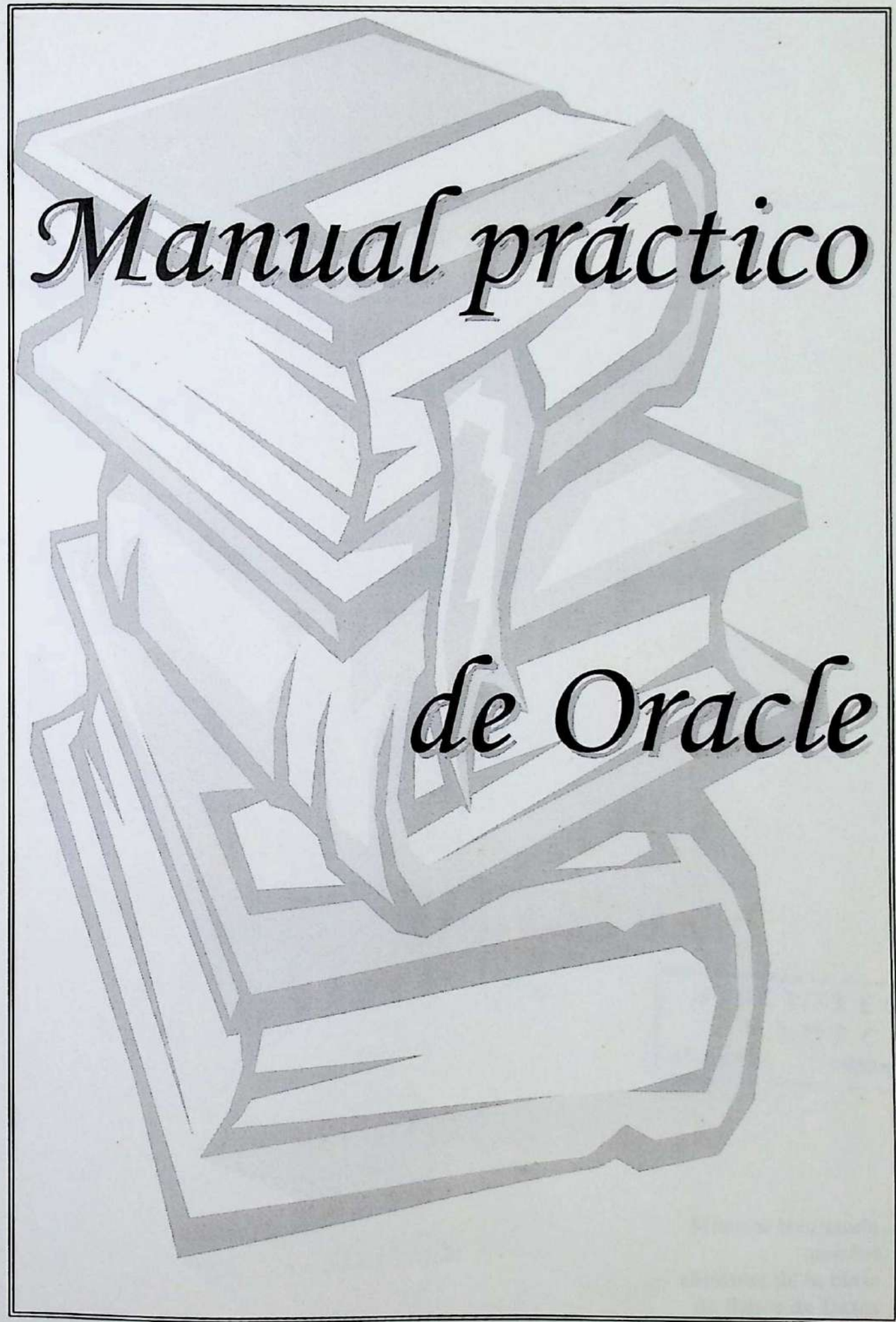


Manual práctico de Oracle

38.363

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

BIBLIOTECA UNADECA ALAJUELA, COSTA RICA

Manual Realizado
por los
alumnos de la clase
de Bases de Datos

Introducción

En el presente manual se trata de dar a conocer los principios básicos de el paquete Oracle, sus funciones y herramientas y como estas se pueden utilizar para desarrollar un sistema básico.

Además se ejemplifican para un mejo entendimiento de las herramientas, también se presenta de una manera sencilla la creación de las bases de datos y como estas se pueden utilizar para realizar búsquedas avanzadas.

Además se muestra la instalación de este paquete y cuales son los requisitos mínimos para instalar el mismo para que se pueda realizar una fácil utilización de este paquete y conocer mas a fondo los principios de básicos de oracle, sus funciones y usos.

Es por esta razón que se espera que al dar lectura a este manual se puede comprender de manera absoluta y se pueda obtener el mejor provecho de esta herramienta de la programación y utilización de las bases de datos.

Indice

54. Report Wizard

55. Modelo de datos (Data Model)

56. Creación de la consulta

57. GruposColumnas de tipo Summary (resumen) y Formula (fórmula)

58. Object NavigatorLive PreviewerData ModelDirección de impresiónSection

Capitulo I

TÉCNICAS DE DISEÑO Y PROGRAMACIÓN DE APLICACIONES CONTRA BASES DE DATOS ORACLE

El objetivo de este documento es marcar unas pautas a tener en cuenta cuando se estén desarrollando aplicaciones que han de trabajar con bases de datos Oracle. Para obtener un buen rendimiento de las aplicaciones, es conveniente manejar una serie de conceptos sobre cómo funciona el optimizador de Oracle, cuándo se pueden usar índices y cuándo interesa evitarlos, o cómo se tratan sentencias en las que aparecen vistas, etc.
Es muy importante para el mantenimiento de una aplicación que el código que se ejecuta aproveche la estrategia que sigue el optimizador a la hora de elegir un plan óptimo. El impacto que tiene en el rendimiento de una base de datos un buen código es mucho mayor que el que pueda tener cualquiera de los parámetros que se usan para configurar la instancia. Además, el coste que supone el optimizar una aplicación en el momento en que se está desarrollando es también muy inferior al que supondrá mejorar algo cuando esa aplicación esté ya en su fase de producción.

Evitar FULL TABLE SCAN no planificados

Un full table scan lee secuencialmente todos los datos de una tabla, tanto si son relevantes para la sentencia que se ejecuta como si no. Hay dos razones importantes para evitar full table scans innecesarios

No son selectivos. Aunque operaciones no selectivas puedan ser apropiadas para grandes tareas por lotes, que manejan muchos datos, son poco apropiadas para aplicaciones online.

Los datos leídos en un full table scan desaparecen de la SGA muy rápido, ya que se sitúan en la parte más volátil del buffer de datos.

Cuándo se realiza un full table scan

En RBO, se realizará un full table scan sobre una tabla cuando en una sentencia SQL se dé alguna de las siguientes condiciones:

No existen índices en la tabla.

En la sentencia no hay ninguna condición sobre las filas que devuelve.

No hay ninguna condición sobre la primera columna de algún índice de la tabla.

Hay alguna condición sobre las primeras columnas de algún índice de la tabla, pero estas condiciones aparecen dentro de expresiones o requieren de alguna conversión implícita.

Hay alguna condición sobre las primeras columnas de algún índice de la tabla, pero estas condiciones son desigualdades o comprobaciones por NULL o NOT NULL.

Si se está usando CBO, Oracle usará un full table scan en todos los casos anteriores y, además, puede decidir hacerlo en alguno de los siguientes:

La tabla no ha sido analizada.

La tabla es pequeña.

Las columnas indexadas no son selectivas.

El modo de optimización está puesto a ALL_ROWS.

Recomendaciones

Para asegurar que una sentencia puede hacer uso de índices, lo primero que debe considerarse es indexar todas las primary key y foreign key. En una aplicación, la mayor parte de sentencias y joins se hacen usando estas columnas.

El orden de las columnas en un índice es una decisión muy importante. Deben considerarse dos factores para decidir sobre este orden. Primero, la columna usada más frecuentemente en las condiciones WHERE debería ser la primera del índice. Segundo, colocar primero la columna más selectiva (aquella con mayor número de valores distintos). La selectividad de la primera columna de un índice es muy importante si se está usando CBO.

Hay que asegurarse de que, en las sentencias, las columnas que se están indexando no aparecen dentro de expresiones (funciones TO_CHAR, UPPER, etc.) o en conversiones implícitas (la columna es un VARCHAR2 y se está comparando con un NUMBER).

Es frecuente el uso de sentencias en las que se pregunta por un campo nulo, para actualizarlo a continuación. Si la tabla es muy grande y van a recuperarse pocos registros, interesa que ese campo tenga inicialmente un valor por defecto (no nulo), y se pregunte luego por ese valor. Por ejemplo, una tabla CLIENTES con un campo TELEFONO. Este campo se deja inicialmente con NULL y se ejecutan sentencias del tipo

select * from clientes where telefono is null;

Esta sentencia siempre hará un FULL TABLE SCAN de CLIENTES. Si se ejecuta frecuentemente, y la tabla es muy grande, el impacto sobre la base de datos puede ser considerable. Para evitar esto, puede definirse este campo con un valor por defecto, por ejemplo, '000000000'. La sentencia anterior se puede reescribir de esta forma:

select * from clientes where telefono = '000000000';

Si hay un índice por el campo TELEFONO, el optimizador puede decidir hacer uso de él (RBO siempre lo usará).

Usar índices selectivos

La selectividad de un índice es la relación entre el número de valores distintos de una columna indexada y el número de registros de la tabla. Si una tabla tiene 1000 registros, y una columna indexada de la tabla tiene 950 valores diferentes, la selectividad del índice es 0.95 (950/1000). La mejor selectividad es 1. Los índices únicos sobre columnas no nulas siempre tienen selectividad 1.

La selectividad de un índice nos da una medida de su utilidad para evitar I/O en la ejecución de sentencias contra la tabla. Si un índice sobre una tabla de1000 registros tiene sólo 5 valores diferentes, entonces su selectividad es muy pobre (5/1000 = 0.005). Para cada posible valor de este índice, habrá un promedio de 200 filas. En este caso, podría ser más interesante realizar un full table scan que acceder vía índice a esta tabla.

Si se está usando CBO, el optimizador no realizará accesos a tablas a través de índices con una selectividad muy baja.

Elección de la primera columna en un índice concatenado

La primera columna de un índice debería ser la columna más selectiva y también debería ser la más usada en las condiciones que aparezcan para las sentencias sobre esa tabla. Si una columna cumple las dos condiciones, entonces ésa debe ser la primera del índice.

Si una columna no selectiva es la más frecuentemente utilizada en sentencias sobre tablas grandes, posiblemente sería necesario revisar el diseño de la aplicación o de los datos.

Si una columna muy selectiva no es muy usada en las condiciones de sentencias, sólo es recomendable crear un índice sobre ella si se trata de una foreign key.

Índices concatenados vs varios índices con una sola columna

Cuando se va a crear un índice concatenado, debe valorarse si la selectividad de ese índice va a ser considerablemente mayor con varias columnas que con una. Por ejemplo, para una tabla de POBLACIONES, un índice por el campo NOMBRE_POBLACION tendrá la misma selectividad que otro por los campos NOMBRE_POBLACION y PROVINCIA (habrá muy pocas poblaciones que se llamen igual en distintas provincias). En este caso, no interesará un índice concatenado, ya que con una sola columna puede localizarse el registro consultado.

Si se tienen varias columnas muy selectivas, podemos pensar en crear varios índices, cada uno de ellos con una columna, si las consultas de nuestra aplicación tienen condiciones de igualdad para esas columnas unidas por AND. Por ejemplo, para sentencias del tipo

```
SELECT *
FROM emp
WHERE job = 'ANALISTA'
AND deptno = 20;
```

podría interesar tener dos índices, uno para cada columna. El plan de ejecución sería el siguiente:

TABLE ACCESS BY ROWID EMP
AND-EQUAL
INDEX RANGE SCAN EMP$JOB
INDEX RANGE SCAN EMP$DEPTNO

Se hace un RANGE SCAN de cada índice y, a continuación, la operación AND-EQUAL, que consiste en obtener la intersección de los RowID que han devuelto los INDEX RANGE SCAN anteriores.
Aquí debe valorarse cómo se harán menos lecturas: si en un INDEX RANGE SCAN del índice concatenado, seguido del acceso a tabla por RowID; o con los dos INDEX RANGE SCAN seguidos del acceso a tabla por RowID.

Joins de varias tablas: NESTED LOOPS y MERGE JOINS

En Oracle hay dos tipos de operaciones para resolver un join: NESTED LOOPS y MERGE JOIN (en Oracle 7.3 se añadió una nueva opción, HASH JOIN). MERGE JOIN es una operación de conjuntos, no devuelve registros a la siguiente operación hasta que todas las filas han sido procesadas. Por el contrario, NESTED LOOPS es una operación de fila, devuelve los primeros registros a la siguiente operación tan pronto como hayan sido procesados. Es importante conocer cómo trabajan cada una de estas dos opciones para determinar cuál es la idónea en función del tamaño de las tablas y la naturaleza de la sentencia.

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

MERGE JOINS

Una operación MERGE JOIN enfrenta los datos resultantes de dos scans. Implica tres pasos:

1. TABLE ACCESS FULL de cada tabla del join.

2. SORT JOIN para ordenar los dos conjuntos resultado del paso anterior.

3. MERGE JOIN para mezclar los resultados del SORT JOIN.

El uso de un MERGE JOIN indica que no existen índices sobre las tablas o que la sintaxis de la sentencia imposibilita el uso de índices. Este tipo de operación no es apropiado para aplicaciones online por varios motivos:

Lentitud al devolver la primera fila de la sentencia. Debido a que es una operación de conjunto (no de fila) no devuelve filas hasta que han sido procesadas todas.

Los bloques se cargan en la SGA haciendo un FULL TABLE SCAN, lo que hace que sean los primeros en ser eliminados cuando se necesite espacio en el buffer de datos.

Puede ser necesario el uso de segmentos temporales para resolver la sentencia, lo que supone un riesgo potencial de contención entre usuarios.

Sin embargo, pueden darse situaciones en que un MERGE JOIN sea la operación más eficiente para resolver un join, como tareas por lotes que traten muchos registros. Un MERGE JOIN será la mejor opción en los mismos casos en que un FULL TABLE SCAN sea la mejor opción: cuando la tabla involucrada sea muy pequeña y cuando la tabla sea extremadamente grande.

Si la tabla es muy pequeña, entonces puede ser más rápido realizar un full table scan que un index range scan seguido de un table access by RowID. Esto es así cuando, por ejemplo, la tabla entera puede ser leída en una única petición de I/O al sistema.

Si la tabla es muy grande, también puede interesar realizar un full table scan por ciertos motivos:

Si estamos seleccionando un rango de valores de una columna y los registros no se encuentran ordenados físicamente según esta columna. En este caso, una combinación

index range scan seguido de table access by RowID puede leer más bloques que un full table scan.

Los bloques cargados en la SGA no se mantienen ahí durante mucho tiempo, no perjudicando la compartición de datos entre procesos.

Puede sacarse provecho del Parallel Query Option de Oracle.

NESTED LOOPS

Esta es la operación más habitual en que Oracle resuelve los joins. Indica que un índice está disponible para hacer el join. Es una operación de fila: cada fila procesada se devuelve a la siguiente operación, en vez de esperar a que se procese el resultado completo. NESTED LOOPS es una operación muy efectiva para aplicaciones online.
Cuando se realiza un NESTED LOOPS, se siguen los siguientes pasos:
Elección de la tabla sobre la que se hará el FULL SCAN. Esta tabla se llamará directora.
FULL TABLE SCAN de la tabla directora.
INDEX RANGE SCAN de la otra tabla.
Si se encuentra alguna coincidencia, un table access by RowID de la segunda tabla.
La elección de la tabla directora en un NESTED LOOPS es crítica para la ejecución de la sentencia. El número de lecturas realizadas por la sentencia depende de qué tabla se tome para realizar el full table scan.

Implicaciones de la tabla directora en un NESTED LOOPS

La clave del buen rendimiento de una operación NESTED LOOPS es el orden en que se realiza el join de las tablas. El número de repeticiones del bucle es el producto del número de registros resultante de la primera tabla por el número de registros de la segunda tabla a los que se accede después. Si se añaden más tablas en el join, la elección de la primera tabla es aún más crítica. Lo conveniente es minimizar el número de registros leídos en los primeros pasos del bucle.

Consideremos un ejemplo.

Tenemos una sentencia en la que aparecen 4 tablas (A, B, C y D), todas ellas del mismo tamaño, con las siguientes cláusulas FROM y WHERE: from D, C, B, A
where A.cod = B.cod and B.cod = C.cod and C.cod = D.cod and A.cod = 123 and D.val = 'VALOR'

Si las tablas A, B, C y D tienen índices para sus columnas cod, un NESTED LOOP será la operación que usará Oracle para resolver la sentencia. Se hará un join de A con B, del resultado de este join con C y del resultado de este último con D. Finalmente, se aplicará la condición que hay sobre D.val.

Si D.val es una columna selectiva, el rendimiento de la sentencia será mejor si hacemos que D entre en el primer NESTED LOOP de la ejecución. De esta forma, menos registros serán devueltos al siguiente NESTED LOOP y así sucesivamente. La anterior cláusula WHERE puede reescribirse así:

from D, C, B, A
where A.cod = B.cod and B.cod = C.cod and C.cod = D.cod and D.cod = 123 and D.val = 'VALOR'

Ahora, se hará un join de D con C, del resultado con B y, finalmente, con A.

Veamos qué puede significar esto con números reales. Supongamos que cada tabla tiene exactamente 100 registros y que solamente hay un registro en D con val = 'VALOR'. Un acceso por índice genera un promedio de 2 accesos a las ramas del índice, más 1 acceso a la hoja por cada fila, además de un acceso por RowID a la tabla por cada fila. Por tanto, para leer 100 filas en un acceso por índice se necesitan 2 lecturas para las ramas del índice, más 100 lecturas del índice, más 100 lecturas de la tabla, haciendo un total de 202 lecturas. Para el join de A con B, se necesitan 100 lecturas por cada una de las 202 lecturas que se hicieron para la tabla A.

Para el join original, tendríamos el siguiente número de lecturas:

Operación Tabla accedida Lecturas Lecturas acumuladas

1ª tabla accedida A 2 + 2 * 100 202

1er. join B 100 * 202 20.402

2º join C 100 * 100 * 202 2.040.402

3er. join D 100 * 100 * 100 * 202 204.040.402

Como se ve, un incremento significativo en cada paso penaliza los pasos posteriores. Veamos qué sucede en el caso de la sentencia modificada:

Operación Tabla accedida Lecturas Lecturas acumuladas

1ª tabla accedida D 2 + 2 * 100 202 (devuelve 1 fila)

1er. join C 1 * 202 404

2º join B 100 * 1 * 202 20.604

3er. join A 100 * 100 * 1 * 202 2.040.604

El orden en que se realizan los joins tienen una implicación decisiva en el rendimiento del NESTED LOOP. En este ejemplo, se puede reducir a una centésima parte el número de lecturas realizadas para resolver la sentencia.

Cómo alterar el orden de los joins

En RBO, si se tiene igual oportunidad de usar índice en todas las tablas de un join, la tabla sobre la que se hará FULL SCAN será la que aparezca en último lugar en la cláusula FROM. Si estamos usando CBO, el optimizador tendrá en cuenta el tamaño de las tablas y la selectividad de los índices para elegir la primera tabla.

Si se conoce la naturaleza de los datos, puede interesar decirle al optimizador que siga un orden concreto para realizar los joins. Esto se hace añadiendo hints (pistas) a la sentencia. Algunos de estos hints son:

ORDERED Se hace el join de las tablas en el orden en que aparecen en la cláusula FROM.

INDEX Proporciona una lista de índices a utilizar.

FULL Da una tabla sobre la que se hará FULL TABLE SCAN. Esta tabla será usada como la tabla directora.

USE_NL Lista las tablas de las que se hará el join usando NESTED LOOP.

Si no es posible mejorar el rendimiento de los joins cambiando el orden de las tablas, puede que interese estudiar la posibilidad de eliminar alguno de estos joins denormalizando nuestras tablas.

Sentencias SQL que usan vistas

Si una sentencia contiene una vista, el optimizador tiene dos modos de resolverla: resolver primero la vista y después la sentencia, o integrar el texto de la vista con la sentencia. Si se resuelve primero la vista, el conjunto resultado de la vista es calculado primero, y el resto de las condiciones de la sentencia se aplican después como un filtro.

Dependiendo de los tamaños de las tablas involucradas, resolver primero la vista puede degradar el rendimiento de la sentencia; si la vista se integra dentro de la sentencia, las condiciones de la sentencia pueden aplicarse dentro de la vista y el conjunto resultado que se obtiene será más pequeño. Sin embargo, en determinados casos puede mejorarse el rendimiento separando operaciones de grupo mediante vistas. Cuando se usen vistas en un join, es conveniente tener en cuenta cómo se van a resolver. Si la vista contiene funciones de grupo (GROUP BY, SUM, COUNT, DISTINCT), no podrá integrarse dentro de la sentencia, sino que ha de resolverse primero.

Integración de la vista en la sentencia

Si la vista devuelve un número de filas muy grande, o si este número de filas va a ser filtrado con condiciones adicionales de la sentencia que usa la vista, el rendimiento se verá mejorado permitiendo que la vista se integre dentro de la sentencia. El optimizador realizará esta operación automáticamente siempre que sea posible.

Para evitar tener vistas que no puedan integrarse en la sentencia, deberán evitarse las funciones de grupo en el SQL de la vista. Estas funciones se añadirán posteriormente a la sentencia.

Forzar a que la vista se ejecute por separado

En ciertos casos, puede interesar que el SQL de la vista no se integre con el resto de la sentencia. Por ejemplo, si se está realizando un GROUP BY en un NESTED LOOPS join de dos tablas, la operación de grupo no se completará hasta que haya finalizado el join de las dos tablas. Esto es así incluso si las columnas que se agrupan son todas de la misma

tabla. En este caso, interesaría realizar el GROUP BY antes que el join, para que el NESTED LOOP procese menos filas. Esto se consigue añadiendo al SQL de la vista la función GROUP BY. De este modo, el optimizador no podrá integrarla dentro de la sentencia y la ejecuta por separado.

Para una vista que no tiene funciones de grupo, también se puede forzar que no se integre dentro de la sentencia, usando el hint NO_MERGE. Con este hint, el optimizador resolverá la vista primero y después el resto de la sentencia.

No "reciclar" las vistas

Cuando se está trabajando con vistas, es muy importante darle el uso para el que fueron creadas. Nunca crear una vista con un propósito y utilizarlas posteriormente en sentencias en las que su rendimiento puede no ser el adecuado. Una vista no debe ser utilizada por el mero hecho de que los registros que devuelve cumplen unas condiciones; hay que tener muy en cuenta con qué otras vistas / tablas se va a hacer un join. Este mal uso de las vistas puede ser crítico en el rendimiento de las sentencias, ya que es el optimizador quien decide cómo y cuando van a resolverse si forman parte de una sentencia más compleja.

Sentencias SQL con subselects

Cuando aparecen subselects dentro de una sentencia, podemos encontrarnos con problemas similares a los que se plantean con las vistas, ya que el optimizador decide cuándo resuelve la subselect y si la integra o no con el resto de la sentencia. Además, ciertas subselects, como comprobaciones de existencia, pueden realizarse de un modo muy eficiente usando la cláusula EXISTS en vez del operador IN.

Cuándo se resuelve la subselect

Si una sentencia tiene una subselect, el optimizador puede resolverla de dos modos:
resolver primero la subselect y a continuación el resto de la sentencia (tipo vista), o
integrar la subselect dentro de la sentencia (tipo join).

Si se resuelve primero la subselect, se calcula primero el resultado de ésta y, a continuación, el resto de las condiciones de la sentencia se aplican como un filtro. Si se

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

integra la subselect con la sentencia, las condiciones de la subselect se unen a las del resto de la sentencia. Si se usan las subselects para comprobaciones de existencia, el rendimiento es mejor si se resuelve por separado. En otro caso, es mejor que la subselect se integre dentro de la sentencia. Cuando la subselect tenga funciones de grupo, siempre se resolverá por separado, limitando las opciones que pudiera tener el optimizador para elegir un plan de ejecución.

Por ejemplo, supongamos que tenemos la siguiente sentencia con una subselect que contiene un DISTINCT:

```
select A.valor_A
from A
where A.cod in
(select DISTINCT B.cod
from B
where valor_B1 = 1 and valor_B2 > 1000);
```

Esta subselect se dice que es de tipo vista, porque tiene una función de grupo y ha de resolverse por separado. Suponiendo que tenemos una clave primaria en A para el campo cod (A_PK), el plan de ejecución sería el siguiente:

```
NESTED LOOPS
VIEW
SORT UNIQUE
TABLE ACCESS FULL B
TABLE ACCESS BY ROWID A
INDEX UNIQUE SCAN A_PK
```

En esta sentencia, la subselect actúa como la tabla directora de un NESTED LOOPS join. El optimizador no tiene opción de elegir qué tabla interesa tomar como la directora del NESTED LOOPS. Sin embargo, la sentencia anterior puede reescribirse para evitar la subselect y tener un join de dos tablas:

```
select A.valor_A
```

```
from A, B
where A.cod = B.cod and B.valor_B1 = 1 and B.valor_B2 > 1000;
```

El nuevo plan de ejecución sería el siguiente:

```
NESTED LOOPS
TABLE ACCESS FULL B
TABLE ACCESS BY ROWID A
INDEX UNIQUE SCAN A_PK
```

La tabla directora del NESTED LOOPS sigue siendo B, pero con la nueva sintáxis el optimizador puede decidir qué tabla elige, en función de los tamaños que tengan en cada momento.

Hints para subselects que devuelvan un valor máximo

Es muy frecuente el uso de sentencias con subselects que devuelven el valor máximo (o mínimo) de un campo. En estos casos, puede reducirse considerablemente el número de lecturas si se utilizan hints para decir al optimizador cómo debe ejecutar la sentencia.

Supongamos una tabla A, que entre otros tiene los campos valor, cod y fecha. Creamos un índice concatenado por los campos cod y fecha. Tenemos la siguiente sentencia:

```
select valor
from A A1
where cod = 1 and fecha = (select max(A2.fecha)
from A A2
where A2.cod = A1.cod);
```

El plan de ejecución para esta sentencia es el siguiente:

```
FILTER
TABLE ACCESS FULL A
```

SORT AGGREGATE

INDEX RANGE SCAN A$COD_FECHA

Para resolver la sentencia, se recorre el índice A$COD_FECHA, y ordena todos los valores para extraer el máximo. Ahora bien, los valores que se están ordenando sabemos que ya están ordenados en el índice, luego podríamos indicarle al optimizador que no realice ese paso y que el índice lo recorra de mayor a menor (por defecto, lo hace de menor a mayor). Reescribimos la sentencia y añadimos un hint para mostrarle al optimizador el camino que debe seguir:

```
select /*+ INDEX_DESC(A A$COD_FECHA) */ valor
from A
where cod = 1 and fecha < sysdate and rownum = 1;
```

El plan de ejecución ahora queda así:

COUNT STOPKEY

TABLE ACCESS BY ROWID A

INDEX FULL SCAN DESCENDING A$COD_FECHA

El número de filas devuelto es uno, y esa fila es la que tiene el máximo valor de la fecha para el código solicitado.

Cómo combinar subselects

Una única sentencia puede contener más de una subselect. Cuanto mayor sea el número de subselects, más difícil será integrarlas dentro de un único join. Para evitar esto, es conveniente combinar varias subselects siempre que sea posible.

Supongamos que tenemos tres tablas A, B y C. La tabla A es muy grande y vamos a hacer una actualización masiva sobre ella. Las tablas B y C son tablas pequeñas, que contienen sólo códigos y descripciones. Tenemos la siguiente sentencia:

```
update A
set valor = 123
where cod1 in (select cod from B where desc = 'COD1') and
```

cod2 in (select cod from C where desc = 'COD2') and fecha < sysdate;

Para cada registro de la tabla A, se ejecutarán las dos subselects. Puede evitarse realizar varias subselects por registro combinando las dos subselects en una sola:

update A
set valor = 123
where (cod1, cod2) = ANY
(select B.cod, C.cod from B, C where B.desc = 'COD1' and C.desc = 'COD2') and fecha < sysdate;

El resultado de esta subselect es el producto cartesiano de los registros de B y C que cumplen las condiciones sobre los campos desc. Si B y C son tablas pequeñas, la sobrecarga que supone este join es despreciable si la comparamos con la mejora que se consigue haciendo un una única subselect.

Validaciones de existencia

Una subselect no siempre devuelve filas, en ocasiones sólo se usan para validar si un dato existe o no. En estos casos, pueden usarse los operadores EXISTS y NOT EXISTS, en vez de IN y NOT IN, para mejorar el rendimiento de la sentencia. De esta forma, se evitan accesos innecesarios a la tabla (no se devuelve la fila que tenemos en la tabla, sólo la condición de que exista o no).

La siguiente sentencia:

select valor
from A
where cod IN
(select cod from B);

puede reescribirse de la siguiente manera, usando el operador EXISTS:

select valor
from A

```
where EXISTS
(select 1 from B where B.cod = A.cod);
```

El operador EXISTS sólo necesita validar que exista un registro devuelto por la subselect. Una sentencia con NOT IN puede ser reescrita para usar el operador NOT EXISTS.
La mejora que se puede conseguir usando (NOT) EXISTS puede ser enorme. El operador (NOT) IN hace FULL TABLE SCANS anidados. Sin embargo, usando (NOT) EXISTS, la sentencia puede hacer uso de índices en los campos que aparezcan en el WHERE de la subselect.

Generar código reutilizable

Cuando Oracle va a ejecutar una sentencia, la compara con las que ya se encuentran en la zona de memoria compartida SQL area. Si hay alguna exactamente igual, toma su plan de ejecución y la ejecuta. Si no está en la SQL area, la carga y crea un plan de ejecución antes de que sea realmente ejecutada.
La SQL area es gestionada internamente mediante un algoritmo LRU (least recently used). Cuando es necesario hacer sitio para una nueva sentencia, la que lleva más tiempo sin ser utilizada se descarga. Para una buena gestión de la memoria compartida, el código de nuestra aplicación debe ser reutilizable.

Usar bind variables

Supongamos una sentencia en la que se pregunta siempre por los mismos campos de una tabla, pero se pone la condición de que uno de los campos tenga un valor dado:

```
select * from emp where dept = 1;
select * from emp where dept = 2;
select * from emp where dept = 3;
select * from emp where dept = 4;
```

Cada una de estas sentencias ocupará un lugar diferente en la memoria compartida, cada una de ellas con su plan de ejecución. Esto se puede evitar con el uso de bind variables:

cuando se ejecute esta sentencia, se puede pasar una variable (con un valor asignado previamente) en vez de una constante.

select * from emp where dept = :dept_no;

Para las sucesivas ejecuciones, Oracle encontrará ya la sentencia en la memoria compartida y puede aprovechar su plan de ejecución, en vez de analizarla nuevamente.

Cuando estamos usando CBO, el plan de ejecución puede cambiar dependiendo del valor que se pase dentro de la variable. Esto es debido a que el optimizador sabe si es conveniente o no utilizar un índice, en base a la selectividad de ese índice, el rango de valores para la columna en cuestión, etc.

Texto homogéneo para los SQL

Las sentencias del tipo

select * from emp where dept = :dept_no;

select * from EMP where DEPT = :dept_no;

no son vistas por Oracle como exactamente iguales, debido al uso de mayúsculas / minúsculas y espaciado intermedio. Por tanto, cada una de ellas ocupará una entrada diferente en la memoria compartida: este código no es reutilizable por Oracle.

Bloques anónimos de PL/SQL

Un bloque anónimo de PL/SQL es un fragmento de código de la forma DECLARE ... BEGIN ... END. Contiene sus variables, cursores y sus sentencias SQL. Aunque sintácticamente están permitidos, no es conveniente que formen parte de una aplicación y estén siendo ejecutados frecuentemente. En su lugar, puede considerarse la posibilidad de meterlos dentro de un PROCEDURE o PACKAGE y que queden almacenados como objetos de bases de datos. La ventaja que tiene el código PL/SQL almacenado frente a los bloques anónimos es que ya están compilados. Además, si el DBA lo considera necesario, un PROCEDURE o PACKAGE puede ser fijado en la memoria compartida para evitar que sea descargado por problemas de espacio, quedando fuera del algoritmo LRU.

Transacciones grandes

Una transacción es un conjunto de operaciones que se ejecutan todas (o no) como una unidad. Para terminar la transacción, se puede ejecutar cualquiera de los comandos commit o rollback. Cuando una sesión comienza una transacción, Oracle le asigna un segmento de rollback siguiendo un algoritmo circular del tipo round-robin. La información necesaria para deshacer esa transacción se va guardando en el segmento de rollback asociado. Varias transacciones pueden estar escribiendo simultáneamente en el mismo segmento de rollback, incluso en el mismo extent, pero nunca en el mismo bloque. Una vez terminada la transacción (commit o rollback), el espacio que ocupa su información de rollback queda marcado como reutilizable. Sin embargo, Oracle no lo utiliza de inmediato, el algoritmo round-robin que se utiliza para asignar lo segmentos de rollback intenta maximizar el tiempo que se conserva esa información .

Los segmentos de rollback no sólo se utilizan para escribir información que se utilizaría para dar marcha atrás a la transacción. También permiten construir una vista consistente de los datos que están siendo modificados por una transacción, tal como estaban en un momento en el tiempo anterior al inicio de esa transacción. Supongamos que se ejecuta una SELECT que tarda mucho tiempo en completarse. Antes de que se termine de ejecutar, se comienza una transacción que modifica bloques que necesita leer la SELECT. Si estos bloques no pueden ser reconstruídos (con información de los segmentos de rollback) a la situación en que estaban en el momento en que empezó la SELECT, se produce un error ORA-01555: Snapshot too old, y la SELECT es cancelada. Esta imposibilidad de recrear el estado anterior de los bloques se debe a que la información de los segmentos de rollback ha sido sobreescrita (la transacción terminó, y ese espacio quedó disponible para ser usado).

Este doble uso que hace Oracle de los segmentos de rollback nos obliga a considerar de una forma especial los procesos que se vayan a ejecutar en nuestra base de datos y que realizan actualizaciones masivas (tardan mucho tiempo en completarse y modifican un gran número de bloques). Si el proceso sólo hace commit (o rollback) al final y genera mucha información de rollback, puede que aborte porque su segmento de rollback no tenga espacio suficiente. Esto supone que todo el trabajo que había realizado la transacción hasta ese punto se pierde y hay que volver a empezar (podría demorar un proceso muchas horas). Por

el contrario, si el proceso hace commit (o rollback) con mucha frecuencia, estamos permitiendo reutilizar el espacio que ocupa su información en los segmentos de rollback, con lo que la probabilidad de un ORA-01555 es mayor. Este error también lo puede dar la propia transacción, si es frecuente que visite repetidamente los mismos bloques.

Conocer este funcionamiento nos puede ayudar en el momento de codificar los procesos que realizan grandes transacciones. Si el rollback que genera cabe en un único segmento, podemos asignárselo al comienzo de la transacción (SET TRANSACTION USE ROLLBACK SEGMENT ...) y no hacer commit hasta el final. De todos modos, no debe suponerse que en ese segmento sólo va a escribir una transacción. Si el espacio que hay para segmentos de rollback es insuficiente, habrá que fraccionar la transacción y hacer commits periódicamente. Esto se consigue definiendo un cursor que seleccione los registros a modificar y recorrerlos en un bucle, tal como muestra el siguiente código:

```
DECLARE
CURSOR C IS select rowid from A
where FECHA > sysdate - 365;
cont NUMBER := 0;
reg C%rowtype;
BEGIN
FOR reg IN C LOOP
cont := cont + 1;
update A set CAMPO = 'VALOR' where rowid = reg.rowid;
IF cont > 10000 THEN
COMMIT;
cont := 0;
END IF;
END LOOP;
COMMIT;
END;
```

Aquí se hacen FETCH's entre COMMIT's (fetch across commits). Esta técnica no es ANSI-SQL, pero Oracle la permite si el cursor no es un SELECT ... FOR UPDATE. El inconveniente es que los registros que se van a modificar no son bloqueados previamente, y

otro proceso podría sobreescribir nuestras modificaciones. Si no se tiene la certeza de que no haya otros procesos que puedan interferir, es conveniente realizar un bloqueo antes de modificar los datos. Esto se puede hacer reescribiendo el código anterior del siguiente modo:

```
DECLARE
CURSOR C1 IS select rowid from A
where FECHA > sysdate - 365;
CURSOR C2(r ROWID) IS select rowid from A
where rowid = r FOR UPDATE;
cont NUMBER := 0;
reg1 C1%rowtype;
reg2 C2%rowtype;
BEGIN
FOR reg1 IN C1 LOOP
cont := cont + 1;
FOR reg2 IN C2(reg1.rowid) LOOP
update A set CAMPO = 'VALOR'
where rowid = reg2.rowid;
END LOOP;
IF cont > 10000 THEN
COMMIT;
cont := 0;
END IF;
END LOOP;
COMMIT;
END;
```

Aquí, los registros son bloqueados de uno en uno (se podría hacer con un rango). El COMMIT se hace después de cerrar el cursor que tiene el SELECT ... FOR UPDATE.

Si el cursor no devuelve los registros en el mismo orden en que se encuentran físicamente, la probabilidad de que se intente leer en distintos momentos el mismo bloque aumenta (aumentando la probabilidad del error ORA-01555). Si forzamos a usar siempre el mismo

segmento de rollback, será el propio proceso quien sobreescriba la información de rollback que va generando, aumentando también la probabilidad de este error.
En resumen, a la hora de programar grandes transacciones hay que elegir con qué problema nos vamos a encontrar. A priori, podremos conocer el volumen de nuestra transacción. Si es desmesuradamente grande, deberíamos decidirnos por fraccionarla programando un cursor. Esto evitará que, después de varias horas, nuestra transacción termine mal y haga rollback de todo el trabajo. Pero nos podremos encontrar con el ORA-01555. No se puede garantizar que este error no va a darse, sólo puede minimizarse la probabilidad de que aparezca. Si nuestro proceso está bien diseñado (así como los datos), que se dé este error no hará que se pierda todo el tiempo (y trabajo) transcurrido: debería bastar con relanzarlo y que continúe por donde se quedó.

Accesos a tablas muy grandes

Cuando nuestra aplicación hace accesos a tablas pequeñas, los bloques leídos en el buffer de datos de la SGA tienen una probabilidad muy alta de ser compartidos por varios procesos. La situación ideal sería aquella en que toda nuestra aplicación, con sus datos, pudiese estar cargada en ese buffer: los procesos nunca necesitarían leer de los ficheros de datos.

Según van creciendo las tablas de nuestra aplicación, esta situación ideal ya no es posible. La posibilidad de reusar bloques ya leídos decrece considerablemente. Además, el tener cargados en memoría muchos bloques de una tabla muy grande, y que permanezcan ahí mucho tiempo, puede perjudicar al resto de usuarios. Esto es lo que sucede cuando se leen bloques vía un INDEX RANGE SCAN poco selectivo. Aunque pueda parecer contradictorio, en ciertos casos, un acceso por índice resulta perjudicial para la gestión de la SGA.
Esto hace que las lecturas de tablas muy grandes necesiten un enfoque diferente cuando se quiera mejorar el rendimiento de la aplicación.

Proximidad de los datos

Si se pretende acceder a grandes tablas mediante índices, interesa que los datos estén físicamente ordenados según el criterio en que se vayan a recuperar. Esto no es posible controlarlo totalmente, pero sí podemos aumentar la probabilidad de encontrar los registros relacionados dentro del mismo bloque. Esto se consigue ordenando los registros que se desea insertar antes de ser insertados.

Supongamos una aplicación que recupera filas de una tabla muy grande, y que siempre interesa recuperar aquellos datos que se encuentran dentro de un rango de fechas, con una sentencia del tipo:

select * from tabla where fecha between :fecha1 and :fecha2;

Si hay un índice por el campo FECHA, el optimizador puede decidir utilizarlo para ejecutar esta sentencia (si tenemos RBO, siempre lo usará). Si los datos están muy disgregados, en el peor de los casos se necesitará leer tantos bloques como registros recupera la select. Sin embargo, si cuando se insertaron los datos estaban ordenados por la fecha, la posibilidad de que con un mismo bloque se recupere más de un sólo registro es considerablemente mayor. La necesidad de lecturas físicas en este caso será mucho menor.

INDEX SCANS vs FULL TABLE SCANS

Si se va a leer de tablas muy grandes, no siempre debe asumirse que un acceso por índice va a dar un rendimiento mejor que un full table scan. UNIQUE SCANS y RANGE SCANS de índices que no sean seguidos de accesos a tabla suelen funcionar bien, pero un RANGE SCAN de un índice, seguido de acceso a tabla por RowID puede resultar peor que un FULL TABLE SCAN. Esto es más cierto cuanto más grande sea la tabla.

Supongamos una tabla de 10.000.000 de filas, y una sentencia del tipo:

select * from tabla where campo between :min and :max;

que va a recuperar 1.000.000 de filas (el 10% de la tabla). Para esta tabla, tenemos un índice por CAMPO. Con RBO, siempre se va a usar ese índice; CBO puede decidir que es mejor acceder directamente a la tabla.

Supongamos que en cada bloque del índice caben 100 registros y que en cada bloque de la tabla caben 10. Del índice habría que leer 10.000 bloques (1.000.000 filas / 100 filas por

bloque). Suponiendo que la tabla estuviese físicamente ordenada por CAMPO (este sería el mejor de los casos), tendríamos que leer 100.000 bloques (1.000.000 filas / 10 filas por bloque). Si la tabla no está ordenada, en el peor de los casos habría que leer 1.000.000 de bloques (1 bloque por fila). En total, si la tabla está ordenada, habrá que leer 110.000 bloques (10.000 del índice, más 100.000 de la tabla); si no está ordenada, se necesita leer 1.010.000 bloques (10.000 del índice, más 1.000.000 de la tabla).

Si se accede directamente a la tabla, un full table scan leerá 1.000.000 de bloques. Esto es mejor que el caso menos favorable accediendo por índice. Además, si se accedió por índice, los bloques se mantendrán en la SGA tanto tiempo como sea posible, posiblemente perjudicando al resto de procesos. Si tenemos un tamaño de bloque de 2K, en el mejor de los casos se intentaría conservar en memoria 220M (2K * 110.000 bloques). En cambio, si se hizo un full table scan, sólo se intenta mantener en la SGA tantos bloques como indique el parámetro db_file_multiblock_read_count (suele tener un valor entre 8 y 64). Los datos que cargaron el resto de procesos pueden seguir conviviendo en la memoria.

Crear tablas completamente indexadas

Cuando se tiene accesos a tablas muy grandes, es interesante no realizar dos operaciones para leer los datos (index range scans y table access by rowid), si se puede resolver la sentencia con una sola operación (index only scans). Si las columnas más frecuentemente seleccionadas de una tabla son relativamente estáticas, podemos considerar crear un índice que contenga todas las columnas que aparezcan en la cláusula WHERE, seguidas de las columnas que aparezcan en la SELECT.

Una sentencia del tipo:

```
select campo1, campo2, campo3 from tabla
where campo4 = :var1 and campo5 = :var2;
```

puede resolverse leyendo sólo del índice, si éste contiene las columnas (por este orden) CAMPO4, CAMPO5, CAMPO1, CAMPO2 y CAMPO3.

Crear Hash Clusters

En un hash cluster, el orden en que se insertan los registros en la tabla no importa; la ubicación física del registro se determina en base a los valores de columnas clave. Se aplica una función hash a los valores de la clave de una fila, y Oracle usa el resultado para saber en qué bloque debe ser almacenada. El uso de un hash cluster hace innecesario un índice cuando se busca por los valores de la clave.

Puede ser conveniente usar hash clusters en la siguientes circunstancias:

Se usan sentencias con condiciones de igualdad sobre los campos de la clave (where ID = :valor). Cuando se ejecuta la sentencia, el optimizador aplica la función hash sobre la clave y calcula directamente el bloque en el que se encuentra el registro.

No hay posibilidad de forzar a que los datos estén físicamente ordenados. Si continuamente se recuperan rangos de valores para una columna, y los datos no están ordenados, la posibilidad de reutilizar bloques ya cargados en la SGA es muy baja. En este caso, el uso de un hash cluster puede ser apropiado.

El espacio de almacenamiento no supone un problema para nuestro sistema, ya que un hash cluster necesita, aproximadamente, un 50% más de espacio que una tabla indexada.

Crear tablas particionadas

Si una tabla muy grande puede verse como varias particiones lógicas (registros que comparten el mismo valor para una columna clave), puede considerarse la posibilidad de partirla en varias tablas más pequeñas. Esas tablas pequeñas pueden consultarse juntas haciendo un UNION ALL de varias selects. Esto es lo que se llama particionamiento horizontal.

El uso de tablas particionadas es interesante, por ejemplo, en grandes tablas históricas que tienen que almacenar datos de varios años. Puede definirse como una tabla particionada, cada partición guardando un trimestre. En este caso, la clave del particionamiento sería un campo fecha. Si la aplicación suele consultar esta tabla restringiendo la columna clave a un valor o un rango, eliminar todas las particiones de la tabla menos una puede suponer una drástica disminución del número de bloques necesarios para resolver la consulta. En cambio, si frecuentemente es necesario leer de más de una partición, puede que la mejora no sea considerable.

Por otra parte, el uso de tablas particionadas supone un beneficio importante para la administración ya que:

en caso de desastre, la recuperación puede ser más rápida que si la tabla no está particionada;

1. Es más fácil hacer backup de una partición que de toda la tabla;
2. Puede balancearse la I/O creando cada partición en un disco diferente;
3. Cuando se realizan borrados masivos de registros basados en la columna clave del particionamiento, puede diseñarse para que baste borrar la partición (esto afecta sólo al diccionario de datos, es muy eficiente y no genera log ni rollback).

Implementar Parallel Query

Cuando se están realizando full scans de tablas muy grandes, puede mejorarse el rendimiento si se hace uso de la opción Parallel Query de Oracle. En Oracle7, no se puede paralelizar un INDEX SCAN.

Seguridad de la aplicación

Cuando se está diseñando una aplicación, debe prestarse una atención muy especial a la seguridad que se va a aplicar tanto a los objetos de la aplicación como a los datos. En Oracle, la seguridad se controla mediante el uso de diferentes roles. Hay roles que deben ser específicos del administrador de la base de datos, otros que permiten crear objetos y gestionarlos, y otros que sólo permiten el manejo de datos. Estos últimos son normalmente definidos para la aplicación.

Gestión de los objetos de la aplicación

Los objetos de una aplicación deben de pertenecer a uno o varios usuarios específicos. No deben usarse con este fin usuarios como SYS y SYSTEM, cuya función no es otra que permitir la administración de la base de datos al DBA.

Cuando se cree este usuario, deberá recibir permisos para poder crear todos los objetos que van a componer su esquema (tablas, índices, triggers, procedimientos, paquetes, etc.) y tendrá un control total sobre ellos. Por tanto, no necesitará privilegios del sistema

específicos como SELECT ANY TABLE, ni otros similares. Igualmente, no deberá recibir privilegios de DBA, pues no los necesita para gestionar su propio esquema.

Gestión de los datos de la aplicación

Una vez creado el esquema con los objetos de la aplicación, no interesa que los usuarios finales se conecten usando el propietario de los objetos. Con este fin, deberán crearse usuarios específicos que tengan únicamente permisos para manejar aquellos datos que correspondan a su perfil. Es decir, habrá usuarios que sólamente puedan leer en ciertas tablas para extraer informes. A ellos se les dará permiso de SELECT para esas tablas exclusivamente. Otros, que tengan que modificar datos, permisos de INSERT, DELETE o UPDATE, según corresponda. Este nivel de seguridad, sólo para los datos, depende ya del diseño de la aplicación y su gestión no tiene por qué corresponder necesariamente al administrador de la base de datos.

Dimensionamiento de las tablas

Cuando se va a crear una tabla en una base de datos que se encuentra en producción, hay que realizar una serie de estimaciones para determinar qué tamaño ocupará esa tabla, qué crecimiento va a tener y cuántos procesos van a trabajar concurrentemente sobre ella. Con esta información se calcularán los valores de ciertos parámetros que se darán a la tabla en el momento de su creación. En muchos casos, cambiar posteriormente el valor de esos parámetros puede ser muy complicado si los datos tienen que estar disponibles permanentemente. Por otra parte, si fueron mal calculados, el rendimiento de los procesos que trabajan con ella puede verse degradado debido a problemas de contención. De ahí, la importancia de que estas estimaciones se hagan correctamente.

INITIAL, NEXT y PCTINCREASE

Para evitar que el espacio que ocupa nuestra base de datos esté muy fragmentado, lo ideal es que cada segmento de datos o índices ocupe una única extensión. El tamaño de esta extensión se puede calcular si sabemos cual será el número de registros que va a albergar. Esto es posible para tablas cuyo número de filas va a ser constante. Si la tabla fuese a tener

un crecimiento constante con el tiempo, se debe dar un INITIAL suficiente para almacenar los datos iniciales y un NEXT que permitiese la entrada de los nuevos datos durante un período fijo (un mes, un año). El valor de PCTINCREASE debe ser siempre 0, para que todas las nuevas extensiones se creen del mismo tamaño.

PCTFREE y PCTUSED

Con la combinación de estos dos parámetros, Oracle gestiona cuándo un bloque está disponible para recibir nuevas inserciones. Inicialmente, el bloque está vacío y disponible para insertar registros en él. Podrán insertarse registros mientras que el espacio libre del bloque esté por encima de PCTFREE. Llegado este punto, en ese bloque sólo pueden borrarse y modificarse registros, quedando el espacio libre disponible para las modificaciones que supongan un incremento en el tamaño del registro. Cuando el nivel de ocupación vuelva a estar por debajo de PCTUSED, el bloque vuelve a estar disponible para nuevas inserciones. La suma de estos dos parámetros debe ser inferior a 100. El valor que se les dé debe calcularse en base a la naturaleza de los registros y al uso que se le va a dar a la tabla.

Un PCTFREE muy pequeño no dejará suficiente espacio para modificaciones y puede provocar migración de filas a otro bloque.

Si el PCTFREE es muy alto, la lectura de un bloque cargará muy pocos registros en la SGA.

Si PCTUSED es muy bajo, disminuye la probabilidad de que un bloque vuelva a ser reutilizado para inserciones.

Un PCTUSED muy alto, puede hacer que el bloque esté continuamente entrando y saliendo de la lista de bloques disponibles.

INITRANS

Cuando un proceso está modificando un registro de una tabla, realiza un bloqueo a nivel del registro y, además, ocupa un slot en la cabecera del bloque Oracle donde está ese registro. Si antes de liberar ese bloqueo (commit o rollback), un nuevo proceso intenta modificar

otro registro que está en el mismo bloque, necesita ocupar otro slot en la cabecera de ese bloque. Si en el bloque no quedan slots libres, ni tampoco hay espacio libre para reservar nuevos slots, el segundo proceso debe esperar hasta que el primero libere el slot que está ocupando. El número de estos slots reservados inicialmente cuando se crea la tabla viene dado por el valor de INITRANS. Este parámetro controla la concurrencia a nivel de bloque para procesos que actualizan registros. Deberá tener un valor tan alto como el número de transacciones concurrentes que vayan a estar modificando el mismo bloque. Su valor por defecto es 1.

FREELISTS

Un proceso que está insertando registros en una tabla necesita obtener una lista de bloques donde pueda guardar los registros. Un bloque está o no en esta lista dependiendo de su nivel de ocupación y de los valores de los parámetros PCTFREE y PCTUSED. Cuando a un proceso se le asigna una lista de bloques libres (una free list) ningún otro proceso que esté insertando podrá hacerlo en los bloques que contiene esa lista. Para evitar contención, un valor interesante para FREELISTS será el máximo número de procesos concurrentes que se espera que realicen inserciones en la tabla.

Capitulo II.

CREACION Y MANEJO DE TABLAS

Creación de tablas

Como expusimos en nuestro artículo anterior, en Oracle cada estructura de información se denomina TABLA las cuales, junto a los índices y al diccionario de datos del sistema, componen la base de datos. Por lo tanto, la creación de las tablas en el proceso de programación en Oracle juegan un papel muy importante. En el momento de crear las tablas se definen características a dos niveles: Tabla y Columna, como se muestra a continuación:

A nivel de Tabla

1. *Nombre*: Nombre de la tabla puede ser de 1 a 30 caracteres.
2. *Propietario:* La tabla tiene como propietario al usuario que las crea En nuestro caso somos el usuario *EIDOS*. Otro usuario que desee usar nuestras tablas debe tener autorización para ello y hacer referencia a la tabla como *eidos.clientes* (*propietario.tabla*)
3. *Cantidad de Columnas:* Una tabla puede tener un máximo de 254 columnas.

A nivel de Columna

Nombre: Puede tener de 1 a 30 caracteres.

Tipo de dato y su ancho

CHAR	Máximo de 255. Por defecto 1.
NUMBER	Máximo de 105 dígitos. Por defecto 44.
INTEGER	Numérico sin decimal. Por defecto 38.
DATE	Hasta el 31 de diciembre de 4712.
LONG	Tipo caracter con tamaño variable hasta 65535 bytes. Permite una sola columna LONG por tabla. No se puede usar en subconsultas, funciones o índices.
RAW	Dato en binario puro (imágenes y sonido) con un ancho máximo de 255.
LONGRAW	Igual que *LONG*, pero para almacenar datos en binario puro.

Restricciones: Su función es definir reglas de validación de la columna.

Para facilitar la continuidad del análisis, usaremos como ejemplo las tablas definidas en el artículo anterior: *Clientes* y *VENTAS.*

La definición de *restricciones* al crear las tablas permite establecer reglas de validación de datos, así como los controles necesarios para mantener la integridad referencial entre tablas a través de las columnas claves. Las restricciones que se pueden definir son:

Valor obligatorio: En Oracle existe el concepto de valor nulo (*NULL*), como un valor indefinido o ausencia de valor y que es diferente al numero 0 o al carácter espacio. Por lo tanto, para que una columna siempre tenga valor (sea obligatoria) se define como *NOT NULL.*

Rango de valores: Sirven para chequear que el valor sea mayor a un valor determinado o para que se encuentre entre dos valores.

Clave Primaria: Columnas que identifican de forma única al registro, es un valor único y no nulo (*NOT NULL*). Por ejemplo: el código del cliente es una *clave primaria* que identifica de forma única e irrepetible a cada cliente.

Clave Externa: Columna de la tabla que hace referencia a un valor que tiene que estar registrado en otra tabla. Por ejemplo: la columna código de la tabla VENTAS es una clave externa que hace referencia a un valor de la columna código (clave primaria) de la tabla *Clientes*.

En la versión 6 de Oracle (que dado lo reciente de la versión 7 aún se usa ampliamente) la única restricción que estaba activa era la de valor obligatorio (*NOT NULL*), siendo las otras restricciones sólo declarativas, o sea, que quedaban registradas en la definición de la tabla, pero no se podían activar. En la versión 6, para garantizar la unicidad de la clave primaria, era necesario crear índices con claves únicas, aspecto éste que retomaremos más adelante. En la

versión 7 de Oracle estas restricciones están implementadas, garantizándose la verificación y correción de datos en cualquier momento sin tener que programar estos controles.

