIO) COUNT(SALARIO) COUNT(CODDEPT)
--------- ------------ -------------- --------------
       20         7850              3              3
       30         4950              2              2
```

En la condición de HAVING se pueden utilizar las columnas generadas y las columnas de agrupación que se indican en la cláusula GROUP BY. Estas mismas columnas se pueden utilizar en la cláusula ORDER BY.

3.6.4 Orden de ejecución de las cláusulas de SELECT.

Cuando la sentencia SELECT contiene todas las cláusulas anteriores, la ejecución se realiza en el orden siguiente:

- Selecciona las filas con la cláusula WHERE.
- Agrupa las filas según la cláusula GROUP BY.
- Calcula los resultados de las funciones de grupo por cada grupo.
- Selecciona los grupos con la cláusula HAVING.
- Ordena los grupos según los resultados de las funciones de grupo.

En la cláusula ORDER BY hay que utilizar funciones de grupo o columnas de la cláusula GROUP BY.

3.6.5 Resumen

Por medio de cláusula GROUP BY de la sentencia SELECT, se agrupan las filas seleccionadas con la cláusula WHERE.

Con las condiciones de la cláusula HAVING se seleccionan las filas de los grupos generados..

Posteriormente la cláusula ORDER BY ordena las filas de los grupos.

3.6.6 Ejercicios

Ejercicio1: Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando las filas con salarios y las filas que hay por cada departamento. Seleccionando sólo los grupos con menos de tres filas por grupo.

Ejercicio 2: Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando las filas con salarios y las filas que hay por cada departamento.

3.7 Consultas dependientes.

3.7.1 Objetivo

En las subconsultas utilizadas hasta ahora, en la cláusula WHERE sólo se han utilizado columnas de las tablas de su cláusula FROM. Esto implica que la subconsulta se ejecuta una sola vez e independientemente de las sentencias anteriores. Cuando en una subconsulta, en su cláusula WHERE se utiliza una columna de una tabla que figura en la cláusula FROM de la sentencia anterior, se dice que la subconsulta está sincronizada o correlacionada.

3.7.2 Consultas sincronizadas o correlacionadas.

Cuando en una subconsulta se utiliza un nombre de columna sin calificar (sin indicar a que tabla pertenece) se interpreta que es de la primera tabla que contenga esta columna, la búsqueda se realiza primero en su propia cláusula FROM y después se va buscando en las cláusulas FROM de las sentencias anteriores.

Estas subconsultas se ejecuta una vez por cada una de las filas de la sentencia principal o anterior.

Ejemplo: Recuperar los empleados que tienen el salario máximo en cada departamento.

```
SELECT nomempl, nomdept, salario
FROM empleado E, departamento D
WHERE E.coddept = D.coddept AND
       salario = (SELECT MAX(salario)
                    FROM empleado
                   WHERE E.coddept = coddept);
```

Resultado:

NOMEMPL	NOMDEPT	SALARIO
MORENO, LUIS	DIRECCIÓN	9000
GARCÍA, PEDRO	FINANZAS	4800
MARTIN, DIANA	COMERCIAL	2850
PEREZ, CARMEN	ORGANIZACIÓN	6000
DIEZ, CLARA	PRODUCCIÓN	5975

Otra forma de realizar esta consulta, es:

```
SELECT nomempl, nomdept, salario
FROM empleado E, departamento D
WHERE E.coddept = D.coddept AND
      (E.coddept,salario) IN
                     (SELECT CODDEPT,MAX(salario)
                        FROM empleado
                       GROUP BY coddept;
```

Resultado:

NOMEMPL	NOMDEPT	SALARIO
MORENO, LUIS	DIRECCIÓN	9000
GARCÍA, PEDRO	FINANZAS	4800
MARTIN, DIANA	COMERCIAL	2850
PEREZ, CARMEN	ORGANIZACIÓN	6000
DIEZ, CLARA	PRODUCCIÓN	5975

3.7.3 EXISTS y su subconsulta sincronizada.

Con EXITS se seleccionan filas de la consulta externa si existen filas en la consulta interna.

Ejemplo: Recuperar los nombres de los departamentos que están localizados en Madrid.

```
SELECT nomdept
  FROM departamento d
 WHERE EXISTS
        (SELECT *
           FROM empleado
          WHERE coddept = D.coddept
            AND codlugt IN (SELECT codlugt
                              FROM lugartrabajo
                             WHERE localid = 'Madrid'));
```

Resultado:

```
NOMDEPT
--------------
DIRECCIÓN
FINANZAS
COMERCIAL
```

3.7.4 Productos externos (Outer joins).

Cuando se combinan dos tablas para recuperar datos de cada una de ellas, en el resultado sólo se incluyen las filas que tienen datos en las columnas que sirven de unión de las dos tablas.

Ejemplo: Recuperar los nombres de los empleados con su lugar de trabajo.

```
SELECT nomempl, nomlugt
  FROM empleado e, lugartrabajo l
 WHERE e.codlugt = l.codlugt;
```

Resultado: .

```
NOMEMPL              NOMLUGT
-------------------- ----------
MORENO, LUIS         CENTRAL
GARCÍA, PEDRO        CENTRAL
LASA, FRANCÍSCO      CENTRAL
MARTIN, DIANA        CENTRAL
LLANOS, CRISTINA     CENTRAL
GIL, ANA             CENTRAL
SANZ, PEDRO          CENTRAL
PEREZ, CARMEN        FABRICA
MORENO, VICENTE      FABRICA
TORRES, SANCHO       FABRICA
GONZALEZ, JUAN       FABRICA
DIEZ, CLARA          FABRICA
DURAN, ADRIAN        FABRICA
```

En el resultado no figuran los empleados que en la columna del código del lugar de trabajo no tiene valor (NULL).

Ejemplo: Recuperar los nombres de los empleados sin código de lugar de trabajo.

```
SELECT nomempl, codlugt
  FROM empleado
 WHERE codlugt IS NULL;
```

Resultado:

```
NOMEMPL              CODLUGT
-------------------- ---------
RUIZ, MARÍA
FERNANDEZ, JOSÉ
```

La solución a esta situación son los productos externos (**Outer joins**).

El producto externo se indica con (+) a continuación de la columna de unión de la tabla más pequeña.

Ejemplo: Recuperar los nombres de los empleados con su lugar de trabajo.

```
SELECT nomempl, nomlugt
  FROM empleado e, lugartrabajo l
 WHERE e.codlugt = l.codlugt(+);
```

Resultado: Se recuperan los empleados que no tiene asignado lugar de trabajo.

```
NOMEMPL              NOMLUGT
-------------------- ----------
MORENO, LUIS         CENTRAL
GARCÍA, PEDRO        CENTRAL
LASA, FRANCÍSCO      CENTRAL
MARTIN, DIANA        CENTRAL
LLANOS, CRISTINA     CENTRAL
GIL, ANA             CENTRAL
SANZ, PEDRO          CENTRAL
PEREZ, CARMEN        FABRICA
MORENO, VICENTE      FABRICA
TORRES, SANCHO       FABRICA
GONZALEZ, JUAN       FABRICA
DIEZ, CLARA          FABRICA
DURAN, ADRIAN        FABRICA
RUIZ, MARÍA
FERNANDEZ, JOSÉ
```

En el resultado aparecen las dos filas de los empleados que no tienen código de lugar de trabajo.

3.7.5 Resumen

Las subconsultas sincronizadas o correlacionadas se ejecutan una vez por cada fila de la consulta principal, al contrario de la subconsultas normales que solo se ejecutan una vez y de forma independiente.

La cláusula EXISTS permite seleccionar filas en una consulta externa si existen filas en la subconsulta sincronizada.

Los productos externos de tablas permiten recuperar las filas de la tabla mayor que no tienen su correspondiente en la tabla más pequeña.

3.7.6 Ejercicios

Ejercicio1: Hallar los nombres de los departamentos en los que hay algún empleado que cumple este año mas de 50 años.

Ejercicio2: Obtener en orden alfabético los nombres de los empleados cuyo salario supera al salario medio de su departamento.

Ejercicio3: Obtener en orden alfabético los nombres de los departamentos que tienen administrativos.

Ejercicio4: Recuperar los empleados que trabajan en Toledo y que tienen un salario menor al medio de la compañía.

3.8 Consultas compuestas

3.8.1 Objetivo

En este apartado se van a utilizar unos operadores que permiten combinar información del mismo tipo, que se encuentra en diferentes tablas. Estos operadores son: UNION, UNION ALL INTERSECT y MINUS.

Estos operadores combinan resultados de consultas en un resultado final. Los nombres de las columnas seleccionadas pueden ser distintos en cada consulta, pero el número y el tipo de columnas debe ser el mismo.

3.8.2 UNION

Este operador combina las filas de las consultas eliminando las duplicadas.

Ejemplo: Recuperar las filas comunes a las tablas empleado y bajas.

