UNIVERSIDAD CATOLICA DE SANTIAGO DE GUAYAQUIL

Facultad de Ingeniería

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

TRABAJO DE SEMINARIO DE GRADUACIÓN

Previo a la obtención del título de:
INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA DE TRABAJO
Configuración de la Base de Datos
(ORACLE 10G)

REALIZADO POR:
Sr. Gabriel W. Guillén Buenaire
Sr. Diego Brito Izquierdo

DIRECTOR DEL TRABAJO DE GRADO:
Ing. Marcos Miranda

GUAYAQUIL – ECUADOR
2010

TRABAJO DE GRADO

Configuración de la Base de Datos Oracle

Presentado a la Facultad de Ingeniería, Carrera de Ingeniería en Sistemas Computacionales de la Universidad Católica de Guayaquil.

REALIZADO POR:
Sr. Gabriel Guillén Buenaire
Sr. Diego Brito Izquierdo

DIRECTOR DEL TRABAJO DE GRADO:
Ing. Marcos Miranda

Para dar cumplimiento con uno de los requisitos para optar por el título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Tribunal de Sustentación:

Ing. Edison Tóala **VOCAL**	Ing. Darwin Cercado **VOCAL**

Ing. Marcos Miranda
DIRECTOR DEL TRABAJO

Ing. Walter Mera **DECANO**	Ing. Vicente Gallardo **DIRECTOR DE CARRERA**

-ABSTRACT-

Con este trabajo se otorgara una fácil y amplia visualización del funcionamiento de la Base de Datos Oracle sobre la cual se trabajo sus limitaciones, configuración y resultados de la misma, permitiéndose observar los cambios posibles sobre ella optimizando su funcionamiento y desarrollo, poniendo a plan futuro para una más extensa aplicación una piloto para el estudio.

El trabajo fue desarrollo a base del aprendizaje obtenido en los módulos de seminario predecesores, para lo que se implemento la aplicación con pruebas conjuntas sobre bases ya tratadas y trabajadas, donde se incorporo investigaciones relacionadas la Oracle Data Base no detallado en los cursos.

Se reflejo que la mayoría de las Base de Datos de uso medio bajo no generan implicación de hallazgos a cambios para el estudio, debido a las configuración por default que se generan por los asistentes de instalación, se potencializa el aplicativo en estructura de almacenamiento de alto índice de transacciones, accesibilidad y de manejo lógico de datos, por complejidad y ordenamiento.

Epígrafe

"Agradezco a mi mama por apoyarme en la parte educativa desde el colegio de una manera anecdótica por ser padre y madre a la vez armándose de fuerza y voluntad para entregarme un nutriente primordial de la vida de cada individuo como es la educación y el conocimiento sin dejar a un lado lo sentimental… Gracias madre."

Gabriel Guillén

"Me gustaría dedicar esta Tesis a toda mi familia. Para mis padres Luis y Yolanda, por su comprensión y ayuda en momentos malos y menos malos. Me han enseñado a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento. Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor y sin pedir nunca nada a cambio.
Para mi mujer María del Carmen, a ella especialmente le dedico esta Tesis. Por su paciencia, por su comprensión, por su fuerza, por su amor, por ser tal y como es. Es la persona que más directamente ha sufrido las consecuencias del trabajo realizado. Agradezco al ente supremo que esta allá arriba, por darme esa dicha y tenerlos a mi lado."

Diego Brito

"Y una dedicatoria especial a nuestro compañero Plutarco Centeno el cual nos acompaño en los 2 primeros módulos con su conocimiento y apoyo como amigo; dedicación de estas dos personas que lo recordaremos SIEMPRE... Descansa en paz".

Tabla de contenido

Índice de Gráficos

INTRODUCCION

Dar a conocer el mundo de Oracle Data Base (Sistema Gestor de Base de Datos), sus alcances, ventajas, desventajas, potencialidades, debilidades (pocas), características, conceptos básicos, generalidades y costo de implementación de aplicaciones.

Todo esto es basado en la estructura de manejo de análisis sobre las base de datos de Oracle ya en funcionamiento, con una operatividad alta y trabajada.

Ya que nuestra sociedad, desde sus inicios a estado sujeta y basada en la información o datos para sus acciones, toma de decisiones, reacciones y en la forma en cómo la mantiene o se dispone en el momento, la misma que después de ser manipulada y/o utilizada por el usuario sirve como consecuencia para conclusiones, respuestas y re información.
Es así que todo, está basado en el tratamiento de la información, por lo que el estado de la misma para un correcto resultado deberá ser veraz, concreto, de fácil acceso y con cierto nivel de seguridad, para lo que se crearon la base de datos y sus aplicaciones de manejo y consultas.

El proyecto tiene una vida activa de 2 meses (el diseño e implementación), mientras que su vida pasiva es de 6 meses netamente en inducción y capacitación de la base de datos Oracle tanto en desarrollo, administración y seguridad (conocimientos obtenidos a través de los módulos de Oracle dictado por la Universidad), donde previo al penúltimo mes se determinó el tema a desarrollar.

El aplicativo se desarrollo con los componentes y valores internos de la Base de Datos por los que se examina la parte interna de operaciones y funciones. Después de su correcto análisis generan resultados ya sean estos adecuados o no adecuados, dando la posibilidad de transformar este valor adecuado en optimo y del no adecuado en optimo.

Donde empezaremos a dar conocer las generalidades de una base de datos Oracle, sus característica, parámetros y sus incidencias así como las consecuencias que puedan ser ocasionadas por una mala asignación en sus valores, los grupos por cómo fueron sectorizados para su estudio y análisis.