Destacadas estas cuestiones veamos, entonces cómo se procede para crear las tablas *Clientes* y *Ventas*.

Tabla Clientes

Objetivo: Ficha con datos para identificar al cliente. Consta del código del cliente (número secuencial), fecha de alta al sistema, nombre, teléfono, dirección y alguna anotación.

Requisitos: Se debe identificar a cada cliente con un código único (clave primaria), registrando su nombre, teléfono y fecha de registro (estos datos son obligatorios). La dirección y anotaciones son campos opcionales.

```
Fuente 1
CREATE TABLE clientes
(
        codigo    integer        NOT NULL
                                        PRIMARY KEY,
        fecha     date   NOT NULL,
        nombre    char(30)   NOT NULL,
        telefono  char(20)   NOT NULL,
        direccion  char(100),
        anotacion  LONG
        );
```

Tabla *Ventas*

Objetivo: Registrar las ventas con al siguiente información: Código del cliente, fecha de la

venta, artículo y valor de la venta.

Requisitos: El número del cliente es una clave externa que hace referencia a la columna codigo en la tabla Clientes. En este caso, todos los datos son obligatorios. Se controla que la columna valor sea mayor a cero.

Fuente 2

```
CREATE TABLE ventas
(
        codigo   INTEGER    NOT NULL
        REFERENCES clientes(codigo),
        fecha    DATE       NOT NULL,
        articulo CHAR(20)   NOT NULL,
        valor    NUMBER(10,2) NOT NULL
                        CHECK (valor>0)
);
```

Las restricciones de Claves Primaria y Clave Externa se definieron a nivel de columna, pero se pueden definir a nivel de tabla, al final de la misma, como se muestra en el fuente 3:

Fuente 3

```
CREATE TABLE clientes
(codigo      INTEGER     NOT NULL,
nombre       CHAR(30)    NOT NULL,
direccion    CHAR(100),
anotacion    LONG,
PRIMARY KEY (codigo));
```

```
CREATE TABLE ventas
(codigo      INTEGER     NOT NULL,
 fecha       DATE        NOT NULL,
 articulo    CHAR(10),
 valor       NUMBER(6,2) NOT NULL
             CHECK (valor>0),
FOREIGN KEY (codigo)  REFERENCES clientes(codigo));
```

La definición de la clave a nivel de tabla es necesaria cuando la misma está formada por más de una columna.

Unicidad de la clave con índices

Para garantizar la unicidad de los valores de la *clave primaria* de la tabla *Clientes* (en la versión 6 donde esta restricción sólo es declarativa y no está activa), se debe crear un índice que garantice la unicidad de la clave principal. Un requisito importante para la unicidad de la clave principal es que las columnas de la clave se definen como *NOT NULL*.

A continuación mostraremos cómo crear el índice *cliente_codigo* para garantizar la unicidad de la clave primaria:

```
CREATE UNIQUE INDEX cliente_codigo
    ON clientes(código);
```

Secuencias: codificación numérica

La codificación numérica del cliente se puede realizar con una secuencia que automáticamente genera los números enteros en orden ascendente, no siendo necesario recordar cuál fue el último número asignado ; esto evita la duplicidad de códigos.

La secuencia es un objeto que genera valores enteros únicos y se emplean para crear claves primarias numéricas, con el uso del siguiente mandato:

```
CREATE SEQUENCE codigo_cliente
                INCREMENT BY 1
                START WITH 1;
```

Para registrar un nuevo código con la secuencia definida anteriormente se usa la pseudo-columna *codigo_cliente.NEXTVAL,* la cual nos dará el siguiente valor que le corresponde a la secuencia, la forma en que esto se realiza se explicará más adelante, cuando analicemos el ingreso de datos.

Para conocer el valor actual de la secuencia, o sea, el último código asignado, se usa la pseudo-columna *codigo_cliente.CURRVAL*, desde la tabla *DUAL* del sistema, cuyo fin es poder consultar pseudo_columnas (como se muestra a continuación):

```
SELECT user,sysdate,codigo_cliente.currval
FROM DUAL;
```

donde:

1.- *user* es el nombre del usuario

2.- *sysdate* es la fecha del sistema

3.- *codigo_cliente.currval* es el último valor asignado a la secuencia.

Ingreso de datos

Una vez creadas las tablas, índices y secuencias, estamos en condiciones de ingresar datos en la tabla.

El ingreso, modificación y eliminación de registros se realiza fundamentalmente con el diseño de pantallas (*formularios*) desde el módulo *SQLFORMS* (que será tema de análisis específico en otro artículo). No obstante, en este artículo veremos el uso de los mandatos *INSERT UPDATE* y *DELETE*.

Para ingresar un nuevo registro debemos ensayar lo que se muestra en la tabla 1:

Nombre de la Tabla (acciones)	**Columnas**
INSERT INTO CLIENTES	
VALUES(	
codigo_cliente.NEXTVAL,	codigo= secuencia
'PINTURERIAS PROPIOS',	Nombre
'45 67 89'	Teléfono
TO_DATE('10/04/95','DD/MM/YY'),	Fecha
'Uruguay 1234',	Dirección
'LIBRERIA'	Anotación
);	

Inserción de nuevos registros

Como se podrá observar, en este ejemplo no se especificó la lista de columnas a insertar, lo que indica que se van a ingresar datos para todas las columnas. Por lo tanto, los valores para

cada columna se tienen que ingresar en el orden en que están definidos en la tabla. Además, es de destacar que la palabra reservada *VALUES* indica la lista de valores a ingresar; que los datos tipo carácter van entre comillas; que la fecha se registra como una cadena de caracteres usando la función *TO_DATE* (encargada de transformar la cadena de caracteres '10/04/95' en fecha, a partir de un formato de fecha especificado -'DD/MM/YY').

También observamos que a la columna *codigo* se le asignó el siguiente valor de la secuencia *codigo_cliente* (*codigo_cliente.NEXTVAL*).

En caso de que sólo se asignaran valores a algunas columnas se debe dar la lista de columnas como se muestra en el fuente 4 correspondiente a la lista de columnas.

Fuente 4

```
INSERT INTO CLIENTES (codigo,nombre,teléfono,fecha)
    VALUES
    (codigo_cliente.NEXTVAL,'CASA AUGE DEPORTES',
    '598768',TO_DATE('15/04/95','DD/MM/YY'));
```

Las columnas a las que se les ingresa información se listan después del nombre de la tabla, en el orden deseado. Las columnas no listadas tendrán valor *NULL*, por ello todas las columnas definidas como obligatorias (*NOT NULL*) deben estar en la lista.

Los siguientes ejemplos muestran posibles errores y sus correspondientes mensajes en el registro de datos:

```
INSERT INTO CLIENTES(codigo,nombre)
    VALUES (codigo_cliente.NEXTVAL,
        'EMPRESA D'
    );
```

Mensaje de error:

ORA-01400: mandatory (NOT NULL) column is missing or NULL during insert.

Intento de registrar cliente con código ya existente:

```
INSERT INTO CLIENTES(codigo,nombre,teléfono)
    VALUES  (1,'EMPRESA TTT','341234');
```

Mensaje de error:

ORA-00001: duplicate key in index

Obsérvese en el caso 2 que la secuencia *codigo_cliente* **no** fue usada al ingresar el valor del código, y sí en el caso 1, provocando error de duplicidad de código. Esto ocurre porque la creación de la secuencia no garantiza la unicidad del código, ya que podemos registrar un código de cliente sin su uso. Sin embargo, la *unicidad* esta garantizada por la definición del índice único visto anteriormente. Si siempre se usa la secuencia la unicidad por supuesto que está garantizada, pero la simple definición de la secuencia no es garantía de su uso.

Listados de registros

A continuación veremos cómo obtener listados para revisar la información registrada, para lo cual seleccionaremos (*select*) registros desde (*from*) una tabla. En realidad el mandato *SELECT* será tema de análisis más detallado en la próxima entrega, por lo que ahora sólo lo trataremos con el objetivo de visualizar los datos ingresados.

Para obtener un listado de todas las columnas y todos los registros de la tabla *Clientes* debemos seguir este procedimiento:

Nombre de la Tabla

```
SELECT * FROM clientes;
```

Donde * = Todas las columnas

El resultado es el que se muestra en la tabla.

Código	Fecha	Nombre	Teléfono	Dirección	Anotación
1	10-Apr-95	Pinturerías propios	45 67 89	Uruguay 1234	Ferretería
2	15-Apr-95	Casa Auge deportes	598768		
3	20-Apr-95	Feria del libro	(0567)845677		
4	30-Apr-95	Club de tenis	905877		

Datos de la tabla *Clientes*

Para listar sólo algunas columnas de la tabla clientes el procedimiento a seguir es:

```
SELECT codigo,nombre FROM clientes;
```

El resultado es el que se muestra en la tabla .

Código	Nombre
1	Pinturerías propios
2	Casa Auge deportes
3	Feria del libro
4	Club de tenis

Datos de la tabla *Clientes* para *Codigo* y *Nombre*

Modificación de registros.

Para modificar valores de la tabla usaremos el mandato *UPDATE*, con el objetivo de modificar el teléfono y la dirección del cliente *Feria del libro*. Para ello, basta con definir:

```
UPDATE clientes
    SET telefono='234567',
    direccion='Andes 945'
WHERE  nombre='Feria del libro';
```

donde:

Clientes es el nombre de la tabla

SET es para indicar el inicio de la lista de columnas y sus nuevos valores.

WHERE garantiza la selección de la fila del cliente.

Es importante destacar que si no se usa la cláusula *WHERE*, se modificará el valor de la columna en todas las filas de la tabla.

Eliminación de registros

La eliminación de registros se realiza con el mandato *DELETE*. El siguiente ejemplo eliminará los clientes con el código cero:

```
DELETE FROM clientes
    WHERE codigo=0;
```

En este caso, si se omite la cláusula *WHERE* serán eliminados todos los registros de la tabla.

SQLPlus

Todas las tareas anteriormente estudiadas se realizan con el módulo *SQLPlus* de Oracle, que trabaja en forma interactiva. A continuación enunciaremos los pasos necesarios para usar SQLPLUS y poder crear tablas, índices o secuencias, así como insertar datos y obtener listados:

1.- Llamar al programa

SQLPLUS

2.- Identificación del usuario

Enter user-name: EIDOS

Enter password:

Si la identificación es correcta se obtiene mensaje de:

Connected to: ORACLE

Si, por el contrario, la identificación es incorrecta se recibe el siguiente mensaje:

ERROR: ORA-01017:

invalid username/password;

logon denied

3.- Indicador en pantalla de que *SQLPlus* está a la espera de la orden:

SQL>_

4.- Escribir los mandatos de creación de tablas, índices y secuencia en un archivo (*TABLAS.SQL*) con el uso del editor.

SQL>edit tablas

5.- Ejecutar los mandatos escritos en el archivo *TABLAS.SQL*

SQL>@tablas

6.- Para salir de *SQLPlus*

SQL>exit

Otras tareas

A continuación examinaremos una serie de mandatos, a nivel de definición de las tablas, gracias a los cuales se puede:

1.- Listar estructuras de las tablas (*DESCRIBE*)

2.- Modificar la estructura de las tablas (*ALTER TABLE*)

3.- Renombrar las tablas (*RENAME*)

4.- Eliminar una tabla (*DROP TABLE*)

5.- Eliminar un indice (*DROP INDEX*)

6.- Consultar las tablas del diccionario

7.- Listado de tablas, Índices y secuencias propiedad del usuario.

Detengámonos en los detalles más significativos de cada una de dichas tareas:

1.- Listar estructura de las tablas (*DESCRIBE*)

Para obtener la estructura (descripción de una tabla) el mandato que se debe emplear es:

```
SQL>DESCRIBE clientes;
```

Name	Null?	Type
NUMERO	NOT NULL	NUMBER(38)
FECHA	NOT NULL	DATE
NOMBRE	NOT NULL	CHAR(30)
TELEFONO	NOT NULL	CHAR(20)
DIRECCION		CHAR(100)
ANOTACION		LONG

Resultado del uso del mandato *DESCRIBE*

Modificar la estructura de las tablas (*ALTER TABLE*)

La modificación de la estructura de las tablas con el uso de *ALTER* permite:

- Añadir nuevas columnas.
- Añadir restricciones a una columna, en este caso la tabla no debe contener datos.
- Modificar el ancho de la columna.
- Modificar el tipo de datos de la columna sólo si la columna no contiene datos o está vacía.
- Modificar al tipo LONG sólo una columna sin restricciones.

El siguiente ejemplo muestra cómo añadir, en la tabla *Ventas*, las columnas *Factura* (para registrar el número de factura) y *Cobro* (tipo carácter con 2 posibles valores, N=NO cobrada, NULL=cobrada) y modificar la columna valor para ampliar su ancho.

```
ALTER TABLE ventas
ADD (
    factura    integer,
    cobro      char
)
MODIFY (
    valor number(10,2)
);
```

Renombrar las tablas (*RENAME*)

Para cambiar el nombre de la tabla *Clientes* a E*MPRESAS* se usa el siguiente mandato:

```
SQL>RENAME clientes TO empresas;
```

Eliminar una tabla (*DROP TABLE*)

Le eliminación de la tabla es como sigue:

SQL>DROP TABLE clientes;

En este caso se eliminan, también, todos los índices de la tabla.

Eliminar un índice (*DROP INDEX*)

SQL>DROP INDEX cliente_codigo;

Consultar las tablas del diccionario.

Toda la información de las tablas está registrada en el diccionario del sistema (*Data Dictionary*), que son tablas especiales que se crean en la instalación de ORACLE (que son administradas por el sistema).

Para consultar la lista de tablas que componen el diccionario se escribe:

```
SQL>HELP DATA DICT
```

Gracias a lo cual se muestra una lista con la información de la tabla .

Nombre de la tabla	Descripción

ACCESSIBLE_COLUMNS	columns of all tables, views, and clusters
ACCESSIBLE_TABLES	tables and views accessible to the user
AUDIT_ACTIONS	maps action type numbers to action type names
ALL_INDEXES	descriptions of indexes on accessible
ALL_SEQUENCES	descriptions of the user's own sequences
ALL_TABLES	description of tables accessible to the user
.....	
USER_TABLES	descriptions of the user's own tables
USER_TAB_COLUMNS	columns of the user's tables, views, and clusters
USER_TAB_GRANTS	grants on objects where the user is the owner, grantor, or grantee

Consulta de las tablas que componen el diccionario

También podemos ver la estructura de una tabla del diccionario como se muestra a continuación:

```
SQL>DESCRIBE ALL_TABLES;
SQL>DESCRIBE all_indexes;
SQL>DESCRIBE all_sequences;
```

Listar las tablas, índices y secuencias definidas por un usuario

Para las tablas:

```
SQL>SELECT TABLE_NAME "TABLA"
    FROM ALL_TABLES
    WHERE OWNER='EIDOS';
```

Resultado: *Clientes* y *Ventas*

Para los índices:

```
SQL>SELECT table_name,index_name
    FROM all_indexes
    WHERE owner='EIDOS';
```

Resultado: *Clientes* (con índice *Cliente_Nombre* y *Cliente_Numero*) y *Ventas* (con índice *Venta_Numero*)

Para las secuencias:

```
SQL>SELECT sequence_name
    FROM all_sequences
    WHERE sequence_owner='EIDOS';
```

Resultado: el nombre de la secuencia usada (*SEQUENCE_NAME*) *Codigo_Cliente*

Como hemos visto, la creación de las tablas constituye el fundamento del diseño de cualquier sistema a desarrollar en Oracle. Una vez definida las tablas el paso lógico siguiente es conocer las técnicas para realizar un adecuado uso de la información contenida en el sistema. Por ello, el próximo artículo lo dedicaremos al lenguaje de Consulta *SQL*.

Oracle basico(II): creación y manejo de tablas

Con el artículo anterior iniciamos una entrega de Oracle Básico comenzando con el tema de creación y manejo de tablas. Ahora pasaremos a estudiar la consulta y selección de registros con el

lenguaje estándar para bases de datos relacionales *SQL* (*Structured Query Languague = Lenguaje de Consulta estructurado*).

La ventaja principal del SQL, desde mi punto de vista, es su capacidad de combinar sencillez y facilidad con potencia y eficiencia, conteniendo un conjunto de herramientas que optimizan las consultas.

Vale la pena destacar que, aunque los conceptos a estudiar son específico de *ORACLE*, también son útiles para cualquier programador que esté trabajando con algún software que contenga SQL.

Sentencia *SELECT*

Como ya sabemos, la herramienta fundamental de SQL es la sentencia *SELECT*, que permite seleccionar registros desde las tablas de la Base de Datos, devolviendo aquellos que cumplan las condiciones establecidas y pudiendo presentar el resultado en el orden deseado.

Primeramente estudiaremos la forma básica de la sentencia SELECT, que esta formado por:

```
SELECT  Lista...
FROM    Tabla, Tabla...
WHERE   Condiciones...
        ORDER BY Expresión,Expresión,...
        ;       Fin de la sentencia.
```

Donde:

La orden SELECT puede contener:

- Columnas: nombre, telefono
- Expresiones y funciones: *SYSDATE-fecha,UPPER(direccion)*
- Pseudo-Columnas del Sistema: *SYSDATE,USER*.
- Asterisco: Todas las columnas.

La orden *FROM* identifica la lista de tablas a consultar. Si alguna de las tablas a consultar no es propiedad del usuario, debe especificarse el nombre del propietario antes que el nombre de la tabla en la forma *nombre_propietario.nombre_tabla.*

La orden *WHERE* decide los registros a seleccionar según las condiciones establecidas, limitando el número de registros que se muestran.

La orden *ORDER BY* indica el orden en que aparece el resultado de la consulta.

Ilustremos lo explicado hasta el momento con el ejemplo del fuente 1, donde consultaremos las ventas realizadas en los últimos 10 días, mostrando el nombre del cliente, artículo vendido y su valor.

Fuente 1

```
SELECT nombre,articulo,valor      Lista nombre del cliente,
                                  nombre del artículo y
                                  el valor de la venta.
    FROM                          clientes,ventas
                                  Tablas con la información de clientes y ventas.
    WHERE clientes.codigo=ventas.codigo
                                  Establece la relación, según código de cliente,
                                  entre las tablas clientes y ventas.
          and sysdate-ventas.fecha>=10
                                  Consulta las ventas de los últimos 10 días.
    ORDER BY nombre               Ordenar el listado por nombre del cliente.
;                                 Fin de la sentencia.
```

El resultado de esta sentencia SELECT sería el de la tabla 1:

NOMBRE	ARTICULO	VALOR

CASA DEPORTES	AUGE	PAPEL	330.0
CASA DEPORTES	AUGE	DISKETTE	33.0
CLUB DE TENNIS		PAPEL	500.5
...			
CLUB DE TENNIS		PAPEL	100.5
FERIA DEL LIBRO		PAPEL	310.0
PINTURERIAS PROPIOS		PAPEL	220.5
PINTURERIAS PROPIOS		DISKETTE	20.5

Resultados de la sentencia SELECT del fuente 1

Obsérvese que las columnas que tienen el mismo nombre en ambas tablas se diferencian escribiendo el nombre de la tabla antes que el nombre de la columna, como en el caso de *ventas.fecha*, *ventas.codigo* y *clientes.codigo*.

Operadores lógicos

Para construir la condición de la consulta necesitamos conocer los operadores lógicos, por eso a continuación damos una lista de los operadores más usados, agrupados en cuatro grupos:

1. Valor único:Comprueban un valor simple.
2. Lista de valores:Comprueban más de un valor.
3. Combinaciones lógicas:Combinan expresiones lógicas.

4. Negación:Invierte el resultado de la expresión con operadores de valor único o de lista de valores.

Valor único

> < >= <= =

Operadores clásicos de comparación:

mayor, menor, mayor e igual, menor e igual, igual a.

!= <> ^=

Operador "Distinto de" en sus tres formas.

IS NULL

Comprueba la ausencia de datos (valor nulo).

No se puede usar la comparación = NULL.

LIKE

Selecciona registros según el reconocimiento de un patrón de consulta.

Lista de valores

BETWEEN valor AND valor

Comprueba que el valor se encuentre en el rango de valores.

```
IN (valor,...,valor)
```

Verifica si el valor pertenece a la lista de valores.

Combinaciones lógicas.

```
AND
```

Retorna Verdadero si todas las condiciones son verdaderas.

```
OR
```

Retorna Verdadero si alguna de las condiciones es verdadera.

Negación

```
NOT
```

Invierte el resultado de una expresión lógica, por ejemplo.

```
IS NOT NULL

NOT BETWEEN valor AND valor

NOT IN  (valor,...,valor)

NOT LIKE
```

38.363

A continuación mostramos algunas consultas con el uso de diferentes operadores lógicos:

- Clientes a los que no se les ha registrado su dirección.

```
SELECT nombre,telefono
FROM        clientes
WHERE       direccion IS NULL;
```

- Clientes dados de alta en los últimos 10 días.

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  fecha BETWEEN sysdate-10
     AND sysdate;
```

001.01 AHA un

- Datos de los Clientes que pertenecen a una lista de clientes.

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre
      IN ('PINTURAS','CASA DE DEPORTES')
;
```

- Clientes dados de altas en lo que va del mes y cuyo nombre comience con la letra P u otra letra mayor o su teléfono contenga el código (0567). Ver fuente 2

Fuente 2

```
SELECT nombre,direccion,telefono,fecha
FROM   clientes
WHERE
fecha BETWEEN
        to_date('01/'||to_char(sysdate,'MM/YY'),'DD/MM/YY')
        AND
        sysdate
        AND (nombre >= 'P' OR  telefono LIKE '%(0567)%');
```

El manejo de fecha es una de las capacidades de mayor variedad e interés en ORACLE por las posibilidades que presenta en el almacenamiento, cálculo y presentación de fechas. Por eso, en el último ejemplo damos un vistazo a algunas funciones útiles en el uso de fechas como son:

```
to_char(sysdate,'MM/YY')
```

Devuelve una cadena de caracteres de la forma mes/año de la fecha actual.

```
'01/'||to_char(sysdate,'MM/YY')
```

Forma la cadena de caracteres *01/mes/año* que representa el primer día del mes. El operador || se usa para unir o concatenar cadenas de caracteres.

```
...
to_date('01/'||to_char(sysdate,
                'MM/YY'
               ),
     'DD/MM/YY')
...
```

Convierte la cadena de caracteres *01/mes/año* al tipo fecha.

Patrón de consulta

Una de las herramientas lógicas más poderosas de SQL es el reconocimiento de un patrón de consulta, instrumento éste que permite la búsqueda por nombre, dirección u otro dato parcialmente recordado. Los patrones de consulta juegan un papel importante en el momento de realizar consultas, ya que es común que necesitemos encontrar un texto y no recordemos exactamente cómo fue ingresado. Con el uso del operador *LIKE* podemos comparar patrones y ubicar un texto, independientemente de la posición en que se encuentre.

Para la definición del patrón de consulta existen dos tipos de caracteres especiales:

% (signo de porcentaje) llamado *comodín*, representa cualquier cantidad de espacios o caracteres en esa posición. Significa que se admite cualquier cosa en su lugar: un caracter, cien caracteres o ningún caracter.

_ (signo de subrayado) llamado *marcador de posición*, representa exactamente una posición e indica que puede existir cualquier caracter en esa posición.

En los fuentes 3, 4 y 5 detallamos tres ejemplos de consulta con el Operador *LIKE*:

Listar los clientes cuya dirección contengan la palabra *URUGUAY* independientemente de su ubicación:

Fuente 3

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  direccion LIKE '%URUGUAY%';

Listar los clientes  cuyos teléfonos tienen comienzan con el código de área el 0722

SELECT nombre,direccion,telefono
FROM   clientes
WHERE  telefono LIKE '(0722)%';
```

Listar los clientes cuyo nombre terminan con la palabra *LIBRO*:

Fuente 4

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre LIKE '%LIBRO';
```

Listar los clientes que tengan la palabra *LIBRO* a partir de la 5ª posición en el nombre.

Fuente 5

```
SELECT nombre,direccion,telefono
FROM   clientes
WHERE  nombre LIKE '____LIBRO%';
```

Agrupamiento de datos

SQL proporciona una forma eficiente para manejar la información con el agrupamiento de datos a través de la formación de grupos y las funciones correspondientes, dando la posibilidad de procesar no solo registros individuales como hemos hecho hasta ahora. También podemos agrupar registros por un criterio determinado, como por ejemplo, agrupar por clientes las ventas realizadas.

Cada grupo tendrá como resultado de la consulta una fila resumen que contiene la información del grupo.

Para la formación de grupos adicionamos, a la forma básica de la sentencia *SELECT* vista anteriormente, la orden *GROUP BY* ubicada antes de *ORDER BY*, como se muestra a continuación:

```
SELECT    Lista...
FROM      Tabla, Tabla...
WHERE     Condiciones
GROUP BY  Expresión, Expresión,...
          ORDER BY    Expresión, Expresión,...
;         Terminador
```

Las funciones para el procesamiento de grupos son:

COUNT(columna) Cantidad de registros en que la columna tiene valores no nulos.

COUNT(*) Cantidad de registros que hay en la tabla, incluyendo los valores nulos.

MIN(columna) Valor mínimo del grupo.

MAX(columna) Valor máximo del grupo.

SUM(columna) Suma los valores del grupo.

AVG(columna) Calcula valor medio del grupo, sin considerar los valores nulos.

La lista de columnas a mostrar en la consulta puede contener las funciones de grupo, así como la columna o expresión usada para formar los grupos en la orden *GROUP BY*. En una misma consulta no se pueden mezclar funciones de grupo con columnas o funciones que trabajan con registros individuales.

Las ventas por cliente es un buen ejemplo para mostrar el uso de los grupos. En el siguiente caso se hace un resumen de ventas por cliente, con la cantidad de ventas, valor mínimo, medio y máximo, así como la suma total de ventas. La formación del grupo será por el nombre del cliente y la columna a cuantificar para cada grupo será el valor de las ventas.

```
SELECT nombre "CLIENTE",
   COUNT(valor) "VENTAS",
   MIN(valor) "MINIMA",
   AVG(valor) "MEDIA",
   MAX(valor) "MAXIMA",
   SUM(VALOR) "TOTAL"
  FROM clientes,ventas
  WHERE clientes.codigo=ventas.codigo
  GROUP BY nombre
;
```

CLIENTE	VENTAS	MINIMA	MEDIA	MAXIMA	TOTAL
CASA AUGE DEPORTES	3	33.0	138.667	330.0	416.0
CLUB DE TENNIS	3	100.5	237.167	500.5	711.5
FERIA DEL LIBRO	3	110.0	203.333	310.0	610.0
PINTURERIAS	3	20.5	90.500	220.5	271.5

Subconsultas

Otro aspecto de fácil diseño y uso que muestra una vez más las posibilidades de SQL son las subconsultas.

Subconsulta es aquella consulta de cuyo resultado depende otra consulta, llamada principal, y se define como una sentencia *SELECT* que esta incluida en la orden *WHERE* de la consulta principal. Una subconsulta, a su vez, puede contener otra subconsulta y así hasta un máximo de 16 niveles.

Las particularidades de las subconsultas son:

1. Su resultado no se visualiza, sino que se pasa a la consulta principal para su comprobación.
2. Puede devolver un valor único o una lista de valores y en dependencia de esto se debe usar el operador del tipo correspondiente.
3. No puede usar el operador *BETWEEN*, ni contener la orden *ORDER BY*.
4. Puede contener una sola columna, que es lo más común, o varias columnas. Este último caso se llama subconsulta con columnas múltiples. Cuando dos o más columnas serán comprobadas al mismo tiempo, deben encerrarse entre paréntesis.

Expliquemos como se construye una subconsulta con el siguiente ejemplo, donde necesitamos saber ¿cuál fue la mayor venta realizada?. Para ello, diseñemos una subconsulta que busque el valor máximo de venta con el uso de la función *MAX(valor)* y una consulta principal que muestre las ventas iguales al máximo valor encontrado por la subconsulta. Veamos el fuente 7.

Fuente 7

```
SELECT nombre,articulo,valor
FROM clientes,ventas
WHERE valor = ( Subconsulta para buscar
        SELECT MAX(valor)       el valor máximo de venta.
```

```
        FROM ventas
      )
  AND clientes.codigo=ventas.codigo
;
```

NOMBRE: CLUB DE TENNIS
ARTICULO: PAPEL
VALOR: 500.5

Otra aplicación clásica de la subconsulta es cuando deseamos saber las ventas de un artículo realizadas por encima de su venta promedio. En este caso, es necesario realizar los pasos mostrados en el fuente 8:

Fuente 8

```
SELECT  nombre,valor "PAPEL"
FROM    clientes,ventas
WHERE   clientes.codigo=ventas.codigo
    AND articulo='PAPEL'
    AND valor >
                  (                          Subconsulta de
                    SELECT AVG(valor)        venta promedio de
                    FROM   ventas            papel.
                    WHERE  articulo='PAPEL'
                  )
ORDER BY valor DESC          Ordenado por valor de venta
;                            en forma descendente
```

NOMBRE	PAPEL
CLUB DE TENNIS	500.5
CASA AUGE DEPORTES	330.0
FERIA DEL LIBRO	310.0

ventas por encima de su venta promedio

Grupos con subconsulta

Hasta el momento estudiamos por separado un conjunto de herramientas de SQL, viendo en cada caso sus posibilidades. Ahora pasaremos a ver la combinación de grupos y subconsultas, lo que multiplica las posibilidades de SQL en cuanto al rendimiento en el diseño de consultas complejas se refiere, las cuales se pueden realizar en forma sencilla y con pocas líneas de código.

Para combinar grupos con subconsulta debemos incluir en la sentencia *SELECT* la orden *HAVING*, que tiene las siguientes características:

1. Funciona como la orden *WHERE*, pero sobre los resultados de las funciones de grupo, en oposición a las columnas o funciones para registros individuales que se seleccionan mediante la orden *WHERE*. O sea, trabaja como si fuera una orden *WHERE*, pero sobre grupos de registros.

2. Se ubica después de la orden GROUP BY.

3. Puede usar una función de grupo diferente a la de la orden SELECT.

El ejemplo a diseñar para nuestra aplicación es la consulta ¿cuál fue el artículo más vendido y en qué cantidad?. En este caso, la orden *HAVING* de la consulta principal selecciona

aquellos artículos (*GROUP BY*) que tienen una venta total (*SUM(valor)*) igual a la mayor venta realizada por artículo (*MAX(SUM(valor))*) que devuelve la subconsulta.

La sentencia *SELECT* y el resultado de nuestra consulta sería la del fuente 9 y la tabla 5:

Fuente 9

```
SELECT articulo "ARTICULO MAS VENDIDO",SUM(valor) "VENTA"
    FROM ventas
    GROUP BY articulo
    HAVING SUM(valor) =
    (                              Subconsulta para buscar
      SELECT MAX(SUM(valor))       el artículo más vendido
      FROM ventas                  con la formación de grupos
      GROUP BY articulo            por artículo.
    )
;
```

ARTICULO VENDIDO	MÁS VENTA
PAPEL	1872

ventas por encima de su venta promedio

Indices

El índice es un instrumento que aumenta la velocidad de respuesta de la consulta, mejorando su rendimiento y optimizando su resultado. El manejo de los índices en *ORACLE* se realiza de forma *inteligente*, donde el programador sólo crea los índices sin

tener que especificar, explícitamente, cuál es el índice que va a usar. Es el propio sistema, al analizar la condición de la consulta, quien decide qué índice se necesita. Por ejemplo cuando en una consulta se relacionan dos tablas por una columna, si ésta tiene definido un índice se activa, como en el caso cuando relacionamos la tabla de clientes y ventas por la columna código para identificar al cliente (*WHERE clientes.codigo=ventas.codigo*)

La identificación del índice a usar está relacionada con las columnas que participan en las condiciones de la orden *WHERE*. Si la columna que forma el índice está presente en alguna de las condiciones éste se activa. No obstante, existen casos en que la presencia de la columna no garantiza el uso de su índice, ya que éstos son ignorados, como en las siguientes situaciones cuando la columna indexada es:

Evaluada con el uso de los operadores *IS NULL* o *IS NOT NULL*.

```
SELECT nombre,articulo,valor
FROM          clientes,ventas
WHERE  nombre IS NOT NULL;
```

Modificada por alguna función, excepto por las funciones *MAX(columna)* o *MIN(columna)*.

```
SELECT nombre,articulo,valor
FROM          clientes,ventas
WHERE UPPER(nombre)>''
;
```

Usada en una comparación con el operador *LIKE* a un patrón de consulta que comienza con alguno de los signos especiales (% _).

```
SELECT nombre,articulo,valor
        FROM        clientes,ventas
        WHERE  nombre
        LIKE '%DEPORTE%'
;
```

Finalmente debemos aclarar que los registros cuyo valor es *NULL* para la columna indexada, no forman parte del índice. Por lo tanto, cuando el índice se activa estos registros no se muestran como resultado de la consulta.

Con lo estudiado en nuestros dos primeros artículos sobre *Oracle Básico*, aprendimos a crear nuestras tablas e índices y a construir las consultas. Por lo tanto, ya tenemos los elementos necesario para analizar las particularidades del *Diseño de Pantallas* con el módulo *SQLFORMS*, con el objetivo de ingresar, modificar, eliminar y consultar las tablas en forma amigable y en la medida de las necesidades del usuario.

Diseño de pantallas con SQLForms

SQLForms es la herramienta de *Oracle* que permite, de un modo sencillo y eficiente, diseñar pantallas para el ingreso, modificaciones, bajas y consultas de registros. El usuario podrá, una vez definida la forma, trabajar con ella sin necesidad de generar códigos, dado que Oracle trae incorporado un conjunto de procedimientos y funciones asociados a las teclas de funciones, como por ejemplo la tecla [F7], que se usa para iniciar una consulta.

El objetivo de este artículo es el estudio de los conceptos básicos de *SQLForms*, a partir de los cuales el lector estará en condiciones de profundizar independientemente con el la documentación existente sobre Oracle, que es completa, voluminosa y con ejemplos muy ilustrativos.

Forma

La forma elegida para el diseño es la de *Cliente-Ventas*, cuyo objetivo, como se muestra en la siguiente figura, es mostrar los datos básicos del cliente y las ventas realizadas:
Las tablas 1 y 2 representan la Forma *Cliente-Ventas*

Código	Fecha	Nombre	Teléfono	Dirección	Anotación
3	27-09-95	Feria del libro	234555	Canelones 1800	

Tabla 1: Bloque Cliente

Fecha	Artículo	Valor
20/09/95	Papel Fanfold	110
11/09/95	Disquete	190
24/08/95	Papel Fotocopia	310

Tabla 2: Bloque Ventas

La forma se organiza en bloques de información, donde cada uno tiene asociado una tabla de datos y las columnas seleccionadas. La forma puede ocupar una o varias pantallas. En el ejemplo, como se puede observar, ocupa una pantalla.

Bloque

En nuestro ejemplo la forma está compuesta por dos bloques: *Cliente* y *Ventas*. A continuación damos la descripción de cada uno de ello, con su correspondiente definición en *SQLForms*.

Cliente

Objetivo: Ficha básica con datos del cliente.
Tabla: CLIENTES.
Registros: Presentación simple, un registro por cliente.
Tipo: Bloque Principal (*Master Block*).
Orden: Por Nombre del cliente (*ORDER BY NOMBRE*).

Pantalla de definición:

Block: CLIENTE Records Array Size:
Table: CLIENTES Displayed: 1 [ ] Prim Key
Sequence Number: 1 Buffered: [ ] In Menu
Lines per: [ ] Column Sec
Default Where/Order By: ORDER BY NOMBRE

Ventas

Objetivo: Ventas realizadas a un cliente.
Tabla: VENTAS.
Registros: Presentación Múltiple, varios registros por cliente, donde cada registro ocupa una línea.
Tipo: Bloque Detalle, cuya información detalla las ventas del cliente representado en el bloque Principal. La relación entre bloques puede establecerse por uno o más campos. En este caso el campo CODIGO del cliente es el que relaciona

ambos bloques. Por eso definimos como condición de relación CLIENTE.CODIGO = VENTAS.CODIGO.

Orden: Por fecha de venta en forma descendiente (ORDER BY FECHA DESC).

Pantalla de definición:

Block: VENTAS	Records	Array Size: 3
Table: VENTAS	Displayed: 5	[ ] Prim Key
Sequence Number: 2	Buffered: 5	[ ] In Menu
	Lines per: 1	[ ] Column Sec
Default Where/Order By:	ORDER BY FECHA DESC	
Master Block: CLIENTE	[X] Delete Details	
Join Condition	CLIENTE.CODIGO = VENTAS.CODIGO	

Campo

Los datos de la forma se llaman campos, pudiendo los mismos representar columnas de la tabla o variables de memoria. La identificación del campo está compuesta por el nombre del bloque y el nombre del campo, como por ejemplo :*CLIENTE.CODIGO* y :*VENTAS.CODIGO*.

En el momento de crear la forma se determinan para cada campo:

Definiciones básicas

1. Nombre.
2. N° de orden.
3. Tipo de dato.
4. Ancho del campo, consulta y visualización.
5. Posición en pantalla.

Definiciones avanzadas

Formato de presentación. El formato de presentación del campo tipo fecha es *DD-MON-YY*

Capitulo III

SQLReport

En principio me había propuesto tratar dos temas en esta entrega: reportes (informes) y menús. Pero, dada la extensión del primero de ellos, decidí dejar el diseño de menús para el siguiente artículo.

El módulo *SQLReport* de Oracle realiza de forma flexible, sencilla y eficiente la creación de reportes, informes o listados permitiendo, entre otras facilidades, la visualización previa por pantalla con una gran variedad en estilos de presentación.

Definiciones básicas

Para adentrarnos en el tema primero veremos las definiciones básicas, fundamento del diseño del reporte en Oracle:

Consulta

Define las columnas y filas de una o varias tablas que serán emitidas en el reporte, así como su orden de presentación. Una consulta puede estar subordinada a otra consulta principal, relacionadas por una o varias columnas.

Grupo

El grupo es una sección del reporte que representa al conjunto de columnas de la consulta, como una unidad, para determinar su ubicación en el reporte y su forma de presentación.

Para cada grupo se definen los siguientes atributos:

- Ubicación.
- Forma de presentación.
- Texto de cabecera y final.
- Título de las columnas.
- Ubicación de las columnas dentro del grupo.

En el momento de la definición de una consulta se crea, de forma automática, un grupo que contiene todos las columnas presentes en la lista de la sentencia *SELECT*. El nombre de este grupo se define como el nombre de la consulta, precedido por los caracteres *G_*. Por ejemplo, más adelante veremos com

Capitulo IV

Instalación del Paquete de Oracle

Para una adecuada instalación del Sistema de Base de Datos que proporciona Oracle 7.0, deben seguirse los siguientes pasos fundamentales:

- Introducir el Disco Compacto de Instalación de Oracle 7.0 en la Unidad de CD-ROM de la computadora.
- Una vez cargado el instalador (puesto que este es auto ejecutable), seleccionar el idioma (en este caso, el castellano) y luego dar aceptar (OK) a las opciones de configuración de las rutas de instalación (Los paths, por ejemplo: C:\ORAWIN95\).
- Cuando la ventana de programas se abra, deben escogerse las siguientes opciones, haciendo colapsar (es decir, abrir cada ítem que posee a su lado izquierdo el signo más (+)) para desplegar las subopciones del mismo. Para un adecuado funcionamiento del Sistema SIDUQ, se deberán escoger los programas que listaremos a continuación:
    - Programa Runtime de la Opción Forms
    - Programa Runtime de la Opción Reports
    - Aplicación SQL Net para Clientes, y
    - Aplicación SQL Net para Protocolo TCP/IP

- Subsecuentemente se desplegará un cuadro de diálogo que nos preguntará que opciones de configuración de protocolos de Red debemos escoger para el funcionamiento de las Bases de Datos Remotas. Debe hacerse click en la marca de verificación ("chulito") en la primera y tercera opción y luego en el botón Aceptar para proseguir con la instalación del paquete.
- Después de escogidas todas las opciones, el programa se encargará de instalar en el disco rígido de la computadora todos los archivos, extensiones y controladores necesarios para el apropiado funcionamiento del Sistema SIDUQ. Una vez

finalizada la copia de archivos, debe de hacerse click en el botón ACEPTAR del cuadro de diálogo que nos indica que la instalación ha finalizado.

- Automáticamente, se regresará a la ventana de selección de programas del Oracle 7.0 (en donde fueron escogidos los ítems para la instalación) y allí para finalizar, debe de hacerse click en el botón SALIR.
- Microsoft Windows se encargará de abrir las ventanas en donde se localizarán los accesos directos (atajos) y una vez abiertas dichas ventanas, debemos hacer click derecho sobre el ícono de ejecución del programa y del menú de cortina escoger la opción copiar. Luego de minimizar las ventanas, se debe hacer click derecho sobre el Escritorio y escoger la opción "Pegar Acceso Directo". Desde aquí podrá ejecutarse directamente cada vez que se requiera.
- Finalmente para que el Forms Runtime pueda ejecutar apropiadamente el acceso a la Base de Datos, debe efectuarse la siguiente modificación al acceso directo:

  - Haga click derecho con el puntero sobre el acceso directo del programa ejecutable.
  - Escoja la opción Propiedades y una vez abierto el el cuadro de diálogo haga click en la etiqueta Acceso Directo.
  - En la caja de texto "Destino", donde está escrito la ruta que Windows sigue para ejecutar el programa, escriba después de esta y seguido con un espacio de por medio, la siguiente sentencia: "C:\regisglg\formas\inicial" y haga click en el botón Aceptar para terminar.

COMO CONFIGURAR EL ODBC DE ORACLE

El ODBC (Open Database Connectivity) es un programa de interface de aplicaciones (API) para acceder los datos en sistemas manejadores de base de datos tanto relacional como no relacional, utilizando para ello SQL (Lenguaje de Consulta estructurado).

El driver procesa llamado de funciones que contienen sentencias SQL que consultan los datos fuentes y retorna resultados a las aplicaciones. Los drivers son también responsables para interactuar con algunas capas de software necesarias para acceder los datos fuentes.

La arquitectura de como trabajan las aplicaciones que se comunican vía ODBC con la base de datos Oracle es la siguiente:

Aplicación

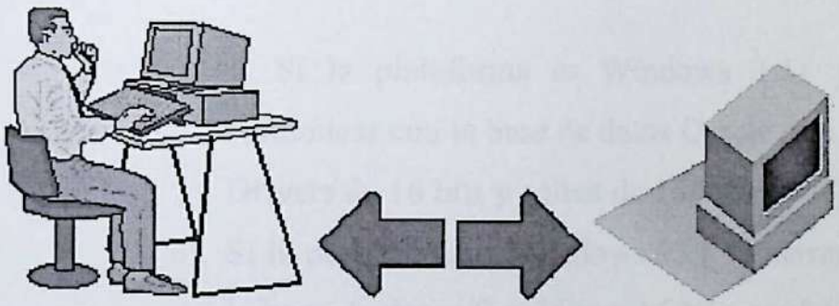
ODBC Drivers

Oracle7 Driver

OCI Layer Oracle7 RDBMS Oracle7 RDBMS

SQLNET Driver SQLNET Listener SQLNET Listener

SOFTWARE DE RED (TCP/IP,DECNET,SPX/IPX)



CLIENTE **SERVIDOR**

¿COMO OBTENERLO ?

1. Visitando en INTERNET la dirección de oracle: www.oracle.com
2. CDROM de Oracle Client Software V7.3.2.2
3. CDROM de Oracle Server for Windows NT.

¿QUE SE DEBE TENER EN CUENTA ?

1. A cuantos bits esta trabajando el equipo donde quiere instalar.

2. Bajo que plataforma esta trabajando. (Windows 3.1, windows95, NT)

3. Espacio suficiente. (55M mínimo).

4. A cuantos bits trabaja la herramienta que se quiere comunicar con la base de datos Oracle.

5. La versión de la Base de Datos Oracle a la cual se quiere conectar.

6. La versión de SQLNET que se esta utilizando.

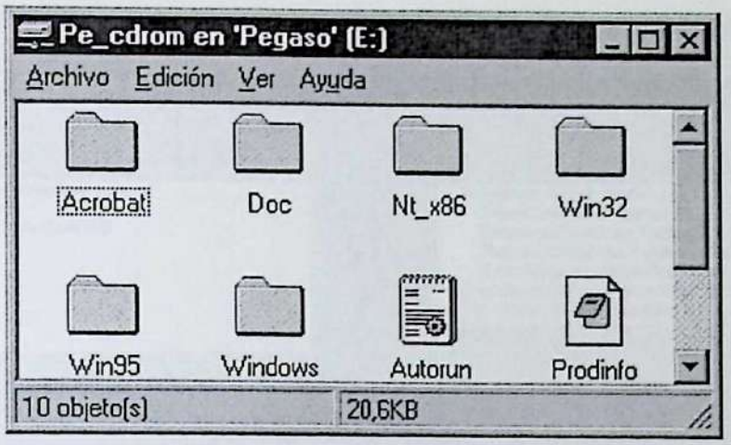
7. Verificar que SQL*NET este configurado en el servidor.

¿CUAL INSTALAR ?

- 1. Si la plataforma es Windows 3.11 y la herramienta que se quiere comunicar con la base de datos Oracle es a 16 bits, se debe instalar el ODBC Drivers de 16 bits y sqlnet de 16 bits.
- Si la plataforma es Windows 95 y la herramienta que se quiere comunicar a la base de datos Oracle es a 16 bits, se debe instalar el ODBC Drivers de 16 bits y sqlnet de 16 bits.
- 3. Si la plataforma es Windows 95 y la herramienta que se quiere comunicar a la base de datos Oracle es a 32 bits, se debe instalar el ODBC Drivers de 32 bits y sqlnet de 32 bits y sqlnet de 32 bits.

¿COMO INSTALAR ?

Teniendo en cuenta que se tiene el CDROM de Oracle Client Software, se tienen tres opciones de donde ejecutar el archivo SETUP.EXE



1. Si se necesita instalar el ODBC Drivers de 16 bits se debe entrar al directorio WINDOWS.
2. Si se necesita instalar el ODBC Drivers de 32 bits se debe entrar al directorio WIN95.

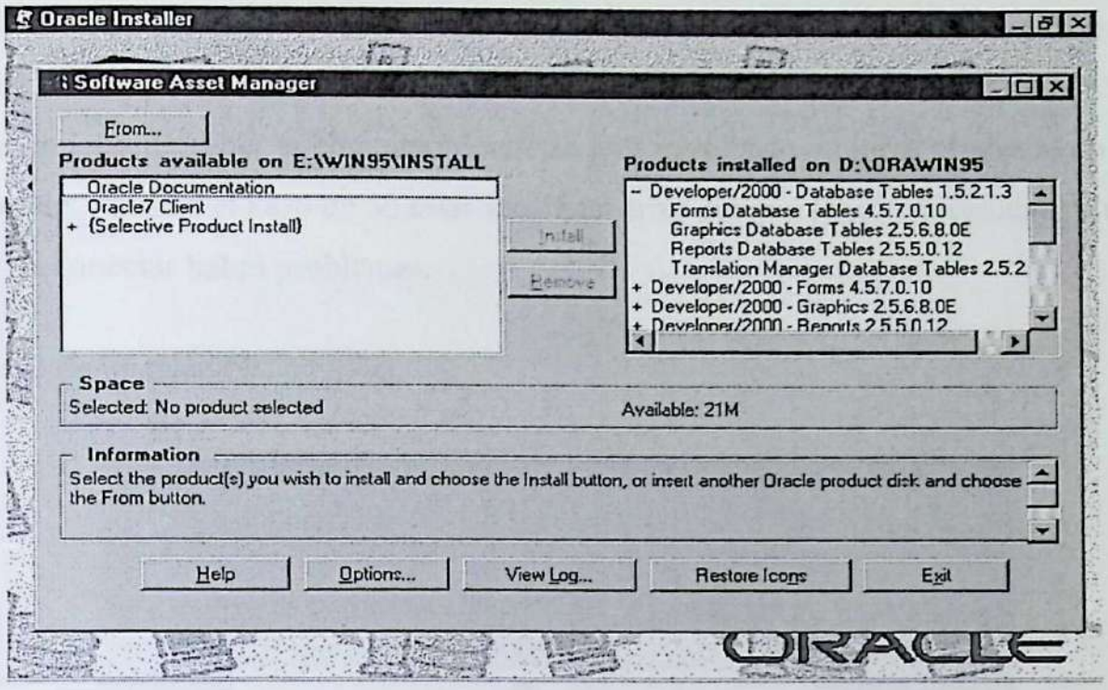
3. Si se necesita instalar el ODBC Drivers sobre la plataforma Windows NT, se debe entrar al directorio NT_X86.

INSTALACION PRACTICA.

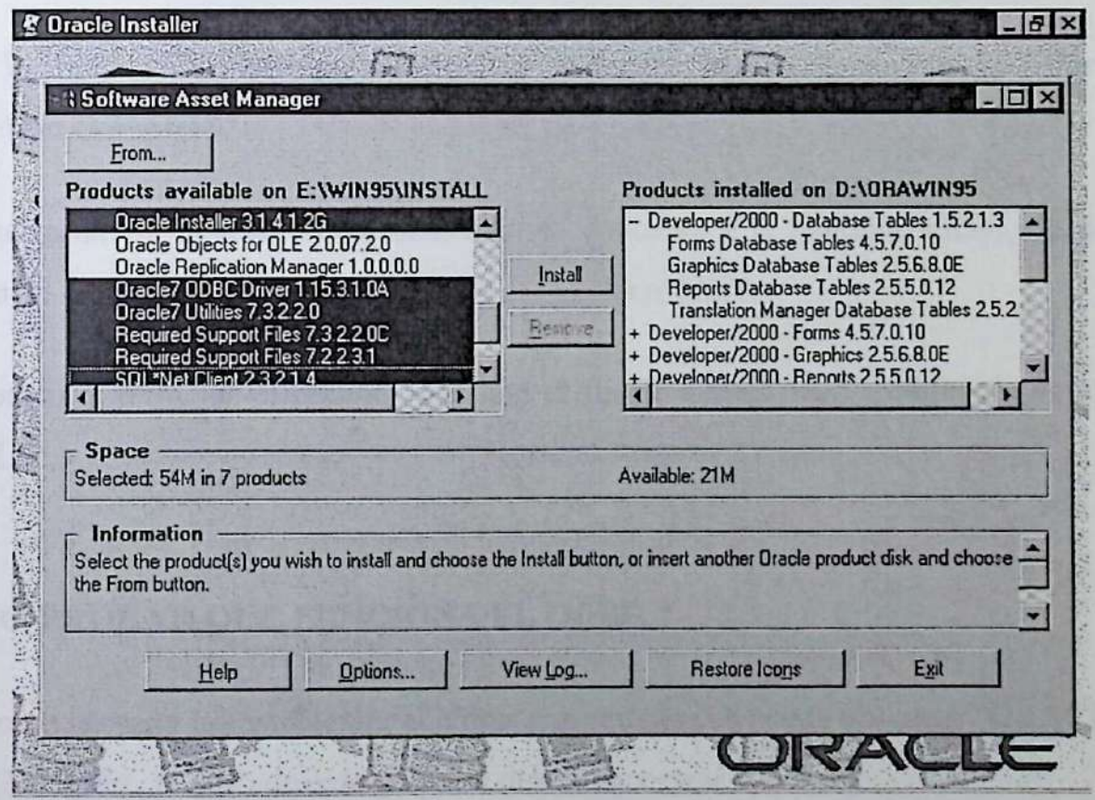
Se va a instalar el ODBC Drivers para Windows 95, por lo tanto escogemos la opción 2.

1. Ejecutar del archivo SETUP.EXE
2. Escoger el lenguaje en el cual se van a instalar los productos.
3. Digitar el nombre de la empresa y el directorio donde se va a instalar.

4. Se deben escoger los productos a instalar. Como solo se desea el ODBC Drivers se debe escoger "Selective Products Install"



5. Escoger los productos a instalar.

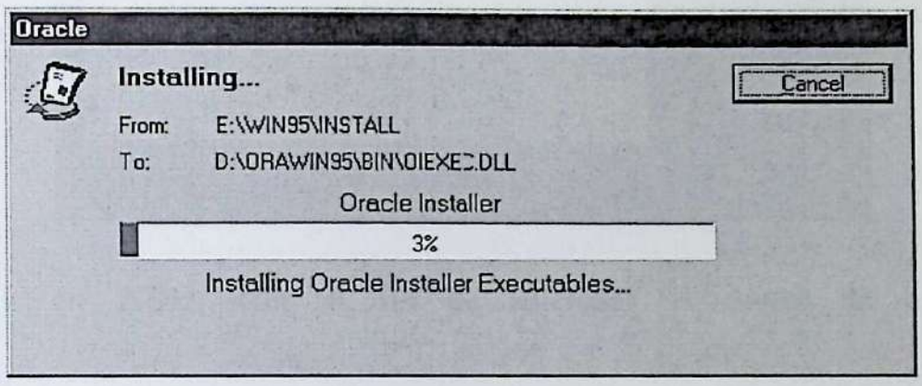


Cuando ya se encuentre todos los productos anteriores resaltados se debe de dar click en INSTALL.

NOTA : Es importante verificar el espacio disponible.

6. Se debe escoger el protocolo de red que se va a utilizar para la comunicación con el servidor en donde se encuentra la base de datos.(TCP/IP, SPX, Names Pipes). Es muy importante saber si el equipo donde se esta instalando ya tiene el protocolo de red configurado, pues en el caso de no estar sacará un error aunque seguirá instalando y cuando se trate de conectar habrá problemas.

7. Proceso de instalación.



8. Si la Instalación termina exitosa, le aparecerá una ventana que le informará : 'Installation Complete'.

9. Al terminar la instalación el instalador lo devuelve a la pantalla inicial, de donde se escogieron los productos. Para salir debe dar click en EXIT.

10. Apagar y reiniciar el equipo, esto con el fin de activar las variables de ambiente de ORACLE.

¿COMO PROBAR QUE FUNCIONA EL ODBC ?