```
SELECT numempl, nomempl
  FROM empleado
 UNION
SELECT numempl, nomempl
  FROM bajas;
```

Resultado:

```
  NUMEMPL NOMEMPL
--------- --------------------
     1010 GARCÍA , FERNANDO
     1011 DIEZ, CARLOS
     1015 LLANOS, CRISTINA
     1021 LOPEZ, DIEGO
     1100 MORENO, LUIS
     1200 HERNANDEZ,CARMEN
     1278 SAN JUAN, VICENTE
     1500 PEREZ, CARMEN
     1578 MORENO, VICENTE
     2001 DIEZ, CLARA
     2011 DURAN, ADRIAN
     2015 GARCÍA, PEDRO
     2120 GONZALEZ, JUAN
     2121 TORRES, SANCHO
     2134 LASA, FRANCISCO
     2167 SANZ, PEDRO
     2190 RUIZ, MARÍA
     2345 MARTIN, DIANA
     2456 FERNANDEZ, JOSÉ
     2565 GIL, ANA
```

Para ordenar el resultado se utiliza ORDER BY pero no se hace referencia a nombres de columnas sino al número de orden de la columna por la que se quiere ordenar.

```
SELECT numempl,nomempl
  FROM empleado
 UNION
SELECT numempl,nomempl
  FROM bajas
 ORDER BY 1;
```

3.8.3 UNION ALL

Este operador combina las filas de las consultas sin eliminar las duplicadas.

Ejemplo: Recuperar las filas de las tablas empleado y bajas.

```
SELECT numempl, nomempl
  FROM empleado
 UNION ALL
SELECT numempl, nomempl
  FROM bajas;
```

Esta consulta recupera dos veces la fila con numempl 1015, ya que existe en las dos tablas.

3.8.4 INTERSECT

Este operador recupera las filas comunes de las consultas. El orden de las consultas no afecta al resultado.

Ejemplo: Recuperar las filas comunes de las tablas empleado y bajas.

```
SELECT numempl, nomempl
  FROM empleado
 INTERSECT
SELECT numempl, nomempl
  FROM bajas;
```

Resultado:

```
  NUMEMPL NOMEMPL
--------- --------------------
     1015 LLANOS, CRISTINA
```

3.8.5 MINUS

Este operador recupera las filas de la primera consulta que no existen en la segunda. El orden de las consultas afecta al resultado.

Ejemplo: Recuperar los empleados que sólo figuran en tabla de bajas.

```
SELECT numempl, nomempl
  FROM bajas
 MINUS
SELECT numempl, nomempl
  FROM empleado;
```

Resultado:

```
  NUMEMPL NOMEMPL
--------- --------------------
     1010 GARCÍA , FERNANDO
     1011 DIEZ, CARLOS
     1021 LOPEZ, DIEGO
     1200 HERNANDEZ,CARMEN
```

```
     1278 SAN JUAN, VICENTE
```

Si se modifica el orden de las SELECT el resultado es diferente.

Ejemplo: Recuperar Empleados que no han sido dados de baja.

```
SELECT numempl, nomempl
  FROM empleado
 MINUS
SELECT numempl, nomempl
  FROM bajas;
```

Resultado:

```
  NUMEMPL NOMEMPL
--------- --------------------
     1100 MORENO, LUIS
     1500 PEREZ, CARMEN
     1578 MORENO, VICENTE
     2001 DIEZ, CLARA
     2011 DURAN, ADRIAN
     2015 GARCÍA, PEDRO
     2120 GONZALEZ, JUAN
     2121 TORRES, SANCHO
     2134 LASA, FRANCISCO
     2167 SANZ, PEDRO
     2190 RUIZ, MARÍA
     2345 MARTIN, DIANA
     2456 FERNANDEZ, JOSÉ
     2565 GIL, ANA
```

3.8.6 Resumen

Por medio de los operadores: UNION, UNION ALL INTERSECT y MINUS se combina información del mismo tipo, que se encuentra en diferentes tablas.

UNION - Combina las filas de las consultas eliminando las duplicadas.

UNION ALL - Combina las filas de las consultas sin eliminar las duplicadas.

INTERSECT - Recupera las filas comunes de las consultas. El orden de las consultas no afecta al resultado.

MINUS - Recupera las filas de la primera consulta que no existen en la segunda. El orden de las consultas afecta al resultado.

3.9 Árboles

3.9.1 Objetivo

En este apartado se van a utilizar dos cláusulas de la sentencia **SELECT** que permiten recuperar las filas seleccionadas en orden jerárquico desde tablas que contienen información de estructura de empresas, arboles genealógicos de familias, componentes de productos, etc.

3.9.2 CONNECT BY y START WICH

Con **star wich** se Indica la condición para seleccionar la fila de inicio del árbol.

Con **Connect by** se Indica la condición entre la fila padre y las filas hijo. Esta condición debe incluir el operador **PRIOR**.

Existen dos formas para la condición:

PRIOR padre operador hijo	Busca hacia abajo en el árbol.
Padre operador PRIOR hijo	Busca hacia arriba en el árbol.

Ejemplo: Recuperar todos los empleados en orden jerárquico. Comenzando por el empleado que no tiene valor en la columna 'jefe'.

```
SELECT numempl, jefe, level
    FROM empleado
   WHERE dgsitlab <> 'J'
   START WITH jefe IS NULL
 CONNECT BY PRIOR numempl=jefe;
```

Resultado:

NUMEMPL	JEFE	LEVEL
1100		1
2001	1100	2
1578	2001	3
2121	1578	4
2456	1578	4
2120	2001	3
2015	1100	2
2134	2015	3
2190	1100	2
1500	2190	3
2011	1500	4
2345	1100	2
2167	2345	3
2565	1100	2

En las consultas de este tipo se genera una pseudo-columna llamada LEVEL. En esta se asigna 1 a la fila raíz, 2 a las filas hijas de la anterior y así sucesivamente.

En el ejemplo siguiente se utiliza la LEVEL para genera la columna Empleado de forma que dé información de la estructura recuperada.

Ejemplo: Recuperar todos los empleados en orden jerárquico. Comenzando por el empleado que no tiene valor en la columna 'jefe'.

Utilizando la función LPAD con LEVEL se indica que se tome una cadena de un espacio y se rellene por la izquierda por el número de espacios determinado por **`3*(level-1)`**. Esto indica cuantos espacios se concatenan a la izquierda de la columna nomempl.

```
SELECT SUBSTR(LPAD(' ',3*(level-1))||
              nomempl,1,28)  Empleado,
       jefe
  FROM empleado
 WHERE dgsitlab <> 'J'
 START WITH jefe IS NULL
CONNECT BY PRIOR numempl=jefe;
```

Resultado: La función SUBSTR limita la longitud de la columna a 28 caracteres.

