# 5

# CONSULTAS Y MANIPULACIÓN DE DATOS

# 3 – Primeras consultas

# Cómo se ejecutan las consultas

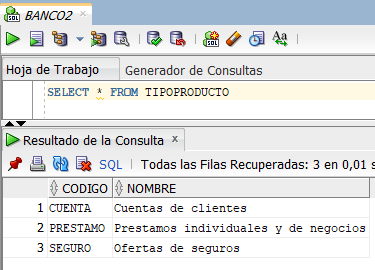
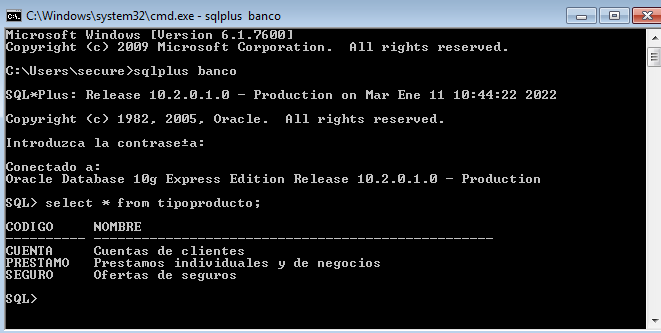
Una vez que el SGBD verifica el usuario y contraseña, si el usuario tiene derechos para ejecutar consultas el servidor comprueba los siguiente antes de ejecutar la sentencia:

* ¿Se tienen permisos para ejecutar la sentencia?
* ¿Se tienen permisos para acceder a los datos?
* ¿La sintaxis es correcta?

Si supera estos tres test, la consulta se pasa al optimizador de consultas cuya función es determinar la forma más eficaz de ejecutar la consulta.

Una vez el servidor ejecuta la consulta el resultado, un ***resultset*** se devuelve a la aplicación que ha ejecutado la consulta (por ejemplo **sqlplus** o **SQLDeveloper**). Un ***resultset*** es una tabla que contiene en sus filas y columnas los resultados de la consulta.

Esa aplicación, **sqlplus o SQLDeveloper** construye visualmente la tabla para dar formato a los resultados.



# Sentencia Select

La sentencia SELECT, está formada por varias clausulas. Sólo dos de ellas son obligatorias (SELECT y FROM), aunque lo habitual es utilizar tres (la cláusula WHERE), del total de seis cláusulas incluidas en las especificaciones de ANSI SQL:

**Select** Columnas a incluir en el **resultset**

**From** Tablas de las que se obtendrán los datos

**Where** Filtra los datos no necesarios

**Group** **by** Agrupa las filas con valores de columna iguales

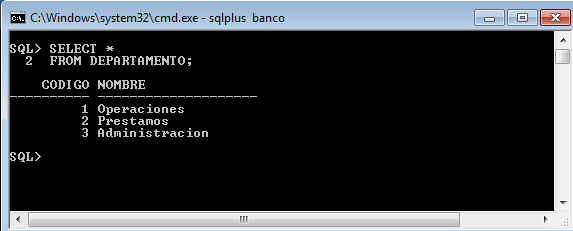
**Having** Filtra los grupos no necesarios

**Order** **by** Ordena las filas a mostrar por una o más columnas

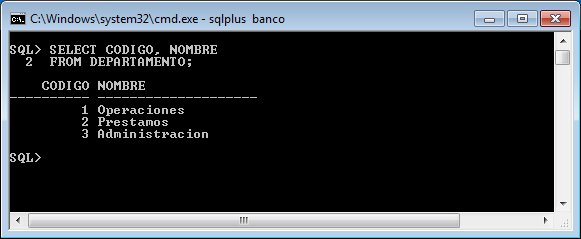
### Cláusula SELECT

La cláusula SELECT es una de las cláusulas que el servidor evalúa en último lugar al ejecutar una sentencia **SELECT**, ya que primero debe de conocer la lista de todas las columnas que se pueden incluir en el **resultset** final, y eso depende de la clausula **FROM**.