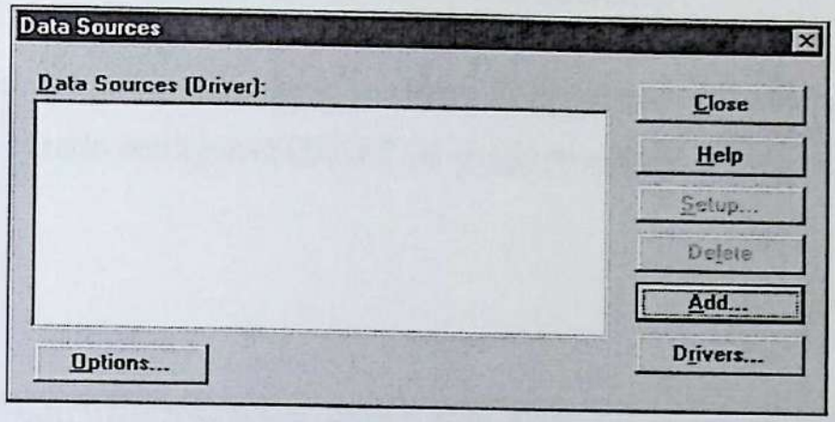
Cuando se termina la instalación, al entrar a programas se puede observar "ORACLE FOR WINDOWS 95". Dentro de este grupo se deben encontrar :

32-bit Administrator
32-bit ODBC Help

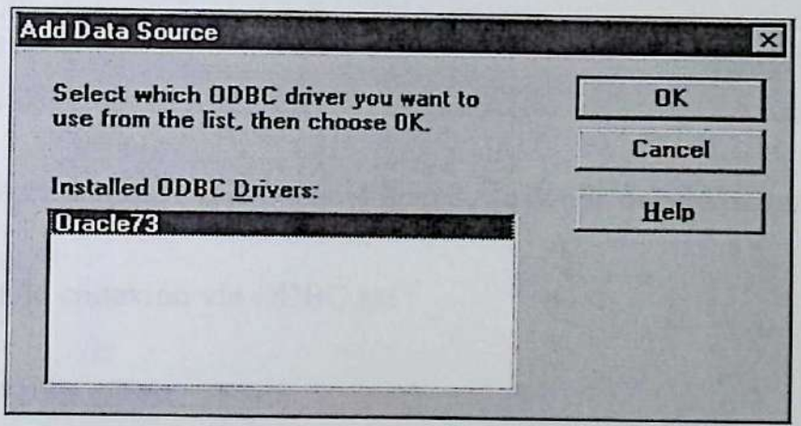
32-bit ODBC Release Notes

32-bit ODBC TEST

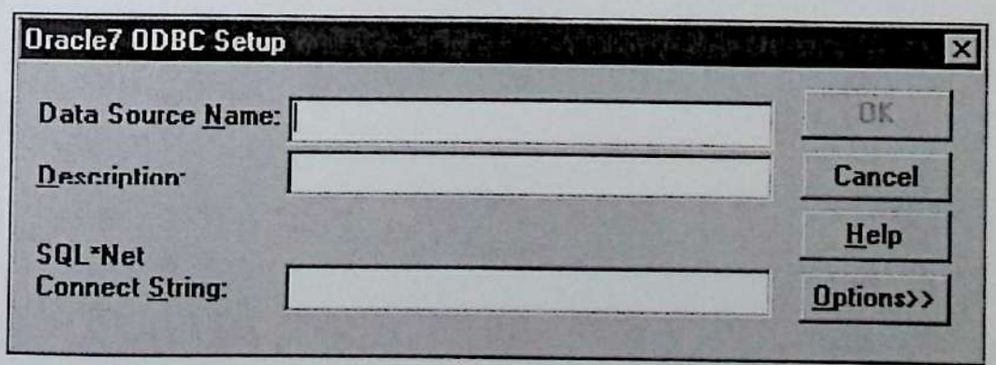
Entrar por 32-bit Administrador así :



Dar click en ADD, con el fin de adicionar el driver de oracle así :



Escoger el driver de ORACLE73.

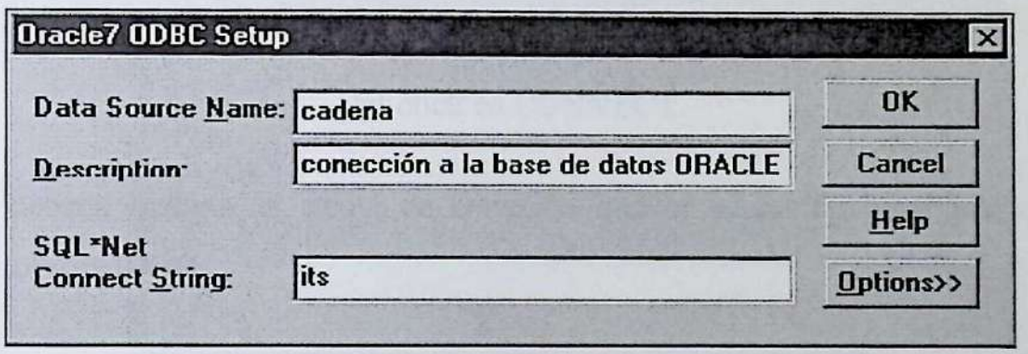


En Data Source Name debe colocar una cadena de caracteres, la cual va a utilizar desde la aplicación para conectarse a la base de datos Oracle.

En Description, se deberá colocar una breve descripción de para que se utilizara ese data source name dado anteriormente, esta información es adicional.

En SQL*Net connect string debe darse la cadena de caracteres utilizada para la conexión a la base de datos cuando configuro SQLNET en el equipo cliente.

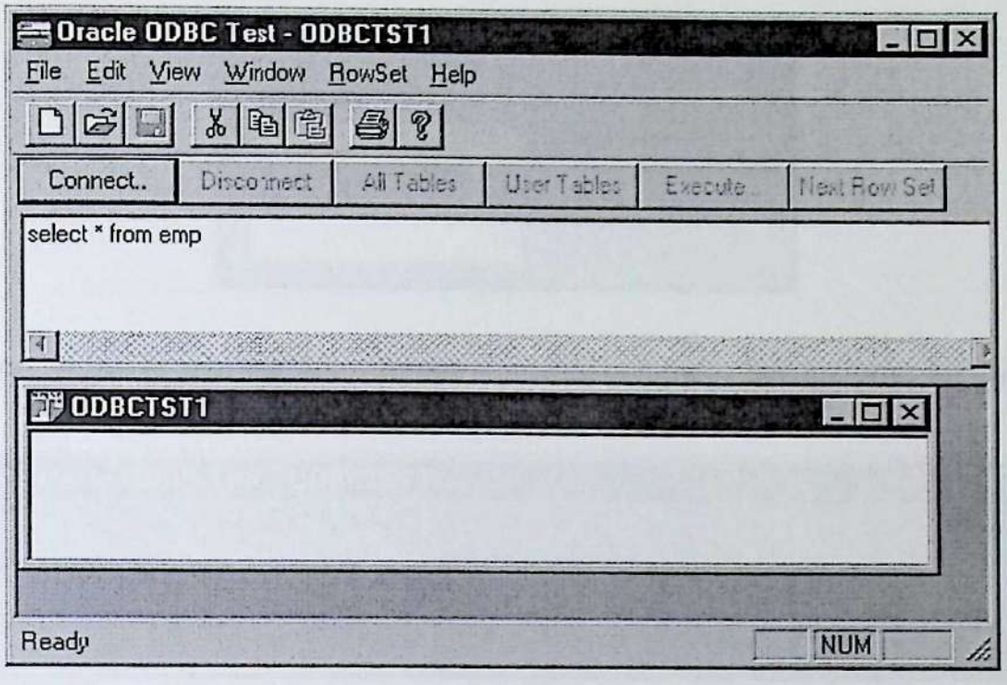
Por ejemplo :



Al dar OK, volverá de nuevo a la pantalla inicial, de donde deberá cerrar la ventana.

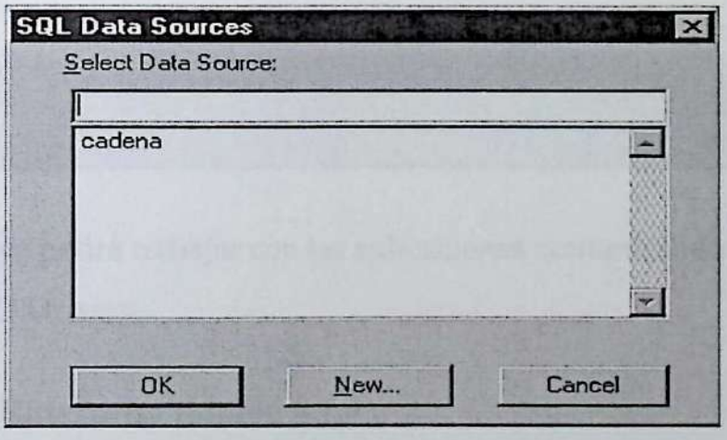
Ahora si se prueba la conexión vía ODBC así :

Deberá entrar a 32-bit ODBC TEST.

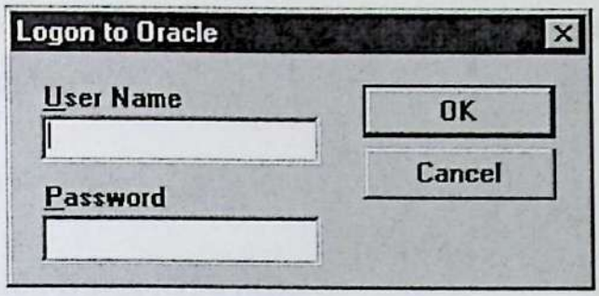


Dar click en CONNECT.

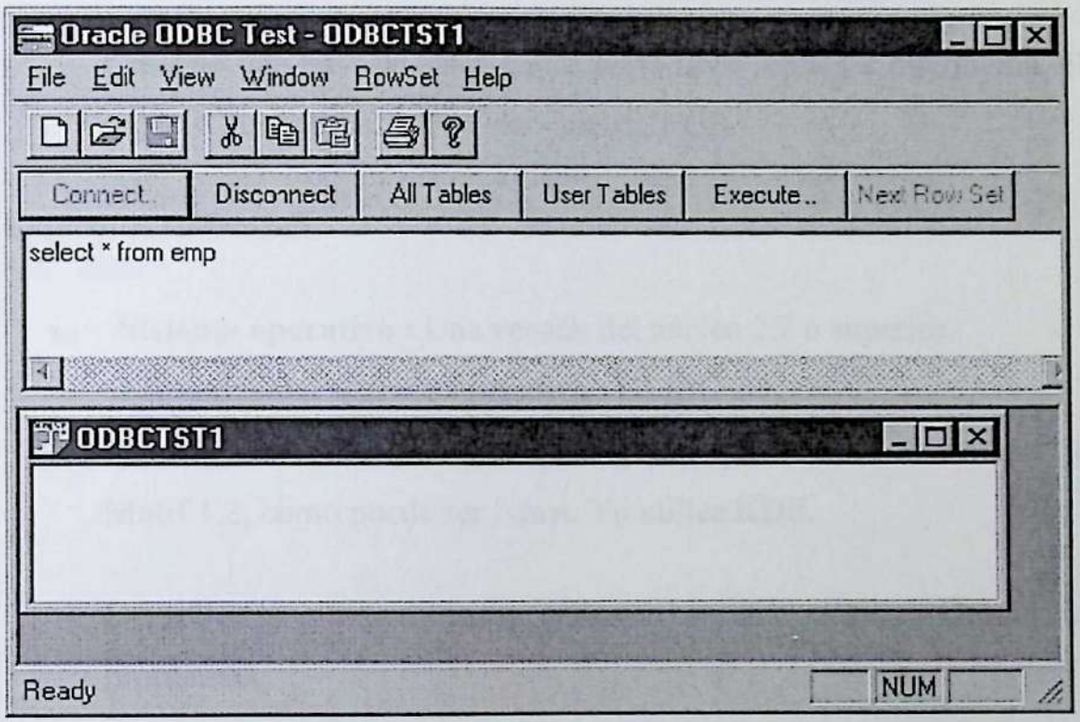
Ahora se deberá escoger el string de conexión que se acaba de crear por el 32-bits administrator de ODBC.



Escribir el usuario y password correspondiente de la persona que quiere conectarse a la base de datos.



Si logra conexión le aparecerá la ventana siguiente con todas las opciones habilitadas.



En este momento ya podrá trabajar con las aplicaciones comunicándose a la base de datos por medio del ODBC Drivers.

Instalación Oracle Enterprise Edition 8.1.6

1. Requisitos para la instalación

Según la documentación de Oracle, los requisitos mínimos para la instalación de la versión 8.1.6 son los siguientes:

Requisitos Hardware :

- **Memoria :** 128 MB mínimo. Yo lo instalé con 128 MB.

- **Espacio de paginación :** Al menos el doble de la RAM. Yo justo tenía el doble.

Espacio en disco :

- **Instalación típica :** 750 MB.
- **Instalación mínima :** 675 MB.
- **Instalación personalizada :** Hasta 1 GB.

Creando una base de datos con soporte Java, réplica e Intermedia, el espacio que utilicé, fué de algo más de 1 GB.

Requisitos software :

- **Sistema operativo :** Una versión del núcleo 2.2 o superior.
- **Librerías del sistema operativo :** GLIBC 2.1.
- **Gestor de ventanas :** Cualquier gestor que soporte la versión de Motif 1.2, como puede ser fvmw. Yo utilice KDE.

Cumpliendo estos requisitos, podremos instalar el motor Oracle sin problemas.

2. Pasos previos a la instalación.

Antes de comenzar con la instalación, debemos comprobar una serie de parámetros de nuestro sistema operativo, y realizar algunas tareas tanto con usuario **root** como con el usuario que instalará el motor, en nuestro caso **oracle.**

Tareas como root

Configuración del Sistema :

- **Memoria compartida :** SHMMAX : Al menos 0.5 * *Memoria física de nuestra máquina.*
  - SHMMIN : 1.
  - SHMMNI : 100.
  - SHMSEG 10.
- **Semáforos :**
  - SEMMNI : 100.
  - SEMMLS : 10 + *el mayor valor del parámetro PROCESSES de todas las bases de datos que se instalarán en el sistema operativo.*
  - SEMMNS : El sumatorío de los parámetros PROCESSES de todas las bases de datos(menos el valor mayor) + dos veces el mayor valor de PROCESSES, y al resultado le añadimos 10 * el número de bases de datos instaladas: por ejemplo, si tuviéramos dos bases de datos :

    BD1 : Parámtero PROCESSES = 50.

    BD2 : Parámtero PROCESSES = 80.

    El resultado sería : (50 + (2 * 80)) + (10 * 2) = 230.

  - SEMOPM : 100.
  - SEMVMX : 32767.

Para comprobar los valores actuales de nuestro núcleo, podemos ejecutar el siguiente comando :

ipcs -m (Nos mostrará los valores de los parámetros de la memoria compartida).

ipcs -s (Nos mostrará los valores de los parámteros para los semáforos).

ipcs -l (Nos mostrará los límites).

Si tenemos que cambiar algún parámetro, debemos editar los ficheros :

/usr/src/linux/include/asm/shmparam.h *Para la memoria compartida.*

/usr/src/linux/include/linux/sem.h *Para los semáforos.*

Hay que tener en cuenta que los valores descritos son para Oracle, es decir, si tenemos otros programas que utilizan memoria compartida y semáforos, debemos añadir a los parámetros calculados, la cantidad que necesiten.

Creación de los puntos de montaje :

En este punto, Oracle recomienda al menos dos puntos de montaje, uno para la instalación del motor y otro para los ficheros de la base de datos, aunque si seguimos la OFA

(*Optimal Flexible Architecture)* debemos tener cuatro puntos de montaje, uno para el software y otros tres para los ficheros de la base de datos (sistema, indices, datos). Aunque se puede hacer la instalación en un único punto de montaje, sobre todo , si es un PC doméstico en el que sólo tenemos un disco.

Yo suelo utilizar la siguiente nomenclatura de directorios:

/oracle/app/oracle/product/X.X.X

Para la instalación del software, donde X.X.X es la versión del motor, 8.1.6.

/oracle/database

Para almacenar los ficheros de la base de datos.

Creación de los grupos linux para los administradores de la base de datos :

En este punto, suelo crear un único grupo linux, al que llamo **dba**, aunque se pueden crear dos, uno para asociar con el SYSDBA y otro para asociar con SYSOPER. Si sólo creas uno, se asociará dicho grupo con los dos roles Oracle. También se propone la creación de un rol para el Instalador de Oracle, yo utilizo el mismo, **dba.**

Creación del usuario Oracle :

Con este usuario instalaremos el motor de la base de datos. Yo suelo crear el típico usuario **oracle**, al que asigno el grupo **dba** y cuyo directorio de inicio será el **ORACLE_HOME**, en nuestro caso ***/oracle/app/oracle/product/8.1.6***.

Una vez creado el usuario, es muy importante asegurarse de que tenga permisos sobre los directorios en los que instalaremos el software y crearemos la base de datos, ya que en caso contrario nos fallará. Yo suelo hacer que oracle sea el propietario de todos los directorios (***chown -R oracle:dba /oracle***).

Tareas como oracle

Permisos para la creación de los ficheros

Hay que asegurarse que el usuario oracle tenga la máscara de creación de ficheros a 022, es decir, permisos de lectura y ejecución para los todos los usuarios y los de su grupo, pero sin permiso de escritura, para hacer esto debemos ejecutar el siguiente comando desde el fichero de parámetros .profile:

umask 022

Definir las variables de entorno

Ahora debemos definir las variables de entorno para el usuario oracle, en la documentación vienen bien explicadas, os pongo las que yo utilizo:

setenvORACLE_HOME "/oracle/app/oracle/product/8.1.6"

Es el directorio en el que está instalado el motor de la base de datos

setenv ORACLE_BASE "/oracle/app/oracle" Se utiliza por la OFA, a partir de aquí se instalan los distintos productos oracle

setenv ORACLE_SID "orcl" El nombre de la instancia de base de datos

setenv NLS_LANG "spanish_spain.we8dec" El juego de caracteres para la instalación

setenv ORA_NLS33 "${ORACLE_HOME}/ocommon/nls/admin/data"

Directorio en el que se encuentra los juegos de caracteres

setenv PATH "${PATH}:${ORACLE_HOME}/bin:/bin:/usr/bin"

Añadimos al PATH, el directorio con los ejecutables de oracle

setenv CLASSPATH "/usr/local/jre:${ORACLE_HOME}/jlib:${ORACLE_HOME}/product/jlib" Variable en la que indicamos el directorio que contiene las clases para el jre (java runtime enviroment), necesario para la ejecución del instaler y los distintos asistentes (net y creación de base de datos)

Instalación

Montar el CD

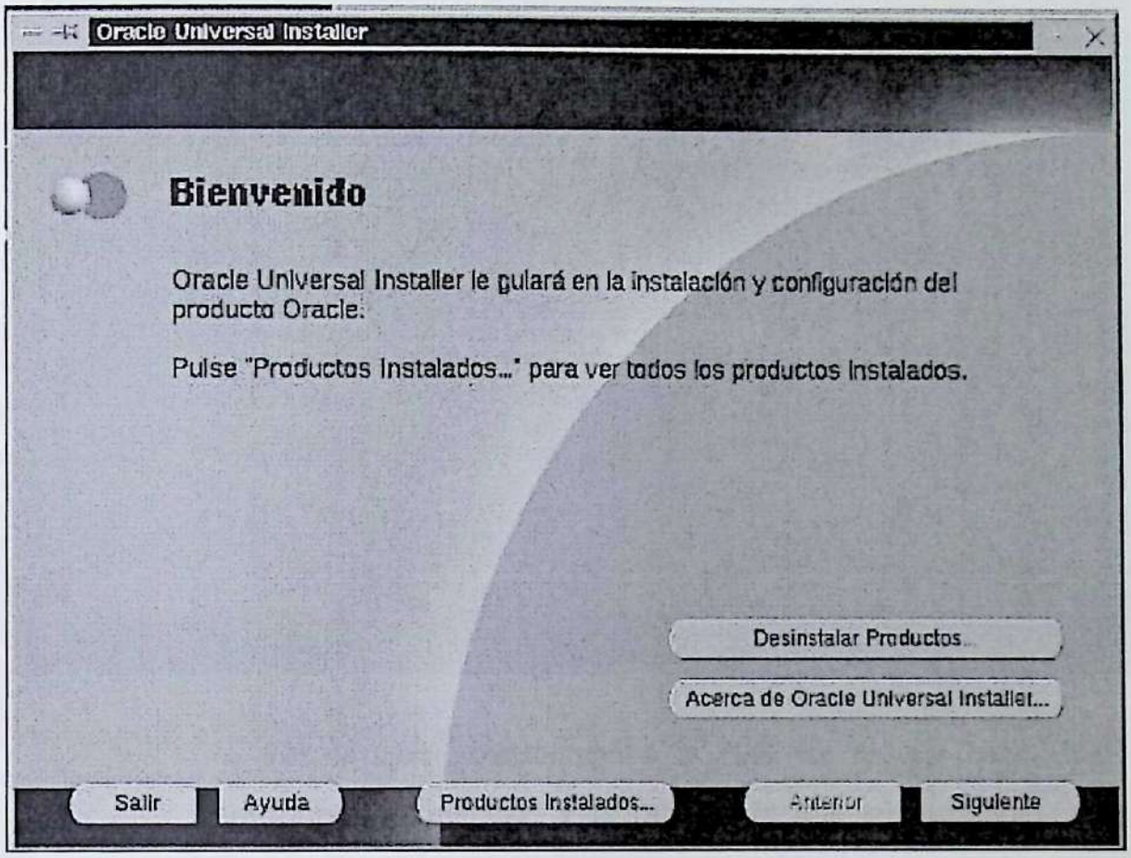
Antes de comenzar con la instalación debemos montar el CD en el que tenemos el motor, en el caso de tenerlo en CD.

Ejecutar el instaler

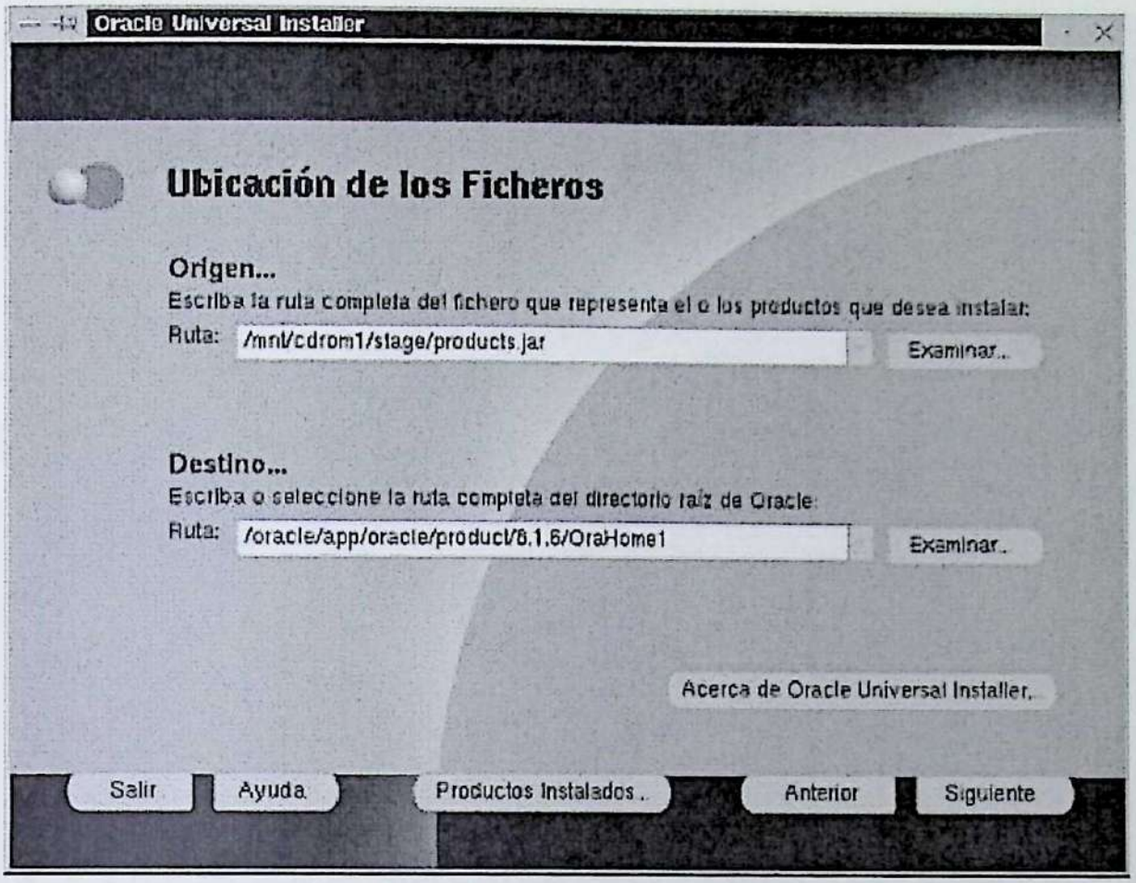
Para ejecutar el instaler, debemos entrar en el entorno de ventanas con el usuario oracle, y una vez dentro ir al directorio en el que tenemos montado el CD, y ejecutar:

./runInstaller

Aparecerá la siguiente pantalla:

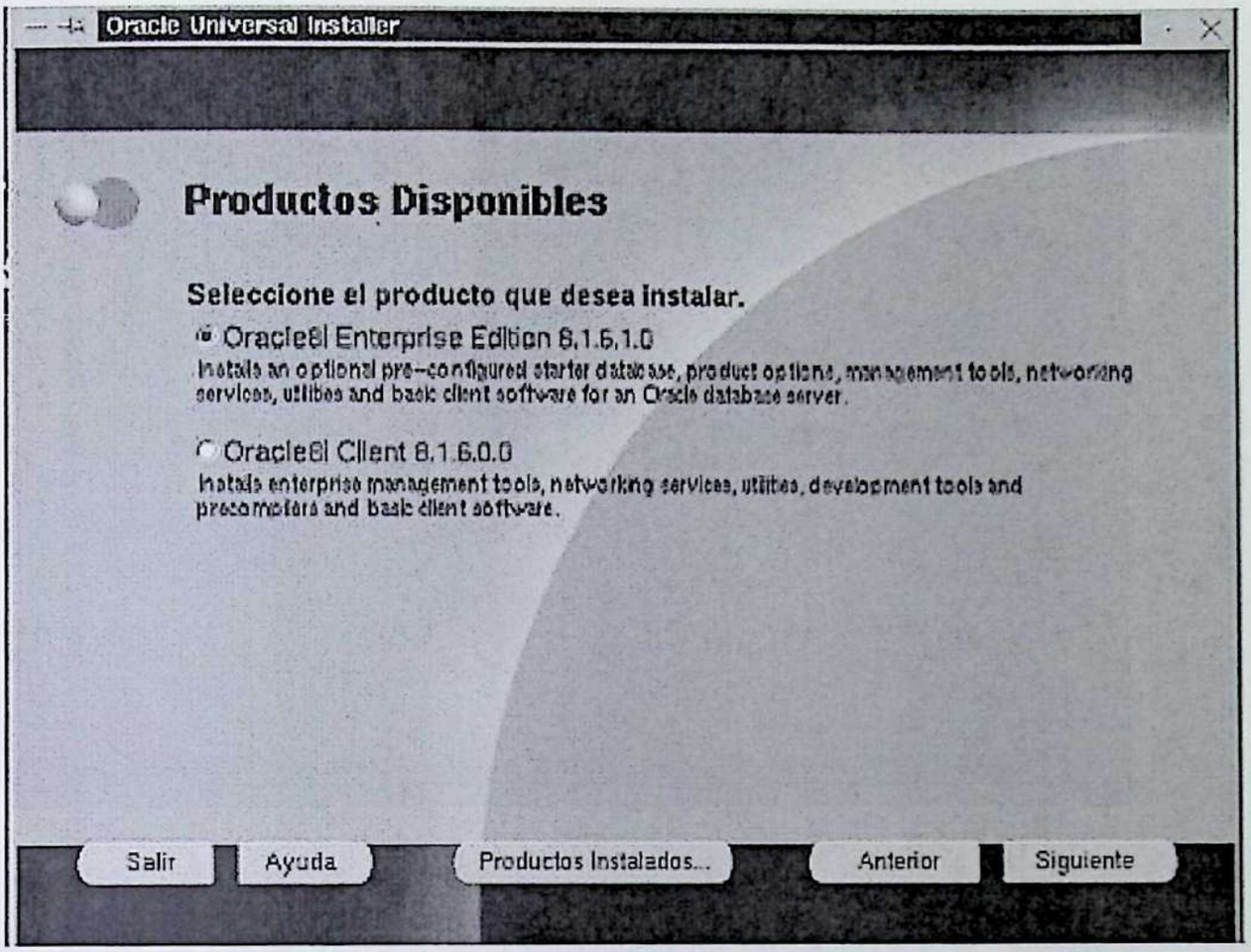


Es la pantalla de bienvenida, pulsamos siguiente y vamos a la siguiente :

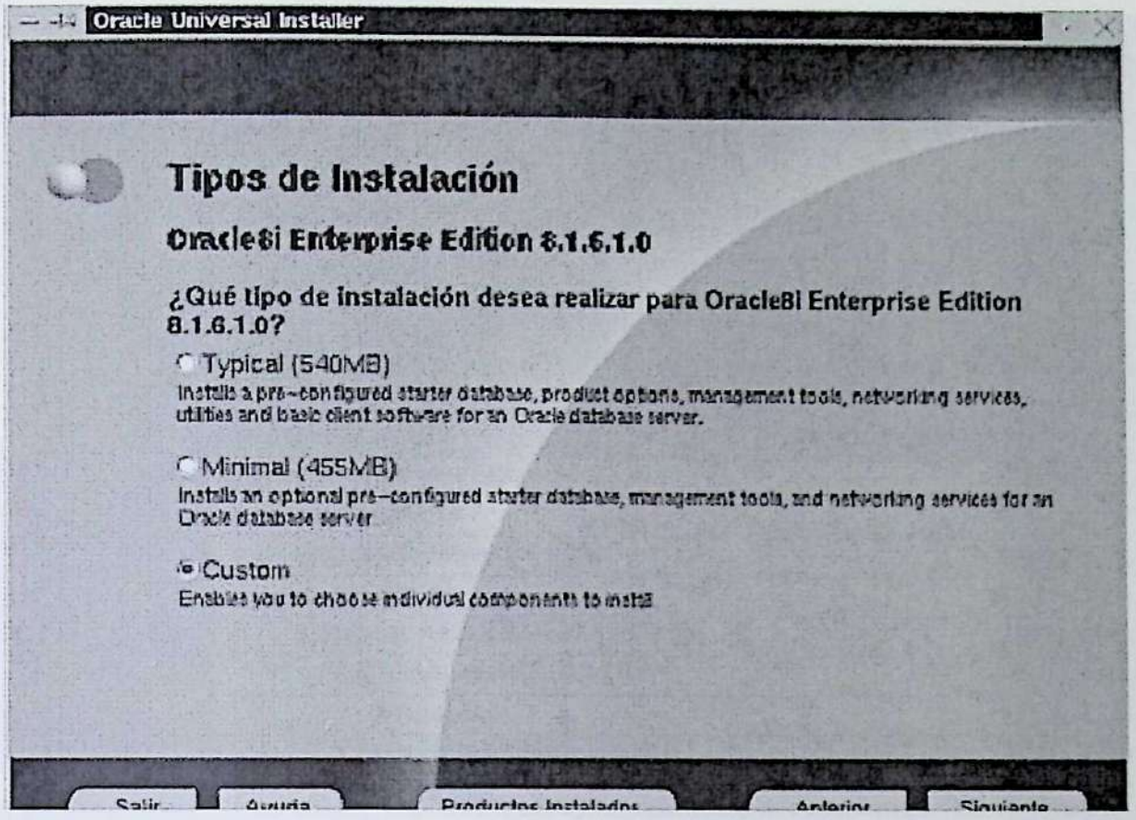


En la que seleccionamos la ruta en la que están los productos a instalar, y la ruta en la que se instalarán, en nuestro caso esta última es : **/oracle/app/oracle/product/8.1.6,** una vez seleccionados los valores correctos pasamos a la siguiente pantalla:

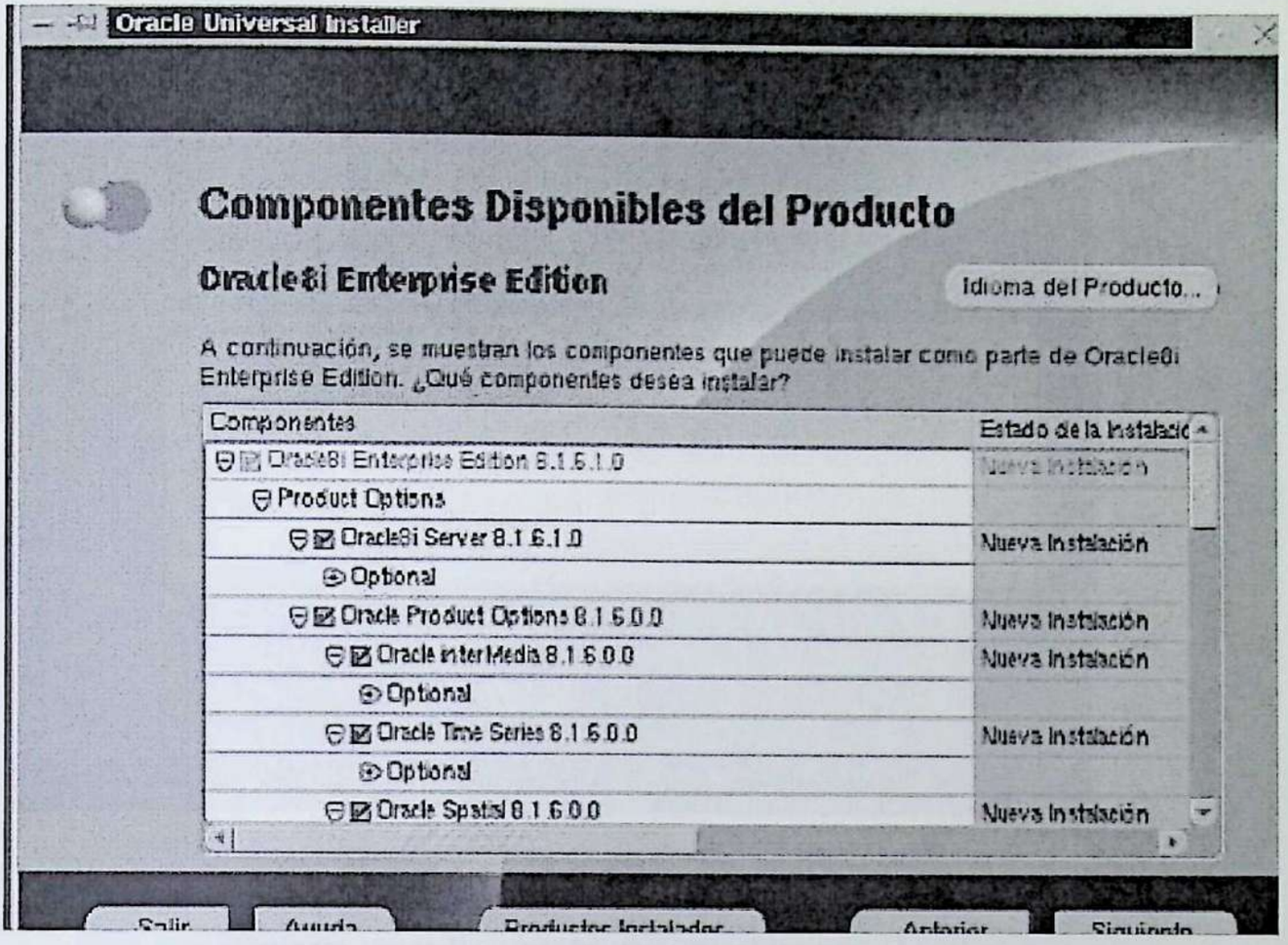
BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA



Aquí seleccionamos la primera opción, que es instalar el motor de la base de datos, continuamos con...



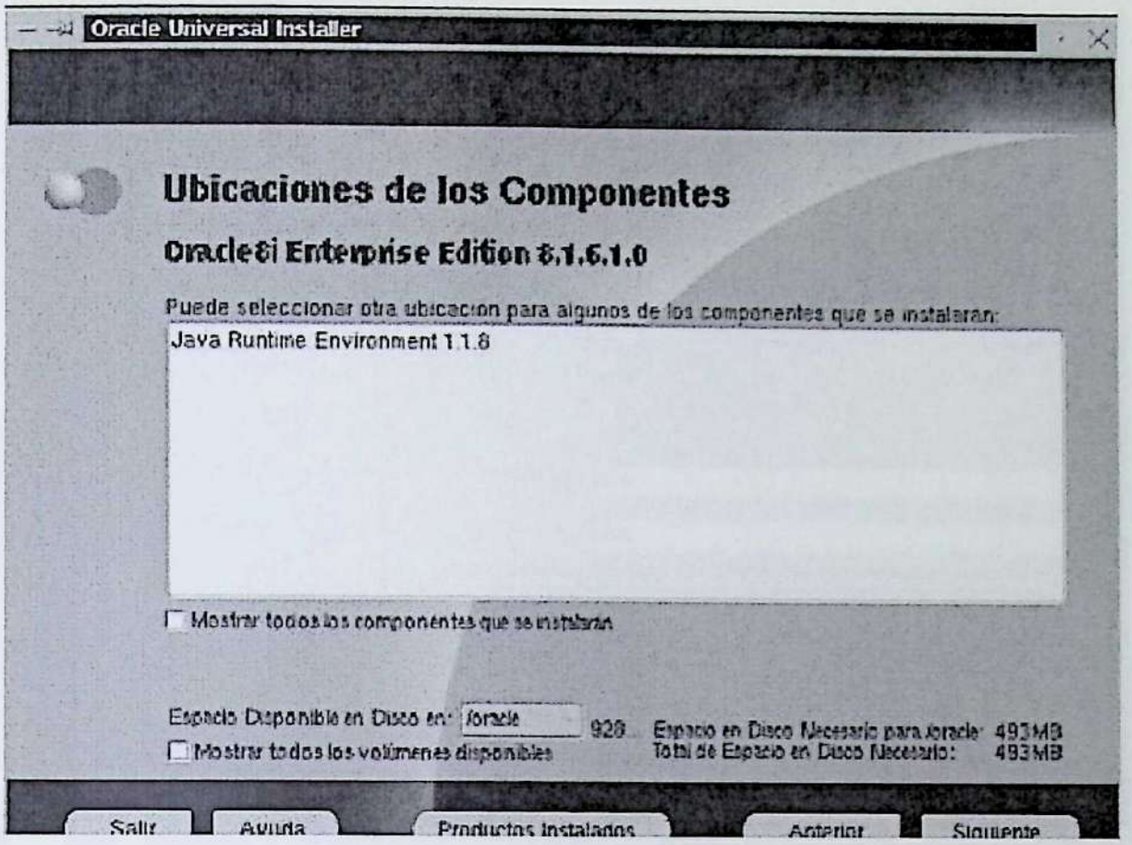
... el tipo de instalación, yo suelo elegir la opción personalizada, para seleccionar los productos que se quieren instalar...



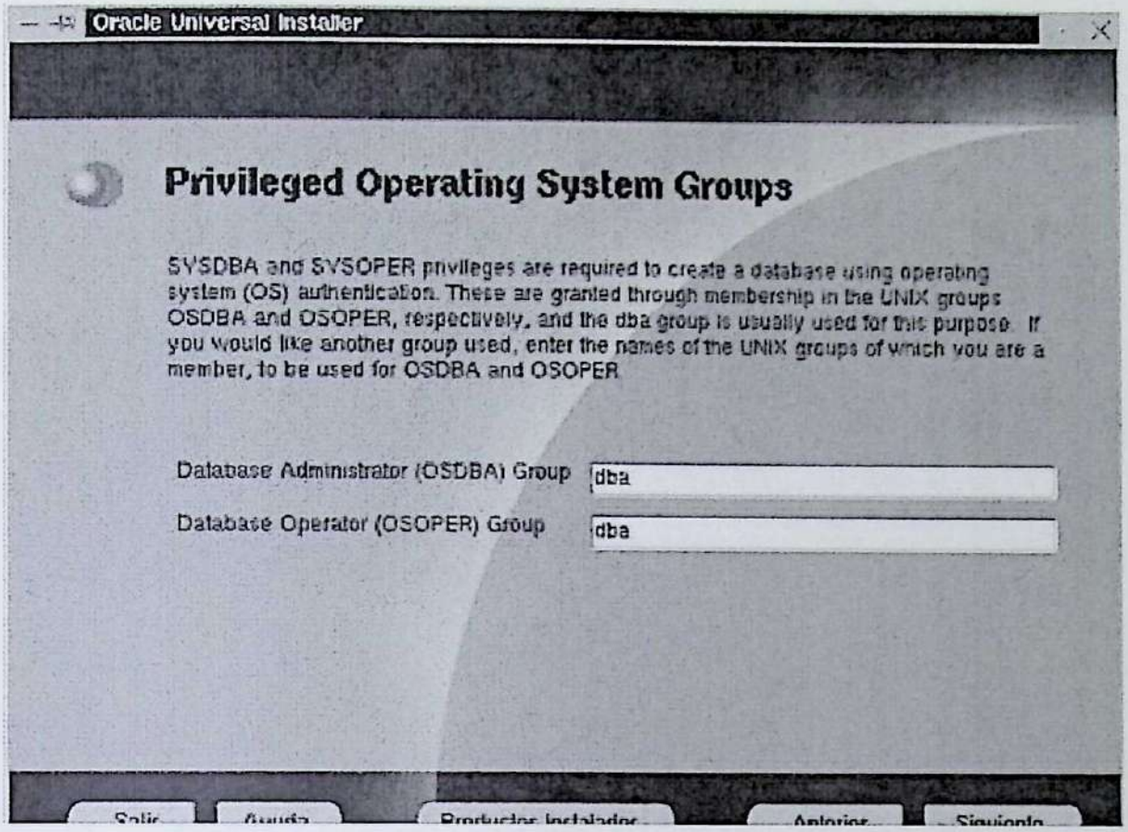
...en la siguiente pantalla seleccionamos los productos a instalar, el que debe estar marcado es Oracle8i Server 8.1.6.1.0, que es el motor de la base de datos, también debemos seleccionar los productos de red, Net8 Client, Net8 Server, para poder conectarnos a la base de datos desde la máquina en la que instalamos y por último Oracle Database Utilities + SQL*Plus.

Si queremos desarrollar en java, deberemos instalar los Oracle Java Products, JDBC Drivers, Oracle SQLJ, Oracle Java Tools.

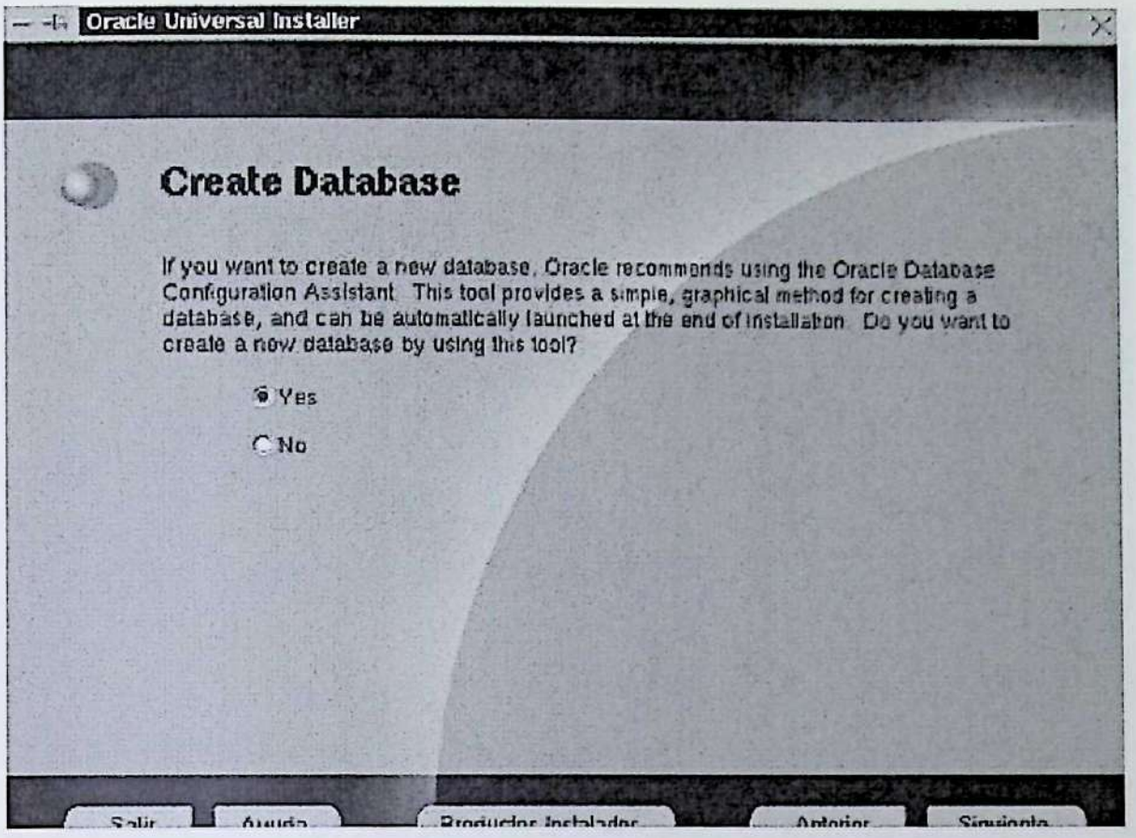
También podemos instalar los distintos cartuchos que nos ofrece Oracle, Oracle Intermedia, Oracle Times Series, Oracle Spatial.



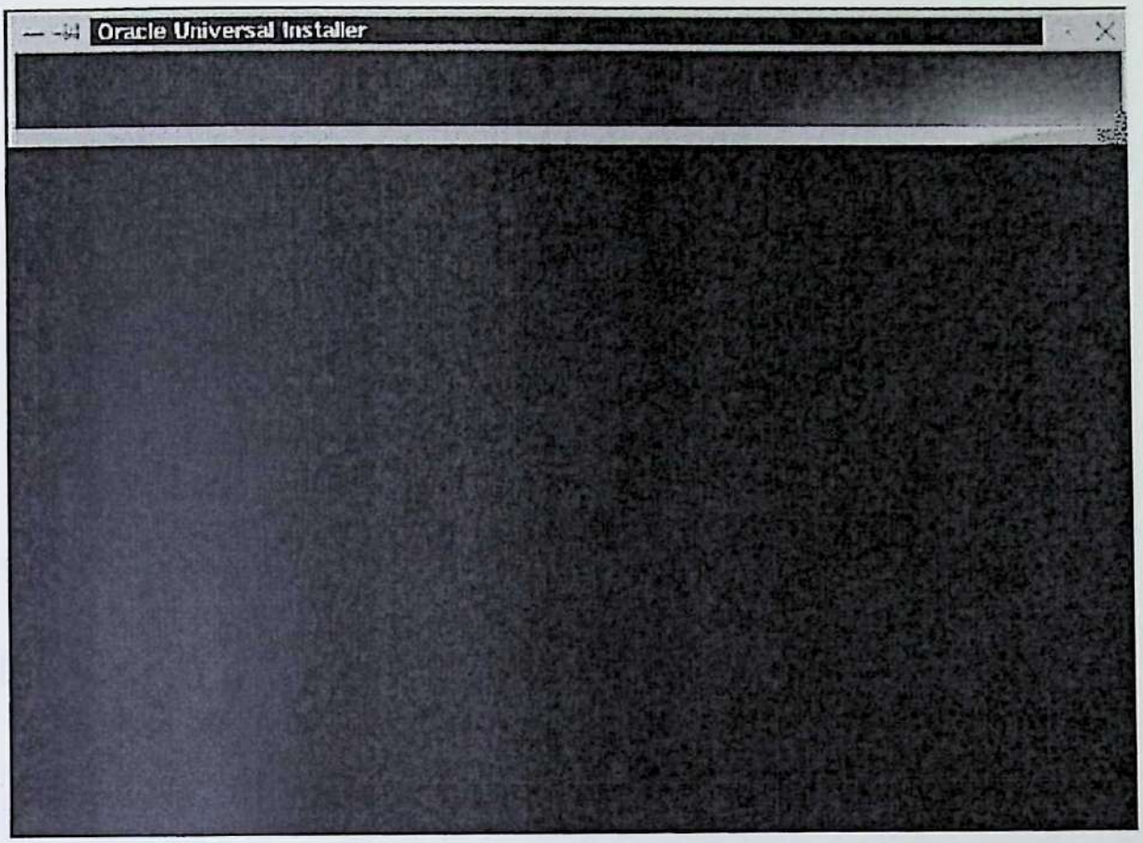
Una vez terminada una revisión del instaler llegamos a la siguiente pantalla, en la que aceptaremos el valor por defecto...



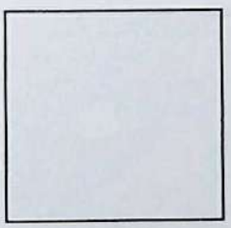
...ahora se nos piden los grupos del sistema que harán de DBA y de operador, en nuestro caso como únicamente creamos uno ,dba , lo marcamos en las dos casillas...



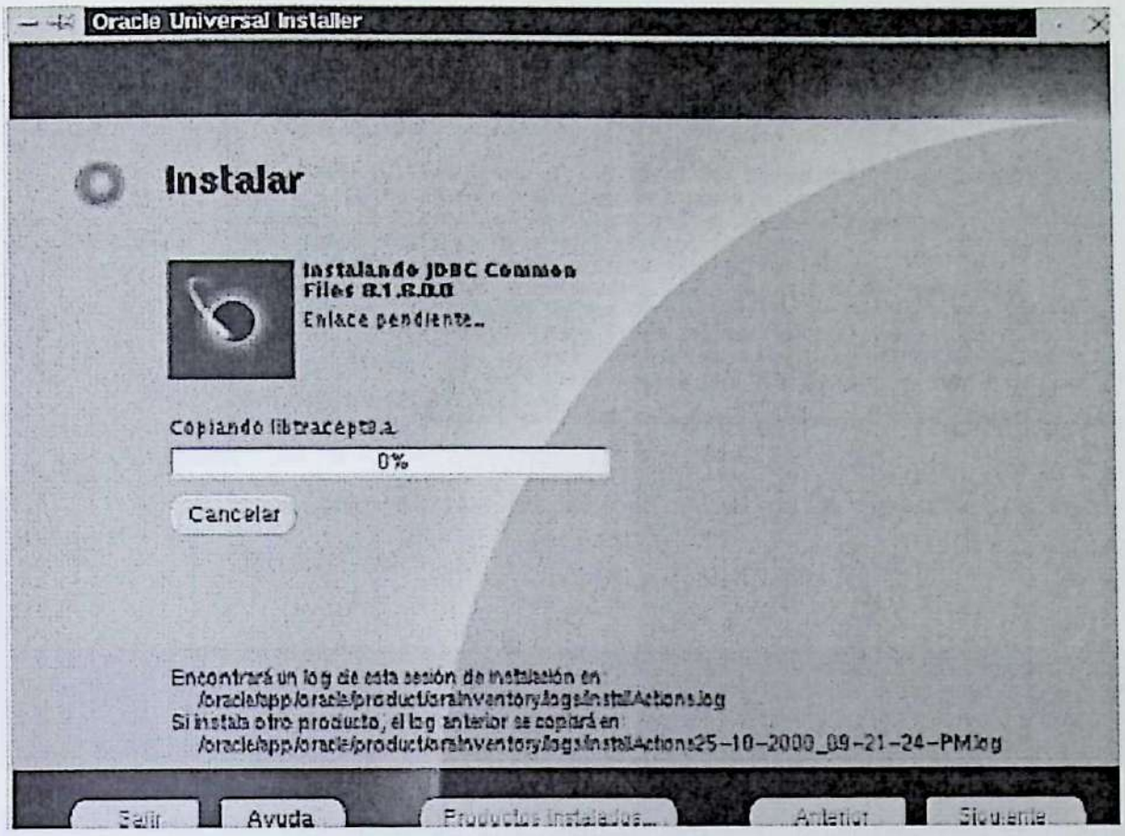
...en esta pantalla seleccionamos que si queremos ejecutar el asistente de Oracle para la creación de una base de datos, ya que nos creará unos scripts con los que crearemos una BD de una manera sencilla...



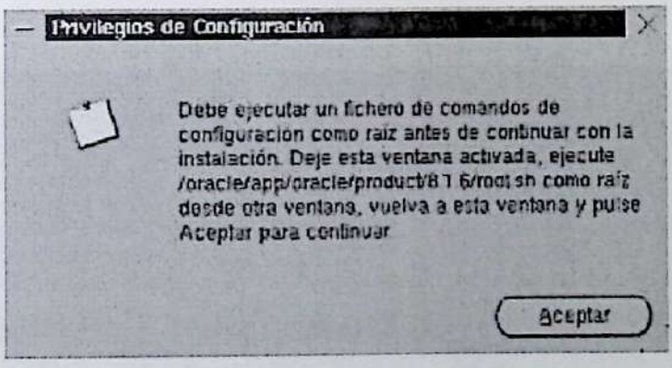
...aquí debemos elegir el nombre que tendrá nuestra base de datos, uno global, que debe ser único en un entorno distribuido, p.e un entorno de réplica, y el SID...



...seleccionamos el directorio en el que se crearan los ficheros de la base de datos...



..., después de ver en una pantalla los productos que se instalarán, pasamos a la pantalla de progreso de la instalación...

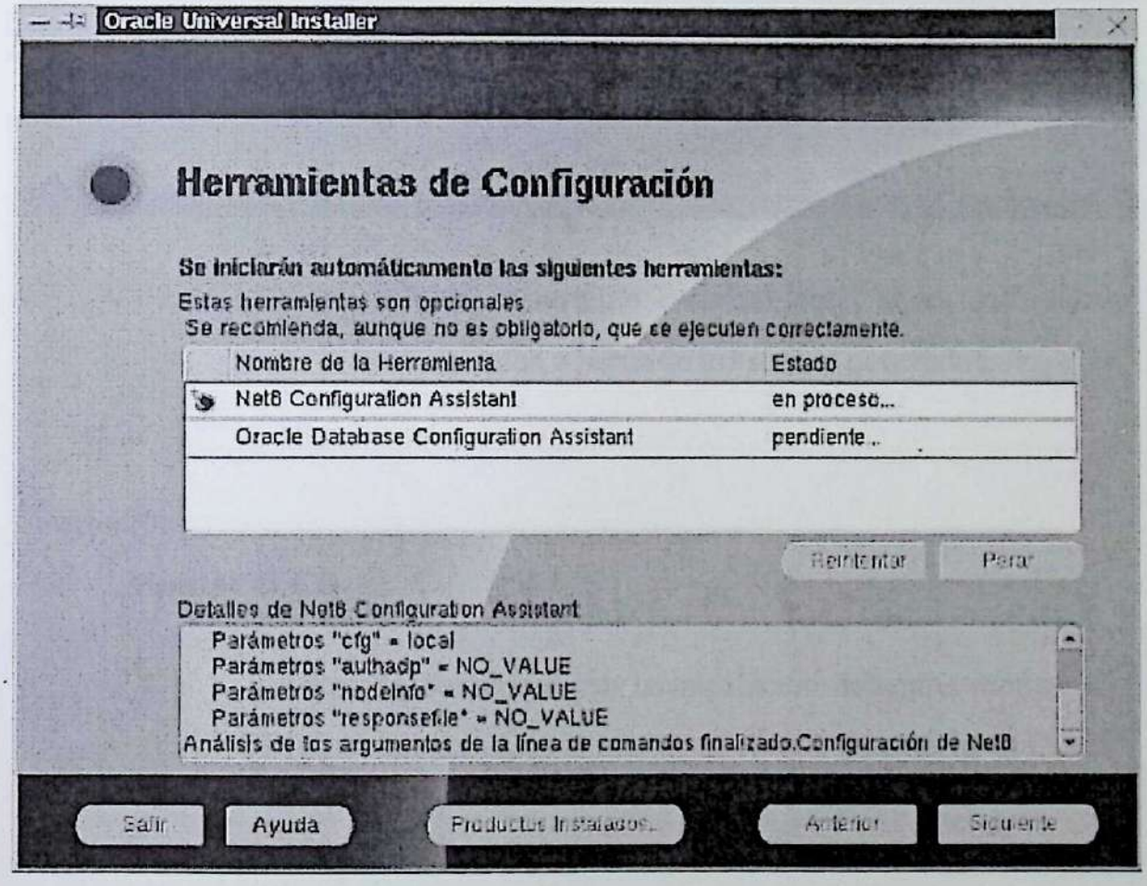


..., una vez terminada la instalación, desde una ventana de comandos, como usuario root, debemos ejecutar el fichero root.sh, que se encuentra en el ORACLE_HOME, para poder seguir con la instalación...