```
EMPLEADO                     JEFE
---------------------------  ---------
MORENO, LUIS
   DIEZ, CLARA               1100
      MORENO, VICENTE        2001
         TORRES, SANCHO      1578
            FERNANDEZ, JOSÉ  1578
         GONZALEZ, JUAN      2001
   GARCÍA, PEDRO             1100
      LASA, FRANCISCO        2015
   RUIZ, MARÍA               1100
      PEREZ, CARMEN          2190
         DURAN, ADRIAN       1500
   MARTIN, DIANA             1100
      SANZ, PEDRO            2345
   GIL, ANA                  1100
```

3.9.3 Eliminación de ramas del árbol y filas

Se pueden eliminar filas del árbol de dos formas: utilizando una condición (**!=** no igual) en la cláusula WHERE o en la cláusula CONNECT BY. Con CONNECT BY se eliminan todas las filas de la rama que comienza con la condición indicada. Con WHERE se elimina sólo la fila indicada con la condición.

Ejemplo: Recuperar todos los empleados en orden jerárquico. Comenzando por el empleado que no tiene valor en la columna 'jefe', eliminando la rama desde el empleado que tiene de jefe a 2190.

```
SELECT SUBSTR(LPAD(' ',3*(level-1))||
              nomempl,1,28)  Empleado,
       jefe
  FROM empleado
 WHERE dgsitlab <> 'J'
 START WITH jefe IS NULL
CONNECT BY PRIOR numempl = jefe AND
                 Jefe   != 2190 ;
```

Resultado: La función SUBSTR limita la longitud de la columna a 28 caracteres.

```
EMPLEADO                     JEFE
---------------------------- ---------
MORENO, LUIS
   DIEZ, CLARA                1100
      MORENO, VICENTE         2001
         TORRES, SANCHO       1578
         FERNANDEZ, JOSÉ      1578
      GONZALEZ, JUAN          2001
   GARCÍA, PEDRO              1100
      LASA, FRANCISCO         2015
   RUIZ, MARÍA                1100
   MARTIN, DIANA              1100
      SANZ, PEDRO             2345
   GIL, ANA                   1100
```

Con WHERE se elimina sólo la fila indicada en la condición.

```
SELECT SUBSTR(LPAD(' ',3*(level-1))||
              nomempl,1,28)  Empleado,
       jefe
  FROM empleado
 WHERE dgsitlab <> 'J' AND
       (jefe != 2190 or
        jefe is null)
 START WITH jefe IS NULL
CONNECT BY PRIOR numempl=jefe;
```

Resultado:

```
EMPLEADO                     JEFE
---------------------------- ---------
MORENO, LUIS
   DIEZ, CLARA                1100
      MORENO, VICENTE         2001
         TORRES, SANCHO       1578
         FERNANDEZ, JOSÉ      1578
      GONZALEZ, JUAN          2001
   GARCÍA, PEDRO              1100
      LASA, FRANCISCO         2015
   RUIZ, MARÍA                1100
         DURAN, ADRIAN        1500
   MARTIN, DIANA              1100
      SANZ, PEDRO             2345
   GIL, ANA                   1100
```

3.9.4 Conexión hacia arriba

Si se coloca la palabra PRIOR al otro lado de la condición de CONNECT BY se recorre la estructura en sentido contrario.

```
  SELECT SUBSTR(LPAD(' ',4*(4-level))||
                nomempl,1,25)  Empleado,
         SUBSTR(LPAD(' ',2*(4-level))||
                nomclab,1,20) Trabajo,
         jefe
    FROM empleado
   WHERE dgsitlab <> 'J'
   START WITH nomclab LIKE 'PRO%'
          OR nomclab LIKE 'ADM%'
 CONNECT BY numempl= PRIOR jefe;
```

Resultado:

```
EMPLEADO                  TRABAJO                   JEFE
------------------------- ------------------- ---------
            DURAN, ADRIAN       PROGRAMADOR         1500
        PEREZ, CARMEN         ANALISTA              2190
    RUIZ, MARÍA             ECONOMISTA            1100
MORENO, LUIS              INGENIERO
            SANZ, PEDRO         ADMINISTRATIVO      2345
        MARTIN, DIANA         VENDEDOR              1100
    MORENO, LUIS            INGENIERO
```

3.9.5 Orden de ejecución de las cláusulas de SELECT.

Cuando la sentencia SELECT contiene estas cláusulas, la ejecución se realiza en el orden siguiente:

- Selecciona las filas con la cláusula WHERE.
- Agrupa las filas según la cláusula GROUP BY.
- Inicia la estructura con START WICH.
- Genera la estructura con CONNECT BY.
- Ordena las filas con ORDER BY.

3.9.6 Resumen

Con las cláusulas **CONNECT BY** y **START WICH** se recuperan las filas seleccionadas en orden jerárquico.

Si la palabra PRIOR de la cláusula CONNECT BY se sitúa junto a la columna hijo, el árbol se genera desde la raíz a los hijos. Si PRIOR se sitúa junto a la columna padre, el árbol se genera desde los hijos al padre.

Con la cláusula WHERE se eliminan filas del árbol pero no sus hijos y con la cláusula CONNECT BY se eliminan ramas completas del árbol generado.

3.10 Manipulación de datos

3.10.1 Objetivo

En este apartado se van a utilizar las sentencias INSERT, DELETE y UPDATE que permiten modificar el contenido de las tablas añadiendo o borrando filas y modificando los valores de las columnas de las filas existentes.

3.10.2 INSERT

Con esta sentencia se insertan filas en una tabla.

Ejemplo: Insertar una fila en la tabla de empleados.

```
INSERT INTO empleado
  VALUES (7731, 'RUIZ, FERNANDO');
```

En la cláusula VALUES se indican los valores a insertar separados por comas y en el orden que tienen las columnas dentro de la tabla.

Los valores de las columnas alfanuméricas y las columnas de fechas deben ir entre apóstrofes.

Ejemplo: Insertar una fila en la tabla de departamentos.

```
INSERT INTO departamento (nomdept, coddept)
       VALUES ('DISEÑO', 45).
```

Si se indican los nombres de las columnas, los valores de la cláusula VALUES deben ir en el mismo orden.

Para insertar columnas de fechas, se utiliza la función TO_DATE indicando la máscara del formato de la fecha. Si no se indica la hora se toma por omisión 12 A.M. (Medianoche).

Ejemplo: Insertar una fila en la tabla de departamentos.

```
INSERT INTO empleado (numempl,nomempl,nomclab,jefe,
                      fchnac,
                      fchingr,
                      salario,comision,coddept)
    VALUES (7731, 'RUIZ, FERNANDO',NULL,2001,
              TO_DATE('20/06/1953','DD/MM/YYYY'),
              TO_DATE('06/08/2000','DD/MM/YYYY'),
              3500,NULL,20);
```

Se puede insertar filas seleccionadas desde una tabla con una sentencia SELECT en otra tabla.

Ejemplo: Insertar en la tabla empledado2 las filas de empleados que tienen un salario mayor de 3500.

```
INSERT INTO empleado2
     SELECT  *
       FROM EMPLEADOS
      WHERE SALARIO > 3500;
```

Las dos tablas deben tener las mismas columnas y estar definidas en el mismo orden.

3.10.3 DELETE

Con esta sentencia se borran filas en una tabla. La cláusula WHERE es necesaria para eliminar sólo las filas deseadas.

Ejemplo: Borrar todas las filas de la tabla empleados.

```
DELETE FROM empleados;
```

Ejemplo: Borrar los empleados del departamento 40, que tienen una comisión < 20.

```
DELETE
  FROM empleado
 WHERE coddept  = 40 AND
       Comision < 20;
```

3.10.4 UPDATE

Con esta sentencia se actualizan las filas seleccionadas de una tabla, con los valores asignados a las columnas en la cláusula SET.

Las filas se seleccionan con la cláusula WHERE.

Ejemplo: Quitar la comisión y aumentar el salario un 10 % a los empleados que tienen una comisión < 10.