PRELIMINARES

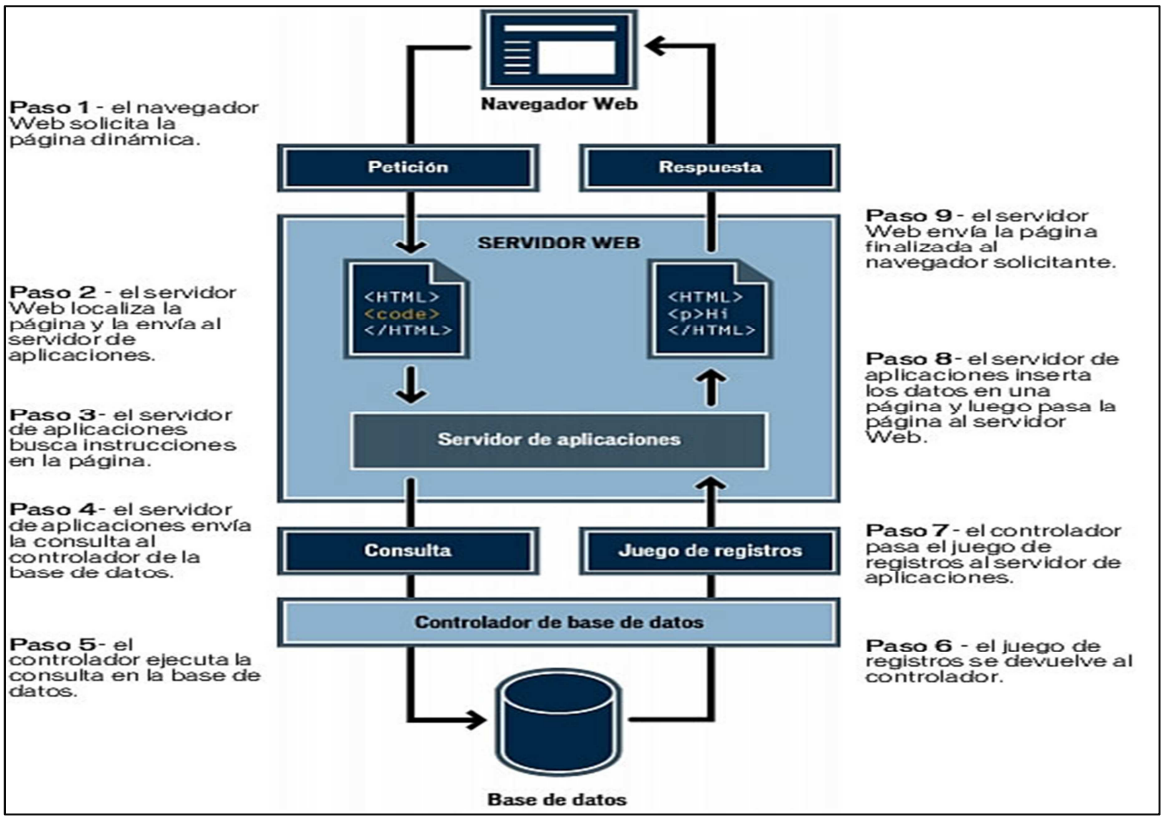
Que es una base de Datos

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Todo esto se basa en la solicitud de información ya sea este proveniente de un aplicativo de una terminal, aunque comúnmente en la actualidad estos requerimientos son iniciados desde una terminal web, donde el proceso de solicitud a la Base de Datos es como se detalla en el grafico siguiente:

Grafico Nº 01: Demanda de una Base de Datos por un Navegador Web



Fuente: http://livedocs.adobe.com/dreamweaver/8_es/using/gs_12_u8.htm

Reseña de Base de Datos Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporatiòn.

Oracle es una potente herramienta cliente/servidor para la gestión de Bases de Datos. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- soporte de transacciones,
- estabilidad,
- escalabilidad y
- Soporte multiplataforma.

Ha sido criticada por algunos especialistas la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Se procederá ahora en centrarnos en que es Oracle exactamente y cómo funciona la programación sobre éste. Oracle como antes se menciono se basa en la tecnología cliente/servidor, pues bien, para su utilización primero sería necesario la instalación de la herramienta servidor (Oracle 10g) y posteriormente se podrá analizar la base de datos desde otros equipos con herramientas de desarrollo como Forms Developer, que son las herramientas básicas de programación sobre Oracle.

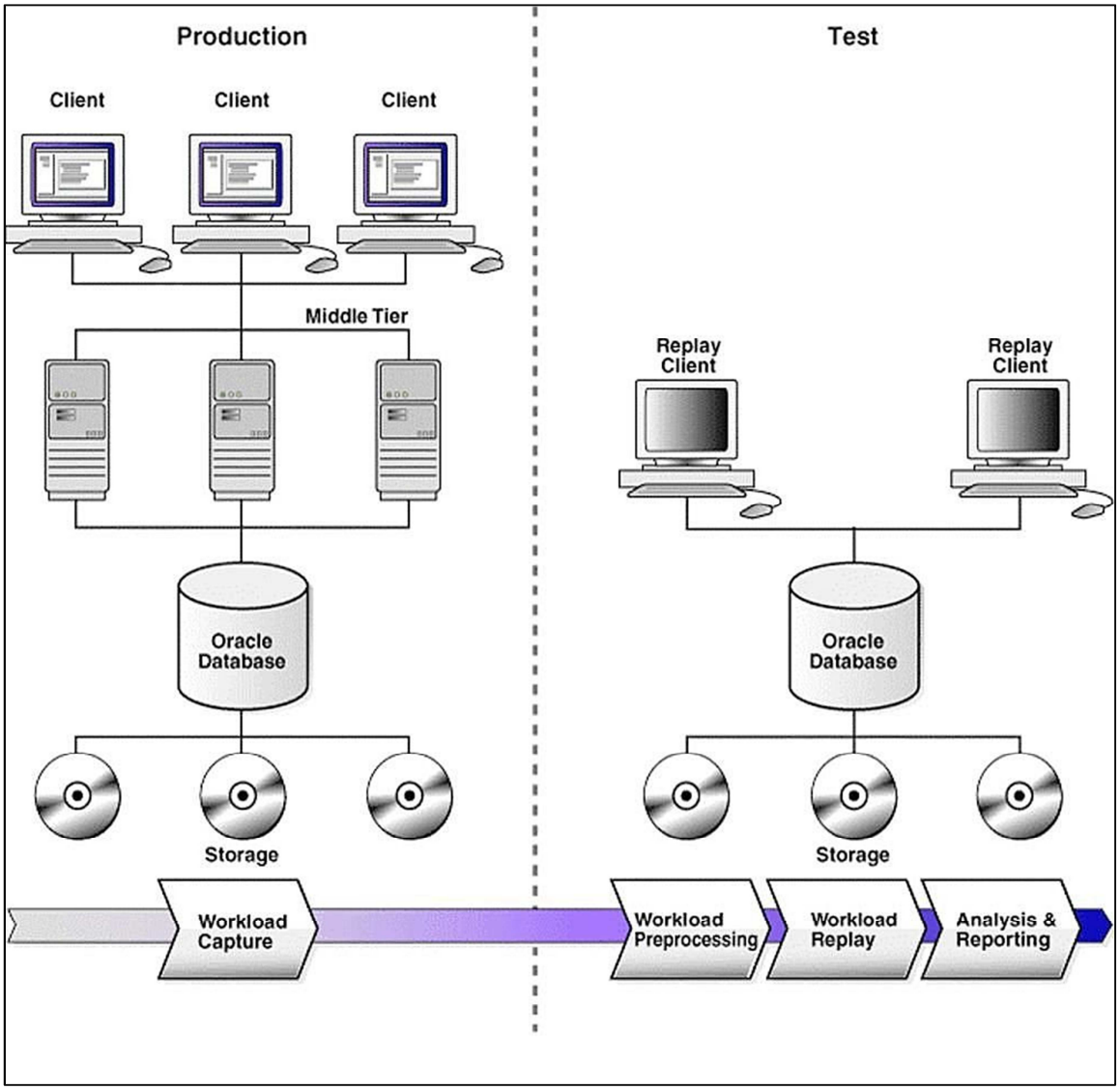
Para desarrollar en Oracle se utiliza PL/SQL un lenguaje de 5ª generación, bastante potente para tratar y gestionar la base de datos, también por norma general se suele utilizar SQL al crear un formulario.

Es posible lógicamente analizar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL.

El Forms Developer es una herramienta que le permite crear formularios de manera local, es decir, mediante esta herramienta se podrá crear formularios, compilarlos y ejecutarlos, pero si se quiere que los otros trabajen sobre este formulario se deberá copiarlo regularmente en una carpeta compartida para todos, de modo que, cuando quieran realizar un cambio, deberán copiarlo de dicha carpeta y luego volverlo a subir a la misma. Este sistema como se puede observar es bastante engorroso y poco fiable pues es bastante normal que las versiones se pierdan y se recorten con frecuencia. La principal ventaja de esta herramienta es que es bastante intuitiva y dispone de un modo que permite componer el formulario, tal y como lo se haría en Visual Basic o en Visual C.

Los problemas anteriores quedan totalmente resueltos con Forms Designer que es una herramienta que se conecta a la base de datos y por tanto se crea los formularios en ella, de esta manera todo el mundo se conecta mediante Designer a la aplicación que contiene todos los formularios y no hay problemas de diferentes versiones, esto es muy útil y perfecto para evitar estropear el trabajo de otros. Pero el principal y más notable problema es la falta de un entorno visual para diseñar el formulario, es decir, aparece una estructura como de árbol en la cual se inserta un formulario, a la vez dentro de éste se inserta bloques o módulos que son las estructuras que contendrán los elementos del formularios, que pueden estar basados en tablas o no.