En este ejemplo de **SELECT**, el asterisco (\*) indica que deben devolverse todas las columnas de la tabla **DEPARTAMENTO**

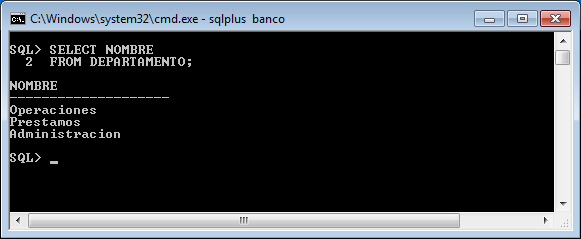


En lugar de indicar todas las columnas se pueden indicar el nombre de las columnas que nos interesan.



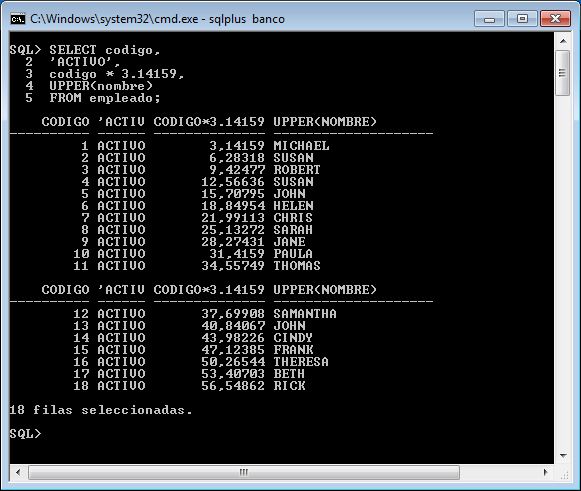
El resultado es el mismo, porque en el segundo **SELECT**, se indican explícitamente todas las columnas de la tabla.

Se permite incluir sólo un subconjunto de las columnas:



Así pues, la cláusula **SELECT** determina cuales de entre todas columnas disponibles, deben incluirse en el ***resultset.***

No se está limitado a incluir sólo columnas de las tablas incluidas en la cláusula **FROM**, se pueden incluir también cosas como esta:

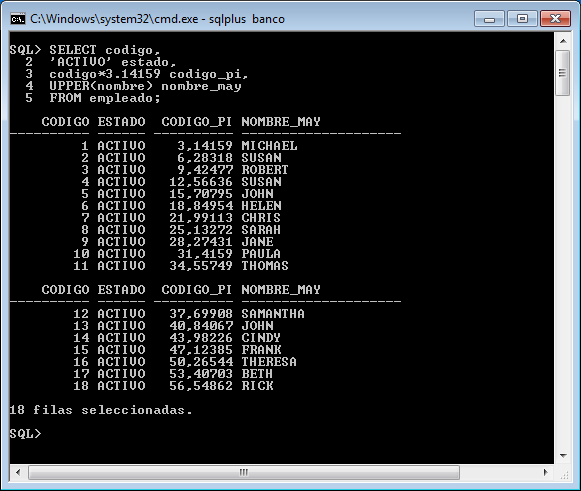


Aunque aun no hemos visto las funciones incluidas en el SQL de Oracle, puede darnos una idea de qué tipo de cosas pueden incluirse en la cláusula SELECT.

#### Alias de columnas

Aunque cuando se devuelven los **resultsets**, Oracle generan etiquetas para las columnas, es posible asignar etiquetas personalizadas.

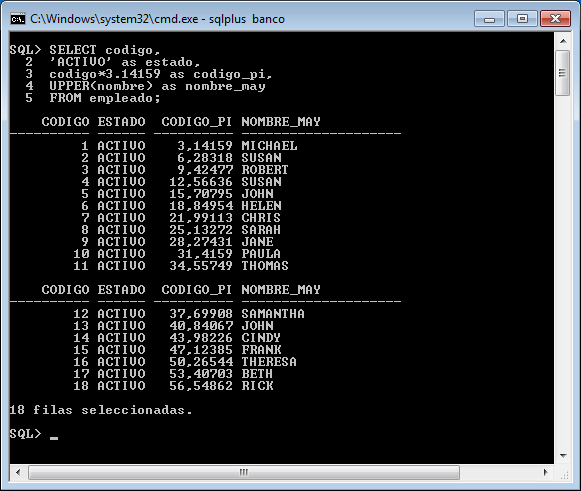
Se puede querer asignar una nueva etiqueta a una columna de una tabla, a los resultados generados por expresiones, o las llamadas a las funciones integradas. Para hacerlo se añade un alias de columna tras cada elemento de la cláusula SELECT.