```
UPDATE empleado
   SET comision = NULL,
       salario = salario * 1.1
 WHERE comision < 10;
```

3.10.5 COMMIT y ROLLBACK.

Con estas sentencias del grupo de Control de Transacción se confirman o deshacen las modificaciones realizadas a los datos dentro de una unidad de trabajo llamada transacción.

Una transacción comienza con el inicio de una sesión o mediante la última sentencia COMMIT o ROLLBACK ejecutada y termina con el fin de la sesión o con la ejecución de una sentencia COMMIT o ROOLBACK.

3.10.5.1 COMMIT

Las modificaciones no se guardan en la base de datos de una forma definitiva hasta que no se validan con un COMMIT.

Mientras un usuario no valida una transacción(COMMIT) sólo él ve los datos actualizados. Cualquier otro usuario que tenga acceso a la tabla, recuperará los datos sin actualizar guardados en la base de datos.

3.10.5.2 ROLLBACK

Con ROLLBACK se deshacen todas las modificaciones realizadas por el usuario desde el anterior COMMIT.

3.10.5.3 Commit implícito y rollback automático

Cuando se realizan alguna de las siguientes acciones se realiza un commit implícito: QUIT, EXIT, CREATE TABLE, DROP TABLE, CREATE VIEW, DROP VIEW, CONNECT, DISCONNECT,GRANT, REVOKE y ALTER.

Si se produce un fallo en el ordenador y existen actualizaciones sin validar, el SGBD hace un rollback automático (deshace las actualizaciones sin validar) cuando arranca de nuevo.

3.10.6 Resumen

Con las sentencias SQL: INSERT, UPDATE y DELETE, se modifica el contenido de las tablas de la base de datos. Hay que tener cuidado con las condiciones de la cláusula WHERE de las sentencias UPDATE y DELETE, para que las actualizaciones **sólo** afecten a las filas deseadas.

Las modificaciones no se guardan en la base de datos de una forma definitiva hasta que no se validan con un COMMIT.

Con la sentencia ROLLBACK se pueden deshacer las modificaciones realizas desde el anterior COMMIT.

3.11 Tratamiento de tablas y vistas

3.11.1 Objetivo

En este apartado se van a utilizar las sentencias SQL que permiten el tratamiento de tablas y vistas. Estas sentencias, que pertenecen al grupo DDL(Data Definition Language), son: CREATE, DROP, TRUNCATE y ALTER.

3.11.2 Creación de una tabla

Para crear una tabla se utiliza la sentencia:

```
CREATE TABLE empleado
       (numempl  NUMBER(4) NOT NULL,
        nomempl  VARCHAR(20),
        nomclab  VARCHAR(15),
        jefe     NUMBER(4),
        fchnac   DATE,
        fchingr  DATE,
        salario  NUMBER(7,2),
        comision NUMBER(3),
        coddept  NUMBER(2),
        codlugt  NUMBER(2),
        dgsitlab VARCHAR(1)
        );
```

En esta sentencia se indica el nombre de la tabla y las definiciones de cada una de las columnas.

Las definiciones de columnas se separan con comas, en las de tipo alfanumérico se indica la longitud, y en las de tipo numérico se puede indicar el número de dígitos y el número de decimales.

3.11.3 Creación de una tabla a partir de otra

Se puede crear una tabla utilizando una sentencia SELECT sobre otra tabla.

```
CREATE TABLE empleado2 AS
SELECT numempl, nomempl, nomclab,
       jefe, fchnac, fchingr,
       salariO, comision, coddept  ,
       codlugt, dgsitlab
  FROM empleado
 WHERE comisión IS NOT NULL;
```

La sentencia anterior además de crear la tabla con las columnas indicadas en la SELECT, inserta las filas seleccionadas. Por lo tanto si se quiere crear sólo la tabla se debe indicar en la cláusula WHERE una condición que no seleccione filas.

3.11.4 Supresión de tablas

Existen dos sentencias SQL que permiten borrar una tabla, son: DROP y TRUNCATE.

Las acciones realizadas con estas sentencia son definitivas, no se permite deshacer las modificaciones realizadas con ROLLBACK.

```
DROP TABLE empleado;
```

Esta sentencia borra el contenido de la tabla, libera el espacio ocupado y borra la definición de la tabla.

```
TRUNCATE TABLE empleado;
```

Esta sentencia borra el contenido de la tabla, libera el espacio ocupado pero no borra la definición de la tabla.

.

3.11.5 Modificación de tablas

En una tabla se pueden hacer dos tipos de modificaciones: cambiar la definición de las columnas o añadir columnas. Para hacer esto se utiliza la sentencia ALTER.

Ejemplo: Aumentar la longitud de la columna nomempl a 30 caracteres de la tabla empleado.

```
ALTER TABLE empleado
     MODIFY (
             nomempl VARCHAR(30)
            );
```

Ejemplo: Añadir la columna fhingdpt (fecha de ingreso en el departamento) en la tabla empleado.

```
ALTER TABLE empleado
     ADD (
            fhingdpt DATE
          );
```

3.11.5.1 Acciones permitidas en la modificación de tablas.

Las acciones permitidas son:

Siempre se puede añadir una columna.

Siempre se puede aumentar la longitud de una columna alfanumérica.

Siempre se puede aumentar el número de dígitos de una columna numérica.

Siempre se puede aumentar y disminuir el número de dígitos decimales de una columna numérica.

No se puede añadir una columna NOT NULL.

Además en columnas que no contienen datos, se puede:

Siempre se puede cambiar el tipo de dato.

Siempre se puede disminuir la longitud de una columna alfanumérica.

Siempre se puede disminuir el número de dígitos de una columna numérica.

3.11.6 Tratamiento de vistas

La creación de una vista (definición de una consulta guardada en la base de datos) ya se ha tratado en el apartado anterior 3.1 Consulta de datos.

En este apartado se van a tratar las restricciones que existen cuando se utilizan las vistas para actualizar las tablas en las que están basadas.

Si la vista está basada en una tabla, se puede utilizar en sentencias INSERT, DELETE Y UPDATE, con las restricciones siguientes:

- No se puede utilizar INSERT si la vista no contiene todas las columna NOT NULL de la tabla.
- No se puede utilizar INSERT O UPDATE si la vista contiene columnas con funciones o cálculos.
- No se puede utilizar INSERT, UPDATE y DELETE si la vista contiene GROUP BY O DISTINCT.

3.11.7 Resumen

El tratamiento de tablas se realiza con las sentencias del grupo DDL (Data Definition Language) CREATE TABLE, DROP TABLE, TRUNCATE TABLE y ALTER.TABLE.

Si una vista está basada en una tabla, se puede utilizar en sentencias INSERT, DELETE Y UPDATE, con algunas restricciones en la definición de la vista.

3.12 Tratamiento de índices de tablas.

3.12.1 Objetivo

En este apartado se van a utilizar las sentencias SQL que permiten el tratamiento de índices. Estas sentencias, que pertenecen al grupo DDL(Data Definition Language), son: CREATE, DROP y ALTER

Un índice de una tabla se genera con los datos de una o varias de sus columnas y la identificación de cada una de las filas.

Aunque no son necesarios para el funcionamiento del SGBD, si existen, aumentan la velocidad de ejecución de las consultas, sobre todo si las tablas son de gran tamaño y las columnas del índice aparecen en la cláusula WHERE en condiciones simples o en combinaciones de tablas. Si en una consulta no tiene cláusula WHERE, los índices no son utilizados por SGBD

También se utilizan los índices para garantizar la unicidad de valores en la columna o columnas que lo componen.

3.12.2 Creación de un índice

Para crear un índice se utiliza la sentencia:

```
CREATE INDEX ind_emp ON EMPLEADO
       (numempl);
```

En esta sentencia se indica el nombre del índice, la tabla sobre la que se crea y la columna o columnas con las que se genera.

.

3.12.3 Creación de un índice único

Para crear un índice único se utiliza la sentencia:

```
CREATE UNIQUE INDEX ind_emp_u ON EMPLEADO
       (numempl);
```

Este índice impide que en la tabla de empleado existan dos filas con el mismo valor en la columna numempl.

Si al crear un índice único, la tabla contiene filas con el mismo valor en la columna(columnas) con la que se quiere generar, el SGBD da un error indicando que el índice no puede ser creado.

3.12.4 Regeneración de índices

Un índice puede ser recreado sin tener que borrar el índice se realiza con la sentencia ALTER.