Grafico Nº 02: Software de Base de Datos



Fuente: http://www.directindustry.es/prod/oracle/software-de-base-de-datos-27142-103502.html

OBJETIVOS

Se propone este trabajo con la finalidad de presentar un aplicativo que logra identificar las virtudes que tiene esta base de datos, los valores por defecto que vienen en la instalación y los optimos valores que deberían presentar estos parámetros, según un testeo que se le realice.

El aplicativo trabaja con un sin número de regulaciones de igual magnitud de los parámetros que contienen una Base de Datos Oracle de la Actualidad, el cual es monitorear y determinar de una manera completa la configuración sobre la cual la Base de Datos fue establecida y sobre la que es trabajada, teniendo así los argumentos y valores suficientes para realizar un análisis, en base a artificios matemáticos establecidos, en conjunto con el comportamiento de la base para poder permitirse generar situaciones aconsejables sobre mejoras que se establecen sobre su funcionamiento, desempeño y desarrollo que se producen, dando como resultado la posibilidad de realizar cambios en la mencionada configuración.

Esto permite re-estructurar la base pos-instalación en sus configuraciones iníciales (valores de los parámetros) de una manera óptima que no genere inconvenientes venideros.

Es pertinente identificar los objetivos específicos que se lograra con este software a desarrollar:

- Establecer los desperdicios de espacio que se pueden generar en asignaciones por default que se producen en las instalaciones en modo típico que se desarrollan por frecuencias por parte de los administradores.

- Reasignar los espacios no usados a secciones y parámetros que se encuentren sobrecargados o saturados por demandas excesivas del sistema o de la concurrencia en altas frecuencias de transacciones que se generan por los usuarios.

- Reorganizar los parámetros de la Base de Datos de Oracle que permiten rediseñar y mantener activa las virtudes de Rollback y flashback en base a anteriores desempeños y requerimientos.

- Determinar los bloqueos que se generan entre los usuarios y medirlos por concurrencia.

ALCANCE

El aplicativo se realizo para trabajar sobre la Base de Datos Oracle 10g, por sus características de almacenamiento, la elaboración de los comandos fue realizada con la herramienta ORACLE DEVELOPER SUIT, la presentación y visualización de las ventanas para el usuario fueron diseñadas en FORMS DEVELOPER 10G, y la generación de imágenes y datos estadísticos que son generados a través de REPORT 10g.

Trabaja sobre toda plataforma que soporte el lenguaje SQL y no es soportada por versiones anteriores a Oracle 9i, por la falta de ciertos parámetros y característica que si se encuentran en el Oracle 10g.

Los valores en los parámetros sobre los que depende una Base de Datos Oracle son determinadas por las funcionalidades de la base y pueden ser estas tomadas en cuenta a partir del archivo INIT.ora por su configuración, ya que existen dos tipos de parámetros los explícitos y los implícitos. Donde todos se generan a partir del archivo ya mencionado, que a la vez crea el SPFILE.ora y tiene las rutas de todos los demás parámetros. (Aproximadamente de 300 parámetros)

La aplicación se creó a base de tres selecciones de parámetros divididas en: Rendimiento, Control y Seguridad en cuanto a Bloqueos.

En la parte de Rendimiento encontramos los parámetros que están relacionados con las funciones y virtudes que da Oracle como:

UNDO que son los siguientes:

- UNDO_RETENTION
- UNDO_MANAGEMENT
- UNDO_BLOCK_PER_SEC

En la parte de Control encontramos los parámetros que están relacionados con las funciones y virtudes que da Oracle como:

FUNCTION OF SGA que son los siguientes:

- LARGE_POOL_SIZE
- SHARED_POOL_SIZE
- DB_CACHE_SIZE
- LOG_BUFFER

En la parte de Seguridad en cuanto a bloqueos encontramos los parámetros que están relacionados con las funciones y virtudes como:

- TIMED_STATISTICS
- MAX_DUMP_FILE_SIZE

Se determina este tipo de selección por la incidencia que mantiene cada parámetro individualmente con el desempeño y operación de la Base de Datos, donde cada valor representa los diferentes estados que mantiene el performance en la línea del tiempo donde este se maneja en periodos cortos de 10 segundos en cada proceso de análisis.

Así mismo se determinaran los bloqueos en lista que se generan en un determinado tiempo especifico, como en un lapso programado por el usuario, listándose por tiempo y niveles; con estos datos tanto de tiempo real como los históricos nos podrán proporcionar información estadística para determinar cuál es el usuario que monopoliza, ciertos objetos de la Base de Datos y a quienes Bloquea, también a los usuarios que por tener bajos privilegios se terminan colocando en la cola de acceso sin poder tomar al objeto requerido, no llegando a cumplir su cometido en el momento que se realiza el requerimiento por parte del usuario.

CONTENIDO

Una Base de Datos Oracle está almacenada físicamente en ficheros, y la correspondencia entre los ficheros y las tablas es posible gracias a las estructuras internas de la BASE DE DATOS, que permiten que diferentes tipos de datos estén almacenados físicamente separados. Está división lógica se hace gracias a los espacios de tablas, *tablespaces*.

1. Base de Datos Oracle.

1.1 Tablespaces

1.1.1 Los Espacios de Tablas, *Tablespaces*

Un espacio de tablas es una división lógica de la Base de Datos. Cada Base de Datos tiene al menos uno (SYSTEM). Un espacio de tablas puede pertenecer sólo a una Base de Datos. Los espacios de tablas se utilizan para mantener juntos los datos de usuarios o de aplicaciones para facilitar su mantenimiento o mejorar las prestaciones del sistema.

De esta manera, cuando se crea una tabla se debe indicar el espacio de tablas al que se destina. Por defecto se depositan en el espacio de tablas SYSTEM, que se crea por defecto. Este espacio de tablas es el que contiene el diccionario de datos, por lo que conviene reservarlo para el uso del servidor, y asignar las tablas de usuario a otro.

Lo razonable y aconsejable es que cada aplicación tenga su propio espacio de tablas.

Hay varias razones que justifican este modo de organización de las tablas en espacios de tablas:

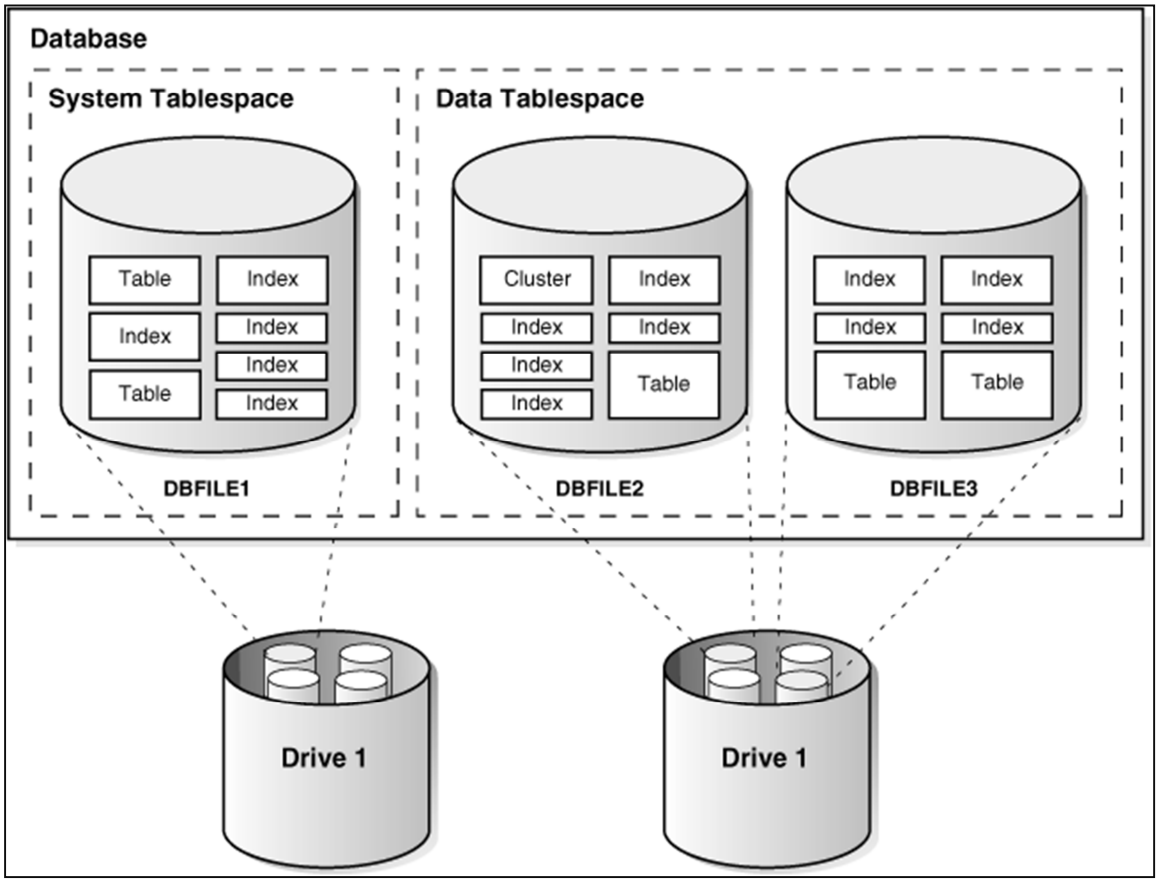
- Un espacio de tablas puede quedarse *offline* debido a un fallo de disco, permitiendo que el SGBD (Sistema Gestor de Base de Datos) continúe funcionando con el resto.
- Los espacios de tablas pueden estar montados sobre dispositivos ópticos si son de sólo lectura.

- Permiten distribuir a nivel lógico/físico los distintos objetos de las aplicaciones.
- Son una unidad lógica de almacenamiento, pueden usarse para aislar completamente los datos de diferentes aplicaciones.
- Oracle permite realizar operaciones de *backup/recovery* a nivel de espacio de tabla mientras la BASE DE DATOS sigue funcionando.

Cuando se crean se les asigna un espacio en disco que Oracle reserva inmediatamente, se utilice o no. Si esta expansión inicial se ha quedado pequeño Oracle puede gestionar el crecimiento dinámico de los ficheros sobre los que se asientan los espacios de tablas. Esto elimina la posibilidad de error en las aplicaciones por fallos de dimensionamiento inicial. Los parámetros de crecimiento del tamaño de los espacios de tablas se especifican en la creación de los mismos.

A continuación se mostrara un esquema de tablespaces para comprensión visual:

Grafico Nº 03: Tablespaces de Oracle Esquema



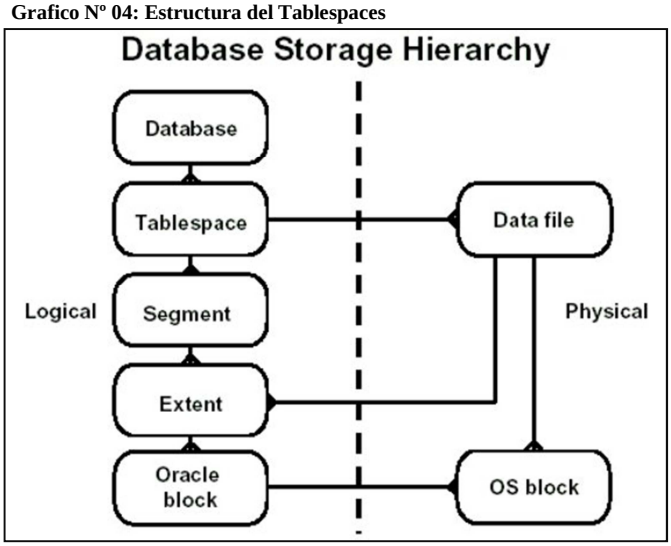
Fuente: www.mcs.csueastbay.edu/.../b14220/schema.htm

Se pueden ver los espacios de tablas definidos en nuestra Base de Datos con el comando SQL siguiente:

SQL> select * from user_tablespaces;

Dentro de cada espacio de tabla se pueden almacenar objetos de distinta naturaleza: tablas, índices, etc. Pero no se deben mezclar por de mas. Se necesita una manera de separarlos, y eso son los *segmentos, como se muestra a continuación*.

Grafico Nº 04: Estructura del Tablespaces



Fuente: http://pruebas.mitoledo.com/wik?ORACLE06DBA0105

Se pueden almacenar más de un segmento por espacio de tabla. Un segmento está contenido en su totalidad en un espacio de tabla. Un segmento está constituido por un conjunto de extensiones, que no son más que grupos de bloques de disco ORACLE contiguos. Cuando se borra un segmento, el espacio es devuelto al espacio de tabla.
Todos los datos de la Base de Datos están almacenados en segmentos. Y existen 4 tipos de segmentos: de datos: almacenan las tablas.

De índices: permiten un acceso rápido a los datos dependiendo de la cantidad de los mismos (árboles B). Las consultas que sólo referencian a columnas indexadas se resuelven en el índice. Establecen un control de unicidad (los índices son automáticos cuando se definen claves primarias). Cada índice ocupa un segmento independiente del segmento de datos y deberían estar en un espacio de tablas distinto al de los datos, para mejorar el rendimiento.

De *Rollback*: son objetos internos de la BASE DE DATOS que permiten efectuar la restauración de las transacciones no validadas asegurando la consistencia en lectura. La estructura de los registros de *rollback* es:

- Identificador de la transacción.
- Dirección del bloque donde está la tabla.
- Número de fila.
- Número de columna.
- Valor del dato antiguo (antes de ser modificado).

Son tan importantes que una Base de Datos no puede arrancar si no puede acceder al menos a un segmento de *rollback.* Si la Base de Datos tiene múltiples espacios de tablas, deben existir al menos dos segmentos de *rollback* y cada segmento de *rollback* debe tener al menos dos extensiones, reutilizables de manera cíclica. Esto segmentos son un objeto compartido de la Base de Datos, aunque se puede asignar un segmento de *rollback* particular a una transacción dada.

Temporales: son creados por Oracle para un uso temporal cuando debe realizar una ordenación que no le cabe en memoria, y en las operaciones: create index, order by, group by, distinct, union, intersect, minus. Son eliminados cuando la sentencia finaliza.

De *bootstrap*: Se crea en SYSTEM y contiene definiciones del diccionario para sus tablas, que se cargan al abrir la Base de Datos. No requiere ninguna acción por parte del Administrador de la Base de Datos. No cambia de tamaño.

La tabla que guarda la información de los segmentos de usuario es user_segments, y se puede visualizar la información sobre los segmentos con la sentencia SQL siguiente:

```
SQL> select * from user_segments;
```

1.1.2 Ficheros

Cada espacio de tablas se compone de uno o más ficheros en disco. Un fichero puede pertenecer sólo a un espacio de tablas. Los ficheros reciben un tamaño fijo en el momento de su creación, y cuando se necesita más espacio se deben añadir más ficheros a espacio de tablas.

Dividir los objetos de la BASE DE DATOS entre múltiples espacios de tablas permite que los objetos sean almacenados físicamente en discos separados, dependiendo de donde estén los ficheros sobre los que se asientan.

1.2 Instancias

Para permitir el acceso a los datos, Oracle utiliza un conjunto de procesos que son compartidos por todos los usuarios. Además, existen estructuras de memoria que son utilizadas para almacenar los datos más recientemente solicitados a la Base de Datos.