Si ahora se observan las cabeceras de las columnas se puede ver que ahora tienen nombres significativos en lugar de los que son generados automáticamente por Oracle.

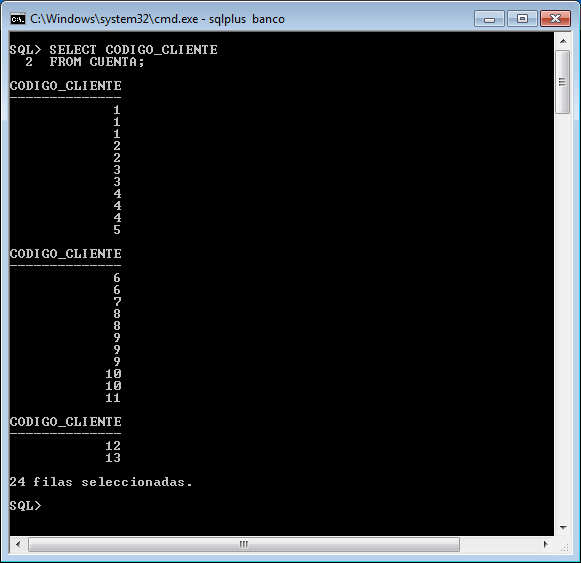
Para hacerlo se han añadido los alias estado, codigo\_pi y nombre\_may, tras la segunda, tercera y cuarta columna. De esta forma el **resultset** es más fácil de entender y es más fácil de trabajar con él, cuando, por ejemplo, se necesita recuperar el dato de una columna en un programa.

También se permite utilizar la palabra **as** antes del alias para mejorar la lectura aunque no hay ninguna diferencia entre usarla o no usarla.

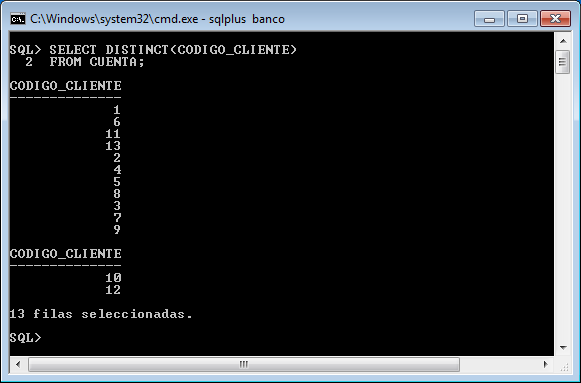


#### Eliminar duplicados

Hay ocasiones en las que las consultas devuelven filas duplicadas. Por ejemplo, si se devuelven los códigos de los clientes que tienen cuentas, aparecen duplicados porque hay clientes que tienen varias cuentas.



Si se desean obtener los distintos clientes que tienen cuentas, en lugar de los códigos de cliente de cada fila, se puede utilizar la palabra DISTINCT para eliminar datos duplicados.



Si no se desean eliminar los duplicados debería indicarse ALL en lugar de DISTICT pero, como es el valor por defecto, la mayoría de los programadores no lo ponen.

Usar DISTINC afecta al rendimiento, pues deben ordenarse los valores para no mostrar los duplicados. Así que no debe utilizarse para eliminar los duplicados que no deberían estar entre los datos porque es un error.

### Cláusula FROM

La cláusula **from** (que puede incluir una o varias tablas) indica las tablas que va a usar la query, y si se incluyen joins, las relaciones de unión entre ellas.

Las tablas no están limitadas a las que estén almacenadas en la base de datos, y como sabemos que una consulta devuelve una tabla, las tablas que se indican o algunas de ellas, puede ser el resultado de una consulta (tabla temporal) o tablas virtuales (vista).

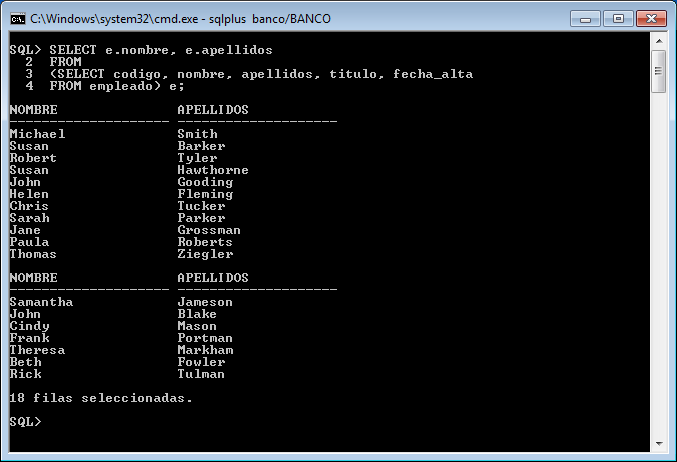
### Tablas generadas por consultas (subconsultas)

Una subconsulta es una consulta que se encuentra contenida dentro de otra consulta.