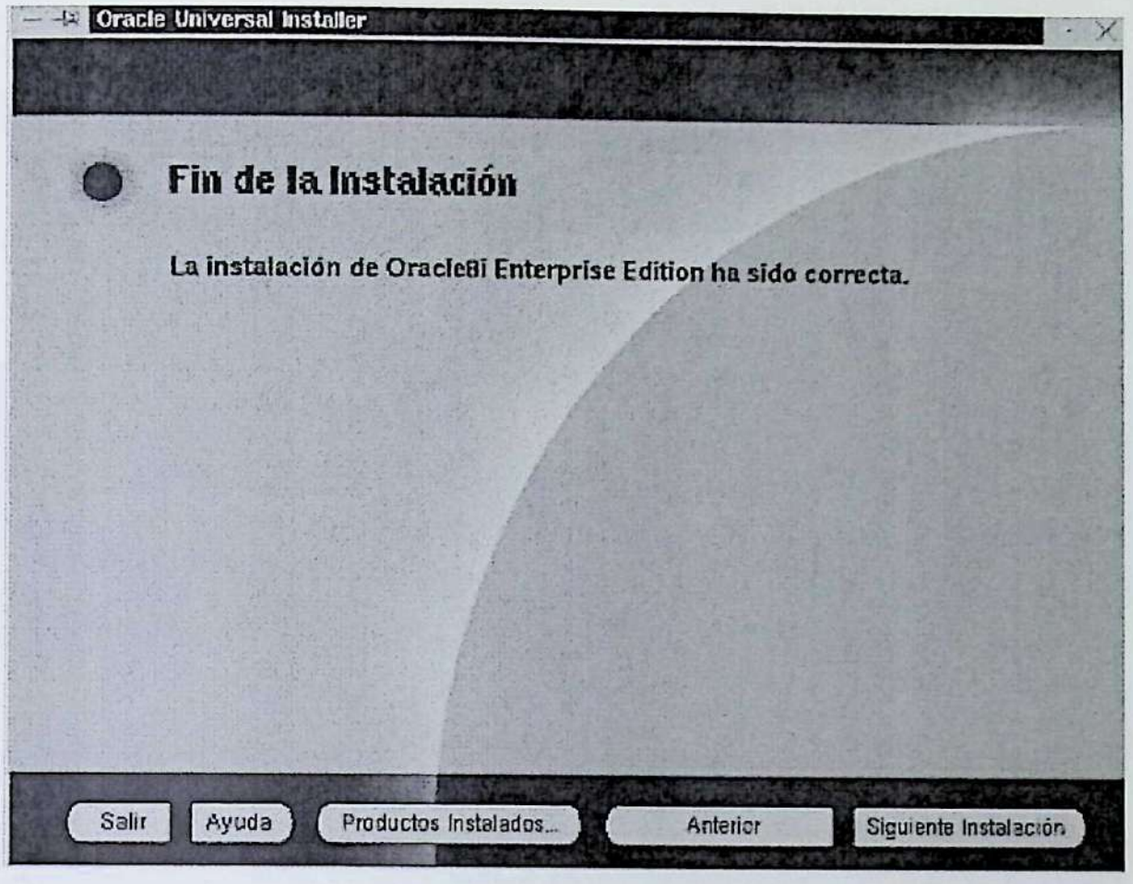
```
[root@lazarillo 8.1.6]# ./root.sh
Running Oracle8 root.sh script...
\nThe following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/app/oracle/product/8.1.6
    ORACLE_SID=   orcl

Enter the full pathname of the local bin directory: [/usr/local/bin]:
Entry will be added to the /etc/oratab file by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
IMPORTANT NOTE: Please delete any log and trace files previously
                created by the Oracle Enterprise Manager Intelligent
                Agent. These files may be found in the directories
                you use for storing other Net8 log and trace files.
                If such files exist, the OEM IA may not restart.
[root@lazarillo 8.1.6]#
```

..., aquí se nos muestra la salida de la ejecución de ese fichero...



... por último pasamos a ejecutar los distintos asistentes, el asistente para configurar Net8 y el asistente para crear una base de datos.



Y la instalación terminó correctamente, ahora sólo falta crear la base de datos a partir de los scripts generados.

Instalación

Montar el CD

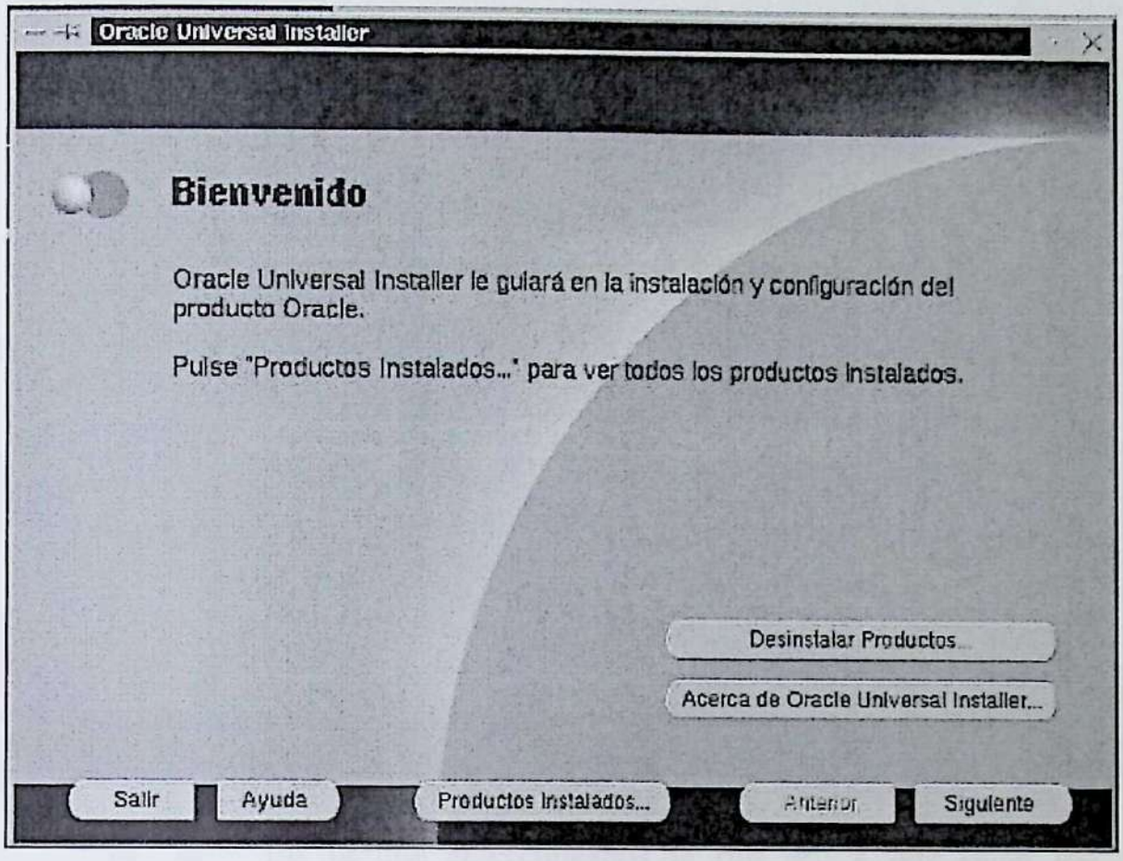
Antes de comenzar con la instalación debemos montar el CD en el que tenemos el motor, en el caso de tenerlo en CD.

Ejecutar el instaler

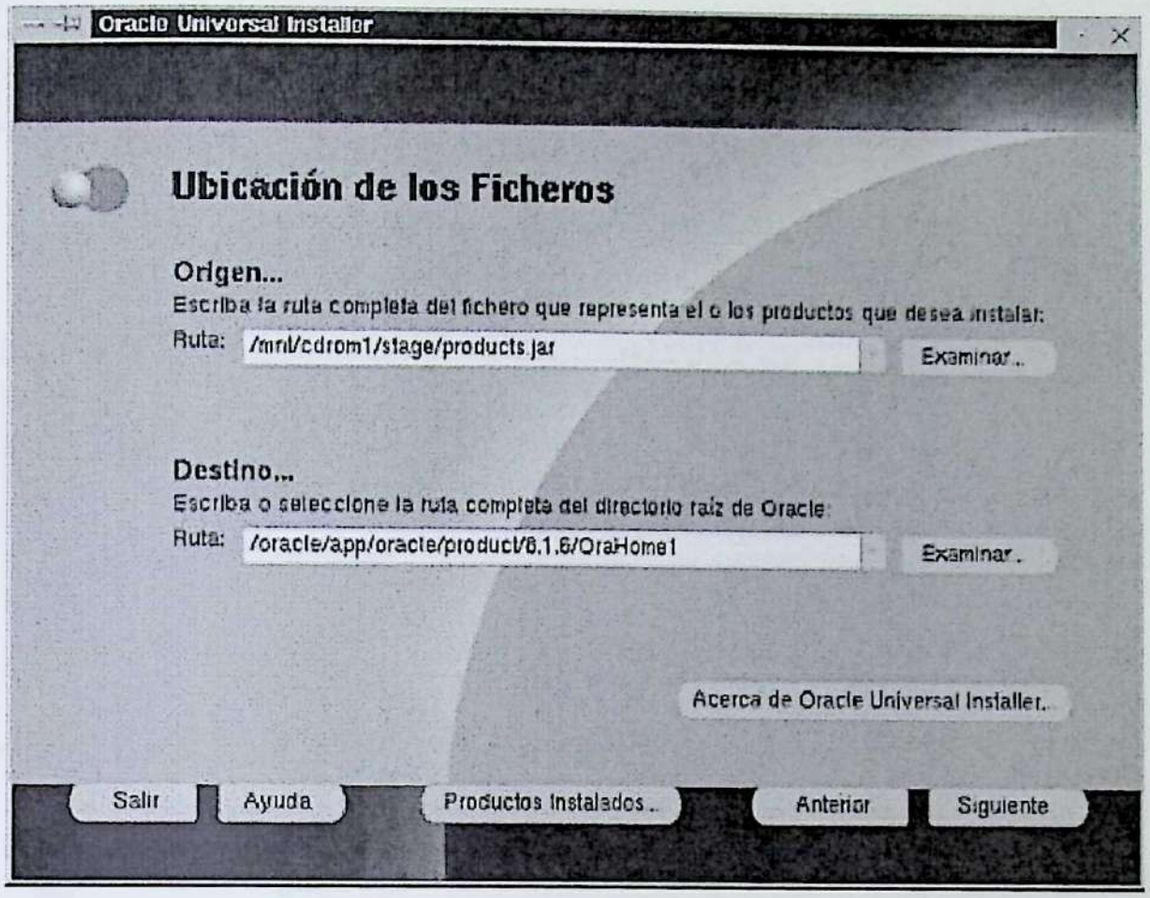
Para ejecutar el instaler, debemos entrar en el entorno de ventanas con el usuario oracle, y una vez dentro ir al directorio en el que tenemos montado el CD, y ejecutar:

./runInstaller

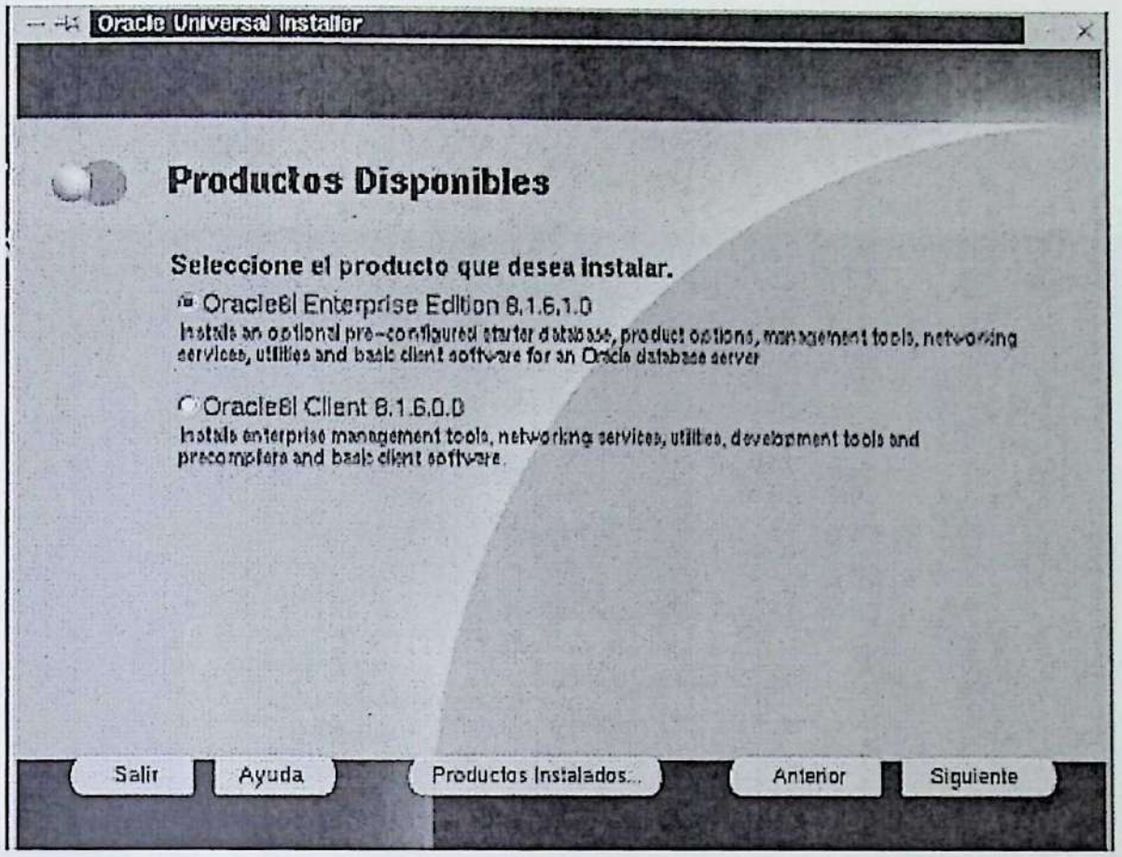
Aparecerá la siguiente pantalla:



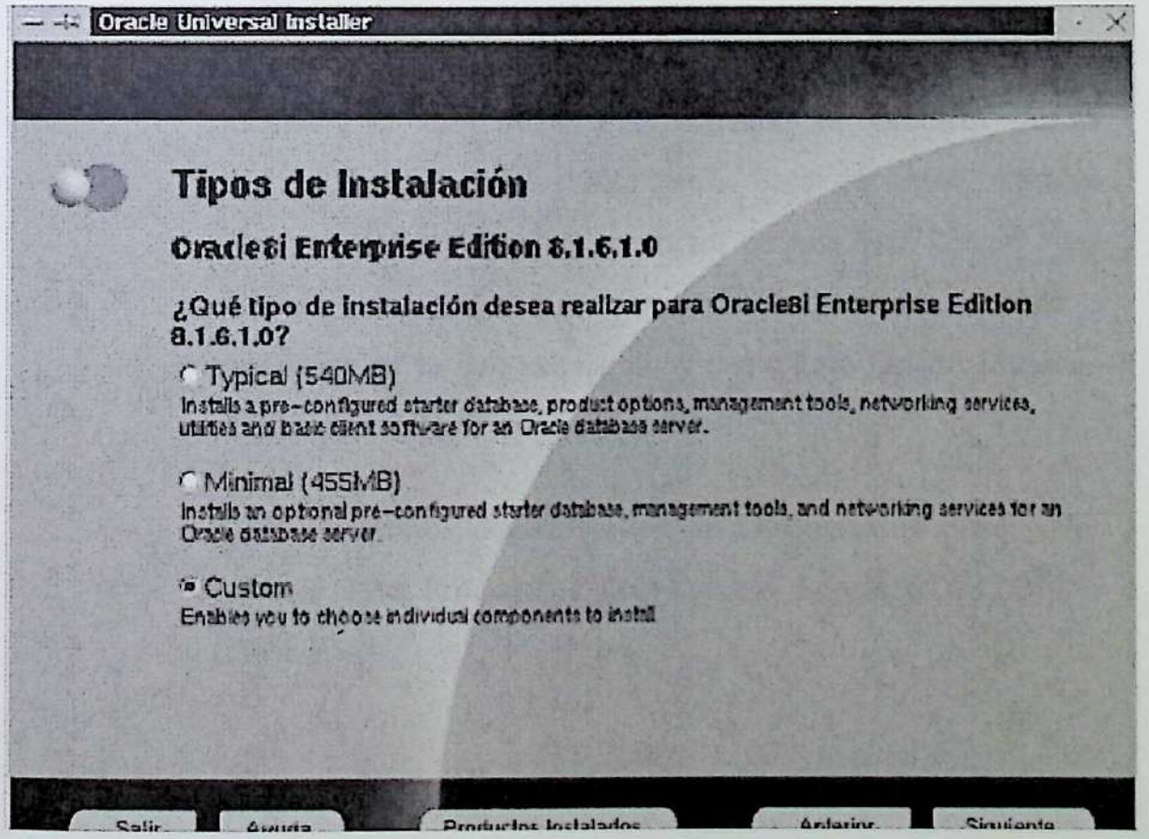
Es la pantalla de bienvenida, pulsamos siguiente y vamos a la siguiente :



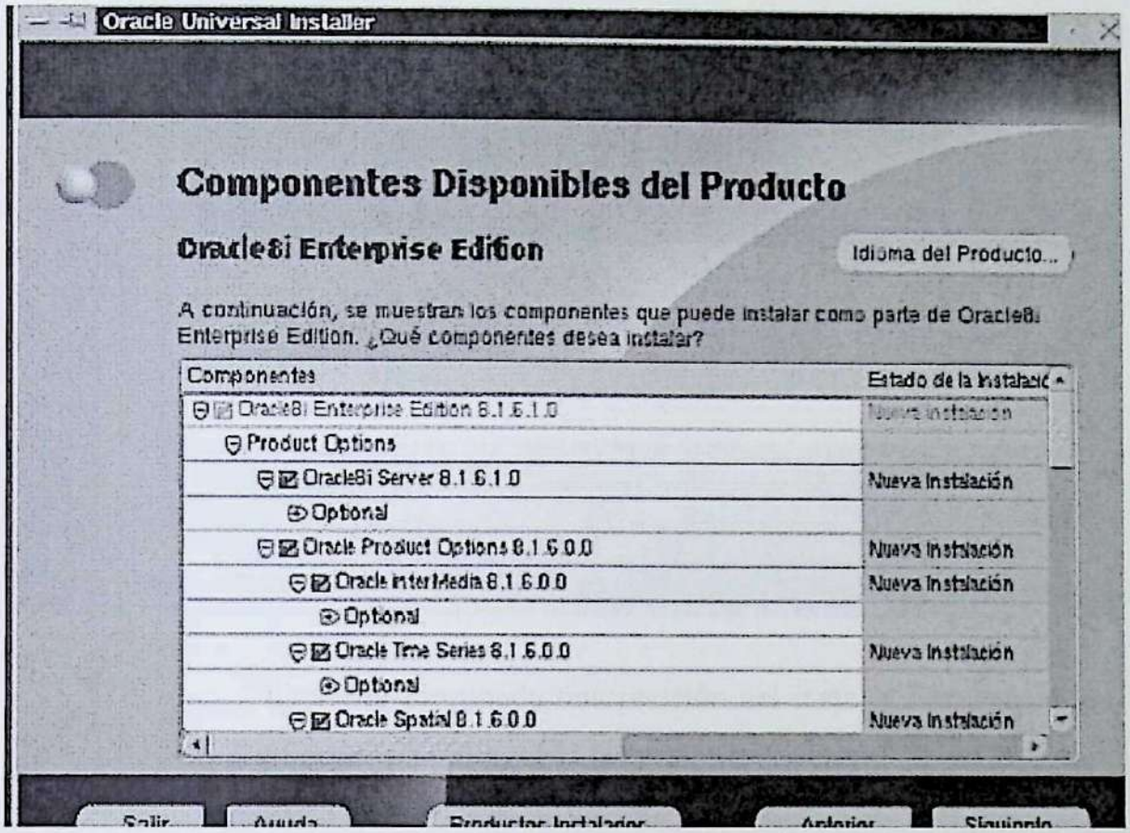
En la que seleccionamos la ruta en la que están los productos a instalar, y la ruta en la que se instalarán, en nuestro caso esta última es : **/oracle/app/oracle/product/8.1.6,** una vez seleccionados los valores correctos pasamos a la siguiente pantalla:



Aquí seleccionamos la primera opción, que es instalar el motor de la base de datos, continuamos con...



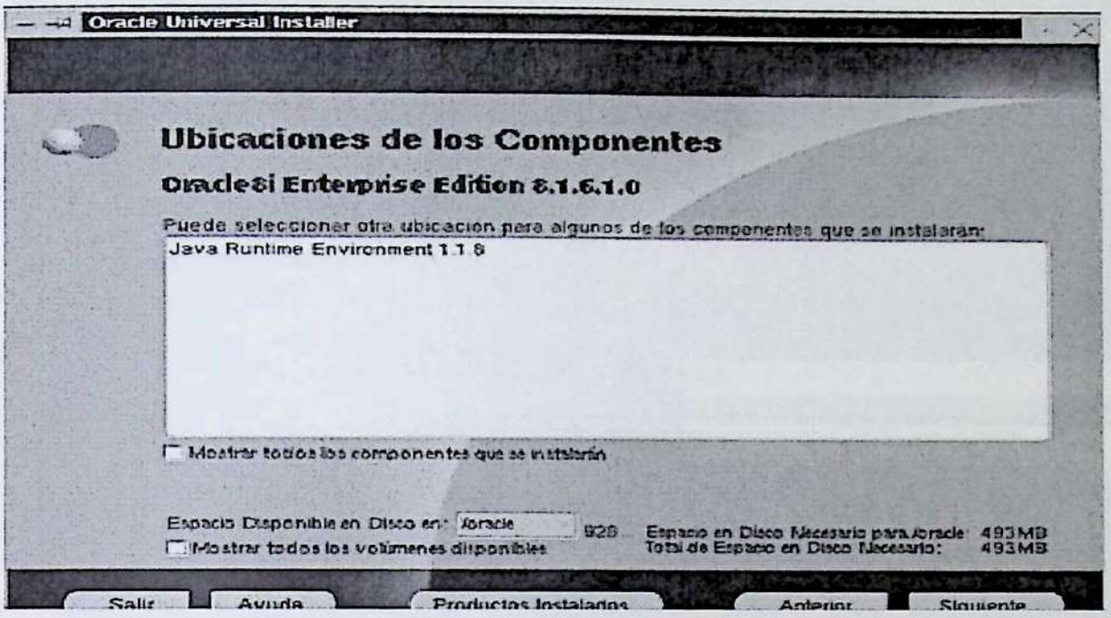
... el tipo de instalación, yo suelo elegir la opción personalizada, para seleccionar los productos que se quieren instalar...



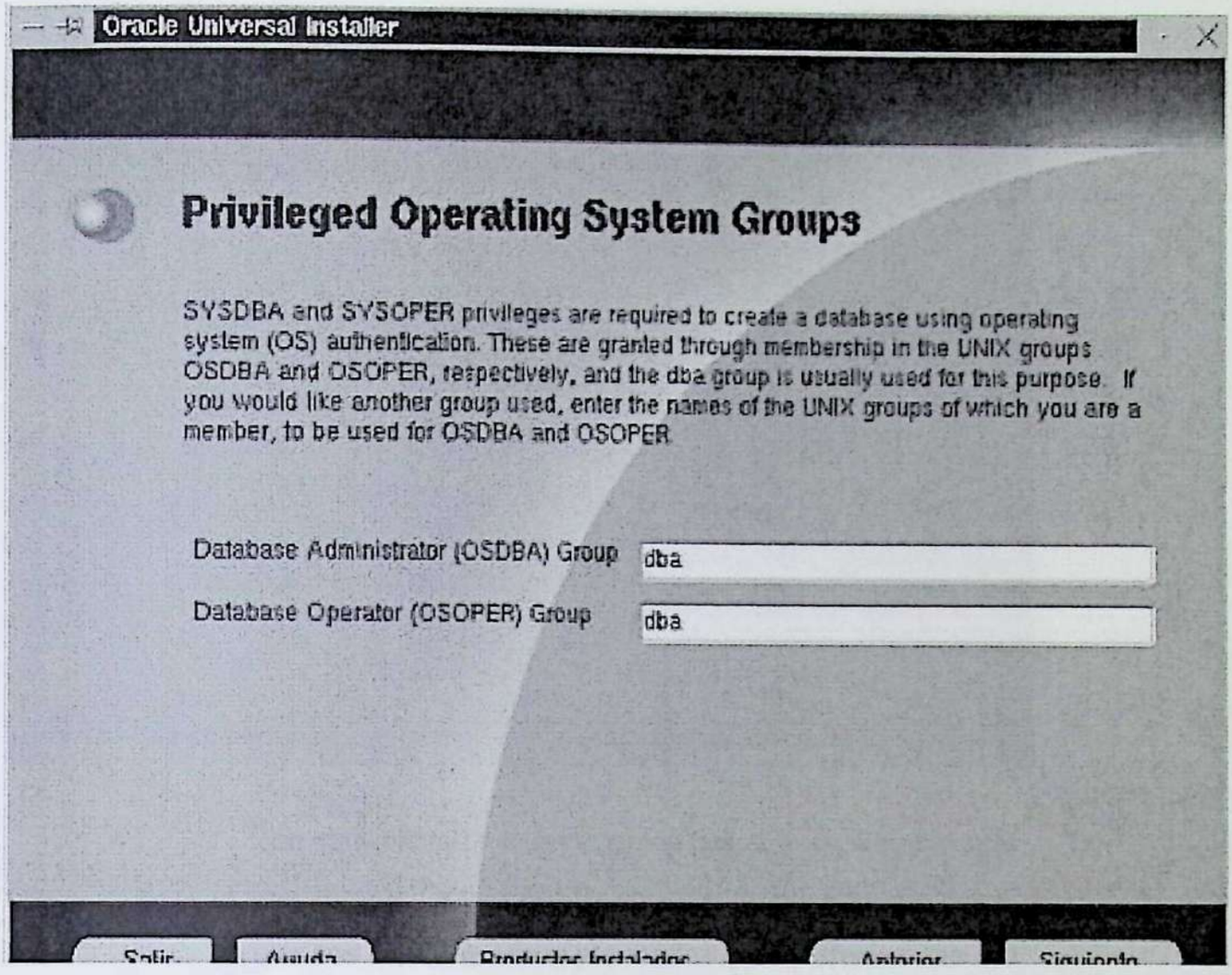
...en la siguiente pantalla seleccionamos los productos a instalar, el que debe estar marcado es Oracle8i Server 8.1.6.1.0, que es el motor de la base de datos, también debemos seleccionar los productos de red, Net8 Client, Net8 Server, para poder conectarnos a la base de datos desde la máquina en la que instalamos y por último Oracle Database Utilities + SQL*Plus.

Si queremos desarrollar en java, deberemos instalar los Oracle Java Products, JDBC Drivers, Oracle SQLJ, Oracle Java Tools.

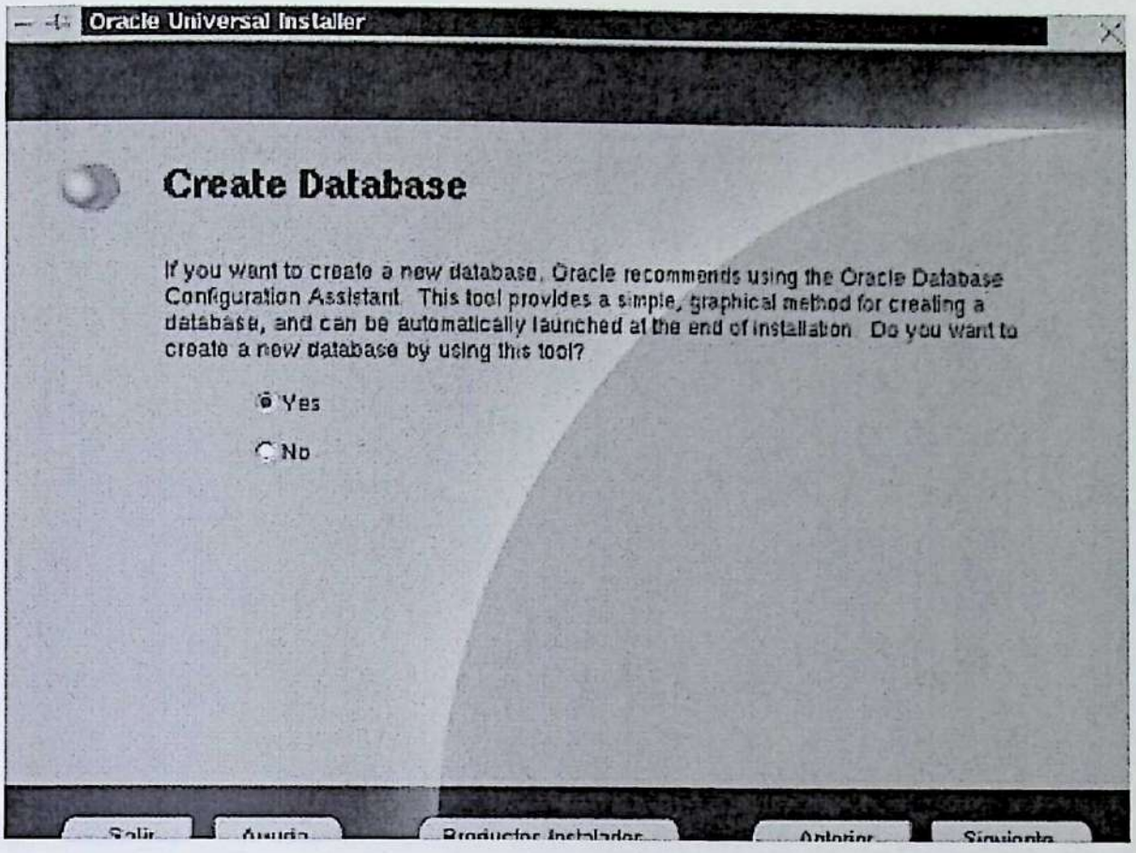
También podemos instalar los distintos cartuchos que nos ofrece Oracle, Oracle Intermedia, Oracle Times Series, Oracle Spatial.



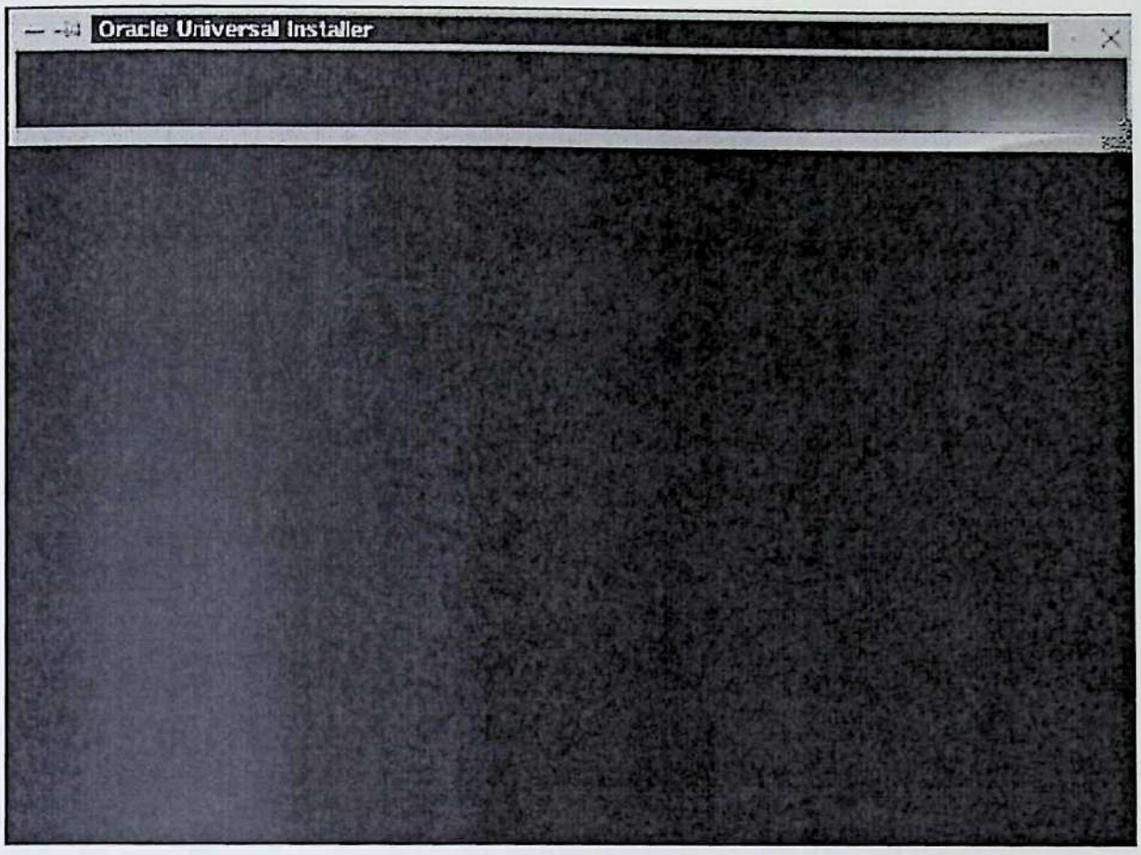
Una vez terminada una revisión del instaler llegamos a la siguiente pantalla, en la que aceptaremos el valor por defecto...



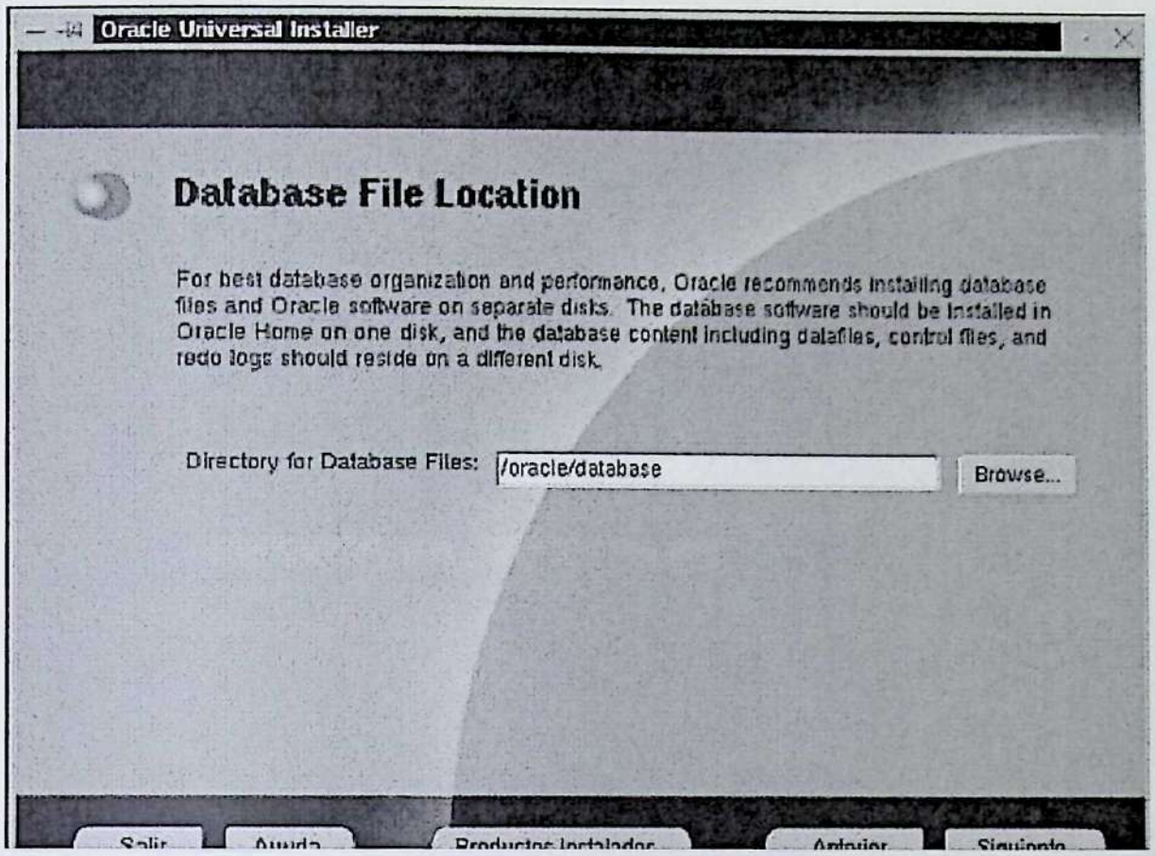
...ahora se nos piden los grupos del sistema que harán de DBA y de operador, en nuestro caso como únicamente creamos uno ,dba , lo marcamos en las dos casillas...



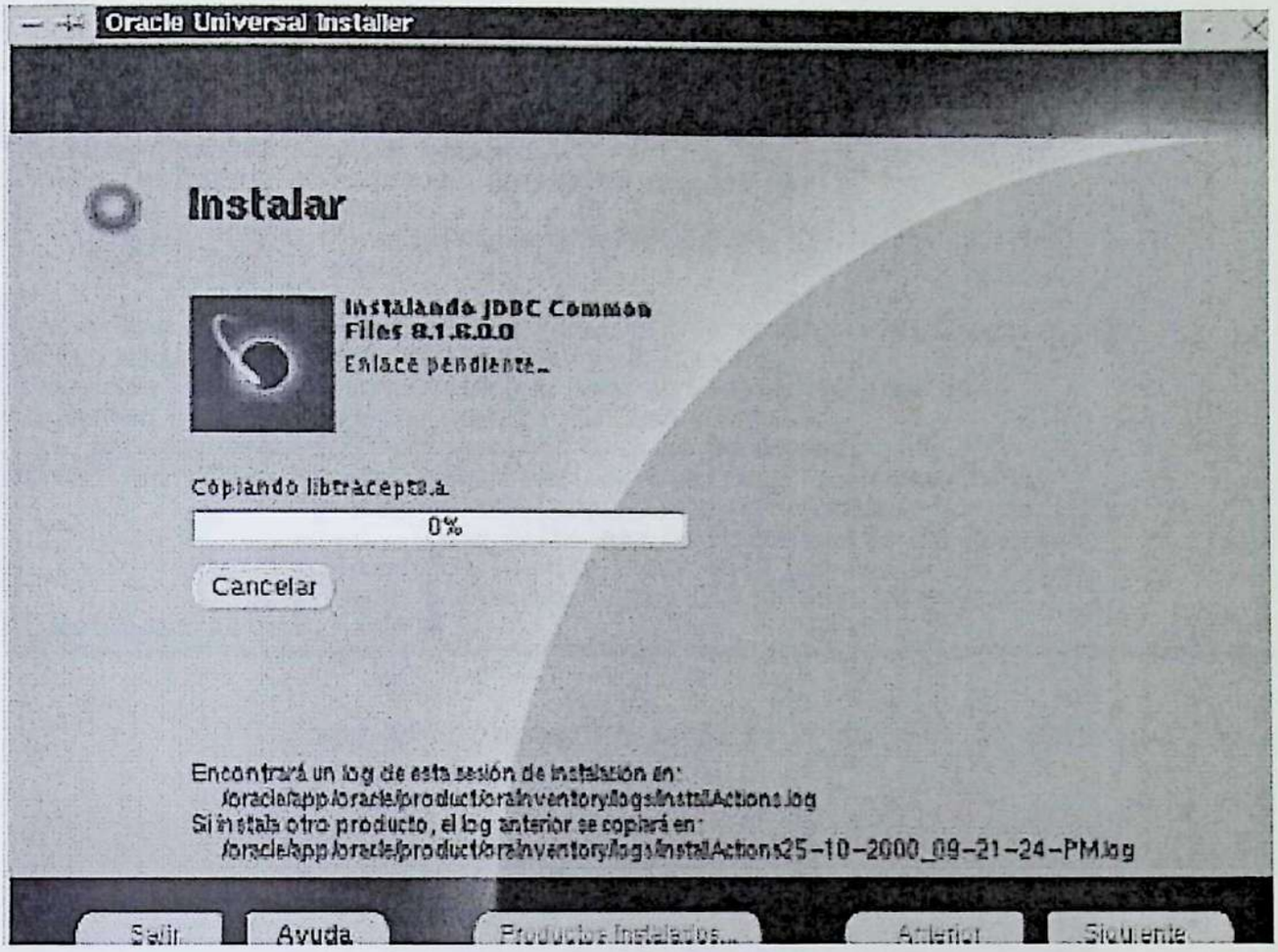
...en esta pantalla seleccionamos que si queremos ejecutar el asistente de Oracle para la creación de una base de datos, ya que nos creará unos scripts con los que crearemos una BD de una manera sencilla...



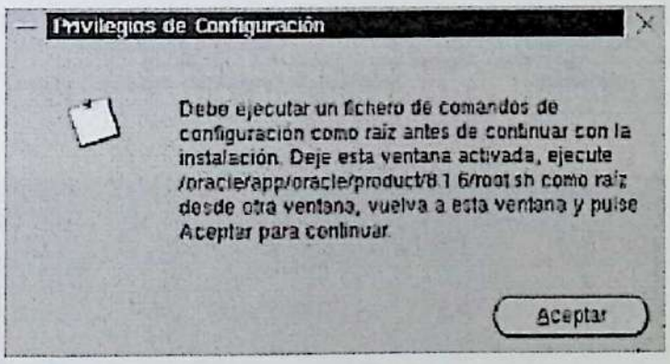
...aquí debemos elegir el nombre que tendrá nuestra base de datos, uno global, que debe ser único en un entorno distribuido, p.e un entorno de réplica, y el SID...



...seleccionamos el directorio en el que se crearan los ficheros de la base de datos...



..., después de ver en una pantalla los productos que se instalarán, pasamos a la pantalla de progreso de la instalación...

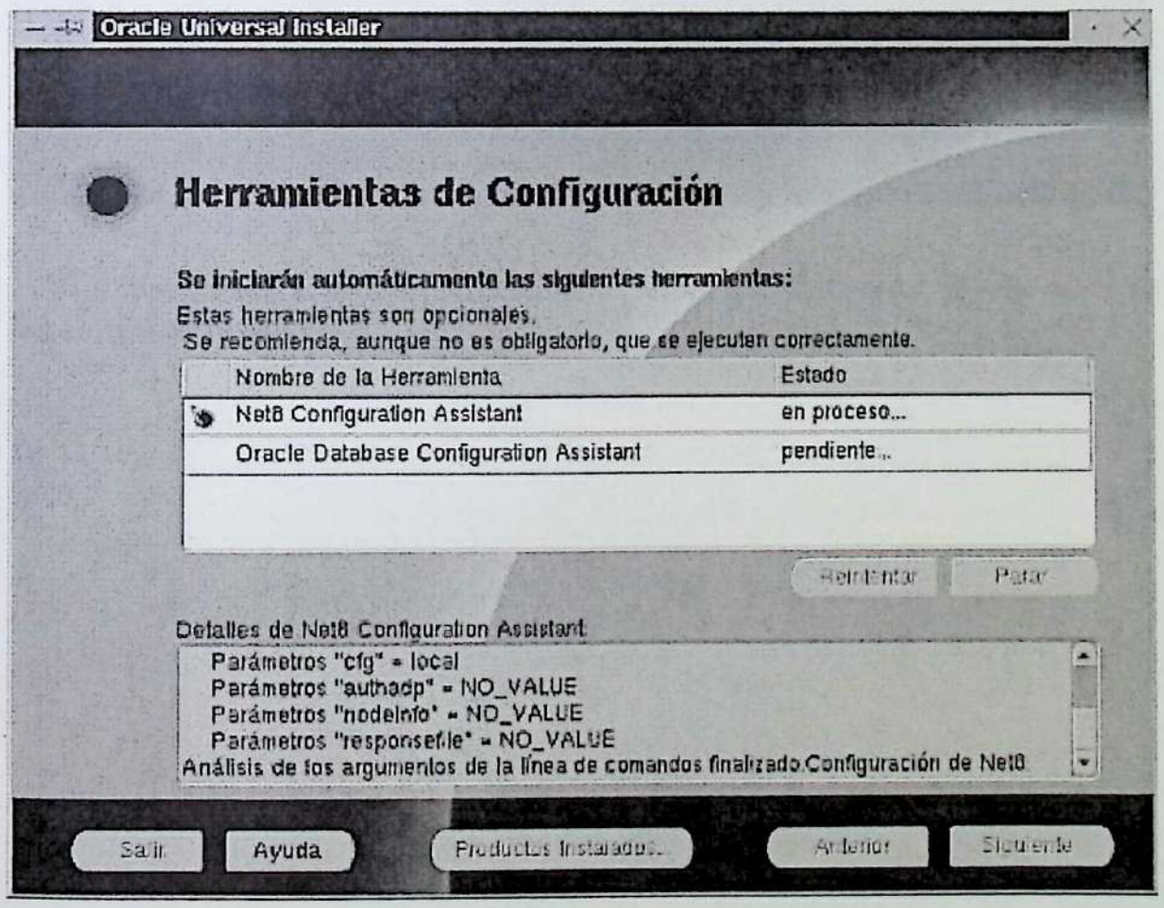


..., una vez terminada la instalación, desde una ventana de comandos, como usuario root, debemos ejecutar el fichero root.sh, que se encuentra en el ORACLE_HOME, para poder seguir con la instalación...

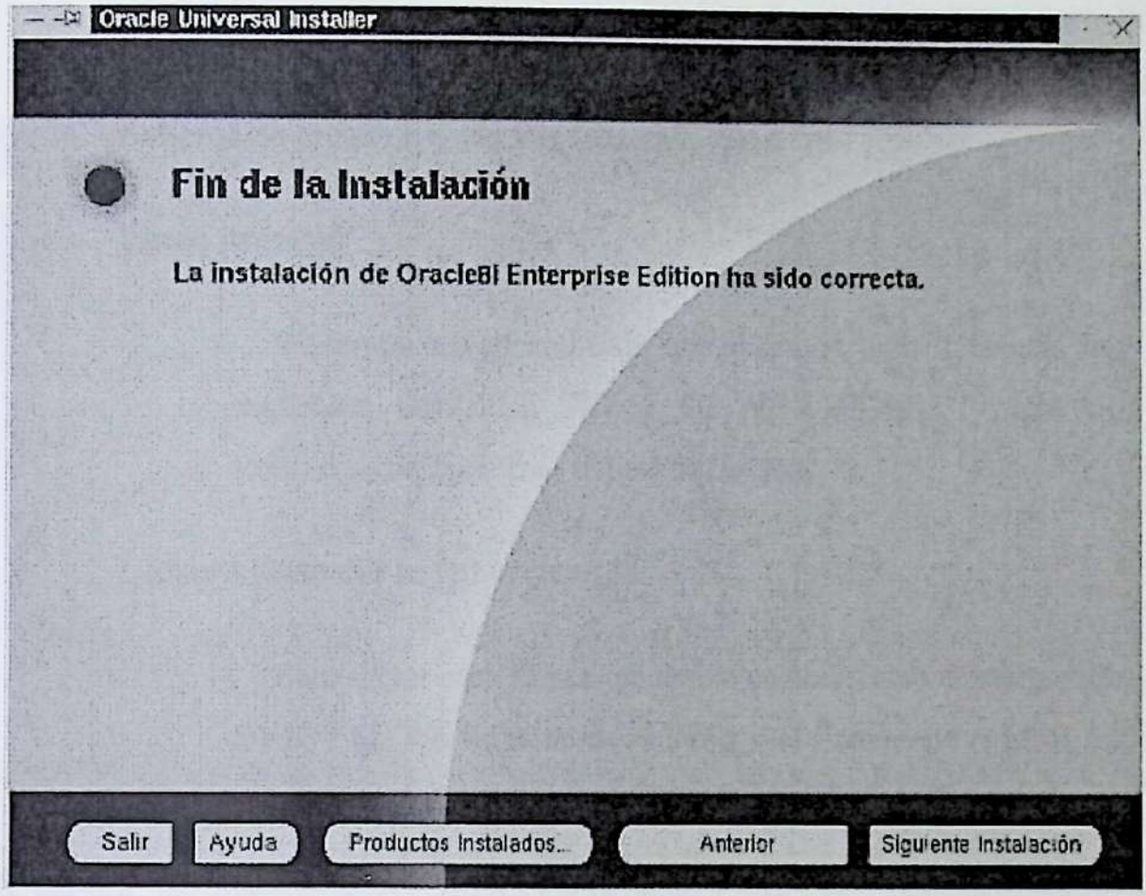
```
[root@lazarillo 8.1.6]# ./root.sh
Running Oracle8 root.sh script...
\nThe following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/app/oracle/product/8.1.6
    ORACLE_SID=   orcl

Enter the full pathname of the local bin directory: [/usr/local/bin]:
Entry will be added to the /etc/oratab file by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
IMPORTANT NOTE: Please delete any log and trace files previously
                created by the Oracle Enterprise Manager Intelligent
                Agent. These files may be found in the directories
                you use for storing other Net8 log and trace files.
                If such files exist, the OEM IA may not restart.
[root@lazarillo 8.1.6]#
```

..., aquí se nos muestra la salida de la ejecución de ese fichero...



... por último pasamos a ejecutar los distintos asistentes, el asistente para configurar Net8 y el asistente para crear una base de datos.



Y la instalación terminó correctamente, ahora sólo falta crear la base de datos a partir de los scripts generados.

Capitulo V

Creación de la base de datos

Como ya he dicho antes, prefiero no lanzar la creación de la base de datos desde el asistente, ya que si algo falla, hay que empezar otra vez. Por eso una vez terminada la instalación voy al directorio donde se han generado los ficheros de creación y los voy lanzando uno a uno.

Pasos previos

Debemos asegurarnos de que tenemos bien definidas las variables ORACLE_SID y ORACLE_HOME, y que de verdad se ha creado el fichero ***sqlorcl.sh***.

Lanzamiento del script orclrun.sh

Este script crea la base de datos, es decir, crea el tablespace SYSTEM, los ficheros de redolog y el fichero de control. El comando que ejecuta es el CREATE DATABASE.

Lanzamiento del script orclrun1.sh

Este script realiza más cosas que el anterior. Lanza la creación del catálogo de la base de datos, crea el resto de los tablespaces que definimos desde el asistente, crea los segmentos de rollback y cambia el tablespace temporal de los usuario administradores de la base (sys, system).

El lanzamiento de este script genera un fichero de log en $ORACLE_HOME/admin/oracle/create/crdb2.log.

Podemos lanzar el siguiente comando para ver si hay errores, en el comando quitamos los errores que se producen por el intento de eliminar un objeto que no existe en la base

de datos, ya que cuando se lanzan los catálogos de la base de datos, Oracle primero elimina los objetos y luego los crea.

cat $ORACLE_HOME/admin/oracle/create/crdb2.log \|grep ORA- \|grep -v 1432 \|grep -v 942 \|grep -v 1434 \|grep -v 1921 \| grep -v 1919 > errores.log

Haciendo un cat sobre el fichero de errores.log, no nos debería aparecer nada.

Lanzamiento del script orclrun2.sh

Este es el último script que se ejecutará en la creación de la base de datos en el caso de no escoger ninguna de las opciones que nos da la instalación (soporte Java, InterMedia...). En el se pasan los catálogos de soporte PL/SQL, traza...

Aquí se producen una serie de errores típicos debido a que no existen los objetos que se intentan borrar, como son 942, 1432, 1434, 1918, 1919, 2289 y 4043.

Lanzamiento del script orclreplicate.sh

Este script se genera por el asistente en el caso de escoger la opción de réplica, y lanza el catálogo necesario para la réplica avanzada (sólo en la versión Enterprise).

Los errores típicos son 955, 1432, 1434.

Lanzamiento del script orcljava.sh

Como su nombre indica, es el encargado de lanzar los catálogos para crear el soporte Java en la base de datos.

Los errores típicos son 942, 1418, 1919.

Lanzamiento del script orclordinst.sh

Este catálogo no estoy muy seguro lo que hace, pero creo que es un catálogo previo a la instalación del módulo Oracle InterMedia.

Un error típico es 1432.

Lanzamiento del script orcliMedia.sh

Crea el soporte para el módulo InterMedia.

Los errores típicos son : 942, 1432, 2289, 4043.

Lanzamiento del script orcldrsys.sh

Crea un tablespace DRSYS.

Lanzamiento del script orclcontext.sh

Soporte para Oracle Context.

Lanzamiento del script orclsqlplus.sh

Crea la ayuda en SQLPLUS.

Lanzamiento del script orclalterTablespace.sh

Cambia el tablespace por defecto del usuario system.

Funciones Oracle

Ora_Bind

(PHP 3, PHP 4)

Ora_Bind -- Vincula una variable PHP a un parámetro Oracle

Descripción

int **ora_bind** (int cursor, string nombre de variable PHP, string nombre de parámetro SQL, int longitud [, int tipo])

Devuelve verdadero si el vínculo se realiza con éxito, y sino devuelve falso. Los detalles de los errores pueden examinarse usando la funciones **ora_error()** y **ora_errorcode()**.

Esta función liga la variable PHP nombrada con el parámetro SQL. El parámtro SQL debe estar en la forma ":name". Con el parámetro optativo tipo, se define si el parámetro SQL se trata de un parámetro de entrada/salida (0 y por defecto), entrada (1) o salida (2). Como en PHP 3.0.1, se puede usar las constantes ORA_BIND_INOUT, ORA_BIND_IN y ORA_BIND_OUT en lugar de los números.

ora_bind debe ser llamada después de **ora_parse()** y antes de **ora_exec()**. Los valores de entrada pueden pasarse por asignación a las variables PHP vinculadas, despues de la llamada a **ora_exec()** dichas variables contendrán los valores de salida, si éstos estuvieran disponibles.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
```

```
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close

(PHP 3, PHP 4)

Ora_Close -- Cierra un cursor Oracle

Descripción

int **ora_close** (int cursor)

Devuelve verdadero si el cierre fué exitoso, o falso de lo contrario. Los detalles de los errores se recuperan usando las funciones **ora_error()** y **ora_errorcode()**.

Esta función cierra un cursor de datos abierto con **ora_open()**.

Ora_ColumnName

(PHP 3, PHP 4)

Ora_ColumnName -- toma el nombre de una columna de resultados Oracle

Descripción

string **Ora_ColumnName** (int cursor, int column)

Devuelve el nomre de un campo/columna *column* en el cursor *cursor*. el nombre devuelto estará en letras mayúsculas.