Una *instancia* de Base de Datos es el conjunto de estructuras de memoria y de procesos que acceden a los ficheros de datos.

Los parámetros que determinan el tamaño y composición de una instancia están almacenados en un fichero llamado init.ora. Este fichero es leído durante el arranque de la BASE DE DATOS y puede ser modificado por el DBA. Cualquier modificación de este fichero no tiene efecto hasta la siguiente vez que se arranque la BASE DE DATOS.

Las estructuras de la BASE DE DATOS Oracle pueden ser divididas en tres clases:

- Aquellas que son internas a la Base de Datos.
- Aquellas que son internas a las áreas de memoria (incluidas la memoria compartida y procesos).
- Aquellas que son externas a la Base de Datos.

1.3 Estructuras Internas de la BASE DE DATOS

Tablas y Columnas

Los datos son almacenados en la Base de Datos utilizando tablas. Cada tabla está compuesta por un número determinado de columnas.

Las tablas propiedad del usuario SYS son llamadas tablas del diccionario de datos. Proveen el catálogo del sistema que permite que la Base de Datos se gestione a sí misma.

Las tablas se pueden relacionar entre ellas a través de las columnas que las componen. La Base de Datos se puede utilizar para asegurar el cumplimiento de esas relaciones a través de la integridad referencial, que se concreta en las restricciones de tablas.

Restricciones de Tablas

Una tabla puede tener asociadas restricciones que deben cumplir todas las filas. Entre las restricciones que se pueden fijar algunas reciben nombres especiales.: *clave primaria*, *clave ajena*.

La clave primaria de una tabla está compuesta por las columnas que hacen a cada fila de la tabla una fila distinta.

La clave ajena se utiliza para especificar las relaciones entre tablas. De modo que un conjunto de columnas declaradas como clave ajena de una tabla deben tener valores tomados de la clave primaria de otra tabla.

Usuarios

Una cuenta de usuario no es una estructura física de la Base de Datos, pero está relacionada con los objetos de la Base de Datos: los usuarios poseen los objetos de la BASE DE DATOS. Existen dos usuarios especiales: SYS y SYSTEM. El usuario SYS posee las tablas del diccionario de datos; que almacenan información sobre el resto de las estructuras de la BASE DE DATOS. El usuario SYSTEM posee las vistas que permiten acceder a las tablas del diccionario, para el uso del resto de los usuarios de la BASE DE DATOS.

Todo objeto creado en la BASE DE DATOS se crea por un usuario, en un espacio de tablas y en un fichero de datos determinado. Toda cuenta de la BASE DE DATOS puede estar unida a una cuenta del Sistema Operativo, lo que permite a los usuarios acceder a la cuenta de la BASE DE DATOS sin dar la clave de acceso.

Cada usuario puede acceder a los objetos que posea o a aquellos sobre los que tenga derecho de acceso.

Esquemas

El conjunto de objetos de un usuario es conocido como esquema.

Índices

Un índice es una estructura de la BASE DE DATOS utilizada para agilizar el acceso a una fila de una tabla. Cada fila tiene un identificador de fila, ROWID, que determina el fichero, bloque y fila dentro del bloque donde está almacenada la fila.

Cada entrada del índice consiste en un valor clave y una ROWID. Cada una de estas entradas se almacena en un árbol B^+.

Los índices se crean automáticamente cuando se define una restricción UNIQUE o PRIMARY KEY.

Clusters

Las tablas que son accedidas juntas frecuentemente pueden ser almacenadas juntas. Para ello se crea un *cluster*. De este modo se minimiza el número de E/S.

Las columnas que relacionan las tablas de un *cluster* se llaman clave del *cluster*.

Vistas

Conceptualmente, una vista puede considerarse como una máscara que se extiende sobre una o más tablas, de modo que cada columna de la vista se corresponde con una o más columnas de las tablas subyacentes. Cuando se consulta una vista, esta traspasa la consulta a las tablas sobre las que se asienta. Las vistas no se pueden indexar.

Las vistas no generan almacenamiento de datos, y sus definiciones se almacenan en el diccionario de datos.

Secuencias

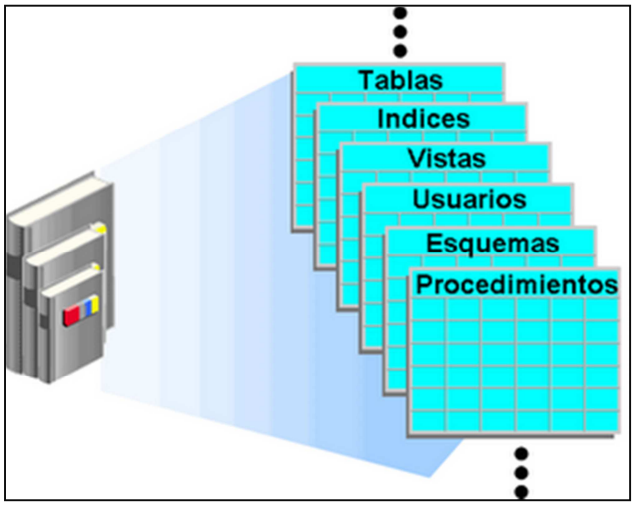
Las definiciones de secuencias se almacenan en el diccionario de datos. Son mecanismos para obtener listas de números secuenciales.

Procedimientos y Funciones

Un procedimiento es un bloque de código PL/SQL, que se almacena en el diccionario de datos y que es llamado por las aplicaciones. Se pueden utilizar para implementar seguridad, no dando acceso directamente a determinadas tablas sino es a través de procedimientos que acceden a esas tablas. Cuando se ejecuta un procedimiento se ejecuta con los privilegios del propietario del procedimiento. La diferencia entre un procedimiento y una función es que ésta última puede devolver valores.

Todo esto se maneja secuencialmente como se muestra a continuación:

Grafico Nº 05: Estructura Interna de la Base de Datos



Fuente: mioracle.blogspot.com/2008_02_01_archive.html

Paquetes, *Packages*

Se utilizan para agrupar procedimientos y funciones. Los elementos dentro de los paquetes pueden ser públicos o privados. Los públicos pueden ser llamados por los usuarios, los privados están ocultos a los usuarios y son llamados por otros procedimientos.

Disparadores, *Triggers*

Son procedimientos que son ejecutados cuando se procede un determinado evento en la BASE DE DATOS. Se pueden utilizar para mejorar y reforzar la integridad y la seguridad de la BASE DE DATOS.

Sinónimos

Para identificar completamente un objeto dentro de una BASE DE DATOS se necesita especificar el nombre de la máquina, el nombre del servidor, el nombre del propietario y el nombre del objeto. Para hacer transparente todo esto al usuario se pueden utilizar los sinónimos. Éstos apuntarán a los objetos y si el objeto cambia de lugar o propietario, sólo habrá que modificar el sinónimo.

Existen sinónimos públicos y privados. Los públicos son conocidos por todos los usuarios de una BASE DE DATOS. Los privados son locales a un usuario.

Privilegios y Roles

Para que un objeto pueda ser accedido por un usuario debe de tener otorgado ese privilegio. Ejemplos de privilegios son INSERT, SELECT, UPDATE, EXECUTE, etc.