Las subconsultas se definen entre paréntesis y se pueden encontrar en varias partes de la sentencia SELECT. Una de ellas es en la cláusula **from**.

Una consulta que se genera dentro de la cláusula **from**, es visible en el resto de la consulta y puede interactuar y relacionarse con el resto de tablas de la cláusula **from**.

Ejemplo:



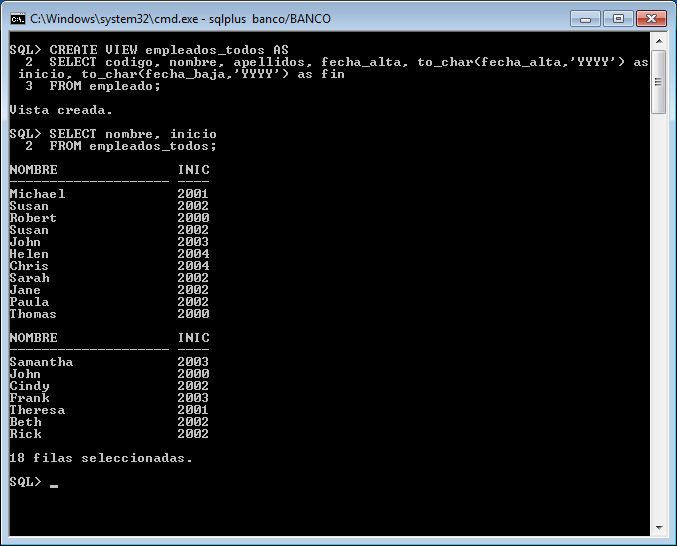
En este caso, una subconsulta se lanza sobre la tabla empleados y devuelve cinco columnas. La consulta que la incluye sólo referencia tres de las cinco columnas. Para hacerlo nombra la tabla temporal resultado de la subconsulta como e (un alias).

### Vistas

Una vista es una consulta que se almacena en el diccionario de datos. Se comporta como una tabla, pero no tiene datos propios y por ello se le llama *tabla virtual*.

Cuando se lanza una consulta que usa una vista en la cláusula **from**, primero se construye la subconsulta que define la vista y a continuación se ejecuta la consulta como si se ha hecho con la subconsulta.

Ejemplo:



Cuando se crea la vista, no se almacenan datos en el objeto empleados\_todos, sino la sentencia SELECT para poder usarla en el futuro.

Una vez que existe la vista, ya puede utilizarse en la cláusula **from** como si de una tabla se tratase.

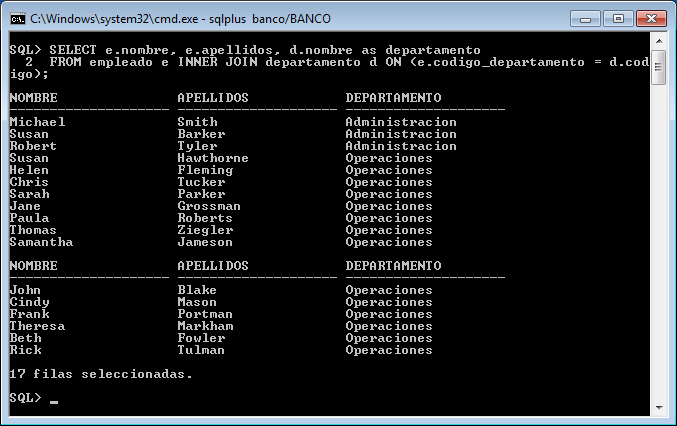
Las tablas se pueden crear por muchas razones, como simplificar el diseño de bases de datos muy complejas u ocultar a los usuarios columnas de las tablas originales.