Ora_ColumnType

(PHP 3, PHP 4)

Ora_ColumnType -- toma el tipo de una columna de resultados Oracle

Descripción

string **Ora_ColumnType** (int cursor, int column)

Devuelve el nombre del tipo de datos del campo o columna *column* en el cursor *cursor*. Se devolverá un tipo de datos, de entre los siguientes:

"VARCHAR2"

"VARCHAR"

"CHAR"

"NUMBER"

"LONG"

"LONG RAW"

"ROWID"

"DATE"

"CURSOR"

Ora_Commit

(PHP 3, PHP 4)

Ora_Commit -- realiza una transacción Oracle

Descripción

int **ora_commit** (int conn)

Devuelve verdadero si es exitosa, de lo cantrario devuelve falso. Puede verse los detalles del error usando las funciones **ora_error()** y **ora_errorcode()**. Esta función realiza una transacción Oracle. Se define como transacción cualquier cambio en una conexión dada, desde la última tarea/retroceso en la ejecucion (rollback), anulación de la ejecución automática de tareas (autocommit), o cuando se ha establecido la conexión.

Ora_CommitOff

(PHP 3, PHP 4)

Ora_CommitOff -- deshabilita el modo automatico de ejecucion de tareas (autocommit)

Descripción

int **ora_commitoff** (int conn)

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

Esta función deshabilita la ejecucion automatica luego de cada instancia **ora_exec()**.

Ora_CommitOn

(PHP 3, PHP 4)

Ora_CommitOn -- Habilita la ejecucion automática de tareas (autocommit)

Descripción

int **ora_commiton** (int conn)

Esta función habilita la ejecucion automatica luego de cada instancia **ora_exec()** en la conexión dada.

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Error

(PHP 3, PHP 4)

Ora_Error -- toma los mensajes de error de Oracle

Descripción

string **Ora_Error** (int cursor_or_connection)

Devuelve los mensajes de error en la forma *XXX-NNNNN* donde *XXX* es la procedencia del error y *NNNNN* es la identificación del mensaje de error.

> **Nota:** El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

En las versiones UNIX de Oracle, pueden encontrarse detalles acerca de un mensaje de error como este: $ **oerr ora *00001*** 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key

Ora_ErrorCode

(PHP 3, PHP 4)

Ora_ErrorCode -- captura el código de error Oracle

Descripción

int **Ora_ErrorCode** (int cursor_or_connection)

Devuelve el codigo numérico de error de la última declaración ejecutada en el cursor o conexion espacificada.

> **Nota:** El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

Ora_Exec

(PHP 3, PHP 4)

Ora_Exec -- ejecuta las declaraciones interpretadas en un cursor Oracle

Descripción

int **ora_exec** (int cursor)

Devuelve verdadero ante la ejecución exitosa, de lo contrario, devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Fetch

(PHP 3, PHP 4)

Ora_Fetch -- extrae una fila de datos a partir de un cursor

Descripción

int **ora_fetch** (int cursor)

Devuelve verdadero (se extrajo una fila) o falso (no hay mas filas, o ha ocurrido un error). Si ocurre un error, los detalles del mismo pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Si no hubo errores, **ora_errorcode()** devolverá 0. Recupera una hilera de datos partiendo de un cursor especificado.

Ora_GetColumn

(PHP 3, PHP 4)

Ora_GetColumn -- toma datos de la fila extraída

Descripción

mixed **ora_getcolumn** (int cursor, mixed column)

Devuelve la columna de datos. Si hay un error, se devuelve falso y **ora_errorcode()** devulve un valor distinto de cero. Note, de igual manera, que la busqueda de un resultado Falso en esta función, puede resultar verdadera, aún en casos en que no ocurran errores:(resultado NULO, cadenas vacias, valor 0 o cadenas "0"). Extrae los datos para una columna o resultado de función.

Ora_Logoff

(PHP 3, PHP 4)

Ora_Logoff -- cierra una conexión Oracle

Descripción

int **ora_logoff** (int connection)

Devuelve verdadero si es exitosa, o falso si hay errores. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Cierra la sesión de trabajo del usuario, y lo desconecta del servidor.

Ora_Logon

(PHP 3, PHP 4)

Ora_Logon -- Abre una conexión Oracle

Descripción

int **ora_logon** (string usuario, string contraseña)

Establece una conexión entre PHP y una base de datos Oracle, con los datos de nombre de usuario y contraseña especificados.

Las conexiones pueden llevarse adelante usando SQL*Net indicando el nombre TNS al *usuario* de este modo:

```
$conn = Ora_Logon("usuario@TNSNAME", "contraseña");
```

Si hubiesen datos con caracteres no-ASCII, habría que asegurarse de que esté presente la variable de entorno NLS_LANG en el sistema. Para los modulos de servidor, deberían definirse en el entorno del servidor antes de iniciarlo.

Devuelve el índice de la conexión si aquella tuvo éxito, de lo cantrario devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Open

(PHP 3, PHP 4)

Ora_Open -- abre un cursor Oracle

Descripción

int **ora_open** (int connection)

Abre un cursor asociado con la conexión.

Devuelve el índice del cursor o Falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Parse

(PHP 3, PHP 4)

Ora_Parse -- interpreta una declaración SQL

Descripción

int **ora_parse** (int cursor_ind, string sql_statement, int defer)

Esta función interpreta una declaración SQL o un bloque PL/SQL y los asocia con el cursor dado. Devuelve 0 si se ejecuta con exito, o -1 ante un error.

Ora_Rollback

(PHP 3, PHP 4)

Ora_Rollback -- retrocede en la lista de transacciones (hace un roll back)

Descripción

int **ora_rollback** (int connection)

Deshace una transaccion Oracle. (Ver **ora_commit()** para la definición de transacción.)

Devuelve verdadero si tiene éxito, o falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Opciones de Perfil en Oracle Applications

En Oracle Applications existen unas partes muy importes a nivel general de toda la aplicación. Estas son los perfiles (Profiles).

Estos perfiles afectan la operación de Oracle Applications, y tambien nos sirven para tener un mejor funcionamiento de Oracle Applications, tanto a nivel de Userios, responsabilidades, etc.

El Administrador del Sistema (System Administrator) es el responsable de la configuracion y el manejo de estos perfiles. A continuacion se dara una descripcion de algunos opciones de perfil.

Cada una de las aplicaciones de Oracle Applications, como:

- *Oracle General Ledger*
- *Oracle Payables*
- *Oracle Inventory*
- *Oracle Assets*
- *Oracle Order Entry*
- *Oracle Payroll*
- *Oracle Purchasing*

- Entre otras.

cuentan con unas opciones de perfil ya sea a nivel de:

- *Site (sitio)*
- *Application (Aplicación)*
- *Responsibility (Responsabilidad)*
- *User (Usuario)*

En los cuales se puede configurar uno o mas perfiles de los niveles anteriores.

PERFIL	**PRIORIDAD**	**DESCRIPCION**
Site	4	Opción que pertenece a todos los usuarios de la instalación
Application	3	Opción que pertenece a todos los usuarios con alguna responsabilidad asociada a una aplicación
Responsibility	2	Opción que pertenece a todos los usuarios con una resposabilidad
User	1	Opción que pertenece a los usuarios individuales identificados en su aplicación

Y cada una de estas es manejada de acuerdo a una prioridad, como se muestra anteriormente.

Ej :

User esta por encima de *Responsibility*, este por encima de *Application*, etc...

Cuando una opcion de perfil es configurada en mas de un nivel, Oracle Applications tomando la prioridad según el ejemplo anterior.

Estas opciones de perfil pueden ser configuradas por la responsabilidad del Programador de la Aplicación (*AOL-Application Object Library*), por la pantalla (**\Navegar\Perfil**), pero son modificadas en la responsabilidad del Administrador del sistema (*System Administrator*), por la pantalla (**\Navegar\Perfil\Sistema**)

Las Opciones de Perfil deben ser usadas como un valor por defecto, para parametros de Programas recurrentes, parametros de Juego de informes o segmentos de flexfields

A continuacion se muestra algunos ejemplos de opciones de perfil en Oracle Applications:

1. Application Object Library (AOL) - Programador de la Aplicacion

Perfil	Concurrent: Request Copies (*Recurrente : Copias Solicitadas*)
Descripción	Con esta opción se puede colocar el número de copias que se va a imprimir por cada documento.

Ej:

Opción	**Valor**
Concurrent: Report Copies	1

Oracle Receivables (Cuentas por Cobrar)

Perfil	AR: Update Due Date (*AR :Modificar Fecha de Vencimiento*)
Descripción	Con esta opción se puede configurar si se puede o no, cambiar la fecha de vencimiento de los items en las facturas de mantenimiento.

Ej:

Opción	**Valor**
AR: Update Due Date	Yes/No

Oracle Assets (Activos Fijos)

Perfil	FA: Print Debug (*AF :Realizar debug en Impresion*)
Descripción	Con esta opción sus mensajes de debug son impresos como archivos de log del concurrent Manager (Manejador Concurrente). Esto se utiliza mas que todo como herramiente para identificar un problema con el código.

Ej:

Opción	**Valor**
FA: Print Debug	Yes/No

Oracle Inventory (Inventarios)

Perfil	INV: Updatable Item Name (*INV : Modificar Nombre de Articulo*)
Descripción	Con esta opción se puede modificar el item

	flexfield o nombre de un item.

Ej:

Opción	**Valor**
INV: Updatable Item Name	Yes/No

Bills of Materials

Perfil	BOM:Check for Duplicate Configuration (*BOM : Verificar configuracion duplicada*)
Descripción	Con esta opción BOM trata de buscar una configuración duplicada al crear un nuevo item.

Ej:

Opción	**Valor**
BOM:Check for Duplicate Configuration	Yes/No

Order Entry (Pedidos)

Perfil	OE: Immediate Inventory Update (*OE : Modificar inmediatamente el inventario*)
Descripción	Con esta opción se controla por defecto el proceso de confirmación de envio

Ej:

Opción	**Valor**
OE:Immediate Inventory Update	Yes/No

Oracle Purchasing (Compras)

Perfil	PO: Legal Requisition Type (*PO :Tipo de requision legal*)
Descripción	Con esta opción se indica si los usuarios pueden ingresar requisiciones internas, o requisiones de compra

Ej:

Opción	**Valor**
PO: Legal Requisition Type	Both/Internal/Purchase

Oracle General Ledger (Contabilidad)

Perfil	Use Performance Module (*Usar modula de desempeño*)
Descripción	Con esta opción se hace que los programas concurrentes de Oracle General Ledger (contabilidad), realicen datos estadísticos para optimizar los programas tales como: Contabilización, Sumarización, Consolidación Asignación de rangos de Presupuestos, Asignación de tasas historicas, MassAllocations y translado y cierre de fin de año. El valor por defecto es Yes (Si)

Ej:

Opción	Valor
Use Performance Module	Yes/No

En cada una de las aplicaciones existe un perfil muy importante para la integración de todas las aplicaciones :

Perfil	Name set of books of GL (*Nombre juego libros GL*)
Descripción	Con esta opción se configura cual es el juego de libros que va a utilizar cualquier aplicación de Oracle Finanacials.

Ej:

Opción	Valor
Name set of books of GL	<Los juegos de libros definidos en GL>

Los usuarios de Oracle Applications pueden usar, modificar o deshabilitar estas opciones de perfil para tener un buen funcionamiento de su aplicación a nivel de toda una instalación. Para mayor información consultar en los ***manuales*** de cada unas de las aplicaciones.

COMANDOS HOST EN PL/SQL Y SQL DINÁMICO CON PIPES

Este artículo presenta una técnica para ejecutar comandos 'host' desde PL/SQL, incluyendo código fuente para su implementación. Adicionalmente la técnica permite ejecutar comandos DDL (Data Definition Language) y DML (Data Manipulation Language) con excepción de los 'selects'.

Una de las ventajas de esta técnica es poder utilizar desde Forms sentencias DDL como por ejemplo un 'Create Table'. Adicionalmente se podrían ejecutar comandos 'host' en el servidor desde herramientas de Developer/2000.

PL/SQL no provee por si solo la capacidad de ejecutar comandos 'host', pero esto se puede lograr mediante el uso de un paquete incluido junto con la opción procedimental conocido como DBMS_PIPE. Este paquete sólo está disponible a partir de PL/SQL versión 2.1, que viene con ORACLE 7.1, si usted posee una versión anterior de ORACLE no podrá poner en práctica el contenido de este artículo. Adicionalmente usted debe tener instalado en su máquina el precompilador PRO*C y un compilador de C, que esté soportado por la versión correspondiente de PRO*C.

El paquete DBMS_PIPE pertenece al usuario SYS y es creado cuando se instala la opción procedimental en su base de datos. Para utilizarlo usted debe tener privilegios de ejecución sobre él. En caso de no tener privilegios consulte a su DBA. En términos generales este paquete permite enviar mensajes entre sesiones conectadas a la misma base de datos. Una de estas sesiones podría ser el bloque PL/SQL que queremos haga el comando 'host' y otra podría ser una sesión creada por un programa en C que corre en el servidor en el que se quiere ejecutar el comando 'host'.

Este artículo incluye dos archivos: demonio.pc y demonio.sql.

Demonio.pc es el código fuente de un programa en C llamado demonio, debido a que es un proceso demonio (daemon) que esta atento a recibir mensajes, y que la mayor parte del tiempo esta 'dormido', en un 'loop' esperando por un mensaje que va a recibir a través de un pipe. Cuando recibe algún mensaje el 'despierta' y procesa el mensaje. El término demonio es común en discusiones sobre el sistema operativo UNIX, y adicionalmente existen muchos procesos UNIX que son demonios.

Demonio.sql es el código fuente de un paquete PL/SQL que deberá estar almacenado en la base de datos. Este paquete tiene procedimientos que utilizan el DBMS_PIPE para enviar y recibir mensajes al y del proceso demonio que corre en el servidor.

En conclusión usted va a tener un paquete en la base de datos (llamado demonio) que va a tener procedimientos capaces de enviar mensajes a un proceso demonio (daemon) en el sistema operativo. Los mensajes enviados desde PL/SQL serán los comandos 'host' que se quieran ejecutar y el proceso demonio en el sistema operativo será quien los ejecuta .Es importante tener presente que siempre que se desee utilizar esta funcionalidad se hace necesario que el proceso demonio este corriendo en el servidor.

Para que todo lo anterior funcione en su sistema, siga las siguientes instrucciones:

- Extraiga del artículo los códigos fuente de demonio.sql y demonio.pc en archivos diferentes en su sistema.

- Conéctese a la base de datos a través de sqlplus, el usuario será el que usted determine va a ser el dueño del paquete demonio, tenga en cuenta que si desea que otros usuarios utilicen el paquete posteriormente deberá darles permisos de ejecución sobre el mismo, para esto consulte a su DBA. Una vez este conectado a la base de datos cree el paquete demonio con 'start demonio'.

- Ya creado el paquete usted debe compilar demonio.pc y volverlo un ejecutable. Este proceso podría variar dependiendo su sistema operativo, en sistemas UNIX usted puede utilizar el 'proc.mk' o el 'proc16.mk' para crear su archivo ejecutable. En sistemas VMS usted puede utilizar el 'script' de encadenamiento 'lnproc'. Para que

el programa compile correctamente se debe utilizar la opción de precompilación 'userid', para que el precompilador sepa como conectarse a la base de datos, adicionalmente debido a que existe código PL/SQL embebido en el programa en C se debe utilizar también la opción de precompilación 'sqlcheck=full'. Por ejemplo si usted utiliza el proc16.mk puede hacer lo siguiente:

- Haga una copia de proc16.mk en un archivo llamado demonio.mk, este archivo lo puede encontrar bajo $ORACLE_HOME/proc16/demo. z

- Edite el archivo demonio.mk y busque la línea donde se define el PCCFLAGS, agregue antes la línea 'USERID=usuario/password' y luego incluya el parámetro user=$(USERID) en la línea de PCCFLAGS , adicionalmente deje el parámetro sqlcheck=full en dicha línea, el resultado al final debe ser algo como esto:

```
USERID=scott/tiger
PCCFLAGS= $(PCCINCLUDE) ireclen=132 oreclen=132 sqlcheck=full
ltype=none                                                    \
user=$(USERID)
```

Salve dichas modificaciones y compile el programa demonio.pc con la instrucción:

```
make -f demonio.mk demonio
```

Tenga en cuenta que en el mismo directorio se deben encontrar el archivo demonio.mk y demonio.pc, el resultado debe ser un archivo demonio que es un ejecutable.

- Una vez creado el paquete demonio en la base de datos y compilado el programa demonio.pc se pueden utilizar teniendo en cuenta que primero se debe ejecutar el programa demonio en el servidor y luego desde PL/SQL se pueden utilizar las funciones demonio.execute_system y demonio.execute_sql para pasar mensajes al proceso demonio en el servidor. También se puede utilizar el procedimiento demonio.stop para terminar la ejecución del programa demonio en el servidor. La

función demonio.execute_system se utiliza para ejecutar comandos en el sistema operativo en el servidor y la función demonio.execute_sql se utiliza para ejecutar sentencias SQL que no pueden ser querys (selects).

Recuerde siempre correr primero el proceso demonio en el servidor, si usted lo desea puede correrlo en background.

Para mas información del paquete DBMS_PIPE consulte el apéndice A del manual 'ORACLE 7 Server Application Developer's Guide'

El siguiente es un bloque PL/SQL anónimo que puede ser ejecutado desde sqlplus y que ilustra la forma en que se puede trabajar:

```
declare
  resultado varchar2(20);
begin
  resultado := demonio.execute_system('ls'); /* Ejecuta el comando ls en el servidor */
 dbms_output.put_line(resultado);
exception
 when others then
 dbms_output.put_line(sqlerrm);
end;
/
```

Este es otro bloque PL/SQL utilizando execute_sql, este bloque crea una tabla llamada aaa y que pertenece al usuario con el cual se conecta el proceso demonio en el servidor a la base de datos, ver código fuente de demonio.pc:

```
declare
 resultado varchar2(20);
begin
```

```
 resultado := demonio.execute_sql('create table aaa (col1 number(3))');
 dbms_output.put_line(resultado);
exception
 when others then
dbms_output.put_line(sqlerrm);
end;
/
```

CÓDIGO FUENTE

Archivo: demonio.sql

Este es el código fuente del paquete demonio, el cual manda mensajes al programa demonio que esta escuchando en el servidor (demonio listener) vía dbms_pipe. Este paquete tiene dos funciones y un procedimiento:

Función execute_sql: Pasa un comando sql a través del primer argumento de la función al demonio listener para que sea ejecutado. El comando sql no puede ser un query debido a que no se pueden retornar las filas que trae el query. El usuario oracle que ejecuta el comando sql es aquel con el cual se conecta el programa demonio en el servidor a la base de datos. Esta función retorna el sqlcode después de la ejecución del comando.

Función execute_system: Pasa un comando de sistema operativo (host command) a través del primer argumento de la función al programa demonio que esta en el servidor (demonio listener) para que sea ejecutado en el sistema operativo. Esta función retorna el resultado del comando en el sistema.

Procedimiento stop: Hace que el proceso demonio en el servidor termine su ejecución. Después de la ejecución de ese procedimiento las llamadas hechas a execute_sql y execute_system van a fallar hasta que se reinicie el demonio listener en el servidor.

Este paquete envía el primer mensaje al proceso demonio en el servidor a través de un pipe llamado 'demonio'. Como parte de este mensaje, se envía el nombre del pipe de retorno

que es el que utiliza el proceso demonio en el servidor para devolver mensajes al procedimiento en la base de datos.

El valor de este pipe de retorno lo proporciona el dbms_pipe.unique_session_name. Gracias a lo anterior se puede lograr que cada sesión en la base de datos utilice su propio pipe, y de ninguna manera una sesión recibirá mensajes de otra.

```
create or replace package demonio as

function execute_sql(command varchar2, timeout number default 10)

return number;

/*
Ejecuta sentencias sql o bloques pl/sql con excepción de querys
Parámetros:
command: La sentencia sql a ser ejecutada
timeout: Número de segundos a esperar para enviar o recibir un mensaje.
(Este parámetro es opcional)
Retorna: El sqlcode después de la ejecución de la sentencia
*/

function execute_system(command varchar2, timeout number default 10)
return number;
/*

Ejecuta un comando host.
Parámetros:
command: El comando a ser ejecutado.
timeout: Número de segundos a esperar para enviar o recibir un mensaje.
(Este parámetro es opcional)
Retorna: El valor pasado por el comando al sistema operativo.
*/
```

```
procedure stop(timeout number default 10);

/* Termina la ejecución del proceso demonio.
Parámetros:
timeout: Número de segundos a esperar para enviar o recibir un mensaje.
(Este parámetro es opcional)

*/

end demonio;
/

create or replace package body demonio as
 function execute_system(command varchar2, timeout number default 10)
return number is
s number;
result varchar2(20);
command_code number;
pipe_name varchar2(30);

begin

/* Usar uniqe_session_name para generar un nombre único para el pipe a través del
cual se retornan los mensajes.*/

pipe_name := dbms_pipe.unique_session_name;

/* Enviar el comando SYSTEM al demonio.
El nombre del pipe utilizado para el envío es demonio*/

dbms_pipe.pack_message('SYSTEM');
dbms_pipe.pack_message(pipe_name);
dbms_pipe.pack_message(command);
```

```
s := dbms_pipe.send_message('demonio', timeout);
if s <> 0 then
raise_application_error(-20010,
'Execute_system: Error mientras se enviaba. Status = ' || s);
end if;
```

/* Chequeando el handshake message. En estos momentos se esta escuchando sobre el pipe que es único para la sesión */

```
s := dbms_pipe.receive_message(pipe_name, timeout);
if s <> 0 then
raise_application_error(-20011,
'Execute_system: Error mientras se recibía. Status = ' || s);
end if;
```

/* Obtener del sistema operativo el código del resultado, este código se muestra utilizando el paquete dbms_output.put_line(). */

```
dbms_pipe.unpack_message(result);
if result <> 'done' then
raise_application_error(-20012,
'Execute_system: Comando no ejecutado.');
end if;

dbms_pipe.unpack_message(command_code);
dbms_output.put_line('Se ha ejecutado el host comando. Resultado = ' ||
command_code);
return command_code;
end execute_system;

function execute_sql(command varchar2, timeout number default 10)
```

```
return number is
s number;
result varchar2(20);
command_code number;
pipe_name varchar2(30);

begin

/* Usar uniqe_session_name para generar un nombre único para el pipe a través del
cual se retornan los mensajes.*/

pipe_name := dbms_pipe.unique_session_name;

/* Enviar el comando 'SQL' al demonio a traves del pipe llamado demonio. */

dbms_pipe.pack_message('SQL');
dbms_pipe.pack_message(pipe_name);
dbms_pipe.pack_message(command);
s := dbms_pipe.send_message('demonio', timeout);

if s <> 0 then
raise_application_error(-20020,
'Execute_sql: Error mientras enviaba. Status = ' || s);
end if;
s := dbms_pipe.receive_message(pipe_name, timeout);

if s <> 0 then
raise_application_error(-20021,
'Execute_sql: Error mientras recibía. Status = ' || s);
end if;

/* Obtener el código de resultado de la sentencia, y mostrar usando
dbms_output.put_line(). */
```

```
dbms_pipe.unpack_message(result);
if result <> 'done' then
raise_application_error(-20022,
'Execute_sql: Comando SQL no ejecutado.');
end if;

dbms_pipe.unpack_message(command_code);
dbms_output.put_line('Comando SQL ejecutado. sqlcode = ' || command_code);
return command_code;
end execute_sql;

procedure stop(timeout number default 10) is
s number;

begin

/* Enviar el comando 'STOP' al demonio. */

dbms_pipe.pack_message('STOP');
s := dbms_pipe.send_message('demonio', timeout);
if s <> 0 then
raise_application_error(-20030, 'Stop: Error mientras se enviaba. Status = ' || s);
end if;

end stop;

end demonio;

/
```

Archivo: demonio.pc

Este es el código fuente del demonio listener para implementar sql dinámico y comandos del sistema desde PL/SQL. Este programa acepta tres tipos de comandos:

STOP:Hace que el programa demonio se desconecte de ORACLE y termine su ejecución.

SYSTEM: Hace que el programa demonio ejecute el siguiente ítem en el pipe como un comando del sistema operativo.

SQL: Hace que el programa demonio ejecute el siguiente ítem en el pipe como una sentencia sql, el usuario Oracle que ejecuta la sentencia sql es el definido en la variable uid.

```
#include <stdio.h>
#include <string.h>

EXEC SQL include sqlca;
EXEC SQL begin declare section;
char *uid = "scott/tiger";
```

/* Usuario/password para conectarse a Oracle. Con este usuario se ejecutarían las sentencias sql en la base de datos */

```
int status;
```

/* Valor retornado por el dbms_pipe.send_message y el dbms_pipe.receive_message */

```
varchar command[20];
```

/* Tipo de Comando que el demonio debe ejecutar */

```
varchar value[2000];
```

/* Sentencia SQL o comando host asociado con el tipo de comando que el demonio debe ejecutar*/

```
varchar return_name[30];

/* Nombre del pipe sobre el cual se van a enviar los resultados */

EXEC SQL end declare section;

/* Manejador de errores para la conexión a Oracle. Si se falla al intentar la conexión,
se requiere salir del programa. */

 void connect_error() {
char msg_buffer[512];
int msg_length;
int buffer_size = 512;

EXEC SQL whenever sqlerror continue;
sqlglm(msg_buffer, &buffer_size, &msg_length);
printf("Error del Demonio al conectarse:\n");
printf("%.*s\n", msg_length, msg_buffer);
printf("Demonio terminado.\n");
exit(1);

}

/* Este es el manejador de errores generales. Note que en este caso no se puede salir
del programa. Se debe imprimir el error y continuar. Esto se debe a que alguno de
estos errores probablemente no afecte operaciones futuras por lo cual el demonio
debería seguir corriendo, esto por supuesto dependiendo del error.*/

void sql_error() {
char msg_buffer[512];
int msg_length;
int buffer_size = 512;
```

```
EXEC SQL whenever sqlerror continue;
sqlglm(msg_buffer, &buffer_size, &msg_length);
printf("Error del Demonio al ejecutar:\n");
printf("%.*s\n", msg_length, msg_buffer);
printf("Demonio continua.\n");

}

main() {

EXEC SQL whenever sqlerror do connect_error();
EXEC SQL connect :uid;
printf("Demonio Conectado.\n");
EXEC SQL whenever sqlerror do sql_error();
printf("Demonio esperando...\n");
while (1) {
```

/* Esperar a recibir un mensaje a través del pipe. */

```
EXEC SQL EXECUTE
begin

:status := dbms_pipe.receive_message('demonio');
if :status = 0 then
dbms_pipe.unpack_message(:command);
end if;

end;

END-EXEC;

if (status == 0) {
```

```
/* En este punto se ha recibido un mensaje a través del pipe. Ahora se
debe determinar el tipo de comando que el demonio debe ejecutar */

command.arr[command.len] = '\0';
if (!strcmp((char *)command.arr, "STOP")) {

/* Se ha recibido el comando STOP. El programa debe terminar. */

printf("Demonio Terminando.\n");
break;

}

else if (!strcmp((char *)command.arr, "SYSTEM")) {

/* Se ha recibido el comando SYSTEM. Se deben desempaquetar los
siguientes dos valores. Estos deben ser el nombre del pipe de
retorno y el comando a pasar al sistema operativo. */

EXEC SQL EXECUTE
begin
dbms_pipe.unpack_message(:return_name);
dbms_pipe.unpack_message(:value);
end;

END-EXEC;

value.arr[value.len] = '\0';
printf("Se ejecutara el siguiente comando '%s'\n", value.arr);

/* Ejecutar el comando. */

status = system(value.arr);
```

/* Enviar un mensaje para indicar que el comando ha sido ejecutado.
Enviar el resultado del comando, para esto se utiliza el pipe
con el cual se paso el primer mensaje. */

```
EXEC SQL EXECUTE

begin
dbms_pipe.pack_message('done');
dbms_pipe.pack_message(:status);
:status := dbms_pipe.send_message(:return_name);
end;

END-EXEC;

if (status) {
printf("Error del Demonio al responder al comando en el sistema");
printf(" status: %d\n", status);
}
}

 else if (!strcmp((char *)command.arr, "SQL")) {
```

/* Comando SQL recibido. Se deben desempaquetar los siguientes
dos valores. Estos deben ser el nombre del pipe de retorno y la
sentencia sql a ser ejecutada. */

```
EXEC SQL EXECUTE

begin
dbms_pipe.unpack_message(:return_name);
dbms_pipe.unpack_message(:value);
end;

END-EXEC;
```

```
value.arr[value.len] = '\0';
printf("Se va a ejecutar el comando sql '%s'\n", value.arr);

/* Ejecutar el comando.*/

EXEC SQL whenever sqlerror continue;
EXEC SQL EXECUTE IMMEDIATE :value;

status = sqlca.sqlcode;

/* Iniciar el manejador de errores, y enviar un mensaje para indicar
que el comando ha sido ejecutado. */

EXEC SQL whenever sqlerror do sql_error();
EXEC SQL EXECUTE

begin
dbms_pipe.pack_message('done');
dbms_pipe.pack_message(:status);
:status := dbms_pipe.send_message(:return_name);
end;

END-EXEC;

if (status) {

printf("Error del demonio mientras respondía el comando sql.");
printf(" status: %d\n", status);

}
}

else {

/* Tipo de comando inválido recibido por el demonio. */
```

```
printf("Error del demonio: recibido comando inválido '%s' .\n", command.arr);

}
}

else {

/* Aquí se entra si se recibe algún error mientras el demonio está
esperando. Si status = 1, es un timeout. El timeout por defecto es de 1000 días. */

printf("Error del demonio mientras se encontraba en estado de espera.");
printf(" status = %d\n", status);

}

}

EXEC SQL commit work release;

}
```

Capitulo VI

Introducción a REPORTS

El componente Reports de Oracle Developer es una de las herramientas de generación de informes más potente y con más prestaciones del mercado. Con Oracle Developer Reports (Reports) puede crearse casi cualquier informe que se pueda imaginar. Esta herramienta es particularmente robusta en un entomo cliente-servidor, pero funciona también satisfactoriamente tanto en el modo de proceso por lotes como en un entomo web, utilizando Oracle Developer Server.

Su productividad no tiene parangón. Un usuario experimentado de Reports puede generar informes de producción estándar, relativamente sofisticados, a razón de uno o dos por día. Bajo la presión de una fecha cercana de entrega, uno de los autores llegó a crear nueve informes de producción relativamente sencillos en un solo día.

Una de las prestaciones más potentes de Reports, que no suele utilizarse, es su capacidad para la especificación dinámica, en tiempo de ejecución, de las partes principales del infonne. Esta característica pen-nite a los desarrolladores diseñar sistemas flexibles de generación de informes que reducen, incluso eliminan, la necesidad de disponer de herramientas de consulta *ad hoc.*

Sin embargo, hay que pagar un precio por toda esta potencia y flexibilidad. Para quien no tenga experiencia con Reports, el diseño de informes con este producto puede llegar a ser increíblemente frustrante. Puede ser realmente difícil, para el usuario novel, imaginarse cómo trabajar con la herramienta Layout Model (modelo de diseño). Para poder aprovechar toda la potencia y flexibilidad de las opciones de generación de informes, hay que *utilizarparámetros léxicos* (cadenas de texto que pueden ser definidas e incorporadas al texto de una consulta en tiempo de ejecución). Pocos desarrolladores se han tomado el tiempo necesario para comprender estas prestaciones, fáciles de usar, por otro lado. Muchos desarrolladores ignoran la importancia de los módulos PUSQL predefinidos, requeridos por muchas de las funcionalidades más complejas de Reports.

Para aquellos usuarios que carezcan de una comprensión completa de tales funcionalidades, Reports es tan sólo otra herramienta para diseñar informes, no más fácil de utilizar ni mejor dotada que otras herramientas de diseño de informes basadas en PC. Sin embargo, aquellos que decidan invertir cierto tiempo en la comprensión de este producto hallarán que no existe en el mercado una herramienta de desarrollo para la generación de informes de producción mejor que ésta.

Reports impone pocas limitaciones. Los autores han empleado Reports para mantener agendas telefónicas corporativas con requisitos de generación de infortnes realmente complejos; igualmente, han utilizado Reports para diseñar informes industriales complejos, con gráficos incrustados. El objetivo de la sección de este libro dedicada a Reports no es sólo enseñarle a crear informes básicos, sino enseñarle también a aprovechar al máximo las más potentes funcionalidades de Reports.

Las herramientas de diseño de informes se suelen clasificar en las tres siguientes categorías:

1. Herramientas fáciles de usar pero relativamente limitadas

2. Herramientas que ayudan a los usuarios experimentados a construir informes relativamente complejos

3. Herramientas de generación de informes de producción capaces de crear cualquier informe, aunque sólo pueda utilizarlas un desarrollador experimentado

Algo de lo que ha venido careciendo la industria es de lo que podríamos llamar una herramienta de generación de infonnes «válida para toda la organización», una que esté al alcance tanto de desarrolladores inexpertos y usuarios finales en el caso de los informes *ad hoc,* como de profesionales de los Sistemas de Información en el caso de los informes de producción complejos. La forma de obtener tal herramienta de generación de informes, válida para toda la organización, consiste en partir de una herramienta de generación de informes de producción dotada de todas las posibles funcionalidades y

superponerle un «asistente» de fácil uso, que permita al usuario crear informes sencillos. Oracle ha conseguido implementar esta idea con Reports, intentando proporcionamos lo mejor de ambos mundos. Claramente, los puntos fuertes de Reports son aquellos que penniten el diseño de informes complejos de producción. Sin embargo, con la nueva interfaz de desarrollo, un usuario avanzado puede emplear también Reports para diseñar un porcentaje razonable de sus informes, con la ventaja añadida de que un desarrollador experimentado puede partir de dichos informes sencillos y tenninar de perfilarlos con las partes más sofisticadas del producto. Ello representa un paso adelante significativo para la industria, ya que Reports facilita la creación de informes tanto a desarrolladores como a usuarios finales, utilizando la misma herramienta en toda la organización.

Aunque Report Wizard (asistente de informes) hace que Oracle Reports sea tan fácil de utilizar para la creación de infonnes normales como cualquiera de los demás productos populares que permiten obtener informes de usuario, Oracle ha elegido no posicionar el componente Reports de Oracle Developer como herramienta destinada al usuario final. Respecto al acceso de los usuarios finales a esta herramienta, Oracle sugiere que los desarrolladores creen infonnes parametrizados y los pongan a disposición de los usuarios.

Antes de la aparición de la versión 2.5 de Reports, Oracle ha tenido un historial tempestuoso con los desarrolladores de informes. Mucha gente opinaba que existían en el mercado otros generadores de informes más fáciles de utilizar o más potentes, o ambas cosas. Sin embargo, dicha percepción de baja calidad y falta de potencia en las herranúentas de desarrollo de Oracle carece de justificación desde la versión 2.5. El omponente Reports de Oracle Developer es tan potente y, a la vez, su uso. es tan sencillo que debería acallar hasta al último de los críticos.

La funcionalidad principal de Oracle Reports no ha cambiado significativamente desde la versión 2.5. La diferencia estriba en que con Reports los informes se pueden crear con mucha mayor rapidez y facilidad, y con menos frustración, que con cualquiera de las versiones anteriores de este producto. Gran parte del esfuerzo dedicado por Oracle a esta última versión se ha invertido en la mejora de su facilidad de uso para el desarrollo.

El equipo de desarrollo de Reports en Oracle era muy consciente de las dificultades que revestía el uso de la versión 2.5 y de las anteriores a ella. Muchos de los cambios introducidos en este producto reflejan la respuesta de Oracle ante la demanda hecha por los desarrolladores de un producto más sencillo de utilizar, más conveniente y más intuitivo. La interfaz de desarrollo de Oracle Reports ha sufrido una gran mejora, incluyendo numerosas nuevas prestaciones destinadas a reducir el tiempo requerido para aprender a utilizar sabiamente el producto.

Una mejora adicional introducida en Reports consiste en la extensión de su funcionalidad merced a la inclusión de un servidor de informes (Reports Server), que acelera el funcionamiento y añade capacidades de planificación y soporte para una arquitectura de tres niveles.

Para aquellos que utilicen una herramienta distinta para generar informes, este capítulo debería convencerles de que merece la pena dar a Reports una oportunidad. En él se hace hincapié en las características de la nueva versión que más afectan a la facilidad de uso y a la rapidez de desarrollo de los informes. Si se necesita una información más detallada, el Apéndice B contiene una descripción del asistente Report Wizard mejorado.

Arquitectura de *Report Builder*

El componente Reports de Oracle Developer dispone de dos interfaces relativamente aisladas entre sí, correspondientes, a su vez, a la forma en que se diseña un infon-ne. En la primera parte de la herramienta, Report Editor - Data Model (editor de informes modelo de datos), el desarrollador define la consulta o consultas necesarias para crear el informe, así como para incluir dicha información en uno o más grupos lógicos. Después, en Report Editor - Layout Model (editor de informes - modelo de diseño), el desarrollador especifica la manera en que dichos grupos aparecen dispuestos en el infonne. El producto dispone de otras dos interfaces: Parameter Form Builder (generador de pantallas de parámetros) y Live Previewer (previsualizador en vivo). La primera permite diseñar una interfaz de usuario sencilla con la que controlar los parámetros de usuario en tiempo de ejecución. Sin embargo, si se desea un control más complejo de

dichos parámetros, debería utilizarse Oracle Fonns. En Live Previewer, el desarrollador puede ver qué apariencia tendrá el informe cuando se ejecute, pudiendo realizar ciertos ajustes en dicha apariencia.

En Reports, el desarrollador suele disponer de dos ventanas abiertas en pantalla: Object Navigator (navegador de objetos) y Report Editor (editor de informes). El editor de informes es una ventana cuyo contenido puede seleccionarse entre las siguientes cuatro interfaces:

- Data Model (modelo de datos)
- Layout Model (modelo de diseño)
- Parameter Form (pantalla de parámetros)
- Live Previewer (previsualizador en vivo)

Los botones que permiten conmutar de una interfaz a la otra están disponibles en la barra de herramientas, de forma que se pueda conmutar con facilidad entre las diferentes áreas de la herramienta. Dadas las inevitables limitaciones de espacio en la pantalla, ésta es una útil característica que mejora la productividad del desarrollador, reduciendo el tiempo invertido en manipular múltiples ventanas en la pantalla.

El modelo de datos

Igual que Fon-ns, Reports separa los grupos lógicos de datos de la disposición física de los datos. Sin embargo, en Reports (a diferencia de lo que ocurre en Forms), la separación entre los modelos lógico y físico es completa. En Reports, el desarrollador crea grupos lógicos y columnas que conecta con marcos y objetos del diseño. En Forms, los grupos lógicos son bloques; en Reports, los grupos lógicos son *grupos. Los* grupos se basan en consultas. El desarrollador organiza los datos obtenidos mediante una consulta

distribuyéndolos en uno o más grupos. Los grupos dentro de una consulta son particiones de datos maestro-detalle, estrictas, jerárquicas. En general, tales particiones lógicas corresponden directamente a los grupos en los que se organiza un informe.

Por ejemplo, supongamos que se desea crear un informe maestro-detalle clásico, presentando los nombres de los empleados que trabajan en un departamento concreto. Se escribiría una única consulta para recuperar la información, como se muestra en las siguientes líneas:

```
SELECT all dept.deptno, dept.dname, emp.empno, emp.ename
FROM dept.emp
WHERE emp.deptno = dept.deptno
```

Si se desea construir un informe maestro-detalle tradicional, pondremos la información de departamento en un grupo padre y la información de empleados en el grupo hijo. Tales grupos suelen corresponder (aunque no es imprescindible) a tablas de la base de datos. Los aspectos específicos relacionados con el diseño de la consulta y con la creación de los grupos se analizarán en el Apéndice D.

Es importante comprender que, tras haber utilizado una consulta para crear grupos, la n-fisión de ésta ha terminado. Existen muy pocas ocasiones en las que el desarrollador haya de preocuparse por la consulta tras haber creado los grupos. El propósito de la consulta es crear los grupos que corresponderán a los datos que necesitemos incluir en el informe.

El modelo de diseño

Para distribuir los objetos en un informe se utilizan *marcos*. Los marcos en Reports son mucho más flexibles que el concepto de bandas empleado por otras herramientas de generación de informes. Cada marco repetitivo de Layout Model (que presenta filas de datos) ha de disponer de su correspondiente grupo. Sin embargo, puede haber grupos que

no estén asociados con ningún marco. Por ejemplo, a la hora de generar algunos informes complejos, puede que necesitemos grupos de detalle, aunque ninguno de los datos contenidos en ellos haya de aparecer en el informe final; sus datos se utilizan sólo para calcular información de resumen.

La Pantalla de parámetros

Los parámetros son variables del informe que el usuario puede alterar durante la ejecución, inmediatamente antes de ejecutar el informe. Se pueden emplear parámetros de sistema para especificar ciertos aspectos de la ejecución del informe, como puedan ser el formato de salida, el nombre de la impresora, el identificador de correo o el número de copias. El desarrollador puede también crear sus propios parámetros para cambiar valores en SQL o PL/SQL en tiempo de ejecución. Por ejemplo, un parámetro podría reemplazar tanto valores literales sencillos como expresiones completas en una consulta.

El previsualizador en vivo

Live Previewer permite ejecutar el informe y mostrar los datos tal y como aparecerían en el informe definitivo, proporcionando una forma fácil de modificar la apariencia del infortne. Live Previewer le permite también ver inmediatamente el efecto de los carnbios introducidos. Mientras está trabajando con Live Previewer, el desarrollador puede realizar las siguientes operaciones:

- alinear columnas
- modificar el tamaiño de las columnas
- desplazar columnas
- insertar números de página

- editar textos
- cambiar colores
- cambiar los tipos de letra
- definir máscaras de formato
- insertar un campo
- insertar un vínculo a un archivo HTML
- insertar objetos comunes

Prestaciones avanzadas e infrautilizadas

Oracle Reports incluye algunas potentes funcionalidades que los desarrolladores rara vez utilizan. Dos de ellas se mencionan brevemente aquí, como ejemplo de las posibilidades que el producto ofrece de cara al diseño rápido y eficiente de informes sofisticados: los parámetros léxicos y las funciones PUSQL predefinidas, que veremos más en detalle, en capítulos posteriores de esta sección.

Parámetros léxicos

Se trata de una de las funcionalidades quizás más potente y, a la vez, raramente utilizada de Reports. Un *parámetro léxico* es sencillamente una cadena de texto susceptible de ser incorporada al texto de una consulta. Ello permite obtener un control completo sobre la consulta en tiempo de ejecución. Esta característica puede utilizarse para crear un sistema de generación de informes flexible.

Para ello, el desarrollador crea una pantalla a mod de interfaz de usuario para la generación flexible de informes. Los usuarios eligen el informe que deseen ejecutar. Después, a través de una serie de listas emergentes, bloques multirregistro y casillas de verificación, los usuarios especifican el informe, los criterios de ordenación y los campos en los que desean ejecutar los saltos, así como una serie de criterios complejos de filtrado. Después, la pantalla crea los parámetros léxicos apropiados para modificar la consulta de informe básica que, a su vez, ejecuta el informe de acuerdo con los criterios especificados por el usuario. En un sistema de este tipo, se pueden diseñar infon-nes que admitan uno o más niveles de ruptura especificados por el usuario. Si no se utilizan rupturas, se puede suprimir su presentación mediante disparadores de formato. Éste es un procedimiento que se puede usar también para que los usuarios puedan especificar las columnas que desean ver reflejadas en el informe.

Un sistema tan flexible de generación de infonnes puede reducir grandemente la necesidad de una herramienta de consulta *ad hoc,* evitando que la organización haya de preocuparse porque los usuarios creen informes no normalizados, lo cual es frecuentemente el caso con las herramientas de consulta *ad hoc.* En estos sistemas, puede ofrecerse a los usuarios la posibilidad de guardar y recuperar sus configuraciones de informe, permitiéndoles obtener con rapidez informes normalizados, incluso aquellos que requieran configuraciones complejas.

Los parámetros léxicos son la característica más importante de Reports, afirmación que razonaremos a continuación. Durante el diseño de un sistema completo de generación de informes, en el análisis original se estimó que harían falta hasta 200 tipos de informe diferentes. El uso de los parámetros léxicos y de una interfaz de usuario flexible permitió satisfacer las necesidades de generación de informes con sólo 12 tipos diferentes de informe. Un beneficio añadido se hizo evidente cuando se cayó en la cuenta de que la estructura de la base de datos cambió considerablemente tres veces durante el curso del proyecto. En cada una de dichas veces, la modificación y comprobación de todo el sistema de informes se realizó en, aproximadamente, tres días. En la instalación de otro cliente, en la que también se había implementado este sistema flexible de

generación de informes, se daba la circunstancia de que las necesidades de los suarios no estaban bien definidas. Éstos hicieron muchas modificaciones, pidiendo gran cantidad de informes adicionales más allá de las especificaciones originales. Sin embargo, la naturaleza flexible del sistema de generación de informes permitió que, salvo en uno de los casos, todo lo que hubiera que hacer se redujera a enseñar a los usuarios cómo utilizar el sistema ya diseñado para obtener su nuevo informe. En el único caso en que esto no fue posible, hubo que añadir una columna a uno de los informes. Todos aquellos que diseñen sistemas de generación de informes de producción deberían utilizar parámetros léxicos y una interfaz flexible basada en Forms.

Funciones PLISOL predefinidas

Oracle proporciona una serie de funciones PUSQL destinadas a ofrecer funcionalidad adicional en Reports. Estas funciones se suministran en un paquete PUSQL denominado «SRW», acrónimo de «SQL Report Writer» (escritor de informes SQL), referencia histórica al nombre de la herramienta original de Oracle para la generación de informes; todos los usuarios de Reports deberían estar familiarizados con este paquete. Este paquete amplía las capacidades de Reports de forma muy parecida a la forma en que las órdenes PUSQL específicas de Forins amplían las capacidades de éste. He aquí algunos ejemplos:

- **SET-MAXROW.** Procedimiento empleado para cambiar el número máximo de filas de una consulta.

- **GET - PAGE-NUM.** Se utiliza para obtener el número de página actual del informe.

- **SET-FIELD-NUM.** Se puede sobreescribir el contenido de los catnpos numéricos obtenidos.

- **SET - FIELD - CHAR.** Se puede sobreescribir el contenido de los campos altanuméricos obtenidos.

- **SET - FIELD - DATE.** Se puede sobreescribir el contenido de los campos de fecha obtenidos.

Se puede encontrar una relación completa de las funciones de SRW tanto en la documentación de Reports como en la ayuda en línea.

Report Wizard

El asistente Report Wizard es la última implementación que ha hecho Oracle de un generador automático de informes. Parte de sus funcionalidades ya existían incluso en la versión 2.5 de Reports, pero la sofisticación y profundidad de Report Wizard van mucho más allá de las de la última versión. Sin embargo, Report Wizard no evita la necesidad de comprender cómo diseñar un modelo de datos o cómo manipular manualmente el modelo de diseño.

En la actualidad, con Report Wizard sólo se pueden crear informes relativamente sencillos. Cualquier modificación más allá de sus valores predeterminados requiere que los desarrolladores comprendan del todo cómo utilizar la herramienta. Incluso aunque Report Wizard se vaya haciendo más sofisticado, los desarrolladores no deben depender completamente de esta herramienta. Cuando se trabaja con generadores automáticos o con asistentes, no sólo es importante saber cómo utilizarlos, sino también cómo diseñar manualmente los informes.

Uno de los usos potenciales de Report Wizard consiste en emplearlo como herramienta de consulta *ad hoc* para usuarios y desarrolladores inexpertos. Con un aumento relativamente pequeño en sofisticación (es decir, en la siguiente versión), Report Wizard podría ser todo lo que tal tipo de usuario necesita. Una ventaja adicional sería que, una vez que un usuario ha intentado diseñar un informe, si éste requiere más sofisticación que la ofrecida por Report Wizard, un profesional de Sistemas de información puede aprovechar el trabajo realizado por el usuario y proceder sencillamente a modificar el informe generado. Sin embargo, como ya se ha mencionado, Oracle no considera Reports como una herran-úenta destinada a usuarios finales.

Report Wizard aparece de dos formas diferentes. La primera vez que se genera un informe, Report Wizard aparece automáticamente invocado por el sistema, guiando al usuario a través del proceso de creación de un informe. Si se accede manualmente a Report Wizard, después de haber creado un informe, seleccionando **Tools, Report Wizard,** se puede acceder a cada una de las partes de Report Wizard merced a una interfaz de fichas.

Modelo de datos (Data Model)

Utilizando exclusivamente Report Wizard, puede diseñar el modelo de datos sólo si el informe contiene una única consulta y una estructura simple de un único grupo, o si se trata de un informe simple de tipo maestro-detalle o matricial. También es posible basar el informe en un grupo de registros construido en Oracle Forms y transferido a Oracle Reports. Cualquier informe que contenga columnas de fórmula, diseños complejos o múltiples consultas, requerirá el uso del modelo de datos del Editor de informes. La sección siguiente proporciona una breve introducción al modelo de datos, que le permitirá diseñar la mayoría de los informes sencillos.

Creación de la consulta

Lo primero que se especifica en el modelo de datos es la consulta. Ésta puede ser tan compleja como se desee, incluyendo combinaciones, uniones y cláusulas «group by». Un desarrollador que desee crear consultas en Reports ha de ser un programador de SQL experimentado. Existen varias fonnas diferentes de introducir una consulta en el modelo de datos. Se pueden crear consultas utilizando la ficha Query (consulta) de Report Builder Wizard o directamente en el editor del modelo de datos, pulsando con el ratón en el icono SQL. Después de crear y compilar la consulta, Reports generará automáticamente un grupo. Un *grupo* es una colección de columnas. Un grupo de Reports corresponde, aproximadamente, a un bloque de Forms. Son estos grupos los que constituyen la base del informe, no la consulta en sí. Las consultas sirven para construir los grupos, utilizados a su vez para rellenar el infonne. La única función de las consultas en Reports es la de crear los grupos. Es importante tener en cuenta esta distinción entre las consultas y los grupos a la hora de proceder a construir el modelo de datos de un informe.

La ventaja de esta distinción reside en que se puede ajustar el resultado de un informe cambiando las consultas que generan los grupos, quizás pasando de dos o tres de ellas a sólo una, o pasando de utilizar una consulta compleja a emplear una más sencilla basada en una vista, sin tener que cambiar (salvo superficialmente) el resto del informe.

En el modelo de datos, es sencillo *vincular* varias consultas una con otra. Las consultas vinculadas constituyen una forma de incluir varias consultas en un n-úsmo informe físico. El vínculo establecido entre las consultas crea un conjunto de datos lógico, potencialmente complejo, en el que se basará el informe. Sin embargo, la acción de vincular varias consultas entre sí crea, en efecto, subconsultas correlacionadas. Por cada registro extraído de la consulta maestra se crea una instancia separada de la consulta de detalle.

Usualmente, se puede aumentar el rendin-iiento reduciendo el número de consultas, siendo la situación ideal aquella en la que sólo se precisa una consulta. Por supuesto, no siempre puede conseguirse tal situación.

Sin embargo, si revisamos el modelo de datos, debemos damos cuenta de que todo lo que estamos haciendo es recuperar información para el informe. Siempre que onsigamos que la misma infonnación llegue siempre a los mismos grupos, podremos manipular las consultas a voluntad para optimizar el rendimiento.

Generador de consultas (Query Builder)

Query Builder es una herramienta de escritura de código SQL del tipo seiíalar-y-pulsar, que puede utilizarse para construir casi todo tipo de consultas, excluyéndose tan sólo las más complejas. Con esta herramienta podemos seleccionar tablas con rapidez, pulsar con el ratón en sus columnas para incluirlas en la consulta y aiíadir condiciones WHERE. Una de las características más útiles de Query Builder reside en su capacidad para obtener del diccionario de datos información sobre integridad referencias (suponiendo que se han definido las restricciones apropiadas) y combinar tablas automáticamente. Se puede acceder a Query Builder tanto desde Report Wizard como desde Data Model. En Report Wizard, accedemos a Query Builder mediante la ficha Data; en Data Model, se accede a esta herramienta cuando se crea o edita una consulta.

Con la misma finalidad y rapidez, se pueden construir consultas complejas. Otra característica útil de Query Builder consiste en que se puede editar directamente el texto de la consulta en el modelo de datos. Luego, cuando se pulsa con el ratón en Query Builder, éste convierte la consulta a su sintaxis gráfica. Por descontado, esta funcionalidad tiene ciertas limitaciones. Query Builder no admite todas las consultas SQL. Por ejemplo, si se crea una consulta compleja que involucro una orden UNION ALL y luego se intenta trabajar con ella en Query Builder, se obtendrá un error, debido a que la herramienta no es capaz de importarla. Sin embargo, para la mayoría de las

consultas, Query Builder es una herramienta excelente y su uso permite un gran ahorro de tiempo.

Una de las mejores características de Query Builder es que proporciona a los desarrolladores una rápida representación visual de la consulta, facilitándoles enormemente la depuración de consultas complejas. Además, aquellos desarrolladores a los que asuste trabajar con el teclado hallarán sencillo y rápido crear consultas simplemente señalando y pulsando con el ratón.

Grupos

Cuando se seleccionan columnas en una consulta, el sistema las ubica automáticarnente en un mismo grupo. Este grupo de campos es con lo que trabaja Reports para crear la disposición del informe.

Los marcos repetitivos de Layout Model corresponden a grupos, no a consultas. Una única consulta puede abarcar, sin embargo, más de un grupo. El grupo que siempre existe es el de *datos principales;* pero es posible tener también uno o más grupos *de salto.* Si se crean dos grupos, uno de ellos será el de datos principales y el otro será el de salto. Dichos grupos pueden crearse mediante Report Wizard, o en el modelo de datos. Cada método se analizará por separado.

Creación de grupos de salto en Report Wizard

Para crear grupos en Report Wizard, en primer lugar hay que seleccionar uno de los estilos de Report Wizard que utiliza un grupo (Group Left [grupo a la izquierda], Group Above [grupo encima], o Matrix With Group [matriz con grupo]). Es a partir de esa selección cuando comienza a estar disponible la ficha Groups (grupos). Cada vez que se selecciona una columna para llevarla del conjunto Available Fields (campos disponibles) a Group Fields (campos del grupo), se crea un grupo de salto. Si hay que combinar las columnas en un único grupo de salto, hay que arrastrar las columnas deseadas hasta el mismo nivel, desapareciendo los niveles que queden vacíos.