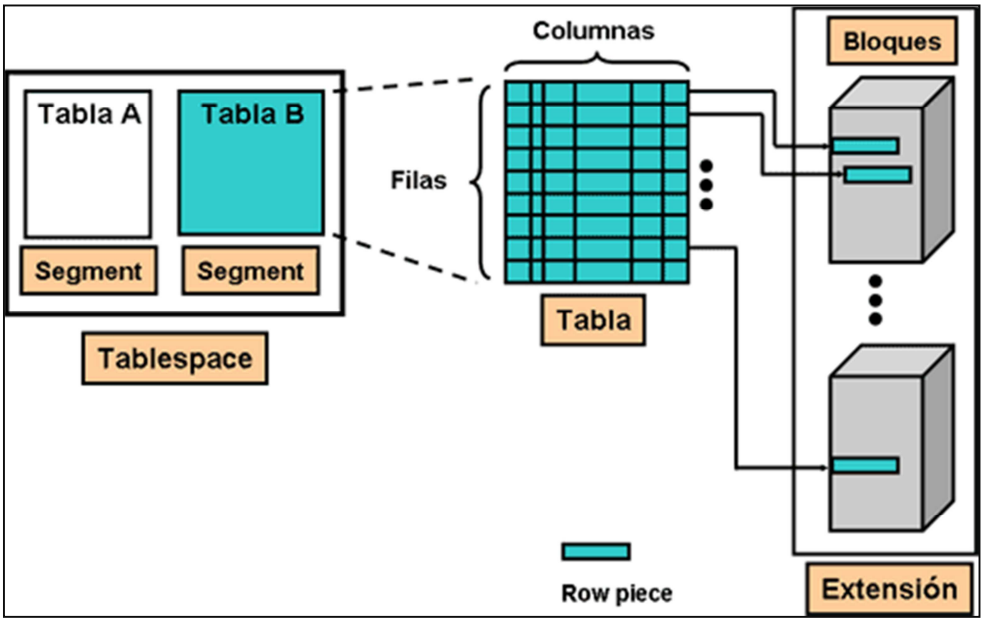
Los roles son grupos de privilegios que pueden ser utilizados para facilitar la gestión de los privilegios. Los privilegios se pueden otorgar a un rol, y los roles pueden ser otorgados a múltiples usuarios.

Segmentos, Extensiones y Bloques

Los segmentos son los equivalentes físicos de los objetos que almacenan datos. El uso efectivo de los segmentos requiere que el DBA conozca los objetos que utilizan una aplicación, cómo los datos son introducidos en esos objetos y el modo en que serán recuperados.

Como los segmentos son entidades físicas, deben estar asignados a espacios de tablas en la BASE DE DATOS y estarán localizados en uno de los ficheros de datos del espacio de tablas. Un segmento está constituido por secciones llamadas extensiones, que son conjuntos contiguos de bloques Oracle. Una vez que una extensión existente en un segmento no puede almacenar más datos, el segmento obtendrá del espacio de tabla otra extensión. Este proceso de extensión continuará hasta que no quede más espacio disponible en los ficheros del espacio de tablas, o hasta que se alcance un número máximo de extensiones por segmento.

Grafico Nº 06: Estructura de almacenamiento



Fuente: http://pruebas.mitoledo.com/wiki/kwiki.cgi?ORACLE06DBA0105

Segmento de *Rollback*

Para mantener la consistencia en lectura y permitir deshacer las transacciones, Oracle debe tener un mecanismo para reconstruir la imagen previa a una transacción incompleta. Oracle utiliza los segmentos de *rollback* para esto.

Los segmentos de *rollback* pueden crecer tanto como sea necesario para soportar las transacciones.

1.4 Estructuras de Memoria Internas

Oracle mantiene dos estructuras principales de memoria: el Área Global de Programa, *Program Global Area, PGA*; y el Área Global del Sistema, *System Global Area* o también *Shared Global Area, SGA.*

El PGA es la zona de memoria de cada proceso Oracle. No está compartida y contiene datos e información de control de un único proceso.

El SGA es la zona de memoria en la que la BASE DE DATOS Oracle guarda información sobre su estado. Esta estructura de memoria está disponible para todos los procesos, por eso se dice que está compartida.

1.4.1 Área Global del Sistema, SGA

Sirve para facilitar la transferencia de información entre usuarios y también almacena la información estructural de la BASE DE DATOS más frecuentemente requerida.

La SGA se divide en varias partes:

- ***Buffers* de BASE DE DATOS, *Database Buffer Cache***

  Es el caché que almacena los bloques de datos leídos de los segmentos de datos de la BASE DE DATOS, tales como tablas, índices y Clusters. Los bloques modificados se llamas *bloques sucios*. El tamaño de buffer caché se fija por el parámetro DB_BLOCK_BUFFERS del fichero init.ora.

  Como el tamaño del buffer suele ser pequeño para almacenar todos los bloques de datos leídos, su gestión se hace mediante el algoritmo LRU.

- ***Buffer Redo Log***

  Los registros *Redo* describen los cambios realizados en la BASE DE DATOS y son escritos en los ficheros *redo log* para que puedan ser utilizados en las operaciones de recuperación hacia adelante, *roll-forward*, durante las recuperaciones de la BASE DE DATOS. Pero antes de ser escritos en los ficheros *redo log* son escritos en un caché de la SGA llamado *redo log buffer*. El servidor escribe periódicamente los registros *redo log* en los ficheros *redo log*.

  El tamaño del buffer redo log se fija por el parámetro LOG_BUFFER.

Área de SQL Compartido, *Shared SQL Pool*

En esta zona se encuentran las sentencias SQL que han sido analizadas. El análisis sintáctico de las sentencias SQL lleva su tiempo y Oracle mantiene las estructuras asociadas a cada sentencia SQL analizada durante el tiempo que pueda para ver si puede reutilizar. Antes de analizar una sentencia SQL, Oracle mira a ver si encuentra otra sentencia exactamente igual en la zona de SQL compartido. Si es así, no la analiza y pasa directamente a ejecutar la que mantiene en memoria. De esta manera se premia la uniformidad en la programación de las aplicaciones. La igualdad se entiende que es lexicográfica, espacios en blanco y variables incluidas. El contenido de la zona de SQL compartido es:

- ✓ Plan de ejecución de la sentencia SQL.
- ✓ Texto de la sentencia.
- ✓ Lista de objetos referenciados.

Los pasos de procesamiento de cada petición de análisis de una sentencia SQL son:

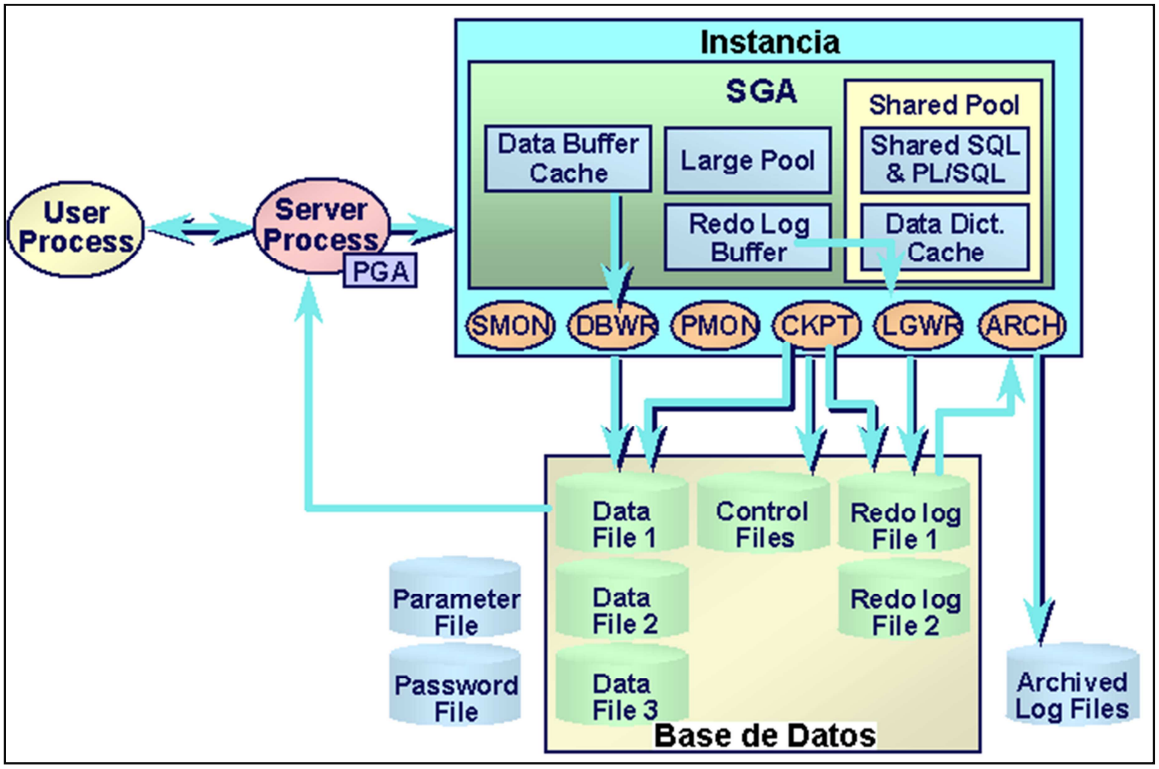
- ✓ Comprobar si la sentencia se encuentra en el área compartida.
- ✓ Comprobar si los objetos referenciados son los mismos.
- ✓ Comprobar si el usuario tiene acceso a los objetos referenciados.