```
ALTER INDEX ind_emp REBUILD;
```

Esta sentencia genera el índice de forma más rápida.

3.12.5 Supresión de índices

El borrado de un índice se realiza con la sentencia DROP.

```
DROP INDEX ind_emp;
```

Esta sentencia borra el contenido del índice, libera el espacio ocupado y borra la definición del índice.

.

3.12.6 Resumen

El tratamiento de índices se realiza con las sentencias del grupo DDL (Data Definition Language) CREATE INDEX, DROP INDEX y ALTER INDEX.

Los índices se utilizan para mejorar los tiempos de respuesta de las consultas en las que se seleccionan filas o se combinan tablas y para asegurar la unicidad de valores en las columnas con las que se generan.

3.13 Tratamiento de Privilegios.

En este apartado se van a utilizar las sentencias SQL del grupo DDL (Data Definition Language) que permiten a los usuarios conceder y retirar privilegios sobre sus tablas y vistas. Estas sentencias son: GRANT Y REVOKE.

3.13.1 Objetivo

En este apartado se van a utilizar las sentencias SQL del grupo DDL (Data Definition Language) que permiten a los usuarios conceder y retirar privilegios sobre sus tablas y vistas. Estas sentencias son: GRANT Y REVOKE.

3.13.2 Asignación de privilegios

Se realiza con la sentencia GRANT.

Notación: GRANT [(]privilegio [,privilegio)] [(columna[,columna])] ON tabla /vista TO usuario

Los privilegios que un usuario puede conceder de sus tablas y vistas son:

- INSERT – Insertar filas.
- UPDATE – Actualizar todas o algunas columnas.
- DELETE – Borrar filas.
- SELECT – Consultar.

Además de las tablas se puede conceder el privilegio ALTER que permite hacer modificaciones en la tabla.

Ejemplo: Autorizar la consulta a la tabla empleado al usuario USU00x.

```
GRANT SELECT
   ON empleado
   TO USU00X;
```

Ejemplo: Autorizar la actualización de las columnas salario y comision de la tabla empleado al usuario USU00x.

```
GRANT UPDATE (SALARIO, COMISION)
   ON empleado
   TO USU00X;
```

Ejemplo: Autorizar la consulta a la tabla empleado a todos los usuarios.

```
GRANT SELECT
   ON empleado
   TO PUBLIC;
```

Ejemplo: Autorizar la consulta y actualización de las columnas salario y comision de la vista wtoledano al usuario USU00x.

```
GRANT SELECT, UPDATE (SALARIO, COMISION)
   ON wtoledano
   TO USU00X;
```

3.13.3 Retirada de privilegios

Se realiza con la sentencia REVOKE.

Notación: REVOKE privilegio [,privilegio] ON tabla /vista FROM [()]usuario[,usuario)]

Ejemplo: Eliminar la autorización INSERT al usuario USU00x en la tabla empleado.

```
REVOKE INSERT
    ON empleado
  FROM USU00X;
```

Ejemplo: Eliminar todos los privilegios al usuario USU00x en la tabla empleado.

```
REVOKE ALL
    ON empleado
  FROM USU00X;
```

3.13.4 Creación de un sinónimo

Cuando un usuario utiliza una tabla o una vista de otro usuario, debe utilizar el nombre completo 'código usuario punto nombre de tabla/vista'. Para simplificar esta notación el usuario puede crear un nombre alternativo llamado sinónimo.

Se realiza con la sentencia CREATE SYNONYM.

Notación: CREATE SYNONYM sinónimo FOR usuario.tabla/vista

Ejemplo : Crear un sinónimo sy_empl a la tabla empleado del usuario USU00x.

```
CREATE SINONYM sy_empl
       FOR USU001.empleado;
```

Se pueden crear sinónimos públicos accesibles a todos los usuarios.

Ejemplo: Crear un sinónimo syp_empl publico a la tabla empleado del usuario USU00x .

```
CREATE PUBLIC SINONYM syp_empl
        FOR empleado;
```

3.13.5 Resumen

Los usuarios pueden conceder y retirar privilegios (SELECT, UPDATE, DELETE INSERT, etc.) sobre sus tablas y vistas a otros usuarios.

Con la sentencia GRANT se conceden y con la sentencia REVOKE se retiran.

Además los usuarios pueden asignar otro nombre a sus tablas, por medio de la sentencia CREATE SYNONYM.

3.13.6 Ejercicio

Realizar las acciones siguientes con dos usuarios y en el orden indicado.

El usuario USUXXX:

- Permitir al usuario USUYYY la consulta y la actualización en su tabla empleado.
- Permitir al usuario USUYYY la consulta y la actualización en su vista wtoledano.

El usuario USUYYY:

- Crear una vista wYYYempleado sobre la tabla USUXXX.empleado seleccionando los empleados del departamento 50.
- Realizar consultas y actualizaciones de la tabla autorizada y de las dos vistas.

El usuario USUXXX:

- Retira todos los privilegios sobre su tabla empleado al usuarios USU YYY.

El usuario USUYYY:

- Realizar consultas y actualizaciones de la tabla autorizada y de las dos vistas.

El usuario USUYYY:

- Crear una vista wYYYtoledano sobre la vista USUXXX.wtoledano.
- Realizar consultas de la vista.

4 Anexos

4.1 Descripción de las tablas utilizadas

4.1.1 Tabla de Empleados

Nombre: empleado

Nombre de columna	Descripción
NUMEMPL	Número de empleado
NOMEMPL	Nombre (apellido, nombre)
NOMCLAB	Nombre de categoría laboral
JEFE	Numero de su jefe
FCHNAC	Fecha de nacimiento
FCHINGR	Fecha de ingreso en la compañía
SALARIO	Salario en miles de pesetas
COMISION	Comisión en tanto por ciento
CODDEPT	Código departamento
CODLUGT	Código del lugar de trabajo
DGSITLAB	Dígito de situación laboral

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
NUMEMPL	NOT NULL	NUMBER(4)
NOMEMPL		VARCHAR(20)
NOMCLAB		VARCHAR(15)
JEFE		NUMBER(4)
FCHNAC		DATE
FCHINGR		DATE
SALARIO		NUMBER(7,2)
COMISION		NUMBER(3)
CODDEPT		NUMBER(2)
CODLUGT		NUMBER(2)
DGSITLAB		VARCHAR(1)

Contenido:

NUMEMPL	NOMEMPL	NOMCLAB	JEFE	FCHNAC	FCHINGR	SALARIO	COMISION	CODDEPT	CODLUGT	D
1100	MORENO, LUIS	INGENIERO		28/03/48	17/11/81	9000		10	91	A
1500	PEREZ, CARMEN	ANALISTA	2190	15/06/54	09/12/82	6000		40	92	A
1578	MORENO, VICENTE	TÉCNICO	2001	30/09/52	03/03/83	4000		50	92	A
2001	DIEZ, CLARA	INGENIERO	1100	23/07/60	02/04/88	5975		50	92	A
2011	DURAN, ADRIAN	PROGRAMADOR	1500	19/11/65	20/09/88	3600		40	92	A
2015	GARCÍA, PEDRO	ECONOMISTA	1100	10/01/60	21/06/89	4800	30	20	91	A
2120	GONZALEZ, JUAN	TÉCNICO	2001	13/01/64	09/03/89	3450		50	92	A
2121	TORRES, SANCHO		1578	29/03/66	23/01/90			50	92	A
2134	LASA, FRANCISCO	OFICIAL	2015	25/08/67	22/03/90	2250		20	91	A
2167	SANZ, PEDRO	ADMINISTRATIVO	2345	10/07/69	12/09/90	2100		30	91	A
2190	RUIZ, MARÍA	ECONOMISTA	1100	11/05/68	08/12/90	3124	20	40		M
2345	MARTIN, DIANA	VENDEDOR	1100	12/09/70	28/10/95	2850	21	30	91	A
2456	FERNANDEZ, JOSÉ	DELINEANTE	1578	02/08/72	01/05/97	2250		50		A
2565	GIL, ANA	SECRETARIA	1100	12/05/74	03/02/99	1950		10	91	A
1015	LLANOS, CRISTINA	SECRETARIA		10/01/44	21/06/66	800		20	91	J

4.1.2 Tabla de Departamentos

Nombre: departamento.

Nombre de columna	Descripción
CODDEPT	Código de departamento
NOMDEPT	Nombre de departamento

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
CODDEPT	NOT NULL	NUMBER(2)
NOMDEPT		VARCHAR(14)

Contenido:

CODDEPT	NOMDEPT
10	DIRECCIÓN
20	FINANZAS
30	COMERCIAL
40	ORGANIZACIÓN
50	PRODUCCIÓN

4.1.3 Tabla de Lugares de Trabajo

Nombre: lugartrabajo.

Nombre de columna	Descripción
CODLUGT	Código lugar de trabajo
NOMLUGT	Nombre
LOCALID	Localidad
DIRECC	Dirección

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
CODLUGT	NOT NULL	NUMBER(2)
NOMLUGT		VARCHAR(10)
LOCALID		VARCHAR(15)
DIRECC		VARCHAR(35)

Contenido:

```
CODLUGT NOMLUGT  LOCALID  DIRECC
------- -------- -------- --------------------------------
     91 CENTRAL  Madrid   C. Antonio López, 125, - DP28044
     92 FABRICA  Toledo   C. Santa Clara, 22, - DP1037
```

4.1.4 Tabla de Bajas

Nombre: bajas.

Nombre de columna	Descripción
NUMEMPL	Número de empleado
NOMEMPL	Nombre (apellido, nombre)
NOMCLAB	Nombre de categoría laboral
FCHNAC	Fecha de nacimiento
FCHINGR	Fecha de ingreso en la compañía
SALARIO	Salario en miles de pesetas
COMISION	Comisión en tanto por ciento
CODDEPT	Código departamento
CODLUGT	Código del lugar de trabajo
FCHBAJA	Fecha de baja

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
NUMEMPL	NOT NULL	NUMBER(4)
NOMEMPL		VARCHAR(20)
NOMCLAB		VARCHAR(15)
FCHNAC		DATE
FCHINGR		DATE
SALARIO		NUMBER(7,2)
COMISION		NUMBER(3)
CODDEPT		NUMBER(2)
CODLUGT		NUMBER(2)
FCHBAJA		DATE

Contenido:

NUMEMPL	NOMEMPL	NOMCLAB	FCHNAC	FCHINGR	SALARIO	COMISION	CODDEPT	CODLUGT	FCHBAJA
1010	GARCÍA , FERNANDO	INGENIERO	28/03/38	17/11/65	2000		10	91	28/03/98
1200	HERNANDEZ,CARMEN	ADMINSTRATIVO	15/06/34	09/12/60	1000		30	92	15/06/94
1278	SAN JUAN, VICENTE	TÉCNICO	30/09/42	03/03/63	1300		50	92	30/11/98
1011	DIEZ, CARLOS	OFICIAL	23/07/32	02/04/60	5975		50	92	23/10/95
1021	LOPEZ, DIEGO		19/11/35	20/09/55	3600		20	92	25/01/97
1015	LLANOS, CRISTINA	SECRETARIA	10/01/44	21/06/66	800		20	91	10/01/99

4.1.5 Tabla de Situación Laboral

Nombre: tsitlab.

Nombre de columna	Descripción
COSITLAB	Código situación anual
NOMSITLAB	Nombre situación anual

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
COSITLAB	NOT NULL	VARCHAR(1)
NOMSITLAB		VARCHAR(10)

Contenido:

```
C NOMSITLAB
- ----------
A Activo
M Maternidad
J Jubilado
E Enfermedad
```

4.1.6 Tabla Dummy

Nombre: dummy.

Definición de columnas:

Nombre de columna	Null?	Tipo de dato
DUMMY		NUMBER

4.2 Soluciones a los ejercicios

4.2.1 Consultas de datos

Ejercicio1:

Recuperar los empleados con las letras OR en la segunda y tercera posición del nombre.

```
 SELECT numempl, nomempl
   FROM empleado
  WHERE nomempl LIKE '_OR%';
```

Resultado:

```
   NUMEMPL NOMEMPL
--------- --------------------
     1100 MORENO, LUIS
     1578 MORENO, VICENTE
     2121 TORRES, SANCHO
```

Ejercicio2:

Recuperar los empleados sin comisión de los departamentos 20 y 40.

```
SELECT nomempl,coddept,comision
  FROM empleado
 WHERE (coddept = 20 OR
        coddept = 40) AND
       comision IS NULL;
```

Resultado:

```
NOMEMPL               CODDEPT  COMISION
-------------------- --------- ---------
PEREZ, CARMEN              40
DURAN, ADRIAN              40
LASA, FRANCISCO            20
LLANOS, CRISTINA           20
```

Ejercicio3:

Recuperar los empleados de los departamentos que tienen un código menor que el correspondiente al departamento Comercial.

```
SELECT numempl,nomempl,coddept
  FROM empleado
           WHERE coddept < (SELECT coddept
                     FROM departamento
                    WHERE nomdept = 'COMERCIAL');
```

Resultado:

```
  NUMEMPL NOMEMPL                CODDEPT
--------- -------------------- ---------
     1100 MORENO, LUIS                10
     2015 GARCÍA, PEDRO               20
     2134 LASA, FRANCISCO             20
     2565 GIL, ANA                    10
```

Ejercicio4:

Recuperar los empleados con el nombre de su lugar de trabajo, el nombre de su departamento y su situación laboral.

```
SELECT nomempl,nomdept,nomlugt,nomsitlab
FROM empleado E,departamento D,lugartrabajo L,tsitlab S
WHERE E.coddept = D.coddept AND
      E.codlugt = L.codlugt AND
      E.dgsitlab = S.cositlab;
```

Resultado:

```
NOMEMPL              NOMDEPT         NOMLUGT    NOMSITLAB
-------------------- --------------- ---------- ----------
MORENO, LUIS         DIRECCIÓN       CENTRAL    Activo
GIL, ANA             DIRECCIÓN       CENTRAL    Activo
GARCÍA, PEDRO        FINANZAS        CENTRAL    Activo
LASA, FRANCISCO      FINANZAS        CENTRAL    Activo
LLANOS, CRISTINA     FINANZAS        CENTRAL    Jubilado
SANZ, PEDRO          COMERCIAL       CENTRAL    Activo
MARTIN, DIANA        COMERCIAL       CENTRAL    Activo
PEREZ, CARMEN        ORGANIZACIÓN    FABRICA    Activo
```

```
DURAN, ADRIAN          ORGANIZACIÓN   FABRICA    Activo
GONZALEZ, JUAN         PRODUCCIÓN     FABRICA    Activo
TORRES, SANCHO         PRODUCCIÓN     FABRICA    Activo
MORENO, VICENTE        PRODUCCIÓN     FABRICA    Activo
DIEZ, CLARA            PRODUCCIÓN     FABRICA    Activo
```

Ejercicio5:

Obtener los empleados del departamento de Martín que tengan su misma comisión.