Creación de grupos de salto en el modelo de datos

Si se desea crear un grupo de salto en el modelo de datos, tan sólo hay que pulsar la columna deseada y arrastrarla fuera del grupo principal de la consulta. Si se arrastra la columna por debajo del grupo de la consulta, ello da lugar a que el grupo de partida se convierta en grupo de salto para el nuevo grupo creado debajo de él. ¿Cuál es el significado de tener rupturas múltiples en una cadena? Para los grupos de salto que se acaban de describir, esto da lugar a una relación de tipo maestro-detalle entre los grupos superior e inferior. Se creará una fila en cada grupo por cada combinación única de todas las columnas del grupo. Por ejemplo, si se incluye toda la información departamental en un grupo, habrá una fila por cada uno de los departamentos. Automáticamente habrá una relación lógica de tipo maestro-detalle entre los dos grupos, con la información de DEPT (departamento) en el grupo superior y la información de EMPLOYEE (empleado) en el inferior. Se pueden tener tantos grupos de salto como sean precisos. Sin embargo, es habitual que tener más de tres o cuatro grupos de salto haga que el informe sea casi imposible de leer. En la práctica, casi nunca hay más de cuatro grupos en una consulta.

Columnas de tipo Summary (resumen) y Formula (fórmula)

Las columnas de resumen y de fórmula permiten presentar información que no forma parte de las consultas base. La información puede estar ya basada en información recuperada (como es siempre el caso de las columnas de resumen), puede utilizar información ya recuperada como parárnetros de invocación (como suele ser el caso de las columnas de fónnula), o, con mucha menor frecuencia, puede ser completamente independiente de cualquier otro dato del informe. Las columnas *de fónnula* son funciones PUSQL que devuelven un valor con la intención de presentarlo. Las columnas *de resumen* son casos especiales de las columnas de fórmula, que permiten la creación rápida de funciones de uso común, como totales de suma, niínimos, máximos, promedios y recuento de campos no nulos.

Columnas de resumen

Las columnas de resumen son agregaciones simples de una única columna del infonne. La columna que se está agregando debe hallarse en un grupo descendiente del grupo en el que se coloque la columna de resumen. Por ejemplo, en el caso de DEPT/EMP no es posible incluir en el grupo de empleados una columna de resumen que calcule el número de departamentos existentes.

Sin embargo, esta restricción no implica que no se puedan emplear columnas de resumen en fórmulas o que no se las pueda presentar a un nivel inferior (por ejemplo, con datos del grupo de los empleados). Si observamos los registros de los empleados, veremos que cada registro puede abarcar una o más páginas. Una de las líneas del informe de empleados muestra el número de empleados que hay en un departamento concreto, el presupuesto total de sueldos de ese departamento y el porcentaje de dicho presupuesto que representa el salario percibido por un cierto empleado. Las columnas de fórmula y de resumen calculan toda la información departamental que reside en el grupo DEPT, pero pueden ser presentadas también junto con la información de los empleados. Éste es otro ejemplo más de la separación entre los modelos lógico y de presentación, una de las características más potentes de Oracle Reports.

Se puede construir una columna de resumen de dos formas distintas:

E **Mediante la richa Totais (totales) de Report Wizard.** El desarrollador puede especificar el campo y la función de agregación que desea calcular. Cuando se sigue este camino, el campo se agregará a cada grupo por encima del nivel del campo, incluyendo el propio informe. Si no se desean todas esas columnas de resumen, siempre es posible borrarlas más tarde.

· **Mediante Report Editor - Data Model.** El desarrollador crea una columna de resumen pulsando en el botón Summary Column de la barra de herramientas de la izquierda y seleccionando después el grupo en el que desea incluir la columna. Si

se desea que se calcule la operación de agregación para todo el informe, basta con pulsar en la zona de color gris, fuera de los grupos.

Si se crean manualmente las columnas de resumen, es preciso especificar para ellas la función que se desea aplicar a los datos. La operación predeterminada en Reports es la suma. Luego hay que especificar, en la paleta de propiedades, cuál es la columna *fuente* sobre la que se desean realizar las operaciones. La columna fuente ha de hallarse en un grupo por debajo del nivel en el que se encuentre definida la columna de resumen. En el ejemplo DEPTIEMP, si ubicamos en el nivel de informe la columna de resumen, dispondrá de todos los campos para poder agregarlos. Ubicándola en el grupo DEPT, sólo podremos seleccionar los campos correspondientes a los empleados.

A continuación deberá especificarse la propiedad *Reset At* (punto de reinicio). En los resúmenes normales, *Reset At* debe definirse siempre en el grupo en el que se encuentre la columna resumen. Si se desea que la columna de totales contenga un total dinámico, hay que hacer que *Reset* At apunte a la página del informe. No es inusual disponer de varios campos de resumen que sólo difieren en la propiedad *Reset* At. En el mismo informe podemos mostrar un total departamental, un total dinánúco para todo el informe y totales de resumen al pie de cada página, todo ello con sólo especificar diferentes valores de la propiedad *Reset At.*

Columnas de fórmula

Las columnas de fórmula se utilizan para calcular y presentar infonnación cuya recuperación de la base de datos no es sencilla. Hasta cierto punto, se trata de una cuestión de estilo: los cálculos se pueden hacer en la consulta base o en columnas de fórmula. En general, las columnas de fórmula son menos eficientes. Sin embargo, su uso añade claridad al informe y su pequeño coste en términos de rendimiento suele merecer la pena.

Las columnas de fórmula que implican el procesamiento de columnas ya recuperadas de la base de datos trasladan el esfuerzo de procesamiento del servidor al cliente, lo cual, dependiendo de las velocidades y cargas relativas de las máquinas servidor y cliente, suele ser deseable. Por ejemplo, una fórmula que presente la suma de dos o más columnas suele ser mejor como fórmula procesada por el cliente. Sin embargo, cualquier fórmula que utilice un cursor implicará viajes de ¡da y vuelta adicionales, y quizás numerosos, al servidor de base de datos.

Con independencia de los rendimientos relativos de servidor y cliente, debemos evitar un exceso de tráfico de red cuando nos preocupe la velocidad en la generación de los informes.

Por tanto, en general, será difícil encontrar cursores en las columnas de fónnula. De forma similar, también hay que tener cuidado con los cursores implícitos. Nótese que, cada vez que se invoca SYSDATE, se está iniciando una consulta de base de datos. Sin embargo, si la base de datos es pequeña y el rendimiento no es un aspecto importante, el uso de columnas de fórmula con cursores puede dar lugar a que el informe sea más fácil de comprender para un desarrollador, facilitando así el mantenimiento.

Las columnas de fórmula sencillas contienen manipulaciones numéricas o de cadena sobre información ya recuperada de la base de datos, por lo que toda la actividad se circunscribe a la máquina cliente. Se puede utilizar este tipo de columnas de fórmula sin penalizaciones para el rendimiento. El desarrollador puede decidir ejecutarlas en la máquina cliente, en Reports Server (el servidor de informes) o allá donde resida la base de datos.

Otros lugares en los que puede ser útil el empleo de columnas de fórmula son aquellos valores obtenidos de la base de datos que el desarrollador desea presentar cual si de un único valor se tratase, tales como números de inventario, nombres de departamento o códigos de departamento que sean concatenaciones de otros campos. En vez de disponer en el modelo de diseño de varios objetos conectados mediante anclas, para aquellos campos que tengan indicadores prefijos o sufijos («+» o «-») es más sencillo crear una

columna de fórmula en el modelo de datos que sea una concatenación de todos los campos de visualización. De esta manera, el modelo de datos sólo necesita contener un objeto, que mostrará la columna de fórmula.

En Report Wizard no se pueden construir columnas de fórmula. Se puede emplear la opción Define Columns (definir columnas) para crear una columna de fónnula en Query Builder, pero dichas columnas sólo pueden utilizar las funciones de agregación de las instrucciones SQL. Esta acción también puede tener consecuencias sobre el rendirniento. El único can-iino para crear una columna de fórmula es la opción Report Editor - Data Model. Pulse en el icono Formula Column (columna de fórmula) y, a continuación, sobre el grupo al que haya de pertenecer la columna de fórmula. Para decidir a qué grupo debe pertenecer la colun-ma de fórmula, ha de considerarse la frecuencia con la que aparecerán las columnas de fórmula. Volviendo al ejemplo DEPT/EMPLOYEE, es preciso comprender qué es lo que representa cada grupo. Para el grupo DEPT, cada fila representa un departamento, mientras que cada fila del grupo EMPLOYEE representa a uno de los empleados. He aquí tres ejemplos:

1. Una columna de fórmula que concatene varios campos de nombre para presentarlos juntos: una fórmula tal tendría su ubicación en el grupo EMPLOYEE, ya que el nombre pertenece al empleado, como indica F-FULLNAME-TX.

2. Una fónnula más compleja, que muestre la desviación estándar del salario de cada departamento: el desarrollador podría estar tentado de ubicarla en el grupo EMPLOYEE, ya que se está manejando información de empleados. Sin embargo, dado que esta fórmula sólo se ha de calcular una vez para cada departamento, hay que ubicarla en el grupo DEPT, como indica F-SAL-STD-DEV-NR.

3. Una fórmula que presente la desviación estándar del salario de todos los empleados de una organización en la cabecera de un infonne: esta fórmula sólo ha de calcularse una vez, constituyendo lo que se conoce como fónnula en el nivel *de*

infonne, y el objeto correspondiente no se ubica en ningún grupo, sino fuera de ellos, en el espacio abierto del modelo de datos.

La ubicación de las diferentes fórmulas en estos sencillos ejemplos es fácil, pero la colocación de una fórmula en el grupo adecuado de un informe complejo puede ser una tarea de mayor dificultad. El desarrollador podrá emplazar correctamente las columnas de fórtnula si medita un poco acerca del proceso.

Modelo de diseño (Layout Model)

El modelo de diseño del editor de informes es la parte de la aplicación en la que se ejerce un control preciso sobre la disposición y apariencia del informe. Incluso para informes relativamente complejos, el uso de Layout Model no es imprescindible. El uso de Live Previewer permite realizar fácilmente casi todas las tareas necesarias para el diseño de informes básicos. Esta sección proporcionará un corto repaso sobre el uso de Live Previewer para realizar modificaciones sencillas en la disposición de los informes.

Live Previewer

El previsualizador Live Previewer, permite al desarrollador realizar cambios en el modelo de diseño y ver el efecto obtenido inmediatamente. Algunos de los cambios en el diseño del informne se pueden realizar incluso en el propio Live Previewer. Cuando los infortnes son sencillos, la combinación de las plantillas con Live Previewer significa que ni siquiera es preciso comprender las complejidades de los marcos de Layout Model. Live Previewer ayuda a los desartolladores inexpertos a crear informes sencillos. Sin embargo, si se han de diseñar informes de producción complejos, sigue siendo preciso comprender cómo funcionan los marcos del editor de diseño.

El uso de Live Previewer permite realizar con facilidad tareas sencillas de edición, entre las que se incluye el cambio de los tamaños de los tipos de letra, la selección de marcos con objeto de poder modificar su separación mutua en la paleta de

propiedades, el cambio del tamaño de los objetos y el control del formato del informe. Esta sección incluye descripciones breves sobre cómo realizar dichas tareas en un informe sencillo.

Cambio del tamaño y tipo de letra de un texto

Si se desea cambiar el tamaño o el tipo de letra de un texto en Live Previewer, hay que empezar por seleccionar en el informe la columna deseada. Pulse en el botón de flecha hacia abajo que hay junto al nombre del tipo de letra o del tamaño de la misma en la barra de herramientas. A continuación, seleccione los valores apropiados para el nuevo tamaño o tipo de letra en las listas desplegables correspondientes. El cambio será visible de inmediato en la pantalla del previsualizador.

Cambío de tamaño de los objetos

Si lo que se desea es cambiar el tamaño de los objetos, pulse sobre uno de ellos para seleccionarlo. Arrastre las asas de selección para ajustar el objeto al tamaño deseado. Nótese que no se puede aumentar el tamaño vertical de un grupo.

Control del formato

En la barra de herramientas de Live Previewer, pueden verse los botones $, %, Commas (separador de miles), Add decimal place (añadir decimales) y Remove decimal place (quitar decimales), que permiten controlar el formato de los valores monetarios, porcentajes, comas, posiciones decimales y cifras

significativas tras la coma, respectivamente. La siguiente es una lista de los botones y de sus funciones:

E**Botón de moneda ($):** Añade el signo del dólar a los números seleccionados.

- **Botón de porcentaje (%):** Añade el signo de porcentaje a los números seleccionados.

- **Botón de comas (,O):** Inserta comas tras cada tres cifras, contando desde el punto decimal.

- **Botón aiíadir decimales (+):** Añade una cifra decimal por cada pulsación del botón.

- **Botón quitar decimales (-):** Elimina una cifra decimal por cada pulsación del botón.

El uso de los marcos

Aunque Live Previewer es muy útil a la hora de realizar algunas modificaciones de diseño, la modificación de los propios marcos sólo se puede llevar a cabo con el modelo de diseño o con el navegador de objetos. Los *marcos* son entidades empleadas para encerrar objetos en su interior y protegerlos contra posibles sobreescrituras o desplazamientos por parte de otros objetos. Por ejemplo, se puede emplear un marco para encerrar todos los objetos que formen parte de un grupo, para rodear las cabeceras de columna o para rodear resúmenes. Cuando se utiliza el diseño predeterminado para un infon-ne, Report Builder crea marcos alrededor de los objetos del informe según lo considera preciso. El desarrollador puede crear manualmente marcos mediante Layout Model.

Existen dos variedades de marco:

- **Marcos de envoltura (no repetitivos):** Los marcos de envoltura agrupan objetos similares y se encargan de mantener sus posiciones relativas dentro del informe. Cuando se ejecuta el informe, los marcos pueden ser visibles o no. Si se desea hacer que un marco sea visible, se puede cambiar su tipo de línea y su color

mediante las opciones **Format Line Width,** para el tipo de línea, y el icono Color, situado en la parte inferior izquierda de la barra de herramientas, para el color. Los marcos no repetitivos deben rodear a varios objetos. Si un marco de este tipo encierra un solo objeto, es un marco superfluo y, a menudo, puede borrarse sin problemas, siempre que no tenga disparadores asociados.

- **Marcos repetitivos:** Este tipo de marco sirve para algo más que para agrupar simplemente unos objetos con otros. Los marcos repetitivos corresponden a los grupos del modelo de datos y, como se podría esperar, una de las propiedades de la paleta es el grupo con el que se asocia el marco. Cada marco repetitivo ha de estar asociado con un grupo del modelo de datos, pero podría haber grupos del modelo de datos carentes de asociación con marco repetitivo alguno. Dentro del marco repetitivo, sólo debe haber objetos pertenecientes a su grupo asociado o a un grupo ascendiente del mismo. Los objetos de un grupo descendiente pueden aparecer también dentro del marco repetitivo, pero sólo si residen enteramente dentro de un marco repetitivo asociado correspondiente a su propio grupo. Dado que un marco repetitivo repite los objetos que contiene una y otra vez, es preciso saber cómo se han de presentar los múltiples objetos a lo largo de la página, esto es, en columnas o en filas.

Object Navigator

La herran-úenta Object Navigator (navegador de objetos) de Reports, mostrada a continuación, es similar a la de Forins. Obviamente, algunas categorías serán diferentes, ya que los infonnes contienen objetos distintos de los que contienen las pantallas. Sin embargo, no es sólo una cuestión de que los objetos se agrupen de forma distinta, sino que el propio navegador de objetos funciona también de manera algo diferente de como lo hace en Forms. En concreto, la metodología de arrastrar y colocar sólo se puede emplear en zonas muy limitadas de Reports.

El objeto de nivel superior de la sección Reports del navegador de objetos es el nombre del informe. A diferencia de lo que ocurre en Forins, el nombre del infortne en el navegador de objetos ha de ser el mismo que el del archivo del informe. En concreto, la única foitna de cambiar el nombre de un informe consiste en guardarlo en otro archivo con el nuevo nombre. El análisis que sigue describe los nodos de alto nivel del navegador de objetos de Reports.

Live Previewer

Aunque éste es uno de los nodos del navegador de objetos, Live Previewer no es realmente un objeto. Si se pulsa dos veces sobre su botón, se abrirá el previsualizador, que ya hemos descrito anteriormente.

Data Model

Esta sección muestra la estructura lógica del informe. Se presenta aquí toda la información del modelo de datos del editor de informes.

Consejo: No debería utilizar la sección Data Model en el navegador de objetos *para otra cosa que no sea la visualización de la estructura de la consulta. La modificación de la estructura de la consulta* **se** *gestiona con mayor eficiencia en el* **propio** *modelo de datos.*

System Parameters. Esta sección contiene los valores de los parámetros del sistema, como el número de copias que se han de imprin-úr y la orientación de impresión del informe.

- **User Parameters.** Los parámetros de usuario se utilizan, principalmente, para almacenar y manipular infonnación en tiempo de ejecución, como parámetros de acoplamiento o léxicos, o para permitir a los usuarios la definición de parámetros de entrada, como por ejemplo rangos de fechas.

- **Queries.** Esta sección contiene las consultas y su texto de consulta. Los grupos asociados con las consultas no se detallan en la sección Queries, sino que tienen su propia sección (Groups).

- **Groups.** Esta sección almacena las agrupaciones de los objetos del informe, las columnas de la base de datos, las columnas de fórmula, las columnas de resumen y las columnas contenedores. Nótese que las columnas de la base de datos asociadas con el grupo se denonúnan «campos» en Report Wizard.

- **Formula Columns.** Esta sección contiene las columnas calculadas definidas en el nivel de informe. Como ya se ha indicado, si se asocia una columna de fórmula con un grupo, su definición se almacenará como subobjeto de dicho grupo.

  Summary Columns. Aquí es donde se pueden hallar las columnas de resumen definidas en el nivel de informe. Como ocurre con las columnas de fónnula, las columnas de resumen definidas en un grupo se almacenan como subobjetos del mismo.

- **Placeholder Columns.** Aquí podemos encontrar las columnas contenedores definidas en el nivel del informe.

- **Data links (vínculos de datos).** Cuando se combinan varias consultas en un informe, en esta sección se almacena la condición de combinación.

Layout Model

Aquí es donde podemos hallar toda la información y los objetos correspondientes al diseño físico del informe. Las cuatro secciones que contiene (Header [cabecera],

Trailer [piel, Body [cuerpo] y Margin [márgenes]) hacen referencia a cuatro lugares distintos en los que se puede especificar dónde se desea que se imprima la información.

- **Header.** La cabecera consiste en una o más páginas impresas antes que el informe en sí. El posible contenido de esta sección sería una página de título del informe o un gráfico que lo resumiera. Cuando se está diseñando un sistema flexible de generación de informes, la cabecera del informe es un lugar adecuado para presentar los criterios de selección utilizados en la creación del informe.

- **'ftailer.** El pie consiste en una o más páginas impresas tras el propio informe. Suele destinarse (si es que llega a utilizarse) para poco más que emitir una

página en blanco con el mensaje «Fin de informe», aunque también puede servir para presentar un resumen del informe o un gráfico.

- **Body.** Se trata de la zona en la que se colocan los objetos principales del informe.

- **Margin.** Report Layout sólo gobierna la parte de las páginas destinada a presentar los datos principales del informe. Los márgenes se pueden emplear para especificar encabezados y pies de página.

Nota: ***Los*** encabezados ***y pies de página*** no ***se especifican*** en ***las secciones***

Header **y** *Trailer, que* **se** *refieren a* **páginas** *completas al* **principio y** *al final* **del** *informe.*

Parameter Form

Se trata de una herramienta de entrada de datos por pantalla de funcionalidad limitada. Puede emerger automáticamente cuando se ejecuta el informe. En dicha pantalla, los usuarios pueden introducir valores o seleccionarlos de listas emergentes, para especificar valores para los parámetros del sistema y del usuario. Parameter Form puede ser útil cuando se ejecute repetidamente un informe durante el proceso de desarrollo y prueba.

Report Triggers (disparadores de informe)

Aquí es donde se ubica el código correspondiente a cualquiera de los cinco posibles disparadores de nivel de infonne asociados con un informe. Observe que los describimos en el orden en el que se disparan en el informe, en lugar de en el orden en el que aparecen en el menú de Report Builder.

- **BEFORE PARAMETER FORM (antes de la pantalla de parámetros).** Cualquier parámetro que se pase al informe ya le ha llegado cuando el informe coniienza su ejecución. No hay nada que se pueda disparar antes del primer paso de la ejecución del informe. Si se quiere tomar los parámetros que ha recibido el informe y manipularlos de forma que aparezcan de forma diferente en la pantalla de parámetros, ésta es la sección en la que se han de hacer las modificaciones. Por ejemplo, cuando se desea pasar a un informe el número de un departamento, pero se desea que el informe presente el nombre del departamento en vez del número, éste es el disparador que se ha de emplear.

- **AFTER PARAMETER FORM (después de la pantalla de parámetros) y BEFORE REPORT(antes del informe).** Estos dos disparadores se ejecutan uno tras el otro, sin que haya ocasión para que tenga lugar suceso alguno entre ellos. Sin embargo, el comportamiento de Reports cuando fallan los disparadores es muy distinto. Si el disparador AFTER PARAMETER FORM falla (devuelve «False»), el informe volverá a la pantalla de parámetros. Es útil situar aquí código destinado a comprobar si los valores de un parámetro son válidos. Aunque

el disparador BEFORE REPORT se activa antes de que se ejecute la consulta, si falla este disparador, no lo hará hasta que Reports intente presentar la primera página del informe. Esto significa que, incluso si algo sale mal en el disparador BEFORE REPORT (significando que puede que no se desee realizar la consulta después de todo), el informe se ejecutará en cualquier caso. Por tanto, el desarrollador no debería situar en el disparador BEFORE REPORT códi o que tuviera alguna oportunidad de abortar el informe antes de que comience a ejecutarse. Este código deberá ubicarse en el disparador AFTER PARAMETER FORM.

Piense que todo lo que se incluya en el disparador AFTER PARAMETER FORM se ejecutará antes que el disparador BEFORE REPORT. Se puede manipular la información de la pantalla de parámetros antes de que llegue al informe. Por ejemplo, esta estrategia se puede emplear para preparar los parámetros léxicos de una consulta. De hecho, dado que aquí se puede escribir cualquier código PL/SQL, se puede realizar cualquier clase de comprobación básica o sofisticado de validez sobre los parátnetros, antes de enviarlos a los informes.

- **BETWEEN PAGES (entre páginas).** Este disparador se ejecuta antes de cada página, exceptuando la primera. No se disparará tras la última página de un infonne. Si un infonne tiene sólo una página, no se llegará a disparar. Se puede emplear este disparador para enviar a la impresora caracteres de control específicos, con el fin de cambiar la orientación del papel o para imprimir a doble cara.

- **AFTER REPORT (después del informe).** Este disparador se ejecuta tras haber impreso el informe o, en caso de que el informe esté saliendo por pantalla, después de que se cierre el informe tras la visualización. Se puede emplear para actualizar una variable global en caso de que, por ejemplo, se esté devolviendo el número de páginas del informe. Puede emplearse también para borrar una tabla temporal utilizada para imprimir el informe.

Program Units (unidades de programa)

Se trata de la misma sección que aparece en Forms, en la que se pueden incluir paquetes, procedimientos y funciones de cliente a los que se hará referencia dentro del informe. Asimismo, se puede crear un paquete y almacenar variables de paquete a las que se pueda acceder en cualquier momento durante la ejecución del informe. Por ejemplo, si se necesita poder acceder a una variable global actualizada por disparadores de formato mientras se está ejecutando un informe, la mejor forma de implementar dichas variables globales es a través de una variable de paquete.

Attached Libraries (bibliotecas adjuntas)

De forma similar a Program Units, Attached Libraries (como en ForTns) puede adjuntar bibliotecas PUSQL externas, que pueden ser invocadas desde el informe.

Templates (plantillas)

Oracle Reports permite al desarrollador aplicar plantillas prefabricadas. En esta sección es donde se crean o modifican las plantillas existentes. Las plantillas de Reports pueden incluir no sólo objetos habituales en los informes, como logotipos, cabeceras de informe, números de página y fechas, sino que son también «inteligentes» en el sentido de que contienen las configuraciones predeterminadas de propiedades que gobiernan la generación del informe. En concreto, se especifica el tipo de letra que se va a emplear tanto para los datos como para las etiquetas. Se pueden especificar diferentes tipos de letra para los marcos de los datos principales y para los marcos de grupo. Oracle ofrece varias plantillas de atractivo diseño, así como la posibilidad para los desarrolladores, de crear y guardar sus propias plantillas.

Estas plantillas ayudarán a reducir enonnemente el tiempo requerido para terminar de perfilar el diseño de los informes. Hasta ahora, existía una cierta tendencia común a no alejarse de los diseños predetertninados, con el objeto de reducir al mínimo el tiempo de desarrollo. Las plantillas para informes permitirán a los desartolladores especificar y reutilizar fácilmente diseños relativamente complejos.

External SOL Libraries (bibliotecas externas SOL)

Si se va a utilizar la misma consulta en un cierto número de informes, es posible almacenaría en un archivo o en la base de datos. Dicha consulta se podrá invocar como texto de consulta en cualquier infonne. La creación y mantenimiento de estas consultas se realiza en esta sección.

PLISOL Libraries (bibliotecas PLISOL)

Ésta es la misma sección que podemos hallar en Forms. En ella se crean y mantienen las bibliotecas PUSQL externas.

Debug Actions (acciones de depuración) y Stack (pila)

Los nodos Debug Actions y Stack están relacionados con las operaciones de depuración. Consulte la documentación de Oracle Developer si desea obtener una descripción de estas características. Es posible depurar el código PUSQL contenido en el informe durante la ejecución de éste. Para ello, hay que abrir la ventana de PUSQL Interpreter (intérprete de PL/SQL) y establecer puntos de interrupción. Con ello, los nodos Debug Actions y Stack contendrán información necesaria para el proceso de depuración.

Built-in Packages (paquetes predefinidos)

Igual que sucede en Forms, esta zona muestra la sintaxis básica de todos los paquetes predefinidos disponibles. El uso de este nodo suele ser una forma sencilla de hallar la sintaxis de una orden en particular.

Database Objects (objetos de base de datos)

ste nodo aparece también en Forms. Permite al desarrollador acceder a tablas, columnas y otros objetos del nivel de la base de datos.

El modelo de datos de Reports

Cuando se diseña un informe con el componente Reports de Oracle Developer, su construcción conlleva dos pasos principales. En el primero de ellos, hay que definir las consultas y declarar la estructura lógica de la información. En el segundo, el desarrollador ha de decidir cuál será el formato bajo el cual aparecerá dicha información en el informe físico. Este capítulo procederá a analizar el primero de ambos pasos, para el que se utiliza la interfaz Data Model del editor de informes.

La interfaz Data Model de Reports es la herramienta con la que se especifica cuál es la información necesaria para crear el informe. Ello incluye la información de resumen y todos los campos calculados que se estimen precisos. El modelo de datos no es sólo el conjunto de información, sino también la estructura lógica que reviste dicha información, incluyendo su organización jerárquica. Por ejemplo, un informe maestro-detalle se representa en el modelo de datos mediante dos grupos de información vinculados.

En el modelo de datos se representa casi toda la información necesaria para crear un informe. Sin embargo, existen algunas excepciones, como son los campos específicos de nivel de sistema, como los números de página, los títulos y la fecha, que pueden

aparecer en los informes sin que hayan de representarse en el modelo de datos. En algunos infonnes complejos, es posible incluir infonnación calculada fuera del modelo de datos y presentada en la disposición física del informe. Sin embargo, con la excepción de los números de página y la fecha del día en curso, en el 99 por ciento de las ocasiones todo lo que aparece en el informe se basa en información contenida en el modelo de datos.

A diferencia de lo que sucede con el componente Forins de Oracle Developer, en Reports existe una clara y precisa separación entre el modelo lógico de datos y el diseño físico de la información. En Forms, los bloques son fundamentalmente agrupaciones lógicas. La disposición física se organiza mediante lienzos y ventanas, elementos que son simultáneamente estructuras lógicas y físicas. Sin embargo, en Reports, existe una distinción muy clara entre los objetos lógicos y los físicos. Es ésta una diferencia fundamental entre la forma de trabajar de Fonns y la de Reports. En Reports existen objetos de *visualización,* o físicos, y objetos *lógicos,* siendo preciso establecer un vínculo entre objetos de un tipo y de otro. En Forms, los bloques representan ambos papeles. Esta circunstancia añade algo más de complejidad a la construcción de los informes. Sin embargo, también permite que la organización de los informes sea más natural.

Preparación de la *consulta: ¡cuanto* más *sencilla, mejor!*

El núcleo de cualquier informe lo constituyen una o más consultas SQL. La mayoría de los informes se basarán en una única consulta y ése es, en realidad, el ideal. Además de su sencillez obvia, un informe que esté basado en una única consulta suele funcionar con el mayor rendin-úento posible. El asistente Report Wizard sólo es capaz de crear informes de consulta única. Sin embargo, de vez en cuando, la única manera de diseñar un infonne es utilizando múltiples consultas. Además, es posible disponer en un informe de varias consultas completamente independientes entre sí y asociarías con diseños también independientes. Esta técnica proporciona la capacidad de presentar

varios infonnes lógicos en el mismo informe físico, ya sea lado a lado, ya sea uno antes que el otro. Esta flexibilidad constituye una de las características más importantes de Reports.

Query Builder

Las consultas se pueden crear con el asistente Report Wizard, o con la interfaz Data Model del editor de informes. En cada una estas herranúentas, existen tres métodos para construir una consulta. Se puede introducir la consulta SQL a través del teclado, se puede copiar de un archivo o se puede emplear el generador de consultas Query Builder. Este generador de consultas es una herramienta a la que podemos acceder ya sea mediante el botón correspondiente de la ficha Data del asistente, ya sea eligiendo el botón SQL situado en la barra de herramientas ubicada en la parte izquierda de la pantalla Data Model. Cuando se está creando la versión inicial de un informe, la forma más fácil de hacerlo es emplear el asistente. Mentras la complejidad del informe sea reducida, se puede seguir utilizando el asistente para modificar el modelo de datos. Sin embargo, si el infonne se compfica más de la cuenta y el modelo de datos requerido deja de ser trivial, será preciso diseñar la consulta correspondiente directamente con la herramienta Data Model.

Si se desea utilizar eficazmente Oracle Reports, es preciso ser un experto en la preparación de consultas SQL. Gracias al uso de Query Builder para construir la consulta requerida, los desarrolladores poco experimentados, carentes de una gran experiencia con SQL, pueden utilizar fácilmente Oracle Reports para construir un informe que funcione. Sin embargo, ese informe puede tardar horas en producir los resultados esperados, cuando se podría haber conseguido que tardase sólo minutos habiendo meditado algo más el diseño del informe. A continuación se describen algunas de las técnicas comunes que se pueden emplear para escribir informes de mejor calidad y rendimiento.

Consultas múltiples

En el modelo de datos se puede disponer de más de una consulta. Las consultas pueden estar vinculadas o no. Las *consultas sin vincular* se utilizan para construir informes independientes adicionales incluidos en el mismo informe físico. Por ejemplo, un informe que resuma los ingresos en el lado izquierdo de la página podría compartir la página con otro informe que resumiera, a la derecha, los gastos correspondientes al mismo período de tiempo. Otro ejemplo podría consistir en presentar el presupuesto de una división seguido de su hoja de balance. En ambos casos se trata de consultas de informe completamente independientes presentadas en el mismo informe físico.

Las consultas vinculadas son un segundo método para incluir varias consultas en el mismo informe físico. El vínculo entre las consultas crea un único conjunto lógico de datos, potencialmente complejo, en el que se basará el informe. Cuando se vinculan dos o más consultas, el efecto obtenido es la construcción de una subconsulta correlacionada, ya que, por cada fila extraída en la consulta maestra, se generará una nueva instancia de la consulta descendiente, con un nuevo valor de la variable de acoplamiento. En realidad, la situación es mucho peor para el rendin-úento que la construcción de una subconsulta correlacionada, ya que, por cada nueva instancia de la subconsulta, se está creando y abriendo un cursor adicional, que se pasa a la base de datos a través de la red. Las ¡aplicaciones de este efecto, de cara al rendimiento, son profundas. Por supuesto, esta situación no afecta demasiado al rendimiento del sistema cuando las tablas manejadas son pequeñas.

Es recomendable, por tanto, intentar limitar el número de consultas. Sin embargo, en algunos casos, el uso de múltiples consultas es inevitable. Por ejemplo, cuando se combinan dos o más informes en un único informe, es preciso que haya una consulta para el primer informe, otra para el segundo, etc., las cuales se siguen unas a otras. En este caso, el informe está recibiendo información de dos consultas distintas e independientes. Hay que tener cuidado para evitar encontramos con demasiadas consultas independientes en el mismo informe, particularmente si se intuye que éste va a abarcar un gran número de páginas. Sin embargo, si el informe se ajusta a una sola página, ésta es una situación aceptable. En una experiencia que tuvieron los autores de este libro, había un informe

que utilizaba consultas múltiples: se trataba de un informe de una página con cuatro consultas de resumen separadas. Se puede diseñar este tipo de informe de fonna que su ejecución sólo tarde unos pocos minutos, sea estable y funcione correctamente.

Precaución: Cuando se espere que un informe sea largo y complejo, debe evitarse el uso de consultas múltiples independientes.

En otra situación, se trataba de preparar una agenda telefónica interna para una gran organización gubernamental. El modelo de datos incluía seis consultas consecutivas e independientes; este informe incluía también objetos OLE tomados de Microsoft Word. El informe resultante abarcaba cerca de 100 páginas. En este caso, la creación y gestión del informe era muy difícil y originaba problemas. Hubo que dividir el informe en otros seis, con ejecuciones separadas, utilizando una interfaz de acceso basada en Oracle Forms.

Si se tienen consultas independientes, es preferible utilizar un número reducido de consultas y sólo combinarlas en un único informe cuando éste vaya a ser sencillo y pequeño. Cuando el informe sea complejo y necesite tres consultas independientes o más, es más seguro gestionar las consultas múltiples en informes separados.

Uno de los errores que suelen cometer los desarrolladores de informes poco experimentados es la creación de modelos de datos extremadamente complejos, con muchas consultas vinculadas entre sí. Una estrategia tal puede arruinar el rendimiento de un informe. La clave para que el rendimiento de un informe sea bueno reside en reducir al mínimo el número de cursores involucrados en el informe. Excepto en los casos más inusuales, es preferible crear el informe con una única consulta, que acceda a la base de datos una vez y devuelva la infonnación adecuada. Ello puede significar el uso de combinaciones externas (outerjoin) o de funciones incrustadas (analizadas más adelante en este capítulo) en la consulta. Si no se es cuidadoso al construir un informe, su rendimiento puede verse afectado muy negativamente. Oracle Reports ofrece un gran control sobre la forma en que se recupera la información de la base de datos. Sin

embargo, tal control significa que el desarrollador tiene ahora la posibilidad de recuperar la información por caminos profundamente ineficientes, si la consulta no está correctamente escrita o si el,diseño de la base de datos no está bien optimizado. Es importante comprender las sutilezas de Oracle Reports si se quieren crear informes eficientes.

Casi siempre se pueden construir informes sencillos maestro-detalle a partir de una única consulta. Utilizaremos el ejemplo sencillo de un informe maestro-detalle basado en las tablas DEPTIEMP.

Se pueden emplear dos métodos para construir este informe:

- Empleando dos consultas (la primera una sencilla instrucción «SELECT FROM dept» y la segunda otra sencilla instrucción «SELECT FROM emp») y luego vincularlas.

- Utilizando una única consulta que devuelva información tanto de DEPT como de EMP y combinando ambas tablas.

La construcción del informe con dos consultas en lugar de con una hará que el tiempo necesario para que se ejecute sea más del doble. Por cada fila de departamento que se recupere, se crea una consulta completamente nueva para la tabla de empleados, que se envía a la base de datos para su procesamiento.

A veces puede ser necesario emplear múltiples consultas para generar correctamente los grupos. La mayor probabilidad de que ocurra algo así se da cuando se requiere el uso de la cláusula GROUP BY para obtener el grupo de nivel superior pero se necesitan datos sin agrupar para el grupo de detalle. Este tipo de situación se da con una base de datos que no esté completamente normalizada, aunque también se pueden llegar a necesitar consultas múltiples con bases de datos normalizadas correctamente. Continuando con el ejemplo DEPT/EMP, si se desea disponer de un informe de consulta

única ordenado por el número de empleados de cada departamento, será necesario crear una función PL/SQL que proporcione el número de empleados de cada departamento. Dicha función se almacena en la base de datos. Luego, cuando se escriba la consulta DEPT, se podrá seleccionar la función como una de las columnas de la consulta, lo que hace que la columna de resumen sea un objeto de la consulta base.

Otra forma de manejar este informe consiste en emplear dos consultas separadas. La primera proporcionará la información del departamento, y se escribiría como sigue:

```
SELECT      deptno,dname,
COUNT(*) no-of-emps
FROM dept, emp
WHERE dept.dept.no = emp.deptno
GROUP BY  deptno, dname
```

La segunda consulta se definiría del modo siguiente:

```
SELECT      empno, ename, deptno
FROM        emp
```

El paso final consistiría en vincular las consultas de DEPT y EMR Esto nos proporciona el modelo de datos apropiado, con los departamentos ordenados por el número de empleados existentes en cada uno de ellos. El desarrollador podría pensar que debería poder servirse sólo de una columna de resumen que se limitara a contar el número de empleados. Sin embargo, no se puede realizar una ordenación basada en columnas de resumen, ya que éstas se calculan en el lado del cliente, una vez que se ha devuelto la fila.

Un informe puede tener tantas consultas como se necesiten. Es posible tener una consulta maestra con varias consultas descendientes. Una consulta descendiente puede, a su vez, ser consulta maestra para otra consulta descendiente.

No existe ningún método para crear una consulta vinculada en el asistente Report Wizard. Sólo puede hacerse manualmente en la interfaz Data Model del editor de informes. Para vincular dos consultas, existen dos opciones:

Pulsar en el botón Data Link (vínculo de datos). A continuación, pulse en una de las cabeceras de consulta (la caja oval encima de la consulta) y arrástrela hasta la otra. Reports hará una suposición sobre la condición de combinación apropiada, consultando las relaciones de clave externa entre las tablas de la base de datos. Si las consultas son sencillas, de una sola tabla, la suposición de Reports será correcta. Sin embargo, si la consulta es compleja, lo más frecuente es que tal suposición sea incorrecta. En cualquier caso, la combinación tarda un cierto tiempo en completarse, dependiendo del entorno. Se deben verificar siempre las combinaciones automáticas, revisando la paleta de propiedades de la flecha de la combinación, para asegurarse de que ésta se ha creado correctamente.

En lugar de pulsar y arrastrar el ratón desde una cabecera de consulta a otra, se puede pulsar en la columna de la consulta *ascendiente* y arrastrar hasta la columna de la consulta *descendiente* con la que se desea establecer la combinación. Esta operación no requiere búsquedas en la base de datos y se completa inmediatamente.

Cuando se vinculan consultas, Oracle Reports utiliza de manera predeterminada *una equi-combinación.* Sin embargo, también es posible crear no-equicombinaciones en aquellas raras ocasiones en las que podrían ser útiles. Los autores de este libro no han encontrado nunca una razón válida para justificar el uso de una no-equicombinación en un entomo de producción, habiéndoles visto tan sólo en ejercicios académicos.

Uso de vistas

Debido a que la información no se actualiza, sino que sólo se visualiza, en el modelo de datos se pueden utilizar vistas muy complejas. La capacidad del lenguaje

SQL de Oracle para incrustar funciones en las *vistas* es una funcionalidad realmente potente, aunque infrautilizada con frecuencia.

En general, se deberían emplear vistas en la mayoría de los informes. Concretamente, en sistemas grandes con muchos informes, un cierto número de vistas se encarga de centralizar gran parte de la lógica y del código del negocio, contribuyendo a permitir una gran reducción del coste y del mantenimiento globales del sistema. Las vistas deberían incorporar incluso aspectos relativamente sencillos. Por ejemplo, se puede añadir una columna con el nombre del departamento a la vista Employee (la de los empleados).

La creación de vistas se debe contemplar como un paso crucial en el diseño de la función de generación de informes de una aplicación. En general, las vistas no están concebidas para atender a un único informe. La idea general consiste en tomar los diseños de todos los informes y meditar sobre cuáles serían las vistas apropiadas para poder atender a todos los informes del sistema. Es aconsejable identificar los objetos y funciones comunes a varios infonnes, a partir de cuya información se podrán construir vistas capaces de proporcionar dicha funcionalidad. Puede que una estrategia de diseño aconsejable sea la de dejar el diseño y construcción de las vistas en manos de desarrolladores con mayor experiencia, lo que permitiría a los desarrolladores menos avezados partir de dichas vistas para construir los informes necesarios.

El diseño de todos los informes basándose en vistas también los aísla, hasta cierto punto, de las tablas físicas. Si se cambia el modelo de datos, todo lo que hay que hacer es modificar la vista, de manera que los informes puedan volver a funcionar.

El empleo de las vistas requiere tomar precauciones de cara al rendin-úento, ya que éste puede verse, a veces, seriamente afectado si se combinan vistas entre sí o vistas con otras tablas. En un proyecto en el que participó uno de los autores del libro, había una vista compleja con varias uniones. La recuperación de la información era muy rápida, hasta que se combinaba con cualquier otra tabla, en cuyo momento era imposible evitar una gran cantidad de exploraciones completas de tabla, comprometiendo el

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

rendimiento. El problema se resolvió volviendo a escribir el informe usando columnas de fórmula que accedían a la vista, con lo que el rendimiento volvió a ser adecuado. El desarrollador debe asegurarse de que las consultas a una vista son eficientes antes de utilizarla en Reports. De esta forma, se asegurará de que no aparezcan problemas de rendimiento debidos a tales vistas y consultas.

Asignación de alias a los nombres de columna

Una de las características de Reports menos convenientes es que todos los nombres de columna son globales. Recuerde que, en Forms, si se desea hacer referencia específicamente a un elemento, debe identificarse de forma unívoca mediante *BLOQUE.NOMBRE-ITEM.* Ello permite que una pantalla compleja gestione fácilmente varios cientos de elementos. En Reports, podría esperarse que se pudiera hacer referencia a las columnas mediante un nombre *GRUPO. COLUMNA.* Por desgracia, en Reports, los nombres de todas las columnas son globales. El efecto de que dichos nombres de columna sean globales consiste en que, por ejemplo, al seleccionar la columna DEPTNO tanto en la consulta DEPT como en la EMP, el nombre del campo DEPTNO recibirá automáticamente el alias DEPTNOI en la segunda consulta. Peor aún: con el paso del tiempo, si se modifican y regeneran las consultas, podemos encontramos con que los alias cambien, haciendo estragos en el código PL/SQL ya escrito. Como consecuencia de todo ello, la mejor estrategia consiste en impedir que Reports asigne arbitrariamente nombres de columna, lo que se consigue asignando dichos alias en la consulta de forma explícita.

Precaución: Merece la pena insistir en que, si se modifica el texto de la consulta de forma que haya más de una columna con el mismo nombre en el informe ello puede invalidar las instrucciones PLISOL que hagan referencia a las columnas por su nombre. Es preferible utilizar siempre alias para las columnas incluidas en las consultas creadas por uno mismo.

Selección de funciones PLISOL dentro de las consultas

Una de las características de Reports (y de la implementación general de SQL por parte de Oracle), que suele pasarse por alto, es la posibilidad de seleccionar una función para que fonne parte de una consulta. Por ejemplo, se puede disponer de una función que devuelva el valor total de un pedido o el número de empleados de un departamento.

Siempre que se disponga de una función que requiera acceder a la base de datos, tal función debe almacenarse en la base de datos. El objetivo es minimizar el tráfico de red. Los cálculos realizados en el cliente son rápidos. Los realizados en el servidor son muy rápidos. Sin embargo, los recorridos de ¡da y vuelta por la red son relativamente lentos. Si se implementa una función basada en una consulta de base de datos que utilice una columna de fórmula, el informe siempre se ejecutará con mayor lentitud que si la fórmula estuviera almacenada en la base de datos.

Si se desean almacenar funciones en la base de datos y acceder a ellas, se han de seguir estos pasos:

1. Ubicar la función en un paquete de la base de datos.

2. En la especificación del paquete, hay que emplear una sintaxis especial con la orden PRAGMA RESTRICT - REFERENCES, que indicará a la base de datos que la función «no escribe información de estado en la base de datos» (WINDS, writes no database states). Sin la inclusión de la orden PRAGMA, Oracle impedirá el uso de la función en cualquier instrucción o vista SQL. El siguiente ejemplo de código muestra la sintaxis correcta:

```
PACKAGE mi-paquete IS
    FUNCTION f-dname(
        ppno IN NUMBER)
        RETURN VARCKAR2;
    PRAGMA RESTRICT-REFERENCES(f-dname, WNDS);
```

Nota: ***Si*** se selecciona en Reports la lista de selección (SELECT),de una consulta *que incluya una función, ésta* tiene que *estar almacenada en la base de datos. Las consultas de Reports no podrán localizar la función sí se la almacena localmente en el informe o en una biblioteca asociada.*

El uso de funciones en las consultas no es lo mismo que utilizarlas como columnas de fórmula en el modelo de datos. Situar las funciones en las consultas SQL tiene ventajas considerables. No existe ningún caso en el que el uso de una columna de fórmula que se base en una consulta de base de datos tenga otras ventajas dignas de mención que la de facilitar la tarea al desarrollador y la legibilidad del informe. Por el contrario, dicho empleo sí tiene ciertas desventajas considerables; concretamente, no es posible efectuar ordenaciones basándose en el contenido de una columna de fórmula, mientras que sí es posible realizarlas basándose en columnas de función contenidas en una consulta SQL.

Es cierto que, si la función no requiere viajes adicionales de ¡da y vuelta a la base de datos, se están transfiriendo algunos ciclos de CPU a dondequiera que se esté ejecutando el informe (cliente o servidor de informes). En el caso poco frecuente de que el servidor de base de datos esté más cargado de trabajo que el servidor de informes, puede que haya alguna justificación para hacer que las funciones sean ejecutadas por el informe. Sin embargo, si colocamos las funciones en el servidor, ya sea en paquetes, ya sea en vistas, conseguiremos poder compartirlas con mucha mayor facilidad con otros informes y pantallas.

Consejo: El almacenamiento de funciones en el lado del servidor puede tener *un profundo ímpacto sobre el rendimiento del informe. En un informe sencillo que podría tardar varias horas en ejecutarse,* **si** *se transfieren al servidor las funciones que hagan un uso intensivo de la base de datos, con frecuencia se con~ sigue un gran aumento en el rendimiento del informe.*

Combinaciones externas

Una de las razones por las que con tanta frecuencia se utilizan y vinculan múltiples consultas reside en que, si se combinan dos tablas mediante una equi-combinación sencilla, puede que perdamos los datos que se han de imprimir en el infortne. Por ejemplo, en un informe normal DEPT/EMP, si se combinan DEPT y EMP, el informe no mostrará aquellos departamentos que carezcan de empleados, mientras que un informe con las dos consultas vinculadas mostrará todos los departamentos. Sin embargo, si se utiliza la combinación externa apropiada, se podrán presentar también los departamentos que carezcan de empleados.

Por supuesto, las combinaciones externas tardan más tiempo en ejecutarse que las consultas sin combinaciones externas. Pero, normalmente, un informe en el que haya una combinación externa funcionará con mucha mayor rapidez que el mismo informe utilizando múltiples consultas vinculadas.

Tablas de informe temporales

Una técnica que se emplea con mucha frecuencia consiste en crear una tabla temporal en la que se almacenan los datos de una parte importante del informe. Para dar forinato a la información contenida en dicha tabla y presentarla se puede diseiíar un informe relativamente sencillo. Cada vez que se ejecuta el informe, se vuelve a crear la tabla temporal y vuelve a llenarse de información. Casi nunca debe emplearse una estrategia como la descrita.

Hay pocas razones que justifiquen el uso de dicha estrategia. La extracción y almacenamiento de toda esa información en una tabla de Oracle es un proceso extremadamente lento y costoso. Es dificil hallar un infonne tan complejo e intrincado

que no pueda resolverse sin el uso de una tabla temporal. Muchos de los informes existentes en aplicaciones reales utilizan tablas temporales innecesariamente.

Un ejemplo en el que una tabla temporal sirvió para mejorar significativamente el rendin-fiento de un informe era un voluminoso informe destinado a la impresión de etiquetas para envíos por correo. En este informe había que recuperar varios conjuntos de etiquetas, cumpliendo cada uno un complejo criterio de búsqueda. El sistema contenía varios cientos de miles de nombres. Los usuarios deseaban especificar subconjuntos aleatorios de esa larga lista. La solución aplicada utilizaba procedimientos almacenados en el servidor que aceptaban el criterio de consulta especificado en una interfaz de acceso basada en Forms. Luego, empleando el paquete incorporado DBMS-SQL para ejecutar SQL dinámico en un procedimiento PUSQL, se generaba el subconjunto aleatorio de nombres y se llenaba una tabla temporal con él. Este ejemplo ilustra lo complejo que ha de ser un informe para que se justifique el empleo de una tabla temporal. Si, durante el diseño de un infon-ne, se le ocurre utilizar una tabla temporal, vuelva a pensárselo. Si puede evitar el uso de una tabla temporal, probablemente su informe funcionará con mucha mayor rapidez.

Cuando se utilizan tablas de resumen y de agregación, a veces las tablas temporales son necesarias para hacer que el informe se ejecute con un rendimiento aceptable. Por ejemplo, en una tabla de resumen, en la que cada fila de la tabla represente las ventas mensuales hechas a un cierto cliente, una tabla temporal mejora considerablemente el rendimiento de aquellos informes en los que se intenten mostrar las ventas totales hechas a los clientes durante un cierto período de tiempo. Sin embargo, esta tabla de resumen sólo se construye una vez al mes, con independencia de la ejecuci del informe. Cualquier informe podría consultar dicha tabla.

Consejo: Se puede limitar el número de filas recuperadas por cualquier consulta *mediante la propiedad* Maximum Rows To Fetch *(máximo número de filas que extraer). Esta propiedad es útil cuando se prueban informes que obtienen información de tablas muy grandes. También puede ser útil para aquellos informes que, por diseño, sólo*

recuperen un número específico de registros. Por ejemplo, un informe que devuelva la información correspondiente a **los** *primeros* **25** *clientes (ordenados según el volumen de las ventas) sólo requeriría ordenar la tabla según el volumen de ventas* **y** *limitar el número máximo de filas a* **25.** *Esta propiedad puede cambiarse también en tiempo de ejecución, mediante el procedi*miento SRW.SET-MAXROW.

Campos sin consulta

A veces es preciso mostrar en un informe datos que no están almacenados explícitamente en la base de datos. Podríamos citar los siguientes ejemplos:

- **Información agregada.** Por ejemplo, el número de personas que pertenecen a un departamento o el volumen total de ventas para un cierto período de tiempo.

- **Información calculada.** Por ejemplo, los ingresos totales percibidos por un empleado, que es la suma de su salario y sus comisiones.

- **Una función de cadena.** Por ejemplo, el nombre concatenado con el apellido.

Reports proporciona la posibilidad de generar columnas de fórmula complejas. Esta posibilidad conlleva una gran flexibilidad para el desarrollador; sin embargo, si se utiliza mal, los objetos resultantes pueden afectar negativamente al rendimiento del informe. Reports proporciona tres tipos diferentes de columnas sin consulta: las columnas de resumen, las columnas de fórmula y las columnas contenedores.

Columnas de fórmula

Las columnas de fórmula son columnas definidas por el usuario que se basan en funciones PL/SQL. Son muy útiles, aunque hay que tomar precauciones cuando se empleen cursores en ellas. Cada vez que se ejecuta una fórmula (lo que implica una vez por cada fila de ese tipo en el informe), se ejecuta una nueva consulta a la base de datos.

Una alternativa que casi siempre mejora el rendimiento consiste en almacenar la fórmula en la base de datos como una función, haciendo que fon-ne parte de la consulta base y reduciendo, por tanto, el tráfico de red.

En el generador de consultas del asistente Report Wizard se encuentra el icono etiquetado Define Columns (definir columnas). Su misión es aplicar funciones a las columnas de la consulta base que se envía al servidor. Por el contrario, las columnas de fórmula se calculan en el cliente. Cuando se utiliza la función Define Columns, el desarrollador está limitado al tipo de columnas que se pueden definir dentro de la lista SELECT de una instrucción SQL. Por el contrario, con las columnas de fórmula se pueden escribir funciones PL/SQL complejas. Por supuesto, siempre se tiene la opción de almacenar en el servidor las funciones PL/SQL y acceder a ellas a través de él, con lo que todos los cálculos realizados en el cliente con una columna de fórmula se pueden también llevar a cabo en el servidor, utilizando una colun-ma definida que invoque una función almacenada. Sin embargo, puede que haya que considerar ciertas posibles ¡aplicaciones de rendimiento y de recursos. En términos generales, el rendimiento se puede mejorar considerablemente ubicando con cuidado el código, concretamente en el lado del cliente (nivel intermedio) o en el servidor de base de datos. La mejora más drástica se podrá apreciar en las columnas de fórmula que utilicen cursores, que pueden estar extrayendo muchas filas de la base de datos para determinar un valor calculado. Tal código se debe almacenar y ejecutar siempre en el servidor. En un cierto ejemplo, un informe que tardaba 12 minutos en ejecutarse utilizando una columna de fórmula que contenía un cursor incrustado pasó a tardar sólo 40 segundos tras pasar el código al servidor.

Columnas de fórmula como indicadores

Otro truco que se puede utilizar con las columnas de fórmula es usarlas como indicadores. Por ejemplo, para que un informe departamental presente el número de hombres y mujeres que trabaja en cada departamento, se incluirían dos columnas de fórmula en el grupo de empleados, una para los hombres y la otra para las mujeres. El valor de la columna correspondiente a los hombres sería 1 si el empleado fuera un hombre y 0 en caso contrario. De igual manera, el valor de la columna correspondiente a las mujeres sería 1 si el empleado fuera mujer y 0 en caso contrario. A continuación, en el grupo maestro se puede incluir un sencillo campo de resumen que cuente el número de hombres y el de mujeres.

Columnas contenedores

Las columnas contenedores actúan como variables globales en el modelo de datos. Se pueden utilizar para seguir la evolución de ciertos valores que se quieran imprimir en el informe. Ese mismo papel puede ser asuniido por las columnas de fórmula en conjunción con variables de paquete PUSQL. La elección entre columnas contenedores o de fórmula en este contexto es, fundamentalmente, una cuestión de estilo de desarrollo.

Un buen uso de este tipo de columna consiste en utilizarlas como contenedores cuando se están creando diseños predetem-finados. Defina campos contenedores sin etiqueta para ayudar a colocar correctamente los campos en un informe.