Si no, la sentencia es nueva, se analiza y los datos de análisis se almacenan en la zona de SQL compartida.

También se almacena en la zona de SQL compartido el *caché del diccionario*. La información sobre los objetos de la BASE DE DATOS se encuentra almacenada en las tablas del diccionario. Cuando esta información se necesita, se leen las tablas del diccionario y su información se guarda en el caché del diccionario de la SGA.

Este caché también se administra mediante el algoritmo LRU. El tamaño del caché está gestionado internamente por el servidor, pero es parte del *shared pool*, cuyo tamaño viene determinado por el parámetro SHARED_POOL_SIZE.

Grafico Nº 07: SGA



Fuente: http://www.ibm.com/developerworks/data/library/techarticle/dm-0401gupta/

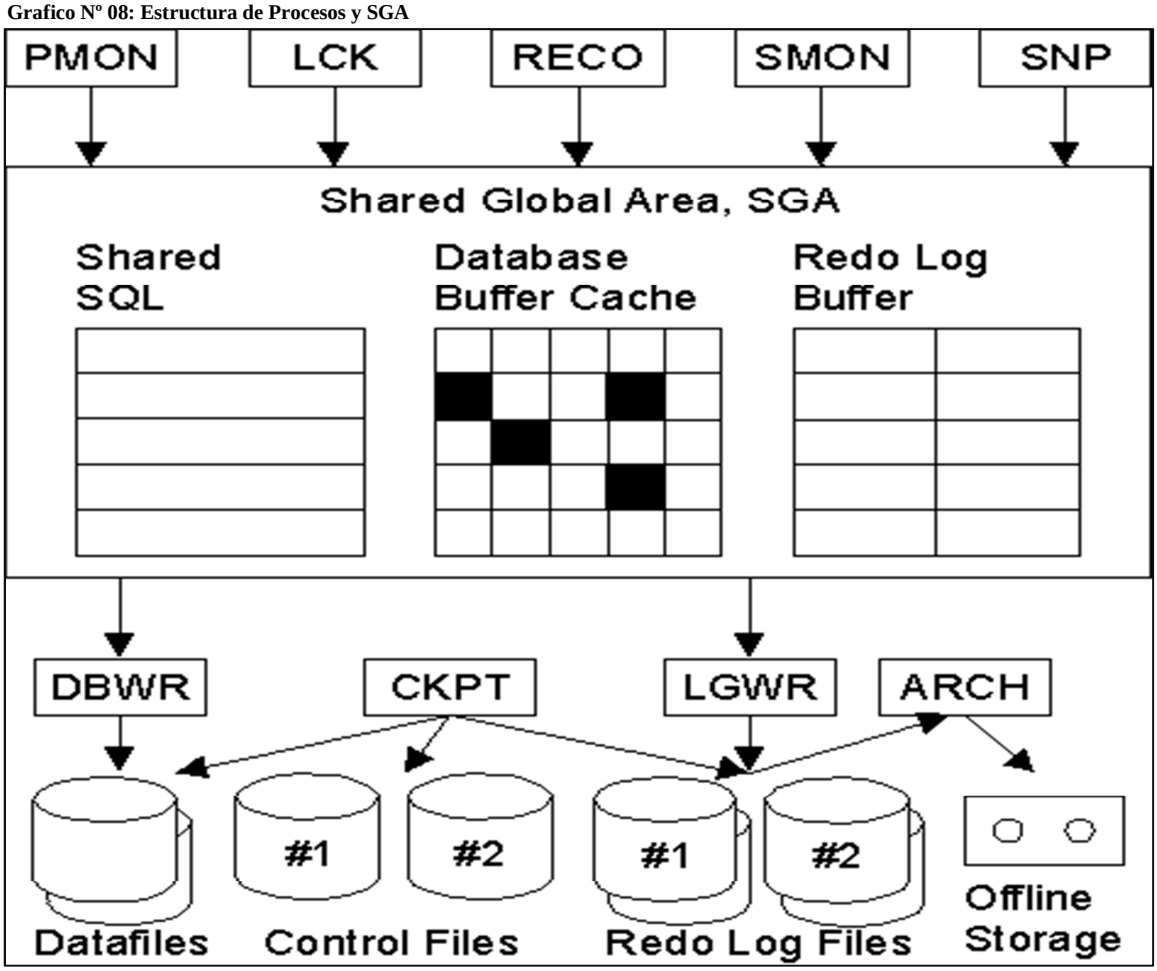
1.4.2 Área Global de Programa

El *Program Global Area* es un área de memoria utilizada por un proceso Oracle. Esta zona de memoria no se puede compartir y es la memoria privada de cada proceso servidor. En esta memoria cada proceso almacena información que sólo es necesaria para su propio funcionamiento como por ejemplo sus variables globales, el estado actual de cada cursor (SQL) que se ejecuta... etc...

1.5 Estructuras de Proceso

El servidor se vale de una serie de procesos que son el enlace entre las estructuras físicas y de memoria. A continuación se describen cada proceso y el papel que juega en la gestión de la BASE DE DATOS. Todo esto se puede ver en la siguiente figura.

Grafico Nº 08: Estructura de Procesos y SGA



Fuente: http://blogs.oracle.com/stevenChan/2009/04/

System Monitor, *SMON*

El SMON es el supervisor del sistema y se encarga de todas las recuperaciones que sean necesarias durante el arranque. Esto puede ser necesario si la BASE DE DATOS se paró inesperadamente por fallo físico, lógico u otras causas. Este proceso realiza la recuperación de la instancia de BASE DE DATOS a partir de los ficheros *redo log*. Además limpia los segmentos temporales no utilizados y compacta los huecos libres contiguos en los ficheros de datos. Este proceso se despierta regularmente para comprobar si debe intervenir.

Process Monitor, *PMON*

Este proceso restaura las transacciones no validadas de los procesos de usuario que abortan, liberando los bloqueos y los recursos de la SGA. Asume la identidad del usuario que ha fallado, liberando todos los recursos de la BASE DE DATOS que estuviera utilizando, y anula la transacción cancelada. Este proceso se despierta regularmente para comprobar si su intervención es necesaria.