```
SELECT nomempl,nomdept,comision,E.coddept
FROM empleado E, departamento D
WHERE E.coddept = D.coddept AND
       (E.coddept,comision) IN
                        (SELECT coddept,comision
                           FROM empleado
                          WHERE nomempl  LIKE 'MARTIN%' )
ORDER BY nomempl
```

Resultado:

```
 NOMEMPL              NOMDEPT          COMISION   CODDEPT
-------------------- -------------- --------- ---------
MARTIN, DIANA        COMERCIAL              21        30
```

4.2.2 Manipulación de texto

Ejercicio1:

Recuperar los empleados del departamento 40, rellenando hasta 20 caracteres por la izquierda con '_*'.

```
SELECT LPAD(nomempl,20,'_*')
  FROM empleado
 WHERE coddept = 40;
```

Resultado:

```
LPAD(NOMEMPL,20,'_*'
--------------------
_*_*_*_PEREZ, CARMEN
_*_*_*_DURAN, ADRIAN
_*_*_*_*_RUIZ, MARÍA
```

Ejercicio2:

Recuperar el apellido de los empleados del departamento 50, desde la columna nomempl.

```
SELECT nomempl, SUBSTR(nomempl,1,INSTR(nomempl,',')-1)
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
NOMEMPL              SUBSTR(NOMEMPL,1,INS
-------------------- --------------------
MORENO, VICENTE      MORENO
DIEZ, CLARA          DIEZ
GONZALEZ, JUAN       GONZALEZ
TORRES, SANCHO       TORRES
FERNANDEZ, JOSÉ      FERNANDEZ
```

Ejercicio3:

Recuperar el nombre de los empleados del departamento 50, desde la columna nomempl.

```
SELECT nomempl, SUBSTR(nomempl, INSTR(nomempl,',')+2)
  FROM empleado
 WHERE coddept = 50;
```

Resultado:

```
NOMEMPL              SUBSTR(NOMEMPL,INSTR
-------------------- --------------------
MORENO, VICENTE      VICENTE
DIEZ, CLARA          CLARA
GONZALEZ, JUAN       JUAN
TORRES, SANCHO       SANCHO
FERNANDEZ, JOSÉ      JOSÉ
```

Ejercicio4:

Obtener los tres últimos caracteres de los nombres de los departamentos ordenados alfabéticamente.

```
SELECT substr(nomdept,LENGTH(nomdept)-2)
 FROM DEPARTAMENTO
ORDER BY 1;
```

Resultado:

```
SUBSTR(NOMDEPT
--------------
IAL
IÓN
IÓN
IÓN
ZAS
```

4.2.3 Manipulación de números

Ejercicio1:

Hallar el número de empleados del departamento 50, el número de comisiones distintas, y la suma y la media de las comisiones.

```
SELECT COUNT(*), COUNT(DISTINCT comision),
       SUM(comision),AVG(comision)
 FROM empleado;
```

Resultado:

```
  COUNT(*) COUNT(DISTINCTCOMISION) SUM(COMISION) AVG(COMISION)
---------- ---------------------- ------------ -------------
        15                      3           71     23,666667
```

Ejercicio2:

Calcular la media de las comisiones.

```
SELECT SUM(comision)/COUNT(*)
  FROM empleado;
```

Resultado:

```
SUM(COMISION)/COUNT(*)
----------------------
             4,7333333
```

Ejercicio3:

Hallar la masa salarial anual (salario + comisiones) de la empresa.

```
SELECT SUM(salario),(SUM(salario) +
SUM(salario*comision/100))
  FROM empleado;
```

Resultado:

```
SUM(SALARIO) (SUM(SALARIO)+SUM(SALARIO*COMISION/100))
------------ ----------------------------------------
       52149                                  54812,3
```

4.2.4 Manipulación de fechas

Ejercicio1 :

Calcular los años de trabajo en la compañía.

```
SELECT nomempl, fchbaja, fchingr,
       TRUNC(MONTHS_BETWEEN(fchbaja,fchingr)/12) años
  FROM bajas;
```

Resultado:

```
NOMEMPL              FCHBAJA  FCHINGR       AÑOS
-------------------- -------- -------- ---------
GARCÍA , FERNANDO    28/03/98 17/11/65         32
HERNANDEZ,CARMEN     15/06/94 09/12/60         33
SAN JUAN, VICENTE    30/11/98 03/03/63         35
DIEZ, CARLOS         23/10/95 02/04/60         35
LOPEZ, DIEGO         25/01/97 20/09/55         41
LLANOS, CRISTINA     10/01/99 21/06/66         32
```

Ejercicio2:

Calcular los Días correspondientes al viernes y al lunes siguientes a la fecha contenida en SYSDATE.

```
SELECT SYSDATE,NEXT_DAY(SYSDATE,'viernes'),
               NEXT_DAY(SYSDATE,'lunes')
  FROM dual;
```

Resultado:

```
SYSDATE  NEXT_DAY NEXT_DAY
-------- -------- --------
21/06/00 23/06/00 26/06/00
```

Ejercicio3:

Recuperar el nombre del mes sin espacios en blanco, el ordinal del día en inglés y en minúscula (dd), y el año de la fecha de ingreso de los empleados de los departamentos con código menor de 30.

```
SELECT nomempl, fchingr,
       TO_CHAR(fchingr, 'fmMonth, ddth, YYYY')
  FROM empleado
 WHERE coddept < 30;
```

Resultado:

```
NOMEMPL              FCHINGR  TO_CHAR(FCHINGR,'FMMON
-------------------- -------- ----------------------
MORENO, LUIS         17/11/81 Noviembre, 17th, 1981
GARCÍA, PEDRO        21/06/89 Junio, 21st, 1989
LASA, FRANCISCO      22/03/90 Marzo, 22nd, 1990
GIL, ANA             03/02/99 Febrero, 3rd, 1999
LLANOS, CRISTINA     21/06/66 Junio, 21st, 1966
```

Ejercicio4:

Recuperar los empleados con código de departamento menor de 30 y su fecha de ingreso con el formato 'Ingresado el 17 de Noviembre de 1979, a las 12:30'.

```
SELECT nomempl,fchingr,
TO_CHAR(fchingr,'"Ingresado el "dd" de "fmMonth" de
"YYYY", a las "HH:MI')
  FROM empleado
 WHERE coddept < 30;
```

Resultado:

```
NOMEMPL              FCHINGR  TO_CHAR(FCHINGR,'"INGRESADOEL"DD"DE"FMMONTH"DE"YYY
-------------------- -------- ------------------------------------------------
MORENO, LUIS         17/11/81 Ingresado el 17 de Noviembre de 1981, a las 12:0
GARCÍA, PEDRO        21/06/89 Ingresado el 21 de Junio de 1989, a las 12:0
LASA, FRANCISCO      22/03/90 Ingresado el 22 de Marzo de 1990, a las 12:0
GIL, ANA             03/02/99 Ingresado el 03 de Febrero de 1999, a las 12:0
LLANOS, CRISTINA     21/06/66 Ingresado el 21 de Junio de 1966, a las 12:0
```

4.2.5 Funciones de transformación

Ejercicio1:

Recuperar los nombres de los empleados eliminando las vocales y cambiando las comas por guiones.

```
SELECT TRANSLATE (nomempl,',AEIOUÁÉÍÓÚ','-')
  FROM empleado
 WHERE coddept in (20,40);
```

Resultado:

```
TRANSLATE(NOMEMPL,',
--------------------
PRZ- CRMN
DRN- DRN
GRC- PDR
LS- FRNCSC
RZ- MR
LLNS- CRSTN
```

4.2.6 Agrupación de filas

Ejercicio1:

Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando las filas con salarios y las filas que hay por cada departamento. Seleccionando sólo los grupos con menos de tres filas por grupo.