### Joins entre tablas

Si aparece más de una tabla en la cláusula **from**, se pueden indicar en esta misma cláusula las relaciones entre las tablas.

No es obligatorio ni en Oracle ni en ningún otro gestor de base de datos, y se profundizará más adelante.

Ejemplo:



Esta consulta muestra datos de dos tablas, la tabla empleado y la tabla departamento. Como en la base de datos, la forma de vincular las dos tablas es a través de la columna codigo\_departamento de la tabla empleado, es lo que se le indica al gestor de base de datos para que devuelva el departamento con el que está asociado cada empleado.

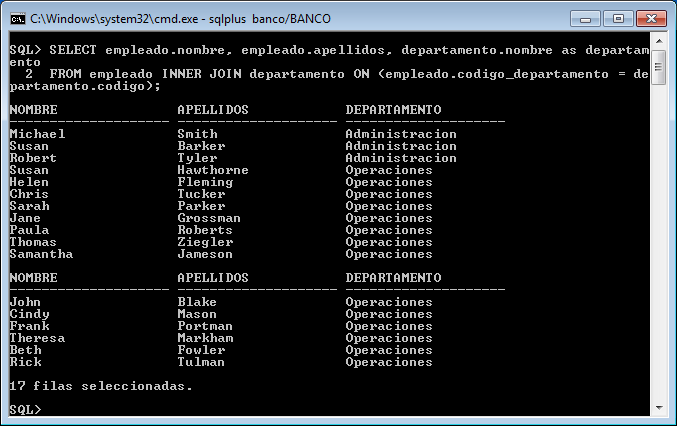
### Alias de tablas

Cuando se usan múltiples tablas en una misma consulta, se necesita una forma de identificar de que tabla es la columna que se indica en el resto de cláusulas de las sentencia.

Se puede hacer de dos formas:

* Indicando el nombre completo de la tabla.
* Dándole a cada tabla un ‘alias’ y usando dicho alias.

En el SQL de la sentencia anterior se ha usado de esta manera. Alternativamente podría haberse usado el nombre completo de la tabla:



El uso de alias genera consultas más compactas sin causar confusión si se eligen los alias con un poco de cabeza.

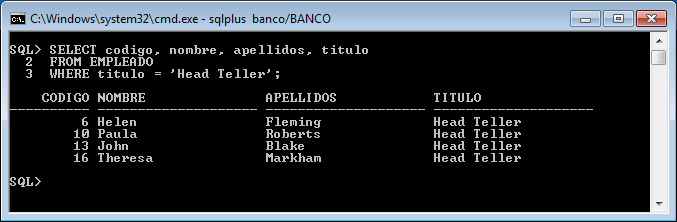
### Cláusula WHERE

Hasta ahora las cláusulas que hemos usado para obtener datos de la tabla empleado han devuelto todas las filas de la tabla, excepto cuando se ha ilustrado el uso de DISTINCT.

La mayor parte de las veces, no se necesitará devolver todas las filas de la tabla, sino que se tendrán que filtrar sólo aquellas que nos interesen o lo que es lo mismo quitar todas aquellas que no nos interesen.

La clausula WHERE se encarga de filtrar las filas que nos interesan y descartar aquellas que no.

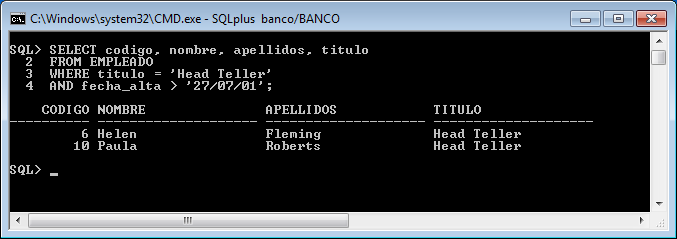
Por ejemplo, posiblemente estés interesado en obtener datos de la tabla empleados pero sólo de aquellos que son los jefes de cajeros. La siguiente consulta devuelve datos de empleados pero sólo de aquellos que son jefes de cajeros:



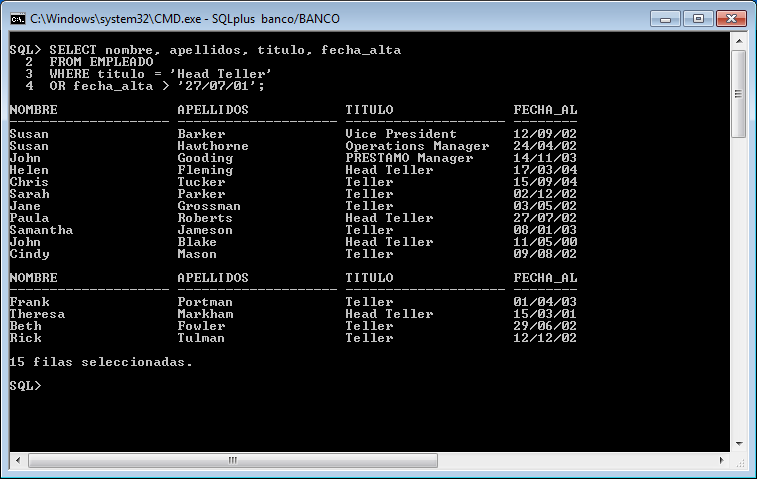
Es con la cláusula WHERE con la que se filtran los 14 de los 18 empleados que no son jefes de cajeros.

Esta cláusula WHERE sólo incluye una condición, pero se pueden incluir tantas como se necesiten separándolas por operadores como **and**, **or** y **not**.

He aquí un ejemplo que ilustra como incluir en la anterior consulta quisiésemos una segunda condición:



Si en la consulta anterior cambiásemos el AND por OR, en lugar de reducir aún más los registros que cumplirían la condición, se incluye dos condiciones que hacen que los registros sean válidos si se cumple cualquiera de ellas.

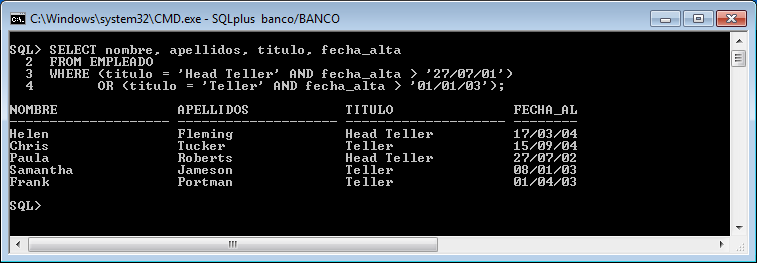


Como puede verse, se incluyen los cuatro registros que son jefes de cajeros y además el resto de empleados cuya fecha de alta es mayor que la fecha indicada.

La condición indicada la cumplen 15 registros de los 18 que componen la tabla empleados, al menos es cierta una de las dos para 15 de los 18 registros.

Así que cuando se separan condiciones con el operador and, todas las condiciones deben evaluarse como verdaderas. Mientras que si se usa el operador or, es suficiente con que una se evalúe como verdadera para que se incluya la fila.

En la siguiente consulta se usan ambos operadores, AND y OR en la misma consulta:



Siempre se deben usar paréntesis para separar grupos de condiciones cuando se mezclan distintos operadores para dar legibilidad al código y no obtener resultados inesperados.

### Cláusula GROUP BY y HAVING

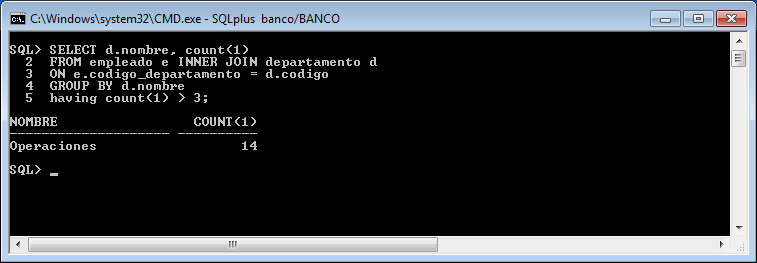
Hasta ahora todas las consultas han devuelto datos sin ninguna manipulación. Sin embargo en ocasiones necesitarás que el sistema gestor de bases de datos te devuelva los datos procesados.

Uno de estos mecanismos es usar la cláusula GROUP BY, que se usa para agrupar filas que compartan los mismos valores.

Por ejemplo, mejor que consultar una lista de los empleados y los departamentos que tienen asignados, podría ser más interesante obtener una lista de departamentos con la cantidad de empleados que tiene asignados.