Otro buen uso de las columnas contenedores se da cuando se rellena un objeto de la pantalla mediante una función predefinida con la orden SRW.SET. La columna contenedora genera un campo asociado en el modelo de diseño. La orden SRW.SET apropiada se invoca en el disparador de fonnato del campo del modelo de diseño, lo que permite rellenar la columna contenedora. Esta es una caractedstica potente, pero que no suele ser necesaria. Un ejemplo de su uso sería un informe que comience en un número de página específico. El número de la primera página llega al informe como parámetro y

el desarrollador emplea una orden SRW.SET para añadir dicho valor al número lógico de página para su presentación. Esta técnica se puede emplear también en alguna función de acumulación compleja en la que, n-iientras el informe está ejecutándose, se actualiza periódicamente la información de una variable de paquete y hay que presentar su contenido. Las líneas que siguen contienen el código de una posible solución de este tipo:

```
FUNCTION upd-page-num

    RETURN BOOLEAN

is

    v teinpnum NUMBER;

BEGIN

    srw.get-page-num(v--tempnum);

    srw.set-field-DUM(O, v-tempnum + ( TO-NUMBER (:p-start-page) l»;
    globals.page-numbers := v-tempnum + TO-NUMBER (:p-start-page) RETURN
    (TRUE);

EM;
```

Objetos del nivel de informe

Cuando se crean informes, a menudo se necesita emplear objetos que, como las columnas de resumen, no dependan de grupos ni, incluso, de consultas. Tales objetos son

los objetos *del nivel de infortne.* Se puede almacenar este tipo de objetos en diferentes lugares, aunque el mejor lugar para hacerlo es en el modelo de datos.

En el modelo de datos, los objetos del nivel de informe son columnas que no pertenecen a consulta alguna. Sin embargo, su uso no está exento de limitaciones. Los objetos del modelo de datos no se pueden manipular mientras el informe está ejecutándose. Si se desea manipular directamente las variables mientras el informe está en ejecución, hay que definir los objetos del nivel de informe como variables de paquete. Las variables de paquete se definen en un paquete creado bajo el nodo Program Units del navegador de objetos. No hay impedimento alguno para obtener y actualizar el contenido de las variables de paquete durante la ejecución del informe. Por ejemplo, se podrían utilizar variables de paquete para la realización de cálculos complejos relacionados con los números de página u otros cálculos de acumulación que la funcionalidad normal de Reports no gestiona con facilidad.

Algunos objetos comunes susceptibles de convertirse en objetos del nivel de informe son los objetos de resumen de totalización y los títulos o pies de informe generados por programa. El sistema para crear objetos del nivel de infonne en el modelo de datos consiste en crear una columna de fórmula, de resumen o contenedora fuera de cualquiera de los grupos.

También se pueden crear objetos del nivel de informe como parámetros de usuario, pero esto sólo es preciso si es necesario manipularlos desde fuera del infoitne. Esta manipulación exterior se puede especificar de dos formas distintas:

- A la hora de ejecutar el informe, el usuario puede especificarla en la pantalla de parámetros

- Pueden especificarse las manipulaciones antes de invocar el informe, pasándoselas a éste en una lista de parámetros a través de una orden RUN - PRODUCT o RUN-REPORT-OBJECT invocada desde Forms, o bien

pasándoselas desde la línea de órdenes cuando se invoca la versión de tiempo de ejecución de Reports.

Consejo: Si intenta pasar parámetros a Reports desde un producto que no pertenece al entorno Oracle, debe ceñirse al número de caracteres disponible en la línea de órdenes (256 caracteres en algunos sistemas operativos). Esto limita enormemente la cantidad de información que puede pasarse al informe. Si necesita pasar grandes cantidades de información a un informe desde una aplicación ajena al entorno Oracle, puede que tenga que incluir la información en una tabla de base de datos o en un archivo de texto, cuyo contenido pueda extraer el informe tan pronto como se inicie.

Grupos

El diseño de un infon-ne no trabaja directamente sobre consultas, sino que trabaja sobre grupos. El grupo actúa como una escala intermedia para los datos, entre la consulta y el diseño final. Los grupos definen las relaciones maestro-detalle dentro de los datos. Por tanto, se puede obtener toda la información necesaria mediante una consulta y proceder luego a «trocear» la consulta en un cierto número de grupos maestro-detalle.

La definición correcta de los grupos en el modelo de datos es un paso fundamental en el desarrollo de un informe. Si no se disponen correctamente los grupos, puede que el infonne funcione mal o proporcione información impredecible.

Inicialmente, cuando se genera una consulta en el modelo de datos sin emplear el asistente Report Wizard, Reports coloca todas las columnas de la consulta en un único grupo. Es entonces cuando el desarrollador ha de asignar a cada columna el grupo que le corresponde. Por ejemplo, si acudimos al sencillo infonne DEPT/EMP, todas las columnas correspondientes a la tabla de departamentos se colocarían en el grupo ascendiente, colocando las colun-mas de la tabla de empleados en el grupo descendiente. Nótese que las columnas de fórmula y de resumen deben incluirse también en los grupos

apropiados. Por ejemplo, si se está creando una columna de resumen para calcular el salario medio de todos los empleados de un cierto departamento, tal columna debería encontrarse en el grupo DEPT. Si lo que se estuviera calculando fuera el sueldo total percibido por un empleado (salario más comisiones), dicha columna debería incluirse en el grupo EMP.

En el caso concreto de las columnas de resumen, no sólo hay que colocarlas adecuadamente, sino que hay que especificar también las reglas que regulan el reinicio de la columna de resumen. El punto de reinicio de una columna de resumen se halla casi siempre en el grupo en el que se la incluye. La única excepción a esto se produce cuando se están creando resúmenes acumulativos. Por ejemplo, si se está mostrando un total dinámico de salarios para toda la organización, el punto de reinicio de la columna de resumen será el informe en sí. El punto de reinicio de una columna de resumen se puede cambiar o establecer en su paleta de propiedades, modificando su propiedad *Reset At.* Esta propiedad gobierna el punto en el que se pone de nuevo a cero el resumen. Las opciones posibles son Report (informe, es decir, nunca se reinicia), Page (página) o cualquiera de los grupos de rango igual o superior al del grupo en el que se halla la columna de resumen.

La estructura de vinculación de grupos de un informe no tiene por qué ser lineal. Considérese el ejemplo de un informe en el que, para un cierto proyecto, es preciso mostrar a las personas asignadas al proyecto cuáles son las responsabilidades que se les han asignado. Considérese también que hay que mostrar los costes asociados con el proyecto.

Columnas de ruptura

Cuando se crean grupos en el modelo de datos, al lado de todas las columnas de un cierto grupo aparecen unas pequeñas flechas azules. Tales flechas indican que la columna se ha marcado como columna *de orden de ruptura.* Las columnas de orden de ruptura tienen dos funciones principales. La primera, y más importante, consiste en que

tal colun-ma indica el identificador unívoco del grupo. Es frecuente añadir columnas no imprimirles a un informe con objeto de garantizar la unicidad dentro de un grupo de ruptura. Volviendo al ejemplo del informe DEPT/EMP, supongamos que se está creando una relación maestro-detalle entre DEPT y EMP y que sólo ponemos DNAME en el grupo DEPT. Si se diera el caso de que dos departamentos tuvieran el mismo nombre, el sistema los combinaría automáticamente. Incluso si se incluye DEP'INO en el grupo DEPT, si esta columna no está marcada como columna de ruptura, no se utilizará como columna del identificador unívoco. Por tanto, los departamentos que tengan el mismo nombre seguirán siendo combinados.

Cuando se diseña un informe, es importante analizar cuidadosamente cuáles son las columnas marcadas como de ruptura. De manera predeterminada, Oracle marca todas las columnas como columnas de ruptura. Sin embargo, esta estrategia tiene como efecto la inclusión de instrucciones ORDER BY innecesarias en la consulta SQL enviada a la base de datos, pudiendo afectar negativamente al rendimiento. Dejar todas las columnas de un grupo marcadas como de ruptura es una práctica de desarrollo descuidada.

Consejo: Defina la propiedad Break Order *(orden de ruptura) de todas las columnas que no sean columnas de ruptura como «None». Puede que esta medida proporcione un rendimiento ligeramente mejor.*

La segunda función de las columnas de ruptura consiste en poderlas emplear como criterio de ordenación para el grgpo de ruptura. Dichas columnas no afectarán a la ordenación del grupo de detalle. Esta debe gestionarse mediante el uso de una orden ORDER BY en la consulta base. La ordenación se realiza de acuerdo con el orden en el que aparecen las columnas en el grupo.

Especificación de grupos

En términos generales, los grupos corresponden a áreas de impresión en el informe. La identificación de un grupo es mucho más compleja de lo que parece. No

siempre es fácil especificar las consultas y sus grupos. Parte del problema reside en que los grupos no necesariamente se corresponden con tablas de la consulta. Habitualmente lo hacen; sin embargo, es frecuente que el número de grupos sea diferente del número de tablas de la consulta o consultas que generan dichos grupos.

Un concepto clave es la necesidad de *asignar un nombre* a los grupos. El nombre del grupo debería ser un objeto del mundo real (normalmente, una tabla) acerca del cual esté almacenando información el grupo. Una posible regla para elegir tal nombre sería hacer que el nombre del grupo coincidiera con la palabra que completase la frase: «Cada fila de este grupo está mostrando información acerca de ... » En el informe DEPT/EMP, los nombres apropiados para los grupos maestro-detalle son «DEPT» y «EMP», respectivamente. No suele ser difícil crear un nombre apropiado para cada grupo. Si un desarrollador se encuentra con problemas para elegir estos nombres, esa es una buena indicación de que el modelo de datos es defectuoso. Sin embargo, en función del diseño del informe, la información contenida en el grupo no tiene por qué consistir en atributos de la tabla de ese objeto. Por ejemplo, en un informe D i se puede mostrar el nombre del departamento al que pertenece cada empleado al lado del nombre de éste. Sigue tratándose de un grupo EMP mostrando información sobre los empleados, pero DNAME no es un atributo de EMR

Es posible tener en el modelo de datos más grupos que zonas distintas haya en el informe físico, pero esto suele ser raro. La única situación en la que podrían existir grupos extra en el informe es cuando se decide no mostrar los datos de más bajo nivel de detalle, aunque siga necesitándoles para realizar cálculos complejos. Es raro que no se muestren los campos que representan las columnas de un grupo, a menos que se trate de los campos de nivel inferior de detalle del grupo. Un error típico consiste en tener grupos extra en el mayor nivel de agregación en las consultas combinadas. Esta situación requiere dos consultas, al igual que en el ejemplo anterior del informe de proyecto que mostraba el personal asociado con un proyecto, junto con los costes asociados. Sería tentador disponer ambas consultas.

Sin embargo, el grupo de proyecto de la segunda consulta es innecesario.

Filtros de grupo

Una de las características más interesantes de Data Model es la posibilidad de filtrar la información de un grupo. Ello se consigue definiendo la propiedad *Filter Type* (tipo de

1. **First.** Su efecto es el mismo que el de especificar el número máximo de filas de la consulta, excepto que ésta es una especificación para el grupo. En la propiedad Number of Records (número de registros) se especifica el número de filas que se desea ver.

2. **Last.** Ocultará todas las filas, excepto las que se encuentran al final de la consulta. Especifique el número de filas que desea visualizar en la propiedad *Number of Records.*

3. **PLISQL.** Permitirá incluir un filtro complejo en el grupo.

Cuando se aiíadió esta capacidad al producto por vez primera, parecía que era una característica prometedora que sería muy útil. Sin embargo, la práctica nos dice que se utiliza raras veces.

El modelo de díseño de Reports

La interfaz Layout Model del editor de informes es una herramienta extremadamente flexible para diseñar la disposición de un informe. La interfaz Layout

Model es el lugar en el que se especifica cómo se presentará el modelo de datos, ofreciendo al desarrollador un control avanzado sobre la fonna en que los objetos interactuarán y aparecerán en el informe. Sin embargo, si se intenta utilizar Layout Model sin una comprensión completa de cómo funciona, puede representar una de las experiencias más desagradables y frustrantes que puede vivir un desarrollador.

Layout Model funciona de manera muy diferente a sus equivalentes de otras herramientas de generación de informes. Muchas de ellas, especialmente las destinadas a usuarios finales, emplean una estructura de bandas. Tienen áreas horizontales predefinidas *(bandas)* en las que se especifica la colocación de las diferentes secciones físicas de un informe. Éste es un concepto que permite obtener buenos resultados con la mayoría de los informes. Sin embargo, la estratificación en bandas horizontales es inherentemente restrictiva, llegando algunas veces a hacer virtualmente imposible crear un informe complejo. Oracle Reports utiliza una filosofía fundamentalmente diferente, basada en dos conceptos subyacentes muy importantes:

- **Independencia completa entre el modelo de datos y el diseiío.** Lo primero que se crea es el modelo de datos, con Data Model. Luego se crea la disposición del infortne, su modelo físico, separadamente. Hay que terminar de trabajar con el modelo de datos antes de crear el modelo físico con Layout Model. La idea consiste en construir rápidamente un modelo físico básico, con objeto de comprobar que el informe no requiere el aporte de más información. Si hace falta añadir información, hay que volver al modelo de datos para incluirla en el modelo lógico. Una razón de peso para completar el modelo lógico antes de trabajar sobre el diseño del informe físico es que, después de haber creado el diseño básico, añadir objetos nuevos es un proceso tedioso. Es conveniente aprovechar Report Wizard para generar la mayor parte posible del informe automáticamente; cuanto más se pueda generar automáticamente, mejor. La mayoría de los informes necesitan sólo del asistente Report Wizard y de la herramienta Additional Default Layout (diseño predeterminado adicional), que se analiza más

adelante, sin obligar al desarrollador a que trabaje manualmente con Layout Model.

- **El uso de marcos.** En vez de emplear bandas horizontales fijas, se pueden especificar en el informe todas las regiones rectangulares o marcos que se precise. Los marcos agrupan a los objetos en el diseño del informe. Asimismo, gobiernan el comportamiento del diseño del informe, controlando la dirección de impresión, las relaciones maestro-detalle y los saltos de página. La clave para hacer un buen uso de Layout Model reside en comprender cómo trabajan e interactúan los marcos y los objetos.

Precaución: Una limitación *clave de* la utilidad de generación ***del diseño predeterminado*** **es** *su* **incapacidad** *para* **especificar** *la máscara* **de** *formato para una columna antes* **de** *realizar un* **diseño** *predeterminado.* **Si** *tal* **cosa** *fuera* **posible,** *prácticamente ningún* **informe exigiría** *trabajo manual en* **Layout Model.**

Toda esta flexibilidad permite crear casi cualquier informe imaginable, aunque, para ello, hay que pagar un cierto precio. Para aprovechar toda esa flexibilidad, es necesario configurar correctamente una serie de propiedades dentro de cada marco. Sin una configuración adecuada de las propiedades, el infon-ne puede proporcionar resultados impredecibles y horas de frustración. Por tanto, es importante comprender cómo funcionan los marcos y cómo hay que utilizarlos para crear informes complejos.

Dado que se pueden colocar objetos y marcos en cualquier parte del infonne, a la hora de imprimir el infonne, Reporis debe emplear reglas precisas para controlar cómo interactúan los objetos en el infonne. Con frecuencia, si la información de un objeto es tan grande que salta a una segunda o tercera línea, los objetos que se hallan por debajo han de desplazarse hacia abajo para dar a dicho objeto el espacio que necesita. Por supuesto, los objetos deben aparecer al lado de sus etiquetas respectivas, por lo que no se los puede desplazar arbitrariamente. El control de la colocación de un objeto en la pantalla es lo que se llama *anclaje* (que analizaremos más adelante, en la sección

«Anclas: cómo mantener las cosas en su sitio»). Cuando se ejecute, el informe aplicará un anclaje predeterminado a todos los objetos del modelo de diseño. Este anclaje predeterminado puede anularse mediante un anclaje explícito. Una de las características de Developer es que puede mostrar el anclaje predetertninado que Reports aplica a los objetos. Para ello, hay que abrir la ventana del navegador de objetos, seleccionar **Navigator, Navigator Options** y activar la casilla de verificación Anchoring Information (información de anclaje) de la ficha Layout. Con ello, el sistema mostrará tanto las anclas explícitas que se hayan introducido manualmente, como algunas de las anclas predeterminadas generadas por Oracle.

Una característica adicional, que hace de Reports una herramienta de generación de informes tan potente, es su empleo del lenguaje PUSQL. Se pueden escribir procedimientos PL/SQL complejos para que gobiernen si los objetos o grupos de objetos se visualizan o no. Se pueden también escribir funciones PL/SQL que devuelvan datos, para luego mostrarlos en el informe.

Oracle Reports no es tan flexible como un lenguaje de tercera generación, como C++. Existen ciertos límites en lo que respecta a las cosas que se pueden realizar en Layout Model. Por ejemplo, no es posible crear objetos o marcos mediante programa en tiempo de ejecución. La capacidad de alterar los objetos es también limitada. Sin embargo, a menos que esté intentando escribir su propia herramienta de consulta *ad hoc,* Reports debería ser capaz de satisfacer cualquier necesidad de generación de informes.

En este capítulo se va a desniitificar Layout Model, de manera que el lector pueda aprovechar las capacidades extraordinarias del producto Reports. Incluso si es un usuario experimentado de Reports, este capftulo contiene algunos consejos y técnicas útiles, que puede que el lector no haya descubierto.

Cómo se trabaja con Layout *Model*

Una vez que comprenda cómo se manipulan los objetos y qué propiedades se pueden cambiar, podrá crear los informes que desee. Lo más importante que hay que comprender respecto de la interfaz Layout Model es que la mayoría de las acciones se realizarán fuera de esta herran-úenta. La combinación de la herramienta Additional Default Layout (analizada más adelante en este mismo capftulo) con las plantillas de Reports permite reducir al máximo la cantidad de veces que habrá que crear o manipular objetos en Layout Model.

Debería evitar realizar manualmente las siguientes acciones en Layout Model:

- **Creación de grupos repetitivos y objetos de diseño.** Esta tarea se realiza más fácilmente con la herramienta Additional Default Layout.

- **Creación de un campo.** Normalmente, es mucho más fácil añadir un campo borrando el marco completo y regenerándolo mediante Additional Default Layout. Ello no evita que se tengan que añadir manualmente campos de resumen y los campos ubicados en los márgenes o en las secciones de encabezamiento o pie del informe.

- **Derinición de anclas manuales.** Se pueden colocar marcos de envoltura alrededor de los objetos para que el anclaje funcione correctamente. El sistema de anclajes se analiza más adelante en este mismo capítulo.

Entre las acciones que podemos llegar a realizar en Layout Model se pueden incluir:

- **Conriguración de la propiedad** *Format Mask* **(máscara de formato) de los campos.** No se puede crear una máscara de formato con Report Wizard, ni se puede especificar una máscara de formato predeterminada mediante una

plantilla. No hay más remedio que seleccionar el objeto en cuestión, ya sea en el navegador de objetos de Layout Model, ya sea en Live Previewer, y aplicarle la máscara de formato requerida. Observe que se pueden aplicar algunas máscaras de formato habituales empleando los ¡conos de la barra de herramientas de Live Previewer (añadir o quitar precisión después del punto decimal y funciones similares).

- **Creación de etiquetas.** Report Wizard generará correctamente la mayoría de las etiquetas, aunque habrá ocasiones en las que el desarrollador tendrá que crearse las suyas propias.

- **Uso de marcos de envoltura alrededor de los objetos.** Con ello se conseguirá que funcione correctamente el algoritmo de anclaje (tal y como se describe más adelante en este mismo capítulo).

- **Conriguración de las propiedades de algunos objetos.** La propiedad que hay que configurar con más frecuencia es el valor *Maximum Records per Page* (número máximo de registros por página) de un marco repetitivo. Su valor suele ser 1 para obligar a que cada registro aparezca en su propia página.

- **Conriguración de la propiedad** *Keep With Anchoring Object* **(mantener con objeto de anclaje).** Tanto con campos como con marcos, el uso de esta propiedad evita a veces la aparición de cabeceras huérfanas.

Colocación de disparadores de formato en los objetos. El uso más frecuente de los formatos consiste en suprimir la impresión de un marco, de un encabezamiento o de un campo.

Ahora que Reports incluye Live Previewer, existe un subconjunto de acciones que se pueden realizar en Layout Model y también en el previsualizador Live Previewer, con la ventaja de que en este último se puede comprobar inmediatamente el efecto de los

cambios realizados. Por esta razón, es mejor llevar a cabo las acciones siguientes en el previsualizador:

- Configuración de las propiedades *Format Mask* (ahora hay disponibles ¡conos especiales en la barra de herramientas de Layout Model, para realizar cambios comunes de máscaras de formato numérico).

- Cambio de anchos de columna.

- Cambio de las propiedades Font (fuente) y *Font Style* (estilo de fuente) de los objetos de texto.

Cambio de los colores de los objetos.

Uso de marcos

Los marcos son de dos clases: *repetitivos,* asociados con grupos en el modelo de datos, y *de envoltura,* que influyen en el anclaje de los objetos que se hallan dentro del marco y agrupan visualmente a los objetos en el informe.

Marcos repetitivos

Los marcos repetitivos se asocian siempre con un grupo, y sólo uno. Todo objeto que aparezca en este tipo de marco ha de pertenecer a dicho grupo, o bien a un grupo ascendiente del grupo con el que está asociado el marco, o bien ha de tratarse de un objeto del nivel de informe. La ilustración siguiente muestra los símbolos de los marcos repetitivos:

Si se coloca en un marco repetitivo un objeto de un grupo que sea un descendiente del grupo con el que está asociado el marco, o un objeto de un grupo asociado con una

consulta no relacionada, cuando se intente ejecutar el informe se obtendrá el siguiente mensaje de error:

Field <nombre del campo> references column <nombre de columna> at a frequency below its group.

Si se tiene un marco repetitivo basado en el grupo Department, en dicho marco repetitivo puede aparecer cualquiera de los objetos de este grupo, obviamente. Asimismo, en el marco repetitivo podría aparecer también cualquier objeto de los grupos Division o Company, o cualquier objeto del nivel de informe.

Dado que este marco se imprimirá una vez para cada departamento, no tiene sentido situar en él un objeto del grupo Employee, ya que hay muchos empleados en cada departamento. En el marco Departament se puede imprimir el nombre de la compañía o de la división junto al nombre de cada departamento, o bien se puede mostrar el salario total para todas las companlas en cada departamento, con independencia de si ello tiene o no sentido en el contexto de un informe específico. Esto es algo que casi nunca se hace en la práctica. Todas las columnas de un marco deberían proceder del grupo con el que está asociado dicho marco, lo que posibilita crear el diseño del marco con Report Wizard o con la herramienta Additional Default Layout.

La inclusión en un cierto grupo de un campo que pertenezca a un grupo de rango superior no ofrece problemas. Por ejemplo, si se incluye la columna DNAME en el grupo Employee, la consecuencia de este sencillo acto será la aparición del nombre del departamento junto con la información de cada empleado del mismo, apareciendo tantas veces en el informe como empleados tenga tal departamento. Si se desea que un cierto dato aparezca tanto en los marcos maestros de un informe como en los de detalle, el campo correspondiente habrá de ser incluido en ambos grupos.

Esto se puede realizar sencillamente seleccionando dos veces la columna correspondiente con un alias en la consulta. En lugar de crear un objeto adicional en el

modelo de datos, también se podría crear manualmente un objeto adicional en el modelo de diseño y asociarlo a la misma columna fuente. Sin embargo, si se crea el objeto adicional en el modelo de datos, ello permite crear objetos de diseño automáticamente, en lugar de tener que crearlos a mano. Si se trata de una columna de fónnula o de resumen, es aconsejable crearla en el nivel superior, reflejando su valor a través de una columna de fórmula que simplemente devuelva el valor del campo de resumen o de fórmula.

Dirección de impresión

Después de haber especificado el grupo en el que se basará el marco, hay que decidir cómo se desea que aparezca la información dispuesta en la página. Habrá que definir la propiedad *Print Direction* en la paleta de propiedades del marco repetitivo. El uso de un marco repetitivo presupone que su grupo devolverá múltiples registros. Si se sabe que el grupo sólo devolverá un registro, la dirección de impresión carecerá de relevancia. Se puede elegir una de entre cuatro posibles opciones:

- **Down (hacia abajo).** Se repetirá el marco a lo largo de la página, desde su parte superior hasta la inferior, continuando la impresión en la página siguiente. Ésta es, con mucho, la opción más habitual. Las tres opciones restantes casi no se utilizan.

- **Across (hacia la derecha).** El marco se repetirá de izquierda a derecha, a través de la página. Cuando no quede más espacio en la página, el proceso se repite en la página siguiente. El tipo de informe que suele requerir esta opción es el que tiene varias columnas con datos similares, como puede ser el caso de presu-

  puestos multianuales en los que cada colun-ma representa los datos de todo un año. Esta opción se utiliza también en informes de matriz.

- **Across / Down.** El marco se repetirá de izquierda a derecha hasta que no quede espacio, momento en el que se repetirá debajo en la misma página, de forma similar a los saltos de línea en un procesador de textos. Esta opción puede ser útil, por ejemplo, cuando el informe incluya múltiples observaciones sobre una muestra de prueba concreta. El marco repetitivo que almacene las observaciones se configurará con esta dirección de impresión.

- **Down 1 Across.** La repetición se produce en primer lugar de arriba hacia abajo. Cuando se alcanza el borde inferior, el marco se repite en el borde superior, de la misma página. Esta opción suele ser útil cuando lo que se está imprimiendo son etiquetas de correo.

Elasticidad vertical y horizontal

La elasticidad hace referencia a la forma en que los objetos se contraen o expanden en respuesta a la cantidad de información contenida en los datos que se han de mostrar. Las propiedades *Vertical Elasticity y Horizontal Elastici* se pueden aplicar a marcos y a campos de datos, tanto en Layout Model como en Live Previewer, pulsando con el botón derecho del ratón sobre el marco o campo y cambiando los parámetros de la paleta de propiedades. Las opciones posibles son «Contract» (contraer), «Expand» (expandir), «Fixed» (fijo) y «Variable». La elasticidad no se aplica a los objetos comunes (líneas, rectángulos, texto estático, etc.).

La expansión o contracción de un objeto sólo se produce en uno de sus lados: el derecho en la elasticidad horizontal y el inferior en la elasticidad vertical. Es importante recordar este detalle. Por ejemplo, si hubiera un espacio en blanco en la parte superior del marco, tal espacio seguirá existiendo con independencia de la opción de elasticidad elegida, pudiendo desaparecer un espacio equivalente situado en la parte inferior del marco.

He aquí las cuatro opciones de elasticidad de las que se dispone:

- **Contract.** Cuando se ejecuta el informe, el tamaño del objeto no puede ser mayor que el asignado en el modelo de diseño. Si quedase espacio vacío, el objeto podría ser más pequeño, pero nunca más grande. Si hubiera más información de la que se puede ajustar en el marco, se truncaría.

- **Expand.** Cuando se ejecuta el informe, el tamaño del objeto debe ser al menos el mostrado en el modelo de diseño. Si el objeto contiene más información de la prevista, su tamaño aumentará para dar cabida a la misma (hacia la derecha, en una expansión horizontal; hacia abajo, en una expansión vertical).

- **Fixed.** El tamaño del objeto en el modelo de diseño es exactamente el mismo que en el informe, cuando éste se ejecuta. No existe ningún símbolo especial para esta opción.

- **Variable.** El marco u objeto se expandirá o contraerá tanto como sea necesario, para acomodar los datos que se vayan a mostrar en el objeto.

Si se configura incorrectamente la elasticidad, podemos experimentar toda clase de problemas difíciles de diagnosticar. Los parámetros que más frecuentemente se emplean para un marco repetitivo son «Fixed» para *Horizontal Elasticity* y «Variable» para *Vertical Elasticity*. Ello permite que el marco se expanda hacia abajo tanto como sea preciso. Si se configura un marco repetitivo (especialmente uno maestro) con elasticidad «Fixed» en la dirección de impresión en la que está tratando de expandirse un marco descendiente, los resultados son impredecibles. Con frecuencia, puede ser de ayuda fijar la elasticidad horizontal o vertical de un marco. Si se está trabajando con un formulario preimpreso, exigiendo que el diseño del informe tenga una gran precisión, lo que más sentido tiene es trabajar con marcos de elasticidad fija. Los marcos de envoltura emplean elasticidad fija aproximadamente un 50% de las veces. Si rodean texto común o campos que no salten de línea, la elasticidad fija puede facilitar la compresión del diseño.

Recuerde que, si se emplea la elasticidad variable o de contracción, lo que se vea en el modelo de diseño puede ser muy distinto de lo que se vea en el previsualizador Live Previewer, debido al espacio adicional de los objetos, que puede desaparecer a medida que los marcos se expanden o contraen.

Campos

La mayoría de los campos suelen estar fijados tanto vertical como horizontalmente. Se suele conocer la cantidad de infonnación contenida en los campos de la base de datos y, por tanto, la que puede mostrar dicho campo. Tal información aparecerá en un lugar específico del informe.

Nota: Si, en Report Wizard, se especifica para un campo un tamaño inferior que el descrito para él en la base de datos, cuando Reports cree el campo, lo generará de manera predeterminada con su elasticidad vertical configurada para expansión (Vertical Elasticity «Expand»), *lo que permitirá que, cuando la información del campo llegue hasta el extremo derecho del espacio asignado, continúe en la línea siguiente.*

Los campos de descripciones y comentarios suelen tener establecida su propiedad *Horizontal Elasticity* como «Fixed» y su propiedad *Vertical Elasticity* como «Expand». Frecuentemente, esta combinación también es útil cuando se quiere disponer de un campo de longitud indeterminada y se necesita poder colocar otro objeto a la derecha de la información presentada. En la sección «Anclas: cómo conservar las cosas en su sitio» se podrá ver un ejemplo concreto de esta situación.

No suele ser necesario que los campos tengan configuradas las propiedades *Horizontal Elasticity y Vertical Elasticity* como «Variable».

Marcos

Las propiedades *Horizontal Elasticity y Vertical Elasticity* de la mayoría de los marcos se configuran como «Fixed» y «Variable», respectivamente. Tanto los marcos repetitivos como los de envoltura han de tener su propiedad *Vertical Elasticity* configurada como «Variable», para poder acomodar la información de cualesquiera objetos del marco que se repitan hacia abajo y que se expandan verticalmente. Si el informe está mostrando sólo uno o dos registros por página, es probable que ello se deba a que la propiedad *Vertical Elasticity* del marco en el que han de aparecer está configurada como «Fixed».

Consejo: Sí se crea manualmente un marco en el modelo de ***diseño,*** sus elas*ticidades vertical* **y** *horizontal se configurarán por omisión como fijas.* **Hay** *que asegurarse de cambiar tal configuración,* **si** *ello fuera preciso.*

Los marcos que se utilizan para agrupar campos y sus etiquetas suelen tener ambas elasticidades fijas. Otras configuraciones para los marcos se utilizan en los informes de matriz, pero por lo demás no se suelen emplear.

Los marcos repetitivos suelen tener ambas elasticidades fijas, a menos que uno de los campos de dentro del marco tenga expansión vertical. En tal caso, el marco debería tener definida la propiedad *Horizontal Elasticity* como «Fixed» y la propiedad *Vertical Elasticity* como «Expand».

Las opciones de elasticidad «Contract» y «Expand» se emplean con menor frecuencia, aunque se dan ciertas excepciones que analizaremos en los ejemplos siguientes.

Configuraciones de elasticidad de ejemplo para campos y marcos

A continuación se enumeran algunos ejemplos en los que se emplea la elasticidad de forma poco habitual.

Hopizontal Elasticity «Contract» y *Vertical* Elasticity **«Fixed».** Normalmente, no se requiere contracción horizontal para un campo. Esta configuración se puede usar en un informe en el que dos campos estén conectados por una línea de puntos. La contracción horizontal permite que el ancho del campo sea el mínimo necesario para mostrar su información. Además, hay que colocar un espacio en blanco al final del campo para asegurarse de que haya una distancia mínima entre el final del campo y el con-úenzo de la línea de puntos, el objeto F - Name tiene una configuración con elasticidad horizontal de contracción y elasticidad vertical fija.

Horizontal Elasticity «Fixed» y ***Vertical Elasticity*** «Fixed». Esta configuración puede emplearse con un marco repetitivo, y se utiliza con mucha frecuencia para formularios preimpresos. Puede servir también para objetos con un máximo de uno o dos registros de detalle, de modo que cada bloque maestro ocupe la misma cantidad de espacio. La paleta de propiedades del marco repetitivo Employee está configurada con este tipo de elasticidad.

- ***Horizontal Elasticity*** «Fixed» y ***Vertical Elasticity*** «Variable». Esta configuración constituye un truco útil cuando un marco de envoltura rodea a un texto común. En general, los objetos se anclan automáticamente a los otros objetos situados por encima suyo. Si se coloca un marco con una elasticidad de este tipo alrededor de un texto común y luego se le pone al texto un disparador de formato que suprima su impresión, el marco se colapsará y todos los objetos que haya por debajo de él se moverán hacia arriba. Si sólo se suprimiera la impresión del texto común, los objetos anclados a él se imprimirían como si se hubiera imprimido el texto común, dejando un espacio en blanco donde debería haber estado el texto.

Flex Mode

Flex Mode (modo flexible) es una gran funcionalidad que facilita el manejo de los marcos. Se puede activar este modo, tanto en Layout Model como en Live Previewer,

pulsando en el icono correspondiente. Con el modo flexible activado, cuando se desplaza o cambia el tamaño de un objeto, todos los demás objetos contenidos en el diseño se desplazarán de acuerdo con ello. Si un campo se halla en un marco y se desplaza el campo, el marco se expandirá para adaptarse al desplazamiento. Cuando se desplazan campos, sólo es posible desplazar los objetos a lo largo de un único eje, vertical u horizontalmente. De otra forma, Reports no sabría cómo desplazar los objetos asociados. Si hay que desplazar el objeto en diagonal, siempre podemos desplazarlo primero en horizontal, volver a seleccionarlo, y desplazarlo en vertical hasta la posición deseada.

Confine Mode

Confine Mode (modo de confinamiento) es una modalidad que hace que todos los objetos se mantengan en el interior de sus marcos ascendientes. Sin embargo, ello no evita el solape de marcos de igual categoría, lo que puede dar lugar a informes mal diseñados. Este modo sólo puede activarse en Layout Model, pulsando en el icono de la barra de herramientas que se asemeja a un candado.

Además, Reports no parece tener problemas con los marcos intersecantes que no sean lógicamente incorrectos. Sin embargo, esta circunstancia permite crear algunos diseños de apariencia extraña. En general, el modo Confine Mode debería estar siempre activado, excepto cuando se desee desplazar un objeto (normalmente un objeto común) de un marco a otro.

Reports 6.0 ha tomado un enfoque distinto del de versiones anteriores. Intentará crear un informe incluso si se ha hecho algo muy ¡lógico, como disponer un marco descendiente solapándose con un marco ascendiente. Incluso si un solo pixel del marco descendiente se solapa con el marco ascendiente, Reports tratará el marco descendiente como si no estuviera dentro del marco ascendiente. Si se mueven los objetos por el infonne sin tener activado el modo de confinamiento, debería verificarse en el navegador de objetos que el diseño sigue siendo correcto.

Diseños predeterminados adicionales

La funcionalidad que más eficientemente minimiza el uso manual de Layout Model es la herramienta Additional Default Layout. Si se aprende a manejar esta herramienta, se conseguirá ahorrar horas de diseño que, de lo contrario, se invertirían en la modificación manual del diseño de los informes. Lo normal, cuando se empieza a trabajar con Oracle Reports, es que un desarrollador invierta menos de una hora en componer la consulta del informe y el resto del día en trabajar con Layout Model. A medida que se aprende a crear informes con eficiencia, el tiempo que se invierte en utilizar Layout Model se reduce considerablemente. Si se utiliza la herramienta Additional Default Layout, el desarrollador puede crear el diseño correspondiente a cada grupo individualmente. Este método se suele emplear para crear un solo grupo, pero no hay nada que impida que se pueda generar todo el informe de esta manera. Si se desea emplear esta herramienta, basta con pulsar en el icono Additional Default Layout, colocado en la parte inferior izquierda de la barta de herramientas vertical.

Pulse con el ratón y arrastre hasta definir la zona en la que desee colocar el diseño generado. Esta zona debe ser lo suficientemente grande como para acomodar en su interior todo el diseño. Si se está generando un grupo descendiente, debería crearse dentro del marco repetitivo ascendiente. Tras definir la zona de diseño, aparece un asistente Report Wizard modificado, en el que no se puede modificar la consulta pero se pueden seleccionar los grupos y columnas que se desea generar. Normalmente, para ajustar el diseño de un informe, se crea un grupo, se lo visualiza en Live Previewer, se borra y se realizan después las modificaciones en el modelo de datos, con Data Model, o se modifican los anchos de los campos en Report Wizard, antes de volver a realizar la generación. Este proceso se repite hasta que el diseño es el adecuado. El empleo de esta estrategia puede reducir considerablemente el tiempo invertido en utilizar Layout Model.

Cómo añadir campos

Cuando ya casi está terminado el informe, el desartollador descubre que le falta un campo. ¿Cómo se añade un campo a un informe? La mayoría de los desartolladores reajustan todo a mano para hacerle sitio al nuevo campo, lo que puede llevar una cantidad de tiempo nada despreciable. En su lugar, se debería borrar todo el marco, el que contenga el espacio en el que tendría que colocarse el campo nuevo, junto con todos los objetos que contenga, procediendo a recrear dicho marco con la funcionalidad Additional Default Layout. Ésta es la forma más sencilla de aiíadir un campo nuevo a un infonue, con mucho. Sin embargo, este método sólo funciona bien si se está creando el informe empleando Report Wizard y plantillas de informe.

Existen algunas raras ocasiones en las que es preciso, añadir manualmente un campo nuevo. Para ello, hay que crear el espacio necesario activando la función Flex Mode y desplazando un objeto o un asa de posicionamiento de un marco. Cerciórese de que se encuentra en el grupo adecuado, cree el catnpo nuevo y asígnele a un origen que sea el mismo que el del grupo en el que se encuentre. Puede asignarle, como origen, una columna de un grupo de nivel superior, aunque esto no es recomendable. Si se asigna un origen erróneo (que suele ser una columna de un grupo situado a un nivel inferior que el del grupo de consulta), se obtendrá un mensaje de error indicando que el campo hace referencia a una columna de nivel inferior al de su grupo.

Desplazamiento de objetos

En general, el método para desplazar objetos por la pantalla es el n-iismo que el de aiíadir campos.

Precaución: Hasta que sepa lo que ***está*** haciendo, el ***desplazamiento*** de ***los objetos puede ser problemático. Debe*** ejecutar ***el informe*** con frecuencia para ***asegurarse de*** que no ha ***corrompido el diseño del*** informe. Recuerde que ***debe guardar el informe*** a menudo. ***Puede,*** también, utilizar la ***opción Undo (deshacer), ubicada en*** el *menú* **Edít** *(edición), para invertir la última acción.*

Si se generan correctamente los objetos a la primera con Additional Default Layout y Report Wizard, no será necesario desplazarlos. **Si** los campos están en un orden equivocado, desplácelos en el modelo de datos y rehaga el diseño del informe con el asistente Report Wizard. Si los campos no están dispuestos correctamente para un forinulario, añada columnas contenedores y regenere el diseño del informe con el asistente. No hay muchas ocasiones en las que el desarrollador tenga que desplazar los objetos manualmente. También es posible emplear Live Previewer para desplazar los objetos.

Consejo: No desactive el botón Lock (bloqueo). Su uso evita que la ***mayoría de los objetos se salgan de sus grupos lógicos. Esto*** no funciona con ***los marcos*** de ***igual categoría,*** aunque ***le protegerá de*** la ***mayor*** parte ***de las*** equivocaciones.

Si tiene que desplazar una etiqueta de un marco a otro, desactive el modo de bloqueo, desplace la etiqueta hasta su nueva posición y vuelva a activar dicho modo, cuidando de que la etiqueta quede completamente envuelta por su nuevo marco ascendiente. Oracle Reports suele comportarse de fonna inteligente a la hora de desplazar las cosas a sus agrupamientos lógicos apropiados. Sin embargo, sigue siendo preciso utilizar el navegador de objetos para comprobar que los objetos desplazados están correctamente colocados. En versiones anteriores de Reports, era preciso emplear con frecuencia las opciones Move Backward / Move Forward (desplazar hacia atrás / hacia delante) del menú Arrange (ordenar) de Layout Model para colocar correctamente los objetos tras el desplazamiento de uno de ellos. En la versión anterior, algunas veces no era posible mover los objetos dentro del marco ascendiente adecuado. El truco para resolver esto consistía en cortar y pegar los objetos y luego recolocarlos. Estas tácticas no parecen ser necesarias con la versión actual de Reports, pero puede ser útil si se siguen teniendo problemas para colocar correctamente los objetos.

Los objetos de diseño están dispuestos en una estructura jerárquica lógica. Los marcos ascendientes repetitivos envuelven a otros marcos descendientes repetitivos. Los objetos dentro de un grupo repetitivo deben pertenecer al marco repetitivo. Los objetos

dentro de un marco de envoltura han de hallarse subordinados a él. Todos los objetos que se encuentren en el mismo lugar de la jerarquía estarán en el mismo nivel de Reports. Oracle Reports ha mejorado con el paso del tiempo, pem-útiendo ahora la colocación automática de los objetos en sus niveles correctos. **Si** se desplaza un objeto en el modelo de diseño, se recolocará automáticamente en el nivel lógico apropiado. La excepción más habitual a este comportamiento se produce cuando se crea un marco de envoltura alrededor de objetos ya existentes. Se podría pensar que tal marco encerraría dichos objetos, pero ése no es el caso. En vez de eso, el marco se creará en el mismo nivel que los objetos. **Si** se desea cambiar la posición lógica del nuevo marco, hay que desplazarlo en el árbol lógico hacia arriba, seleccionando **Arrange, Move Backward** o pulsando F8. Cuando realice esta operación, revise con extremo cuidado el árbol del navegador de objetos, para poder saber cuándo se ha desplazado el marco hasta su posición lógica correcta. Reports suele gestionar apropiadamente el posicionamiento automático de los objetos en los niveles adecuados. Pero, si se crean objetos manualmente, es imprescindible asegurarse completamente de que se encuentran en el nivel adecuado. En caso contrario, se puede utilizar una de las siguientes opciones de menú y teclas de función.

- **Arrange, Bring to Front 0** F5 (desplazar hacia delante)
- **Arrange, Send to Back 0** F6 (desplazar hacia atrás)
- **Arrange, Move Forward** o F7 (desplazar hacia abajo)
- **Arrange, Move Backward** o F8 (desplazar hacia arriba)

Anclas: cómo mantener las cosas en su sitio

La colocación de los objetos en un informe es una tarea fundamentalmente diferente de la colocación de elementos en Oracle Forms. Cuando se coloca un objeto en una pantalla, ésa es su ubicación. En un informe, se espera que la posición de los objetos colocados en él se reajuste en función de la manera en que los demás objetos del infonne se expandan o contraigan. Los generadores de infonnes sencillos, que utilizan la filosofía de las bandas, tratan esta situación situando los objetos en bandas horizontales que atraviesan la pantalla, y que se expanden o contraen según sea preciso. En Reports, no existe la restricción de construir los informes con simples bandas, sino que se puede emplear una rica selección de objetos rectangulares que se expanden o se contraen vertical u horizontalmente. Los usuarios de Oracle Reports esperan que el producto les proporcione diseños apropiados. Para conseguir esto, los objetos se anclan entre sí automáticamente. En general, un objeto se ancla a su vecino más próximo que pueda expandirse o desplazarse hacia él. El algoritmo de anclaje predetenninado se basa en la identificación del objeto que mejor «empuja» a cada objeto concreto. Los detalles de este algoritmo se pueden encontrar en la ayuda de Reports, bajo el epígrafe «Body Algorithm» (algoritmo del cuerpo del informe). Sin embargo, si el infonne no aparece como se esperaba, debido a que los objetos se están solapando entre sí, se puede cambiar la forma en que los objetos se anclan unos a otros, definiendo anclas explícitas o colocando un marco de envoltura alrededor de los objetos que no se muestren correctamente.

Se pueden utilizar anclas para situar un objeto descendiente en relación con su ascendiente, tanto vertical como horizontalmente. El objeto ascendiente mostrará el símbolo de ancla en Layout Model. Las anclas ayudan a definir las posiciones relativas entre los objetos, ya que algunos objetos del diseño pueden ver alterado su tamaño cuando se ejecute el informe. El posicionamiento se basa en el tamaño de los objetos tras la ejecución del informe. De manera predeterminada, Reports intenta anclar los objetos de la forma que parece más razonable. Cada objeto está anclado a su vecino imnediato por la izquierda o por arriba. Como ya se ha indicado, se puede mostrar parte de la información de los anclajes seleccionando **Navigator, Navigator Options** y activando el comnutador *Anchoring Information* (información de anclaje) en la ficha Layout. Es

posible anular el anclaje predeterminado seleccionando la herramienta de anclaje de la barra de herramientas vertical de Layout Model y pulsando sobre parte del objeto descendiente. Pulse después sobre la parte del objeto ascendiente con la que se desea que esté anclado el objeto descendiente. Nótese que el anclaje ha de establecerse entre partes concretas de los objetos ascendiente y descendiente. Esto sólo es importante si el ascendiente o el descendiente no tienen fijo su tamaño vertical u horizontalmente. Normalmente, la mejor estrategia consiste en anclar una esquina de uno con una esquina del otro. Sin embargo, es casi imposible dibujar las anclas de esta forma. Lo normal será que se dibujen las anclas de cualquier manera y luego se pulse sobre el anclaje con el botón derecho para que aparezca la paleta de propiedades, donde cambiaremos el campo de porcentaje a «O» o «100» en ambos lados.

Ejemplo de un ancla

¿Por qué es preciso reemplazar las anclas predetemiinadas? Es posible generar cientos de informes sin tener que sustituir nunca un ancla predeterminada. Sin embargo, a veces no existe otro camino para hacer que los campos se comporten adecuadamente. Un ejemplo para ilustrar esto es el de dos campos, uno de nombre y otro de apellido, que se desea que se muestren juntos con independencia del tamaño del campo del nombre.

Se puede anclar el final del campo de nombre (first - name) con el comienzo del campo de apellido (last-name), con un espacio apropiado entre ambos. Ello proporciona la ilusión de que sólo se está imprimiendo un campo. Una solución mejor consiste en incluir una columna de fórmula que concatene ambos campos con un espacio intermedio, lo que elimina el problema.

Si el diseño del infonne requiere muchas anclas, probablemente no se esté realizando cuidadosamente. Casi siempre hay que evitar las anclas explícitas. Incluso en el ejemplo anterior, se puede evitar utilizar un ancla creando en el modelo de datos una columna de fórmula, que concatene el nombre con un espacio y con el apellido, o concatenando las columnas en la consulta. Este sistema hace que el informe sea más

limpio y sencillo de mantener. Cuando esté diseñando informes, piense siempre en estrategias alternativas que eviten el uso excesivo de anclas.

Disparadores de formato

Sólo hay una clase de disparador asociado con un objeto de diseño: el disparador de formato. Un disparador de formato es una función booleana. Si la función devuelve el valor «True» (verdadero), el objeto aparecerá en el informe. Si la función devuelve un valor «False» (falso), el objeto no aparece. Esta función es útil en muchos contextos. Tomemos, por ejemplo, un informe maestro-detalle: No hay necesidad de mostrar los campos maestro si no existen detalles.

Se puede incluir un disparador de formato en el marco que rodea los objetos del grupo maestro, ya sea en el propio marco repetitivo, ya sea en el marco que rodea al marco repetitivo, y suprimir la impresión si no hay registros de detalle. La forma de averiguar que no hay registros de detalle consiste en introducir en el grupo maestro una columna de resumen (NO - OF - RECORDS), que lleve la cuenta de los registros de detalle asociados. A continuación, escriba el siguiente disparador de formato para el grupo:

```
IF no-of-records
THEN
   RETURN FALSE;

ELSE

   RETURN TRUE;

end IF
```

Cuando se ejecute esta rutina, si no hay registros descendientes, se suprimirá la impresión del marco ascendiente maestro y de todos los objetos incluidos en él.

Uso y construccion de plantillas de Reports

Una plantilla de Oracle Developer Fortns es, esencialmente, una pantalla parcialmente creada. En Forms se puede crear una plantilla a partir de cualquier archivo FMB, como resultado de lo cual, el desarrollador obtiene la flexibilidad que proporciona poder incluir en ella disparadores, objetos y bibliotecas, cuyo nivel de complejidad puede elegir a voluntad. Una plantilla de Oracle Developer Reports funciona de manera muy diferente.

Una plantilla de Reports es un archivo especial que contiene objetos que se copian en un informe en función de información y propiedades preseleccionadas, que gobiernan la forma en que se crea un diseño mediante los asistentes. Cuando se aplica una plantilla a un informe existente, Reports intenta tomar las decisiones adecuadas acerca de qué objetos deberían volver a copiarse y cómo disponer el informe. Por desgracia, la arquitectura de Reports no incluye la posibilidad de asignar indicadores de «modificación» a cada uno de los objetos del informe. Por esta razón, si se vuelve a aplicar una plantilla a un infonne podemos obtener un comportamiento inesperado del producto.

Consejo: Una plantilla sólo se debe aplicar a un informe una vez. Las aplicaciones adicionales de la plantilla que se deban realizar se harán mediante la herramienta Addítional Report Layout, que se encuentra en la barra de herramientas de Layout ModeL Si cambia la plantilla en la que se basa un informe, verifique manualmente cada aspecto de la aplicación de la misma, con objeto de asegurarse de que se ha obtenido el resultado esperado.

Se analizarán los siguientes temas:

- La estructura de plantillas de Reports.
- La construcción de una plantilla de ejemplo.
- El establecin-iiento de normas para las plantillas de informe y la construcción de un grupo de plantillas necesario para un entorno de producción de Reports.

Las plantillas de Reports constituyen una funcionalidad extremadamente valiosa añadida a Oracle Developer. Por vez primera, se tiene la posibilidad de generar con gran rapidez informes elegantes. Incluso se podrán crear informes complejos sin precisar de una gran manipulación del modelo de diseño. Casi ninguna de las modificaciones manuales que se realizan en el modelo de diseño debería conllevar algo más que la especificación de un modelo de diseño predeterminado adicional y su generación mediante una plantilla. La única excepción a esto se produce cuando se requiere un disparador FORMAT en un marco o en un objeto.

Si no se dispone de un conjunto de plantillas de informes que puedan asistir en el desarrollo de los informes, se está dejando de lado una de las mejoras de productividad más importante de Oracle Developer. Con las versiones anteriores de Reports se podían generar informes complejos con rapidez, pero había que llevar a cabo muchas y profundas manipulaciones manuales, para conseguir que su apariencia fuese elegante. Con la versión actual, tal manipulación no es necesaria, ya que, utilizando una plantilla, se,, pueden generar informes complejos. Si el desarrollador está realizando una cantidad considerable de modificaciones manuales en los informes, ello es señal de que no se están aprovechando las ventajas inherentes al uso de las plantillas de Reports.

Generalidades

Una de las grandes diferencias entre las plantillas de Reports y las de Forins consiste en la capacidad de las plantillas de Forms de emplear objetos reverenciados. Si cambian los estándares la interfaz de usuario (GUI), el cambio se puede realizar rápida y fácilmente en Forms, haciendo que dicho cambio se refleje en las pantallas existentes. Con Reports, una vez que se genera un informe a partir de una plantilla, deja de existir conexión alguna entre el informe y la plantilla. No es posible hacer que un cierto cambio se propague a varios informes. El único camino disponible para compartir cualquier elemento en Reports consiste en utilizar objetos de base de datos, como las funciones y los procedinúentos de un paquete, o mediante bibliotecas, que se pueden asociar a múltiples informes. Éstos son los únicos métodos para hacer que los cambios afecten simultáneamente a más de un infonne. Sin embargo, si se utiliza Report Wizard para construir completamente los informes, es posible generar un elevado porcentaje de los informes. Estos informes se pueden entonces regenerar rápidamente, utilizando una plantilla distinta.

Las plantillas de Reports tienen, en realidad, dos propósitos. Por un lado, su misión es la misma que la de las plantillas de Forms, ya que se pueden definir elementos comunes a todos los infonnes, se pueden especificar distintos disparadores, añadir parámetros de usuario, asociar bibliotecas e incluir objetos comunes en los diseños de los informes. El otro propósito de las plantillas de Reports es controlar la forma en que se genera el diseño del infonne a través del asistente Report Wizard o de la herramienta Additional Default Layout. Se pueden especificar los tipos de letra y muchas otras propiedades utilizadas en campos, etiquetas y grupos.

Reports incluye un cierto número de plantillas de informe de ejemplo. Sin embargo, la mayoría de ellas son útiles sólo para realizar demostraciones de las prestaciones del producto, no estando pensadas para su empleo en producción. A modo de ejemplo, muchas de tales plantillas incluyen como fondo una foto de la sede corporativa de Oracle.

Limitaciones

Las plantillas de Reports tienen algunas lin-útaciones:

- No se pueden colocar objetos en la sección Body (cuerpo) de la utilidad Template Layout Model (modelo de diseño de plantilla). Ésta no es una restricción particularmente problemática, pero obliga a colocar en los márgenes los títulos, números de página, fechas y cualquier objeto de plantilla.

- En Template Data Model (modelo de datos de plantilla) no se pueden crear objetos globales, lo que se puede contrarrestar creando parámetros de usuario y variables de paquete en la plantilla.

- Ya no hay forma de especificar objetos Header (cabecera) y Trailer (pie) predeterminados en la plantilla de infon-ne, aunque se podía hacer en Reports 3.0.

Estructura de la plantilla

En la plantilla hay disponibles varias secciones que se corresponden con las secciones de un informe:

- Data Model (modelo de datos)

- Layout Model (modelo de diseño)

- Report Triggers (disparadores de informe)

- Program Units (unidades de programa)

- Attached Libraries (bibliotecas asociadas)

Sin embargo, tales secciones no coinciden del todo con las secciones de un informe. Además de dichas secciones, existe también una paleta de propiedades para todo el informe. Si se cambia la configuración predeterminada en la paleta de propiedades de la plantilla, cada informe creado con ella dispondrá de las nuevas configuraciones. Los parámetros que tienen más probabilidades de cambio son *Height* (altura) y *Width* (anchura) de la página. Si se está imprinúendo con el formato Portrait (vertical) en vez de con el de Landscape (horizontal), será preciso cambiar los valores de altura y anchura en la paleta de propiedades del modelo de diseño.

Editor de plantillas: el modelo de datos

En una plantilla, cuando se utiliza Data Model, el desarrollador puede especificar parámetros de usuario y parámetros de sistema predetenninados. Por tanto, ello permíte asignar valores predeterminados a los parámetros de sistema. Por ejemplo, se puede especificar que el informe se ejecute en el modo de vista preliminar asignando el valor «Preview» a la propiedad DESTYPE *Initial Value* (valor inicial de tipo de destino) en la paleta de propiedades.

Nota: En la ayuda ínteractiva, bajo el epígrafe «System Parameters», se puede *encontrar una descripción de otros parámetros de sistema.* **El** *sistema de ayuda no contiene descripciones en las secciones dedicadas a cada parámetro individual.*

Se pueden añadir parámetros de usuario para permitir el uso de parárnetros léxicos o variables de acoplamiento. Es posible añadir al modelo de datos cualquier parámetro común del que se desee disponer en todos los informes, como pueden ser rangos de fechas o números de registros. Reports incluirá automáticamente estos parámetros en todos los informes generados mediante esta plantilla. Tenga en cuenta que los parámetros de usuario y de sistema son los únicos objetos que se pueden especificar en el modelo de datos de una plantilla de informe. No es posible crear consultas de

plantilla predeterminadas, ni incluir ningún otro objeto de los que se crearían en el modelo de datos del informe.