```
SELECT coddept,SUM(salario),COUNT(salario),COUNT(coddept)
    FROM empleado
   WHERE codlugt in (SELECT codlugt
                       FROM lugartrabajo
                      WHERE localid = 'Madrid')
   GROUP BY coddept
  HAVING COUNT(coddept) < 3;
```

Resultado:

```
  CODDEPT SUM(SALARIO) COUNT(SALARIO) COUNT(CODDEPT)
--------- ------------ -------------- --------------
       10        10950              2              2
       30         4950              2              2
```

Ejercicio 2:

Agrupar los empleados que trabajan en Madrid, por su código de departamento, sumando los salarios y contando las filas con salarios y las filas que hay por cada departamento.

```
SELECT coddept,SUM(salario),COUNT(salario),COUNT(coddept)
    FROM empleado
   WHERE codlugt in (SELECT codlugt
                       FROM lugartrabajo
                      WHERE localid = 'Madrid')
   GROUP BY coddept
   ORDER BY SUM(salario);
```

Resultado:

```
  CODDEPT SUM(SALARIO) COUNT(SALARIO) COUNT(CODDEPT)
--------- ------------ -------------- --------------
       30         4950              2              2
       20         7850              3              3
       10        10950              2              2
```

4.2.7 Consultas dependientes

Ejercicio1:

Hallar los nombres de los departamentos en los que hay algún empleado que cumple este año mas de 50 años.

```
SELECT nomdept
  FROM departamento d
 WHERE EXISTS (SELECT *
                 FROM empleado
                WHERE cosdept = d.coddept
                  AND (TO_CHAR(SYSDATE,'YYYY') -
                       TO_CHAR(FCHNAC,'YYYY')) >= 50);
```

Resultado:

```
NOMDEPT
-----------
DIRECCIÓN
FINANZAS
```

Ejercicio2:

Obtener en orden alfabético los nombres de los empleados cuyo salario supera al salario medio de su departamento.

```
SELECT nomempl,salario
  FROM empleado E
 WHERE SALARIO > (SELECT AVG(salario)
                    FROM empleado
                   WHERE coddept = E.coddept)
ORDER BY nomempl;
```

Resultado:

```
NOMEMPL                SALARIO
-------------------- ---------
DIEZ, CLARA               5975
GARCÍA, PEDRO             4800
MARTIN, DIANA             2850
MORENO, LUIS              9000
MORENO, VICENTE           4000
PEREZ, CARMEN             6000
```

Otra forma de realizar la consulta es:

```
SELECT E1.nomempl
  FROM empleado E1, empleado E2
 WHERE E1.numempl = E2.numempl
 GROUP BY E1.numempl, E2.numempl, E1.salario
HAVING E1.salario > AVG(E2.salario)
 ORDER BY 1;
```

Ejercicio3:

Obtener en orden alfabético los nombres de los departamentos que tienen administrativos.

```
SELECT nomdept,coddept
  FROM departamento D
 WHERE EXISTS (SELECT ' '
                 FROM  empleado E
                WHERE D.coddept  = E.coddept AND
                      nomclab= 'ADMINISTRATIVO')
ORDER BY nomdept
```

Resultado:

```
NOMDEPT          CODDEPT
-------------- ---------
COMERCIAL             30
```

Ejercicio4:

Recuperar los empleados que trabajan en Toledo y que tienen un salario menor al medio de la compañía.

```
SELECT nomempl,nomdept,salario
  FROM empleado E,departamento D
 WHERE E.coddept = D.coddept AND
         codlugt = (SELECT codlugt
                      FROM lugartrabajo
                     WHERE localid = 'Toledo' AND
                           salario < (SELECT AVG(salario)
                                        FROM empleado));
```

Resultado:

```
NOMEMPL              NOMDEPT          SALARIO
-------------------- -------------- ---------
DURAN, ADRIAN        ORGANIZACIÓN        3600
GONZALEZ, JUAN       PRODUCCIÓN          3450
```

Otra forma de realizar la consulta es:

```
SELECT nomempl,nomdept,salario
FROM empleado E,departamento D
WHERE E.coddept = D.coddept AND
        coslugt = (SELECT codlugt
                     FROM lugartrabajo
                    WHERE localid = 'Toledo') AND
        salario < (SELECT AVG(salario)
                     FROM empleado);
```

4.2.8 Tratamiento de privilegios

Ejercicio:

Realizar las acciones siguientes con dos usuarios y en el orden indicado.

El usuario USUXXX:

- Permite al usuario USUYYY la consulta y la actualización en su tabla empleado.

```
GRANT SELECT, UPDATE
   ON empleado
   TO USUYYY;
```

- Permite al usuario USUYYY la consulta y la actualización en su vista wtoledano.

```
GRANT SELECT,UPDATE
   ON wtoledano
   TO ORA002;
```

El usuario USUYYY:

- Crea una vista wg50empleado sobre la tabla USUXXX.empleado seleccionando los empleados del departamento 50.

```
CREATE VIEW wg50empleado AS
 SELECT *
   FROM USUXXX.empleado
  WHERE coddept = 50;
```

- Realiza consultas de la tabla autorizada y de las vistas.

```
SELECT nomempl
  FROM USUXXX.empleado;
```

```
SELECT nomempl
  FROM USUXXX.wtoledano;
```

```
SELECT nomempl
  FROM wg50empleado;
```

El usuario USUXXX:

- Retira todos los privilegios sobre su tabla empleado al usuarios USU YYY.

```
REVOKE ALL ON empleado
  FROM USUYYY;
```

El usuario USUYYY:

- Realiza consultas de la tabla USUXXX.empleado autorizada y de las vistas.

Resultado: No tiene privilegios sobre la tabla.

```
SELECT nomempl
  FROM USUXXX.empleado;

SELECT nomempl
  FROM USUXXX.empleado
            *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Resultado: No tiene privilegios sobre la tabla USUXXX.empleado.

```
SELECT nomempl
  FROM wg50empleado;

select nomempl from wg50empleado
                    *
ERROR at line 1:
ORA-04063: view "ORA002.WG50EMPLEADO" has errors
```

```
SELECT nomempl
  FROM USUXXX.wtoledano;
```

- Crea una vista wYYYtoledano sobre la vista USUXXX.wtoledanos.

```
CREATE VIEW wYYYtoledano AS
 SELECT *
   FROM USUXXX.wtoledano;
```

- Realiza consultas de la vista.

```
SELECT nomempl
  FROM wYYYtoledano;
```

4.3 Equivalencias entre los tipos de datos ANSI, DB2 y oracle

ANSI

Tipo de datos ANSI	Tipo de datos ORACLE
CHARACTER(n)	CHAR(n)
CHAR(n)	
CHARACTER VARYING(n)	VARCHAR(n)
CHAR VARYING(n)	
NATIONAL CHARACTER(n)	NCHAR(n)
NATIONAL CHAR(n)	
NCHAR(n)	
NATIONAL CHARACTER VARYING(n)	NVARCHAR2(n)
NATIONAL CHAR VARYING(n)	
NCHAR VARYING(n)	
NUMERIC(p,s)	NUMBER(p,s)
DECIMAL(p,s)	
INTEGER	NUMBER(38)
INT	
SMALLINT	
FLOAT(b)	NUMBER
DOUBLE PRECISON	
REAL	

- El tipo FLOAT es un número en punto flotante con una precisión binaria b.

- El tipo DOUBLE PRECISION es un número en punto flotante con una precisión binaria 126.

- El tipo REAL es un número en punto flotante con una precisión binaria 63.

DB2

Tipo de datos DB2	Tipo de datos ORACLE
CHARACTER(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
DECIMAL(p,s)	NUMBER(p,s)
INTEGER	NUMBER(38)
INT	
SMALLINT	
FLOAT(b)	NUMBER

Los tipos de datos TIME y TIPESTAMP se pueden expresar con el tipo de dato DATE de Oracle.

4.4 Bibliografía

Título: SQL Reference.
Autor: ORACLE

Título: SQL Para usuarios y programadores.
Autor: J. Benavides Abajo.
Editorial: Paraninfo

Título: Oracle8 Guía de aprendizaje.
Autor: Michael Abbey.
Editorial: Oracle Press