Usando la cláusula GROUP BY esto es posible. La cláusula HAVING se utilizará para filtrar de todos los grupos que se obtendrían, aquellos que interesan, de forma similar a como WHERE filtra las columnas.

Estas en una consulta en la que se obtiene la cuenta de los empleados por departamento con más de tres empleados.



El uso de estas cláusulas se verá más adelante.

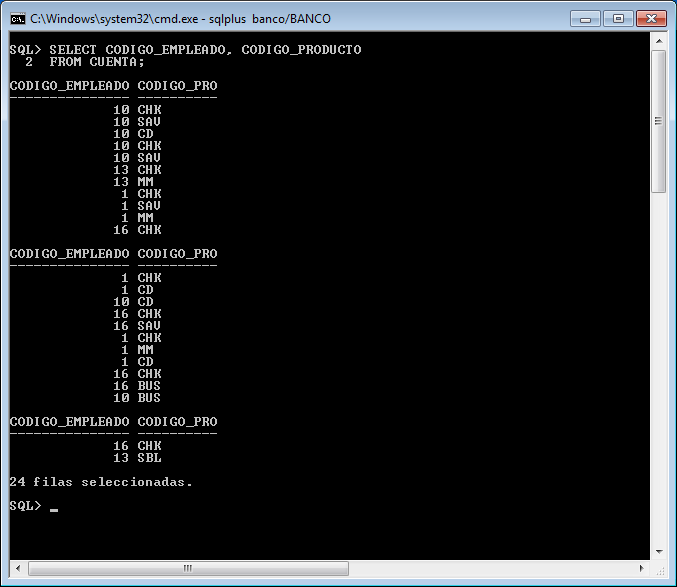
### Cláusula ORDER BY

Como hemos hablado en el tema del modelo relacional, los datos contenidos en una tabla no tienen ningún orden, y por eso los datos devueltos por una consulta no tienen porqué respetar ningún orden.

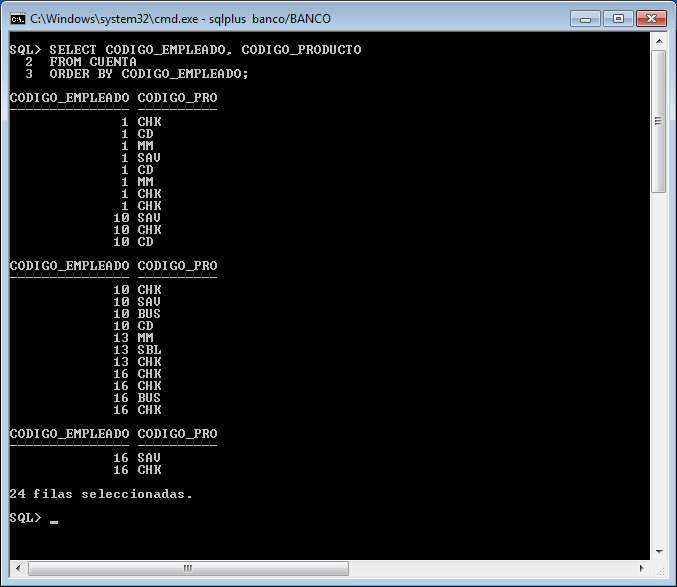
Si se quiere que los resultados se muestren en un determinado orden, debe indicarse con la cláusula ORDER BY.

La cláusula ORDER BY puede ordenar el resultado según los datos de una columna o según una expresión basada en los datos de una columna.

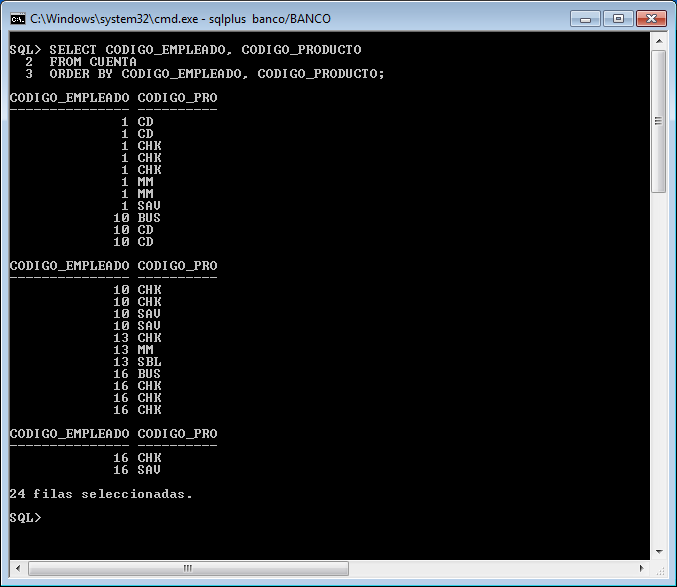
En la consulta siguiente pueden verse los empleados que han contratado cuentas y que tipo de producto es cada cuenta contratada. No obstante no respeta ningún orden.