Editor de plantillas: el modelo del diseño

La sección Layout Model de la plantilla es el núcleo de la plantilla del informe. Es aquí donde se especifican las propiedades que influyen en la forma en que el asistente Report Wizard y la herramienta Additional Default Layout generan los diseños de informe. Usando una plantilla de informe, se pueden especificar propiedades generales para cualquier informe. También se pueden definir configuraciones específicas para cada uno de los diferentes tipos de informe (Tabular, Group Above, Matrix, etc.). Se puede especificar la apariencia de los diferentes atributos visuales utilizados para los grupos de salto de un informe maestro-detalle; por ejemplo, es posible especificar que el tamaño de letra empleado para el grupo maestro sea mayor que el del grupo de detalle. Las opciones predeternúnadas en el nivel de informe se definen en la sección Body. Sin embargo, para tipos específicos de informe o para niveles específicos dentro de un informe, se podría desear disponer de formatos distintos para niveles diferentes. Por ejemplo, en un informe maestro-detalle-detalle que contenga departamentos, ubicaciones y empleados, se puede ajustar el tamaño de letra para los departamentos a 20 puntos, el de las ubicaciones a 15 puntos y el de la información detallada sobre los empleados a 12 puntos.

En la sección Layout Model del editor de plantillas, sólo se pueden ubicar objetos nuevos en la sección Margin (margen) de la plantilla. En versiones anteriores de Reports, se podían colocar también objetos en las secciones Header y Trailer de la plantilla; sin embargo, dicha funcionalidad ha desaparecido del producto. Cuando se aplique una plantilla a un informe, cualquier objeto que aparezca en los márgenes de **la** misma se copiará en el informe.

El cuerpo del informe en la plantilla funciona de manera distinta que el margen. No se pueden crear objetos nuevos en el cuerpo de la plantilla, sino que se introducen cambios en las propiedades de los objetos en el navegador de objetos, y se observa

pasivamente el efecto de dichos cambios en el modelo de diseño. Se pueden seleccionar y modificar objetos en el modelo de diseño e invocar su paleta de propiedades para modificar las propiedades, pero no es posible crear ni elin-finar objetos en el modelo de diseño.

Tras haber establecido los valores predeterminados en la sección Main (principal) del modelo de diseño, la sección Override (suplantar) permite *suplantar* dichos valores. Esta característica permite cambiar las configuraciones predeterminadas elegidas para las propiedades asociadas con cada uno de los estilos de informe. Este tipo de acción debe llevarse a cabo individualmente para cada estilo de informe. Cuando se especifican en el área de suplantación secciones dedicadas a estilos específicos de informe, éstos heredarán automáticamente las propiedades configuradas en la sección Main.

Cuando se crean plantillas, se está trabajando en dos áreas: el navegador de objetos y la utilidad Layout Model del editor de plantillas. El sistema más conveniente consiste en trabajar en el navegador de objetos n-fientras se mantiene abierta la ventana Layout Model, para observar los efectos de los cambios realizados en el navegador de objetos.

Section

La primera área que se puede manipular en la sección Layout Model del editor de plantillas es la paleta de propiedades Section. Aquí es donde se especifica un cierto número de propiedades, entre las que se incluyen la altura *(Height)* y anchura (Width) de una página. A diferencia de las demás propiedades de las plantillas, las propiedades de Section no se pueden suplantar. Ésta es una limitación del producto, ya que se podría desear disponer de un informe del tipo Group Left (agrupación izquierda) basado en un tamaño de papel de 1 1" x 8.5" (orientación apaisada). Para conseguir esto, sería necesaria una plantilla distinta.

Body

En esta sección se describe la forma en la que se desea que trabaje el editor de diseño, especificando aspectos tales como los tamaños de letra y los colores de diferentes clases de objetos. Si se está desarrollando un informe maestro-detalle complejo y varios niveles o categorías, el desarrollador puede instruir a Report Wizard sobre la manera en que se desea disponer cada nivel del informe. Esta capacidad permite construir informes de apariencia elegante y sofisticado con mucha rapidez. La sección Templates, Layout Model, Body, Default permite especificar los parámetros que describen cómo se dispondrá todo en el informe. La sección Templates, Layout ModelBody, Override permite cambiar tales configuraciones para tipos específicos de informe y para partes concretas de tipos específicos de informe (es decir, el marco maestro puede ser distinto de los marcos de detalle).

Sección Body-Default. La paleta de propiedades de la sección Default gobierna el comportanúento de todos los marcos. Se puede configurar, por ejemplo, la cantidad de espacio horizontal que habrá entre los campos, mediante la propiedad *Interfield (Horizontal).* Cualquier parámetro que se defina aquí puede suplantarse en la sección Override, con el fin de cambiar un marco concreto para un tipo especfflco de informe (es decir, maestro-detalle-detalle).

En el navegador de objetos, en la sección Templates, Layout Model, Section, Body, Default, se pueden configurar las propiedades que gobiernan la forma en la que se disponen los objetos en un grupo repetitivo, controlando el comportamiento de una sección de un informe. Por ejemplo, se puede utilizar la paleta de propiedades Default para configurar el espacio entre marcos y campos. Por desgracia, no es posible configurar en una plantilla el espacio existente entre los registros de un grupo repetitivo, lo que sería una opción útil.

En la sección Default, hay disponibles cinco categorías de propiedades que se pueden configurar:

Frames. Esta paleta de propiedades controla las propiedades visuales de cuatro tipos de marco: Section (sección), Headings (encabezamientos), Fields (campos) y Summaries (resúmenes), así como la forma en la que se crean tales marcos. Cada vez que el asistente crea una sección de un informe, muestra cada uno de los cuatro tipos de marco (tres, si no hay resúmenes).

- **Field Labels / Headings.** Esta paleta de propiedades permite especificar la apariencia visual de las etiquetas de datos y de los encabezamientos de columnas. Es posible configurar tres tipos de subencabezamientos: Character (alfanumérico), Number (numérico) y Date (fecha). Cada uno de ellos dispone de su propia paleta de propiedades. En la práctica, la única diferencia que se suele desear entre ellos es la que concierne a la propiedad Character *Justification.* Su configuración suele ser «Left» (izquierda) para las etiquetas y encabezan-úentos de campos de caracteres y fechas, y «Right» para las etiquetas de campos numéricos y sus encabezamientos.

- **Fields.** Están disponibles las mismas tres subsecciones que para Field Labels Headings: Character, Number y Date. Estas paletas de propiedades se utilizan para configurar las propiedades visuales de cada tipo de infonnación, como son el tipo de letra *(Font),* el color *(Color)* y el estilo de relleno *(Fill Style).* No es posible especificar aquí una máscara de fonnato *(Fonnat Mask).*

- **Summary Labels.** Están disponibles las mismas propiedades que en Field Labels / Headings.

- **Summaries.** Se pueden especificar las propiedades visuales de los campos de datos de resumen para los tipos Character, Number y Date.

Dado que la mayoría de los informes suelen ser del tipo maestro-detalle, probablemente deseará que cada nivel del informe tenga una apariencia ligeramente distinta. Existen varias estrategias para conseguirlo. Se pueden aplicar las configuraciones predeterminadas al infortne de detalle, suplantándolas para el maestro, o

viceversa. En cierta forma, es más intuitivo que la configuración predeterminada gobierne la apariencia del detalle y suplantarla para el maestro, ya que habrá ciertos infonnes que no sean del tipo maestro-detalle y que asumirán la configuración predeterminada.

Suplantaciones. Las suplantaciones permiten especificar propiedades de plantilla para tipos específicos de informe (e incluso para diferentes niveles de salto de dichos informes), que reemplazarán a las propiedades definidas en la sección Default. En la sección Templates-Layout Model-Section-Body-Override se pueden especificar propiedades concretas para cada tipo de informe y, cuando sea apropiado, para diferentes niveles de salto dentro de cada tipo de informe. De manera predeterminada, esta sección empleará los parámetros definidos en la sección Default. Cualquiera de las propiedades configuradas en Default puede suplantarse en la sección Overrides y el nuevo valor se puede aplicar a cualquier tipo de informe o nivel de salto de un informe.

Para suplantar las propiedades configuradas en la sección Body-Default para un cierto tipo de informe, basta con acudir a la paleta de propiedades de la sección Override de cada tipo de informe. Por ejemplo, para suplantar la configuración de los informes del tipo Tabular, habrá que acceder a la paleta de propiedades de la sección Templates-Layout Model-Section-Body-Override en el navegador de objetos, y efectuar en ella los cambios deseados.

Las suplantaciones permiten especificar qué parámetros se desea configurar para cada tipo de infon-ne, incluso si la configuración es distinta para cada uno de ellos. Una de las desventajas de estas anulaciones consiste en que hay que configurarlas separadamente para cada tipo de informe. No se puede especificar el comportamiento general de todos los registros maestros. Aun si los registros maestros de los tipos de informe Group Left y Group Above, por ejemplo, han de tener las mismas propiedades, habrá que definirlas separadamente. Un aspecto interesante y útil de la capacidad de suplantación consiste en que, para cada sección, podemos disponer de las propiedades de la correspondiente paleta, así como de las disponibles en las secciones Frames, Field

Labels / Headings, Fields, Summary Labels y Summaries. Ello permite suplantar cualquiera de las propiedades especificadas en la sección Default.

Una limitación de esta capacidad consiste en que no se pueden suplantar los objetos de los márgenes ni dotar a cada tipo de informe de objetos diferentes en sus márgenes, ya que estos objetos se definen una vez para toda la plantilla. Si se desean tener objetos diferentes en los márgenes para tipos distintos de infonne, hay que definir plantillas distintas para cada tipo de informe.

Las propiedades de la sección OverTide se heredan de la sección Default según el comportamiento tradicional de orientación a objetos. En concreto, si se cambia una propiedad en la sección Default, la sección Override la heredará. Puede que se desee configurar explícitamente una propiedad en la sección Override con el n-úsmo valor que el que figura en la sección Default. La razón para actuar así es que, incluso si se cambia la propiedad en la sección Default, su configuración no será suplantada para un informe específico. Un ejemplo puede ser la configuración de distancias entre marcos en un informe de etiquetas de correo: podrían ser las mismas que en la configuración predeterminada, pero incluso si hubiera que cambiarlas en dicha configuración, no se querrían cambiar para el informe de etiquetas de correo. Este objetivo se consigue de manera parecida a como se consigue en Forms: configure el valor de la propiedad del objeto heredado de forma que difiera de la configuración predeterminada y luego vuelva a configurarla con el valor original. Con ello se romperá el vínculo con la sección Default y se fijará el valor de la propiedad en la sección Override con el último valor introducido.

Disparadores de informe

Se pueden crear disparadores específicos de informe en una plantilla. ¿Qué es lo que ocurre cuando se aplica a un informe una plantilla que contiene disparadores? Si el informe no contiene ya líneas de código para el disparador de la plantilla, Oracle Reports

copia (no hereda) el código del disparador de la plantilla. El efecto de esta operación puede comprobarse si se cambia la plantilla y se la vuelve a aplicar al informe: la operación no cambiará las líneas de código del infoitne correspondiente al disparador, porque ahora el informe ya contiene líneas de código para el disparador. Ésta es una de las razones para no aplicar una plantilla dos veces.

Observe que este comportamiento es distinto de la forma en que Oracle Designer genera los disparadores. En Designer, y para cada disparador, se puede tener parte del código generado automáticamente y parte del mismo creado manualmente. La sección de código del disparador en Reports no es tan sofisticado. La regla consiste en que, si el informe no contiene código correspondiente a un disparador cuando se aplica la plantilla, entonces el código de la plantilla se copiará en el infonne. La plantilla no interactuará con los disparadores de informe bajo ninguna otra circunstancia.

Los disparadores de informe escritos por el desarrollador no tienen que compilarse necesariamente con limpieza; pueden consistir incluso en fragmentos de código. Si hubiera porciones de código que se tuvieran que modificar para cada informe, puede que sea incluso útil que el disparador no se compile, forzando al desarrollador a modificar las líneas de código, al construir el informe, con objeto de eliminar los errores de compilación.

Unidades de programa

Las unidades de programa se comportan de forma similar a los disparadores de infoitne. Si se incluye una unidad de programa en una plantilla y se aplica ésta a un informe, dicha unidad de programa se copiará en el infonne, tal y como es de esperar. Si el informe tiene ya una unidad de programa con el mismo nombre, tal unidad quedará inalterada. Si la unidad de programa aparece en el informe por la aplicación de una plantilla y se hacen modificaciones en la unidad de programa del informe y se aplica de nuevo la plantilla, los cambios realizados no se verán afectados.

Precaución: Sí se elimina la plantilla ***de¡ asistente Report*** Wízard, declarando ***que el informe*** no ***está basado*** en una plantilla, tal operación eliminará ***las*** uní***dades de programa contenidas*** en la plantilla, ***incluso*** aunque ***se las hubiera modificado.***

Bibliotecas asociadas

En el momento de escribir este libro, la funcionalidad Attached Library (biblioteca asociada) en las plantillas es aún limitada. Cuando se crea un informe basado en una cierta plantilla, si ésta tiene una biblioteca asociada también la tendrá el informe. Sin embargo, si se borra la biblioteca asociada del informe y volvemos a hacer que Report Wizard aplique la plantilla al informe, Reports no podrá volver a asociar la biblioteca. **Si** se aplica a un informe una plantilla con una biblioteca asociada y luego le aplicamos otra sin bibliotecas asociadas, la aplicación de la segunda plantilla no eliminará la biblioteca asociada del informe.

Si se crea un informe basándose en una plantilla dotada de una biblioteca asociada, el informe resultará dotado de dicha biblioteca. Sin embargo, si se decide cambiar las plantillas o eliminar la biblioteca asociada, el programa podría no comportarse según lo esperado. Tras la realización de pruebas bajo circunstancias diferentes, se dieron algunos casos en los que Reports asoció la biblioteca, otros en que no lo hizo o, incluso, otros en que interrumpió la ejecución del programa. Cuando se realice cualquier acción que no consista en crear un informe por primera vez mediante el uso de la plantilla, es muy recomendable verificar si están asociadas las bibliotecas apropiadas.

Construcción de una *plantilla de* ejemplo

En esta sección describiremos todas las acciones precisas para la creación de una plantilla de Reports operativo. Esta plantilla corresponde al sistema mediante el que los autores de este libro construyen informes de producción y debería servirle al lector como marco de trabajo para crear sus propias plantillas.

Antes de construir una plantilla, se debe tener experiencia en el uso del generador de informes (Report Builder). El objetivo es construir una herramienta que aumente la productividad en la construcción de informes. **Si** no se tiene una idea clara sobre cómo construir informes, mal se puede crear una plantilla útil.

Un hecho que hay que reconocer acerca de las plantillas es que se puede crear una única plantilla capaz de manejar múltiples clases de infortne. Con una sola plantilla, se puede trabajar en diferentes clases de informes (tabular y de pantalla), dado que la plantilla permite especificar parámetros para cada tipo de informe.

El primer paso para la creación de una nueva plantilla es pulsar en la opción «Templates» del navegador de objetos y pulsar luego en el botón Create.

Modelo de datos

Es necesario modificar algunos parámetros en la sección Templates, Data Model del navegador de objetos.

Parámetros del sistema

Se trata de los parámetros básicos de sistema que necesita un informe, que incluyen un conjunto de valores predeterminados. Si se está trabajando en el modo cliente-servidor y se están generando informes para impresión, posiblemente sea una buena idea cambiar la propiedad *Initial Value* del tipo de destino (DESTYPE) a «Preview», para ver los informes en pantalla antes de proceder a imprimirlos. Se ha de configurar la propiedad DESTYPE *Initial Value* como «Printer» si se desea que los

informes salgan directamente por la impresora. Si se desea generar los infonnes para poderlos utilizar en la Web, hay que especificar la propiedad DESTYPE *Initial Value* como «File», la propiedad DESFORMAT *Initial Value* como «PDF», «RTF» o «HTML», y hacer que la propiedad DESNAME *Initial Value* contenga la ruta predeterminada al archivo en el que se vayan a almacenar los informes. Se puede suplantar el valor predeterminado de cualquiera de los parámetros del sistema con objeto de que la plantilla se adapte a las necesidades puntuales de cada informe. Cada una de las propiedades debe configurarse en la plantilla con las opciones deseadas.

Parámetros de usuario

Aquí se puede especificar una lista de parámetros de usuario externos al informe. El valor de dichos parámetros puede definirse en tiempo de ejecución mediante el uso de una pantalla de parámetros (Parameter Forin) o puede ser transferido al informe desde Forins a través de una lista de parámetros. Por último, pueden ser definidos en la línea de órdenes, si se están invocando los informes desde otro producto. No hay límite o restricción para los tipos de datos (caracteres, fecha, numérico) que se pueden emplear en estos parámetros, ni en cuanto al número de parámetros que se puede especificar. Los parámetros que se sugieren a continuación son los que utilizan los autores de este libro cuando han de construir un número reducido de informes. Estos parámetros se pasan al informe dinámicamente en tiempo de ejecución. He aquí los parámetros disponibles:

DISPLAY HEADER (HEAD-DISP). Se trata del título del informe. Conviene que sea un campo de caracteres de longitud variable, con una longitud máxima de 1.000 caracteres y un valor inicial del estilo de «Título de informe de ejemplo».

- **Start page.** Su valor predeterminado es 1, indicando el número de página por el que comenzará el infortne.

El resto de los parárnetros definidos por el usuario son para la creación dinán-úca de informes:

FLEX-FROM. Se emplea para añadir tablas a la cláusula FROM de la consulta.

- **FLEX-WHERE.** Se utiliza para añadir datos a la cláusula WHERE.
- **Breakl.** Se usa para que el usuario pueda elegir columnas de salto en el informe.
- **Break2.** Se usa para que el usuario pueda elegir columnas de salto en el infon-ne.
- **Breakl-disp.** Se usa para asignar etiquetas a las columnas de salto.
- **Break2-disp.** Se usa para asignar etiquetas a las columnas de salto.
- **FLEX-DISP.** Se trata de un campo de presentación de gran tamaño (2.000 caracteres) que contiene una descripción narrativa de los parámetros pasados al infortne.

Modelo de diseño

Sólo existen unos pocos objetos que se puede desear añadir al diseño. Los números de página, las fechas y horas, y los títulos de ejecución para los informes son los objetos que se añaden con mayor frecuencia. Estos objetos han de colocarse en el margen del informe. Si el editor de plantillas no está abierto, pulse con el botón derecho del ratón en cualquier punto de la sección Templates del navegador de objetos y seleccione Template Editor. Podrá ver los tipos de letra predeterminados para la forma en que se dispondrán los objetos. Pulse en el botón Margin, para acceder a la zona de diseño de los márgenes.

Fecha y hora

Si se desea poner una fecha en la esquina superior izquierda de un informe, el primer paso es abrir el modelo de diseño del informe. Luego, pulse en el botón Insert Date and Time (insertar fecha y hora) de la barra de herramientas superior. Con ello se activará el asistente para insertar la fecha y hora, en el que se puede elegir los formatos de fecha y hora. El formato de fecha utilizado habitualmente es DD-MM-YYYY.

Título del informe

En lugar de predefinir el título del informe es posible pasar el tftulo del informe como parámetro de usuario. Esto permite utilizar el mismo diseño de infonne para múltiples infoitnes.

El objeto F-HEADER abarca la mayor parte del ancho del margen superior. El origen de su contenido es el parámetro HEAD-DISP del informe de ejemplo. El tftulo para una instancia concreta del informe se pasa desde la pantalla. Si el título de un cierto informe ha de ser estático, haga simplemente que ese título sea el valor predetem-únado del parámetro HEAD-DISP. Su propiedad *Horizontal Elasticity* debe configurarse como «Fixed», de manera que, si un título es demasiado largo, no invada la zona de la fecha, a la izquierda del informe, o la del número de página, a la derecha del mismo. La propiedad *Vertical Elasticity* puede configurarse como «Variable».

Número de página

Una de las formas de gestionar la numeración de las páginas consiste en emplear la numeración de páginas predeterminada suministrada por Reports. Es suficiente para la mayoría de los informes.

Para emplear la numeración predeterminada, siga estos pasos:

1. Abra el modelo de diseño del editor de plantillas, pulsando con el botón derecho del ratón en cualquier punto de la zona de plantillas del navegador de objetos y seleccione «Template Editor».

2. Pulse en el botón Margin de la barra de herramientas superior. La barra de herran-úentas cambiará de contenido y aparecerá una línea gruesa alrededor del cuerpo del informe.

3. Puede utilizar el botón Insert Page Number (insertar número de página) ubicado en la barra de herramientas, para invocar al asistente Insert Page Number Wizard y especificar las opciones apropiadas.

Si se han seguido los pasos anteriores, el sistema generará cierto texto común en la posición especificada. Nótese que este procedimiento incrustará el valor de un campo (en este caso, el campo Page Number). Cuando se ejecute el informe, mostrará el texto «Page» <número de página>. Nótese que el texto del objeto común es «Page&<PageNumber>». Esta técnica funcionará en cualquier parte del informe. En cualquier campo de texto se puede incrustar el valor de cualquier campo de informe, precedido del símbolo «&». Esto puede ser útil porque el texto común puede imprimirse en cualquier ángulo, mientras que el de los campos de texto sólo se puede imprimir horizontalmente. Es igual de sencillo crear texto común manualmente, sin emplear el asistente.

Visualización de los parámetros del informe en tiempo de ejecución

El último campo de margen empleado en la plantilla de ejemplo se encuentra en la esquina inferior izquierda de la página, debajo del cuerpo. Se emplea para mostrar un texto que describe los parárnetros de usuario que se han pasado al informe, y sólo se imprime en la primera página. Es recomendable que este campo abarque toda la parte inferior de la página. La propiedad *Horizontal Elasticity* se configura como «Contract», y la propiedad *Vertical Elasticity* como «Variable». El origen de su contenido es FLEX-DISP. En este caso, el campo dispone de un contorno hecho con una línea negra.

Por si se diera el caso de que el informe no recibe información alguna, se ha de incluir en el informe el siguiente disparador FORMAT, que suprimirá del todo la impresión del campo si se produce tal circunstancia:

```
v-return BOOLEAN := TRUE;
BEGIN
    IF :flex-disp IS NULL THEN
        v-return := FALSE;
    END IF:
    RETURN (v-return);
```

Unidades de programa y bibliotecas asociadas

Las unidades de programa y las bibliotecas asociadas pueden contener todos los procedimientos y funciones generales PL/SQL que se empleen para la generación de los informes. Es preferible incluir todos los procedimientos descritos hasta ahora en una biblioteca asociada, antes que almacenarlos como unidades de programa. Ello se debe a que (aparte del comportamiento irregular de las plantillas con respecto a esta sección), una vez que se ha obtenido una instancia de un informe mediante la plantilla correspondiente, no queda ninguna relación de herencia entre el informe y dicha plantilla. Si se cambia la plantilla, el informe no cambiará automáticamente de acuerdo con el cambio de la plantilla. Las funciones y procedimientos almacenados localmente en unidades de programa de la plantilla se copiarán en cada infonne obtenido mediante dicha plantilla. Si, más tarde, se halla un error en alguno de dichos procedimientos, habrá que

actualizarlo manualmente en cada uno de los informes obtenidos mediante aplicación de la plantilla original. Si se colocan estas funciones y procedimientos en una biblioteca asociada, todo lo que hará falta será modificar el contenido de la biblioteca y volver a compilarla. Todos los informes que la utilicen se actualizarán automáticamente.

Cambio de la configuración predeterminada de la plantilla

Con objeto de que el informe, una vez generado, tenga la apariencia deseada, se pueden modificar las configuraciones de diseño para cada uno de los tipos de informe del asistente Report Wizard. Para ello, hay que situarse en la sección Templates-Layout Model-Section-Body-Default del navegador de objetos y pulsar en Default, con lo que aparecerán las secciones Frames, Field Labels / Headings, Fields, Summary Labels y Summaries. El desarrollador puede modificar los tipos de letra predeterminados y controlar los atributos visuales tanto de los datos normales como de las columnas de resumen.

Las instrucciones siguientes sirven para cambiar los atributos visuales para que sean iguales a los empleados en la plantilla de ejemplo:

- Para cambiar el formato de las etiquetas predeterminadas para los datos, hay que expandir el nodo Default de la sección Body. En el nodo Field Labels / Headings que aparece, pulse con el botón derecho del ratón sobre Character y haga que las propiedades de tipo de letra sean «Arial», «Bold» en *Font Style* y 10 puntos en *Size.*

- En la sección Fields-Character, cambie las propiedades de tipo de letra a Times New Roman y 10 puntos.

- En la sección alfanumérica Summary Labels, cambie las propiedades de tipo de letra a Arial / Bold Italic / 1 0 puntos.

- En la sección Summaries, cambie las propiedades de tipo de letra, en *Character*, a Times New Roman / Bold / 10 puntos.

Ahora debería guardar la plantilla y crear un sencillo informe basado en la plantilla recién creada, utilizando Report Wizard. Si se modifica la plantilla y se vuelve a ejecutar Report Wizard, puede que los cambios realizados se reflejen en el informe o puede que no. Ello se debe a que Reports intenta conservar todos los cambios que se hayan realizado en el informe tras su generación. Al no existir un indicador individual de «modificado» para cada objeto, Oracle Reports intenta tomar decisiones lógicas sobre lo que debe sustituirse y lo que no.

Si desea ver reflejados todos los cambios en el informe, borre todos los objetos en Layout Editor y vuelva a generar el diseño cada vez que haga un cambio en la plantilla. La razón de sólo eliminar los objetos en el modelo de diseño y no comenzar de nuevo es que, para un informe complejo, la definición del modelo de datos es un proceso laborioso. Una vez definido el modelo de datos, se pueden realizar cierto número de intentos para generar el diseño adecuado, ya sea utilizando distintas plantillas, ya sea cambiando los anchos de las columnas. Puede que se dé cuenta de que algunas de las etiquetas son incorrectas, o de que los campos no son lo bastante anchos. Puede que sea más fácil utilizar Report Wizard para regenerar el diseño que manipularlo en Live Previewer o Layout Model.

Existen otras muchas propiedades que también se pueden manipular. Sin embargo, las que acabamos de citar son las únicas que es necesario cambiar con respecto a sus configuraciones predeterminadas.

Una vez especificadas las configuraciones predeterminadas, se pueden configurar propiedades independientemente para cada tipo de informe. Por ejemplo, el tipo de informe más habitual es Group Above. Dado que la plantilla de ejemplo incluye hasta

dos niveles de salto, merced a las columnas de salto, el desarrollador debe configurar los tamaños de letra predeterminados apropiados para dichos grupos de salto.

1. Abra la paleta de propiedades de Templates-Layout Model-Section-BodyOverride-Group Above-Section (Level l). En la sección Field Labels / Headings, cambie el tamaño del tipo de letra (Font size) a 14 puntos y a 12 puntos en Section (Level 2).

2. Un informe maestro-detalle-detalle que se base en esta plantilla tendrá tres niveles. Vaya al área Layout Model-Section-Body-Override de la plantilla y pulse en Group Above. Pulse después tres veces en el icono Create, con objeto de crear tres secciones.

3. En Section (Level l), cambie los tamaños de letra de Field Labels / Headings y de Fields a 14 puntos. En las áreas Summary Labels y Summaries de Section (Level l), cambie los tamaños de letra de los tipos de datos Character, Number y Date a 14 puntos.

4. Cambie el tamaño de letra de todos los tipos de datos de Field Labels Headings de Level 2 a 12 puntos.

5. No cambie el tamaño de letra de *Character* en Summaries para el nivel 2, ya que este campo se hallará en la parte inferior de la columna para los objetos de nivel 3 y ha de tener su misma apariencia.

6. En el área Templates-Layout Model-Section-Body-Default, cambie la propiedad *Inter-Field (Horizontal) a* «. 1 » (en lugar de .1 74).

Éstos son los únicos cambios que han de hacerse. Sería útil poder cambiar la propiedad *Vert. Space Between Frames* (espacio vertical entre marcos) para los marcos

repetitivos, pero la plantilla no permite modificar dicha propiedad. Un cambio como el indicado ha de realizarse sobre el propio informe, tras haberío generado.

Por último, se puede añadir una línea negra que separe las secciones de nivel 1 de un informe de las secciones de nivel 2, en un informe de tipo Group Above. Para ello, hay que configurar la propiedad *Borders* del área Layout Model-Section-BodyOverride-Group Above-Section (Level 1)-Fraines-Section Frame como «Bottom Only» (sólo inferior).

Definición de estándares para *las plantillas de informe*

No existe una única estrategia correcta para la creación de plantillas de informe. Sin embargo, hay algunas estrategias básicas que ayudan a aumentar al máximo la eficiencia del desarrollo de informes.

Cuando se están configurando estándares de interfaz gráfica de usuario (GUI) en la plantilla de ejemplo, se nos presenta de nuevo la vieja discusión sobre si conviene realizar modificaciones importantes en el informe tras su generación, o si es mejor sacrificarse para adaptarse a lo que el producto puede generar. El uso de las plantillas perniite preparar infortnes de apariencia mucho mejor con el diseño predetenninado. Sin embargo, si se desea reducir al mínimo la cantidad de trabajo manual realizado con Layout Editor, hay que ser cuidadoso al establecer estándares de interfaz gráfica de usuario.

En general, los estándares visuales tradicionales de generación de infonnes suelen conducir a la obtención de informes más fáciles de mantener, de lectura más fácil para los usuarios y capaces de mostrar más datos en una misma página. El uso de elementos especiales para resaltar aspectos concretos, la aplicación de diferentes formatos en función del contenido del informe y la inclusión de algunas líneas en el mismo, aparte de añadir poco o nada a la facilidad de uso del informe, representan un aumento del

tiempo de desarrollo potencialmente grande. Como en el caso de Forms, se debe intentar establecer estándares que eviten aumentar grandemente el tiempo de desarrollo del informe. Siempre podemos encontrarnos con informes clave de carácter ejecutivo que requieran la aplicación de unos formatos y de un control muy precisos. No se debe olvidar que tal sofisticación se obtiene a cambio de un cierto precio.

Además, ha de reconocerse que, durante la construcción de informes de producción, la mitad del tiempo se invierte en utilizar Report Wizard para preparar la disposición de todo el informe. La otra mitad del tiempo se invertirá en emplear la herramienta Additional Default Layout para construir una parte del informe. Ello significa que serán necesarias varias plantillas. Por ejemplo, un cierto informe del tipo Group Above podría tener hasta cuatro niveles. Si se especifica un informe Group Above de cuatro niveles en la sección Override de una plantilla y se intenta aplicar ésta a un informe maestro-detalle de dos niveles, la plantilla aplicará sólo los niveles 1 y 2. Sin embargo, puede que el objetivo sea aplicar los niveles inferiores de detalle, en vez de los superiores. Además, en un entorno complejo, puede que quiera diseñar los marcos de sus informes de uno en uno. Por ello, es útil crear un cierto número de plantillas distintas. Para un informe del tipo Group Above de cuatro niveles, podría ser necesario disponer de un total de seis plantillas:

- De cuatro niveles.
- De tres niveles.
- Dedos niveles.
- Plantillas individuales de nivel 1 para cada uno de los otros informes multinivel (4, 3 y 2).

Una misma plantilla puede incluir parámetros de suplantación para más de un tipo de infonne. En la práctica, sin embargo, el mantenimiento de tal plantilla puede resultar

confuso. Es preferible disponer de un conjunto diferente de plantillas para cada tipo de informe que genere el asistente Report Wizard. Si se desea ver satisfechas la mayoría de las necesidades de generación de informes, serán necesarias las seis plantillas de Group Above anteriormente mencionadas, cuatro plantillas de informe Mailing Label para los tamaños de etiqueta habituales y una plantilla para informes de tipo matricial (Matrix). Un informe sencillo de tipo Tabular o Fonn utiliza la misma plantilla de cuatro niveles indicada para Group Above, ya que se pueden utilizar los parámetros predetern-únados de dichos informes. Los tipos de informe Group Left y Matrix with Group no se generan con la suficiente frecuencia como para que merezca la pena crear plantillas para ellos. Si fuera preciso, bastarían algunas modificaciones manuales para adaptar las plantillas existentes a la generación de tales informes.

Creación de un Informe de producción

Este ejemplo ilustrará cierto número de principios importantes relacionados con el desarrollo de informes. No sólo utilizaremos el asistente Report Wizard, sino que también tendremos que crear parte del informe a mano.

Analizaremos tanto los puntos fuertes de Query Builder como sus limitaciones, junto con alguno de los procesos mentales que han de seguirse durante la construcción de un informe complejo. Este informe se ha creado tomando como ejemplo un informe de producción real. Se trata de un informe de complejidad superior a la media.

Las tablas de demostración suministradas con Oracle Developer Release 6. Estas tablas se pueden-instalar empleando el menú Inicio de Windows y seleccionando la opción Install Demo Database Objects (instalar los objetos de base de datos de demostración) del grupo de programas Oracle Developer Release 6. Esta operación dará lugar a la creación de una pequeña cantidad de tablas de ejemplo y de otros objetos de

base de datos. Antes de poder llevar a cabo esta operación, hay que crear un usuario que tenga los privilegios apropiados para la creación de tablas. Por omisión, la utilidad creará las tablas en el esquema SCOTT. Por desgracia, hay diferentes versiones de producción de Oracle Developer Release 6, suministradas con versiones distintas de la base de datos de demostración; por tanto, es posible encontrarse con que algunas de las tablas y columnas tienen nombres distintos de los que se han utilizado en este capítulo. En los sitios web de los autores podrá e-n-.ontrar la versión exacta de la base de datos de demostración.

En este tutorial, construiremos un informe que será, probablemente, más complejo que los que tendrán que crear la mayoría de los desarrolladores. Este tutorial demuestra que Oracle Report Builder permite construir eficientemente incluso lof'informes más complejos. Es posible crear informes sencillos y ponerlos en producción, en una hora o menos. Un desarrollador con experiencia que conocemos fue capaz de crear nueve informes de producción en un solo día, siendo alguno de ellos bastante complejo.

Este tutorial dista de ser sencillo. Sin embargo, demostrará algunas de las características más potentes de la herramienta, características que no suelen hacerse evidentes durante la creación de infonnes más simples. Muchas herramientas de generación de informes serían incapaces de crear este informe.

Preparación para la creación del informe

El primer paso para construir cualquier informe consiste en conocer con exactitud qué tipo de informe se desea crear. Se debe esbozar en un papel, o por lo menos en la cabeza, cuál será el aspecto final que deseamos para el informe. En el caso que nos ocupa, se deten-ninó que la estructura del informe.

Nótese que se trata de un informe maestro-detalle en el que el maestro tiene dos detalles independientes. En este caso se muestra una lista de productos. Para cada producto, el informe presenta cuántas unidades de un cierto producto ha consumido cada

cliente, así como cuántos productos fueron vendidos por cada vendedor. Este informe se basa en un modelo de datos que incluye un grupo maestro y dos grupos de detalle, cuyo diagrama se muestra en la ilustración siguiente. Report Wizard no puede generar directamente el infortne completo, aunque sí puede generar una parte importante del mismo.

Una idea recomendable consiste en incluir, en la documentación del informe, las partes esenciales del ERD. La complejidad de la sección del modelo de datos mostrada en la ilustración anterior es, en realidad, bastante habitual en un infonne. La única diferencia a la hora de crear informes de producción consiste en que, en un sistema de producción, el desarrollador utilizará vistas, probablemente, en vez de trabajar directamente contra las tablas base.

Al llegar a este punto, es necesario establecer de alguna manera cuáles van a ser los grupos de consulta.

Observe que aparecen cinco grupos en dos consultas. El grupo PROD es el grupo maestro, mientras que los grupos CUST y EMP son los grupos de detalle. Los grupos SALE y EMP SALE muestran la información detallada del nivel de transacción utilizada para generar cantidades e importes de resumen. Nótese también la diferencia entre la estructura de los grupos en la consulta y la disposición de las entidades en el modelo de datos. No hay nada que obligue a que exista una similitud estructural entre la manera en que se agrupa la información en el informe y el agrupanúento utilizado en el modelo de datos subyacente. A partir de este momento, estamos en disposición de construir la primera parte del informe.

Paso 1: elección del estilo del informe

Después de iniciar Report Builder, hay que seleccionar File, New, **Report** en el menú, para crear un informe nuevo. Aparecerá un cuadro de diálogo, en el que habrá que seleccionar la opción correspondiente al asistente Report Wizard. Si aparece la página

«Welcome to the Report Wizard» (bienvenido a Report Wizard), pulse en el botón Next (siguiente). Si no desea ver esta página cada vez que ejecute Report Wizard, desmarque la casilla de verificación correspondiente, antes de pulsar en el botón Next.

Report Wizard aparece de dos formas distintas. La primera vez que se genera un infonne, este asistente guía al usuario durante todo el proceso de creación del informe. Si se invoca Report Wizard después de haber creado el informe (seleccionando Report Wizard en el menú Tools), se podrá acceder a cada una de las páginas de dicha herramienta. El Apéndice D contiene detalles adicionales sobre Report Wizard.

En la ficha Style (estilo) se tiene la posibilidad de elegir uno de entre varios estilos de informe. El estilo de informe gobierna el diseño del mismo. El Apéndice D contiene los descripciones de tales estilos. Para los propósitos de este tutorial, seleccione Group Above.

Paso 2. conexión con la base de datos

Si no ha establecido aún la conexión con la base de datos, pulse el botón Connect (conectar). La conexión se ha de realizar empleando el usuario que posea las tablas de ejemplo.

Paso 3. creación de la consulta; selección de las tablas

Lo primero que hay que hacer es crear la consulta de los clientes. Pulse en el botón Query Builder, mantenga pulsada la tecla CTRL y seleccione las tablas CUSTOMER, ITEM, PRODUCT y ORD en la ventana Select Data Tables (seleccionar tablas de datos). Pulse en el botón Include (incluir). Cierre la ventana Select Data Tables. Observe que las tablas pueden ser combinadas automáticamente. Esta información de combinación se obtiene a partir de las restricciones de integridad referencias almacenadas en la base de datos. Puede que sea preciso combinar alguna de las tablas a mano. Utilice Query Builder para crear una combinación entre dos tablas:

pulse sobre la columna de clave externa de la tabla descendiente y arrástrela hasta la columna de clave primaria de la tabla ascendiente. Otra opción adicional consiste en pulsar, en la barra de herramientas, el botón Set Table Relationship (configurar relación entre tablas; se trata del botón que muestra dos tablas con una flecha de interconexión), y escribir las relaciones. Este tutorial requiere el establecimiento de las combinaciones siguientes:

ITEM.PRODID=PRODUCT.PRODID

ITEM.ORDID=SALES-ORDER.ORDER - ID

ORD.CUSTID=CUSTOMER.CUSTID

Paso 4: creación de la consulta;'sélección *de columnas*

Seleccione las columnas siguientes del informe, pulsando en la casilla de verificació contigua al nombre de la columna. En la tabla PRODUCT, seleccione PRODID DESCRIR En la tabla CUSTOMER, seleccione CUSTID y NAME. En la tabla ITEM, seleccione ACTUALPRICE, QTY e ITEMTOT. Se podrán ver las correspondientes marcas de verificación al lado de las columnas seleccionadas. Pulse en OK para generar la instrucción de consulta SQL.

Paso 5. creación de la consulta; *modificación* del código *SOL*

Uno de los problemas que podemos hallar al emplear Reports consiste en que todeoi los nombres de columna son globales. En la instrucción de consulta SQL el nombre de la columna que almacena el nombre del cliente es «Name», mientras que el nombre de la columna destinada a guardar la descripción del producto es «Descrip.» Ninguno de ellos es un nombre claro de columna. Más adelante, durante la creación de nuestro informe, será útil tener claro a qué hace referencia cada nombre. Por tanto, es

necesario emplear alias para sustituir a tales nombres, lo que se lleva a cabo modificando el código SQL generado mediante la adición del alias «Cust - name» para la columna de nombre de cliente, y del alias «Prod-Desc» para la columna de descripción del producto. Pulse Next.

Paso 6.- creación de la consulta; selección de los campos de ruptura

Declare los campos (columnas) que van a constituir el grupo maestro, desplazando las columnas PRODID y PROD-DESC hasta la caja de la derecha. De manera predeterminada, esto da lugar a la creación de dos grupos de ruptura. Pulse y arrastre PROD-DESC hasta colocarlo encima de PROD - ID. El icono de cursor del ratón cambiará su forma por la de una mano, desapareciendo el grupo del nivel 2 cuando se libere el botón del ratón.

Cree un segundo nivel desplazando CUSTID y CUST - NAME hacia la derecha y arrastrando CUST-NAME hacia arriba, hasta el grupo CUSTID. Pulse en el botón Next.

Paso 7: decisiones de visualización

Hay que decidir qué es lo que se va a presentar en el informe. En este caso, seleccione todo (utilizando el botón que muestra una doble flecha hacia la derecha). Si los campos no se encuentran en el orden en el que deseamos que aparezcan, podemos arrastrarlos y colocarlos en el orden deseado en la ventana. El orden en el que aparezcan los campos en la pantalla es el mismo en el que aparecerán en el informe. Pulse en Next.

Paso 8. columnas de resumen

Especifique que desea generar una suma sobre ITEMTOT y QTY. Pulse en Next.

Paso 9: etiquetas

En este punto del proceso, se podrían cambiar la anchura de los campos o las etiquetas del informe. Para los propósitos de este ejemplo, podemos dejar las etiquetas como están. Pulse en Next.

Paso 10. selección de plantilla

Para este ejemplo, seleccione No Template (sin plantilla). Pulse en el botón Finish (terminar).

Paso 1 1: adición de una consulta

No se puede utilizar el asistente Report Wizard para añadir consultas adicionales. Hay que utilizar la interfaz Data Model del editor de informes para añadir manualmente la segunda consulta, correspondiente a la información de los empleados. Para ello, pulse en el botón SQL y, luego, sobre un espacio en blanco de la pantalla de Data Model. Aparecerá la ventana SQL Query Statement (instrucción de consulta SQL). Pulse en Query Builder y seleccione las tablas CUSTOMER, EMPLOYEE, ITEM y SALESORDER. No es preciso seleccionar PRODUCT, ya que utilizaremos el grupo correspondiente a los productos de la primera consulta. Pulse en Include y luego en Close.

Si la base de datos dispone de integridad referencias, las tablas se combinarán automáticamente. En caso contrario, habrá que crear manualmente las siguientes combinaciones:

EMPLOYEE.EMPLOYEE-ID=CUSTOMER.REPID

CUSTOMER.CUSTID=SALESORDER.CUSTOMER-ID

SALESORDER.ORDERID=ITEM.ORDERID

Paso 12. selección de las columnas

Realice las siguientes operaciones en Query Builder:

- En la tabla EMPLOYEE, seleccione EMPLOYEE-ID, LAST NAME, FIRST NAME y MIDDLE INITIAL.

- En la tabla ITEM, seleccione QTY, ACTUALPRICE, ITEMTOT y PRODID, de manera que se disponga de algo con lo que combinar esta consulta con la anterior.

- Pulse en OK y, luego, en el botón OK de la ventana SQL Query Statement.

Paso 13: asignación *de alias a los nombres de columna*

Hemos de evitar los problemas derivados de la asignación automática de nombres que realiza Reports, para impedir la duplicación. Para ello, debemos editar el código de la consulta, visualizando las propiedades de Q2 y añadiendo los alias tras los nombres de columna siguientes:

- ACTUALPRICE recibe el alias EMP-SALE-PRICE

- QTY recibe el alias EMP-SALE-QTY

- PRODID recibe el alias EMP-PROD-ID

- ITEMTOT recibe el alias EMP-ITEMTOT

Paso 14: creación de los grupos de ruptura

En el modelo de datos, mantenga pulsada la tecla mayús y pulse en EMP SALE PRICE, EMP SALE-QTY y EMP PROD ID. Después, arrastre estos tres campos hacia abajo para crear un grupo diferente.

Paso 15. cambio de los nombres de grupos y *consultas*

Visualice la paleta de propiedades de cada uno de los grupos y cambie su nombre. Proceda, igualmente, a cambiar los nombres de las consultas empleando sus respectivas paletas de propiedades.

Paso 16. creación de columnas de resumen

Defina columnas de resumen para EMP - ITEMTOT y QTY, en el grupo EMP, igual que Report Wizard hizo en el grupo CUST. Para ello, pulse en el icono Summary Column y luego en el grupo EMP. Se creará una columna nueva con el nombre CS-1.

Pulse dos veces en la columna recién creada para acceder a su paleta de propiedades, cambie la propiedad *Name* a «EMP - PRICE». La propiedad *Source* ha de ser «EMP-ITEMTOT» y el parámetro *Reset At* ha de ser «EMP». Con ello haremos que la columna sume los precios para cada grupo de EMP. Siguiendo el mismo procedimiento, cree una segunda columna resumen en el grupo EMP y denomínela EMP-QTY, cambiando igualmente el contenido de sus propiedades *Source y Reset At* por «QTY» y «EMP», respectivamente.

Paso 17. vinculación de consultas

Vincule las dos consultas, pulsando en el icono Data Link (vínculo de datos), que es el icono que muestra dos clips enganchados o dos eslabones de cadena. Pulse en la

cabecera de la consulta Product Sales y arrastre hasta la cabecera de la consulta Emp Sales. Si la base de datos dispone de la integridad referencias apropiada, Report Builder seleccionará habitualmente la combinación adecuada. En este caso, lo que deseamos combinar es el campo PROD-ID de la consulta de ventas de producto con el campo EMP-PROD-ID de la consulta de ventas por empleado. Si no disponemos de la integridad referencias adecuada, poden»s vincular ambas consultas pulsando en el botón Data Link, después sobre PRODID, en la consulta de ventas de producto, y arrastrando el ratón hasta EMP 'PROD-ID, en la consulta de ventas por empleado. Puede ver la condición de combinación pulsando en la doble flecha que conecta ambas consultas, con lo que aparecerá la paleta de propiedades correspondiente.

Con esto hemos terminado de definir el modelo de datos, por lo que podemos comenzar a trabajar en el diseiño del informe. Es conveniente guardar en este punto el trabajo realizado hasta aquí.

Paso 18. creación del diseño del informe

Pulse en el botón de Layout Model. Pulse y arrastre hasta abarcar todo el diseño generado, con objeto de seleccionarlo. Pulse el botón Additional Default Layout y seleccione un área de aproximadamente, 20 cm de ancho por 5 cm de alto. Aparecerá una versión abreviada de Report Wizard, en la que deberá seleccionar Style y luego la opción Group Above.

Paso 19.- selección de grupos y modificación de configuraciones

En la ficha Groups, seleccione sólo PROD y CUST, pulsando en ambos casos en la opción Down, con lo que se indica a Reports que ambos se repetirán a lo largo de la página hacia abajo.

En la ficha Fields, seleccione todo excepto los campos Sum, ITEMTOT, ACTUAL-PRICE-PER-REPORT y SUM-QTY-PER-REPORT.

En la ficha Labels, se pueden cambiar las etiquetas del informe, si así se desea. Por ejemplo, cambie las etiquetas correspondientes a los campos de precio (ITEMTOT) y cantidad, de manera que presenten los mensajes «Total Amt.» y «Total Qty.», respectivamente. Pulse en Finish y ejecute el informe en Live Previewer.

Paso 20. cambio del diseño mediante Live Previewer

Cuando se imprime el informe, la zona dedicada a la descripción del producto es demasiado estrecha. Podemos corregir este defecto en Live Previewer pulsando en el campo y arrastrándolo hasta que tenga el ancho apropiado. También podemos modificar las etiquetas en Live Previewer, pulsando sobre ellas y alterando el texto.

Paso 21: adición de información al Informe

Cree un área para la información sobre los empleados pulsando en el espacio en blanco que aparece bajo F - PROD-DESC en Layout Model. Se seleccionará el marco grande situado alrededor suyo. No se puede arrastrar hacia abajo el borde inferior para ampliar el área, ya que existe otro marco rodeando al que deseamos aumentar. Oracle Report Builder posee una característica útil para tratar esta situación. Pulse en el botón Flex Mode. A partir de este momento, podemos pulsar en el asa inferior y cambia,r el tamaño del marco. Deslice el asa aproximadamente dos centímetros y medio hacia abajo, con objeto de proporcionar espacio suficiente para insertar la infortnación de empleados.

Mientras prepara la disposición de la infortnación de empleados, observe que ésta se encuentra almacenada en tres campos distintos: Last Name (apellido), First Name (nombre) y Middle Initial (inicial intermedia). La costumbre americana dicta que el formato utilizado para visualizar el nombre presente primero el apellido, seguido por una coma, el nombre, un espacio y la inicial intermedia, Por ejemplo, «Sniith, John A». Se puede escribir una función que se encargue de hacer esto.

Paso 22.- creación de una función

Pulse en el icono de Data Model. Cree una columna de fórmula pulsando en el icono

Forinula Column y, luego, en el grupo EMR Pulse dos veces en la nueva columna CF-l, para acceder a la paleta de propiedades. En ella, cambie el nombre de la columna por «Full Name», el tipo de datos *(Datatype)* a «Character» y el ancho a 220. En la propiedad Comments (comentarios), puede añadir un comentario descriptivo, como puede ser «Employee Full Name» (nombre completo del empleado), de cara a la documentación. A continuación, pulse en el icono PUSQL, para visualizar una ventana de edición de PUSQL. En ella se puede escribir cualquier programa PUSQL arbitrariamente complejo. En este caso, podemos añadir una función sencilla:

```
Function Name Formula return char is BEGIN
    RETURN :last-name 11 ',' 11 First-name 11 11 :middle-initial; EM;
```

Pulse en el botón Compile para asegurarse de que se ha introducido correctamente el código, con los dos puntos al comienzo del nombre de cada variable y un punto y coma al final de la línea.

Paso 23. adición de la información de empleados

Pulse en el botón Additional Default Layout y pulse y arrastre el área predeterminada adicional en el espacio en blanco debajo del campo F-PROD-DESC. Aparecerá el asistente Report Wizard.

Configure la propiedad Style como Tabular. En la ficha Groups, seleccione el grupo EMP utilizando el botón Down>. En la ficha Fields, seleccione NAME, MP-NAME, EMP-PRICE y EMP-QTY. En la ficha Labels, cambie el contenido de las etiquetas para NW-PRICE y EM-QTY por «Total Amt.» y «Total Qty.», igual que en el

grupo de clientes. Cambie la etiqueta para NAME a «Employee», ajustando su anchura a 15 para utilizar mejor el espacio disponible.

Paso 24: aplicación *de la* plantilla

Pulse en la ficha Template. El archivo de plantilla será por on-usion con el **que** se haya trabajado. Se puede elegir cualquier plantilla de entre las de la lista proporcionada, o bien es posible aplicar cualquier archivo de plantilla que el desarrollados haya creado. Las plantillas utilizadas para crear este informe se pueden encontrar en los sitios web de los autores.

Pulse en Finish. Ello dará lugar a la creación de la sección de empleados **del** infonne. Si no se alinean correctamente las secciones de clientes y empleados, puede pulsar y arrastrar para definir un recuadro que englobe toda la parte del infonne no ahneada, para ajustarla en caso necesario.

Consejo: Pulse ***y*** arrastre empezando en cualquier zona fuera de los marco& *con el fin de evitar seleccionar a uno de ellos.*

Puede que haga falta desactivar la función Snap to Grid (ajustar a la cuadffcula), en el menú View, para poder alinear la información de empleados con la de clientes.

Paso 25.- adición de totales

Los totales mostrados en la sección de clientes pueden ser también útiles en la sección de empleados del informe. La forma más fácil de hacer esto consiste en duplicar los tres campos (seleccionarlos todos y pulsar la combinación CTRL-D) en el modelo de diseño, arrastrándolos después hasta el lugar adecuado.

Paso 26. ajuste del diseño del informe

Pulse en Live Previewer para ver el estado actual del informe. Probablemente, deseará añadir algo de espacio entre productos. Identifique el marco repetitivo asociado con el grupo de productos e introduzca un cierto espacio libre entre cada marco. En el navegador de objetos, bajo el nodo Layout Model, localice el marco R-PROD. También puede seleccionarlo en Layout Editor. Contendrá el símbolo de marco repetitivo (que muestra una flecha). Pulse sobre el marco repetitivo dos veces para visualizar su paleta de propiedades. En la sección de marco repetitivo, cambie la propiedad *Vert. Space Berween Frames* (espacio vertical entre marcos) a «.2». Pulse INTRO para guardar el valor de la propiedad; Live Previewer mostrará el efecto imnediatamente. Cambie tarnbién la propiedad *Page Protect* (protección de página) del marco R-PROD a «Yes», de manera que la visualización de la información referente a un producto no quede a caballo entre dos páginas.

Paso 27.- finalización del informe

En Live Previewer, mantenga pulsada la tecla mayús y pulse en cada uno de los cuatro campos de cantidades. Pulse en el botón Currency (moneda), con objeto de aplicar a dichos campos un fonnato de moneda. Pulse dos veces en el botón Add decimal place (asñadir posición decimal), de manera que la información de tales campos aparezca en dólares y centavos. Si se ha seguido correctamente este tutorial. Dependiendo de la plantilla utilizada, puede que sean distintos los tipos de letra y los colores de los campos y de las etiquetas.

Conclusión

En el trabajo presentado anteriormente se demuestra la importancia de tener los conocimientos básicos del paquete Oracle.

También se da a conocer el componente Reports de Oracle Developer que es una de las herramientas de generación de informes más potente y con más prestaciones del mercado para brindarles un mejor conocimiento de lo que se puede hacer con este paquete que actualmente es necesario que todo informático conozca.

Además de trato de enfatizan en las funciones que Oracle posee dando ejemplos prácticos de cómo se utilizan para obtener mejores resultados al usar esas herramientas.

Bibliografía Consultada

Direcciones en Internet:

www.cybercursos.net

www.oracle.es

www.microsoft.com/latam/sql

Revistas Consultadas:

Revista COMPU MAGAZINE, Número 51, Octubre '92

Revista COMPU MAGAZINE, Número 50, Septiembre '92

(y diversos apuntes conseguidos de distintas publicaciones)

Libros Consultados:

Oracle 7 Manual de Referencia

Koch, George.

Osborne/McGraw-Hill

1994

Oracle Manual de Referencia.

Koch, George.

Osborne/McGraw-Hill.

1992

Mastering Oracle.

Cronin, Daniel.

Hayden Books.

1990

UNIVERSIDAD TECNOLÓGICA DE SANTIAGO

UTESA

RECINTO SANTO TOMÁS DE AQUINO

SUSTENTANTES:

Gilda Isabel Valera

Frank Joel Inoa

Elaine Altagracia Herrera

Profesor:

Ing. Luis Nuñez

Materia: Base de Datos

Trabajo enviado por:

Isabel Valera
isabelvalera55@hotmail.com

http://www.monografias.com/especiales/mecanografia

http://www.monografias.com/especiales/mecanografia

Bibliografía

Oracle 7 Manual de Referencia
Koch, George.
Osborne/McGraw-Hill
1994

Oracle Manual de Referencia.
Koch, George.
Osborne/McGraw-Hill.
1992

Mastering Oracle.
Cronin, Daniel.
Hayden Books.
1990