Si quisiésemos analizar cada uno de los empleados, sería deseable que los datos estuviesen ordenados por la columna código de empleado:



Y si además quisiésemos que los productos contratados por cada empleado también estuviesen en orden, usaríamos dos criterios para la ordenación, ordenando primero por el primer criterio y los empates por el segundo.

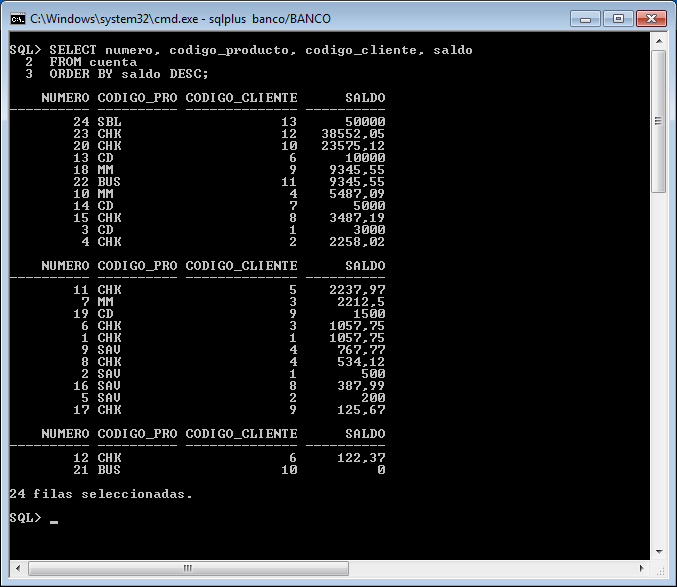


### Ordenación ascendente y descendente

Cuando se indica una ordenación se puede especificar si esa ordenación debe ser ascendente o descendente con los operadores **ASC** y **DESC**.

La ordenación por defecto es ascendente, así que sólo se debe indicar DESC si se desea una ordenación descendente, aunque muchos operadores lo incluyen siempre para mejorar la legibilidad.

Por ejemplo, si queremos ordenar las cuentas según las que tengan un balance mayor.



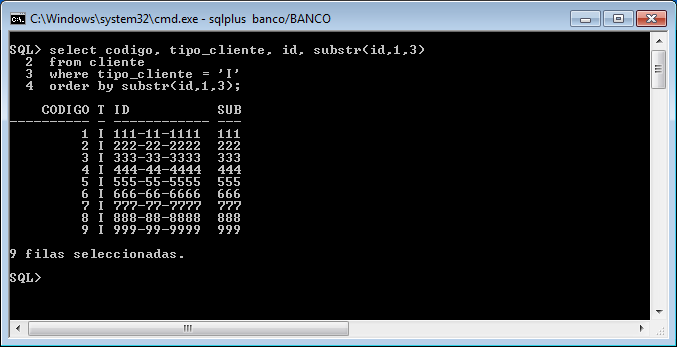
Las ordenaciones descendentes se usan normalmente para rankings, como por ejemplo para obtener los X mejores saldos.

Oracle permite limitar el número de filas que muestra la consulta con la condición que se puede incluir en el where: **rownum = n** y a partir de la versión 12c con la cláusula

### Ordenación usando expresiones

En ocasiones el criterio de ordenación puede no ser el valor de alguna de las columnas sino una expresión.

Por ejemplo en este caso se va a ordenar a los clientes por sus tres primeros caracteres de su carnet de identidad:



### Ordenación por número de columna

Si se están ordenando los resultados en función de una de las columnas de la cláusula SELECT, en lugar de indicar el nombre de la columna, puede ordenando indicando la posición de la columna en la cláusula ORDER BY.