Database Writer, *DBWR*

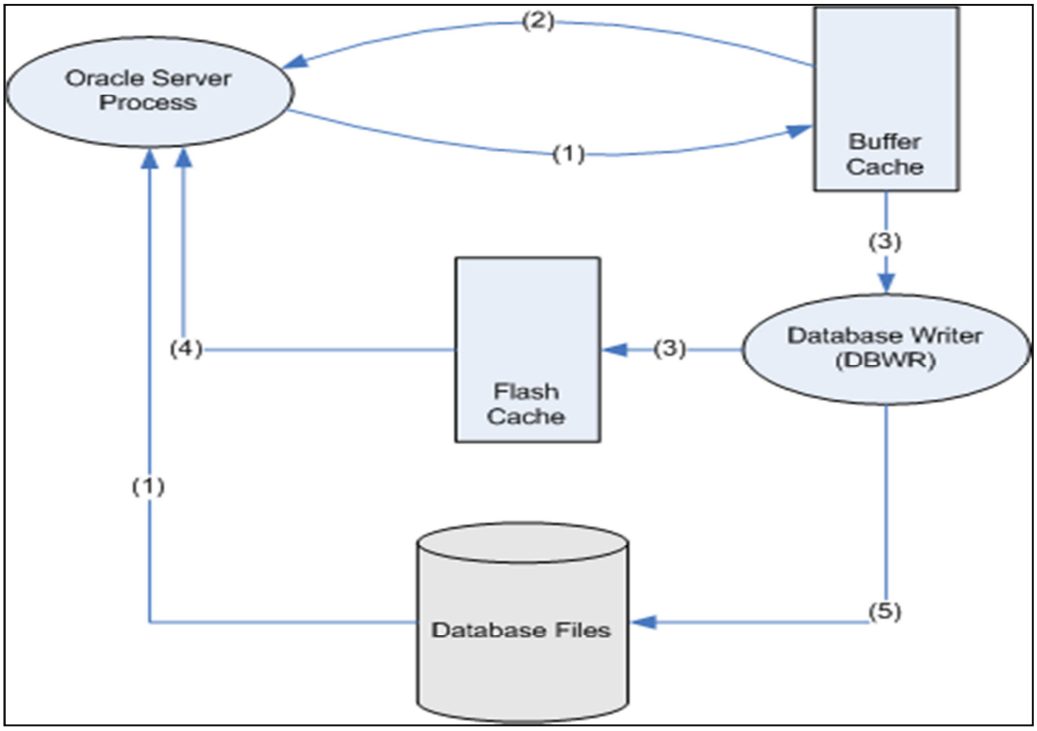
El proceso DBWR es el responsable de gestionar el contenido de los buffers de datos y del caché del diccionario. Él lee los bloques de los ficheros de datos y los almacena en la SGA. Luego escribe en los ficheros de datos los bloques cuyo contenido ha variado. La escritura de los bloques a disco es diferida buscando mejorar la eficiencia de la E/S.

Es el único proceso que puede escribir en la BASE DE DATOS. Esto asegura la integridad. Se encarga de escribir los bloques de datos modificados por las transacciones, tomando la información del *buffer* de la BASE DE DATOS cuando se valida una transacción. Cada validación no se lleva a la BASE DE DATOS física de manera inmediata sino que los bloques de la BASE DE DATOS modificados se vuelcan a los ficheros de datos periódicamente o cuando sucede algún *checkpoint* o punto de sincronización, esto es una grabación *diferida porque:*

- Los bloques del *buffer* de la BASE DE DATOS (bloques del segmento de *rollback* y bloques de datos) menos recientemente utilizados son volcados en el disco continuamente para dejar sitio a los nuevos bloques.
- El bloque del segmento de *rollback* se escribe SIEMPRE antes que el correspondiente bloque de datos.
- Múltiples transacciones pueden solapar los cambios en un sólo bloque antes de escribirlo en el disco.

Mientras, para que se mantenga la integridad y coherencia de la BASE DE DATOS, todas las operaciones se guardan en los ficheros de *redo log*. El proceso de escritura es asíncrono y puede realizar grabaciones multibloque para aumentar la velocidad.

Grafico Nº 09: Secuencia Iterativa de la Base de Datos



Fuente: http://www.napolifirewall.com/OR

Log Writer, *LGWR*

El proceso LGWR es el encargado de escribir los registros *redo log* en los ficheros *redo log*. Los registros *redo log* siempre contienen el estado más reciente de la BASE DE DATOS, ya que puede que el DBWR deba esperar para escribir los bloques modificados desde el buffer de datos a los ficheros de datos.

Conviene tener en cuenta que el LGWR es el único proceso que escribe en los ficheros de *redo log* y el único que lee directamente los buffers de *redo log* durante el funcionamiento normal de la BASE DE DATOS.

Coloca la información de los *redo log buffers* en los ficheros de *redo log*. Los *redo log buffers* almacenan una copia de las transacciones que se llevan a cabo en la BASE DE DATOS. Esto se produce una a cada validación de transacción, y antes de que se comunique al proceso que todo ha ido bien, cuando se llena el grupo de *buffers* de *redo log*, cuando el DBWR escribe *buffers* de datos modificados en disco.

Así, aunque los ficheros de Data Base no se actualicen en ese instante con los buffers de BASE DE DATOS, la operación queda guardada y se puede reproducir. Oracle no tiene que consumir sus recursos escribiendo el resultado de las modificaciones de los datos en los archivos de datos de manera inmediata. Esto se hace porque los registros de *redo log* casi siempre tendrán un tamaño menor que los bloques afectados por las modificaciones de una transacción, y por lo tanto el tiempo que emplea en guardarlos es menor que el que emplearía en almacenar los bloques sucios resultado de una transacción; que ya serán trasladados a los ficheros por el DBWR. El LGWR es un proceso único, para asegurar la integridad. Es asíncrono. Además permite las grabaciones multibloque.

Checkpoint, *CKPT*

Este proceso escribe en los ficheros de control los *checkpoints*. Estos puntos de sincronización son referencias al estado coherente de todos los ficheros de la BASE DE DATOS en un instante determinado, en un punto de sincronización. Esto significa que los bloques sucios de la BASE DE DATOS se vuelcan a los ficheros de BASE DE DATOS, asegurándose de que todos los bloques de datos modificados desde el último *checkpoint* se escriben realmente en los ficheros de datos y no sólo en los ficheros *redo log*; y que los ficheros de *redo log* también almacenan los registros de *redo log* hasta este instante. La secuencia de puntos de control se almacena en los ficheros de datos, *redo log* y control. Los *checkpoints* se producen cuando:

- Un espacio de tabla se pone inactivo, *offline.*
- Se llena el fichero de *redo log* activo
- Se para la BASE DE DATOS
- El número de bloques escritos en el *redo log* desde el último *checkpoint* alcanza el límite definido en el parámetro LOG_CHECKPOINT_INTERVAL
- Cuando transcurra el número de segundos indicado por el parámetro LOG_CHECKPOINT_TIMEOUT desde el último *checkpoint*.
- Está activo si el parámetro CHECKPOINT_PROCESS tiene un valor verdadero.

Archiver, *ARCH*

El proceso archivador tiene que ver con los ficheros *redo log*. Por defecto, estos ficheros se reutilizan de manera cíclica de modo que se van perdiendo los registros *redo log* que tienen una cierta antigüedad. Cuando la BASE DE DATOS se ejecuta en modo ARCHIVELOG, antes de reutilizar un fichero *redo log* realiza una copia del mismo. De esta manera se mantiene una copia de todos los registros *redo log* por si fueran necesarios para una recuperación. Este es el trabajo del proceso archivador.

Recoverer, *RECO*

El proceso de recuperación está asociado al servidor distribuido. En un servidor distribuido los datos se encuentran repartidos en varias localizaciones físicas, y estas se han de mantener sincronizadas. Cuando una transacción distribuida se lleva a cabo puede que problemas en la red de comunicación haga que una de las localizaciones no aplique las modificaciones debidas. Esta transacción *dudosa* debe ser resuelta de algún modo, y esa es la tarea del proceso recuperador. Está activo si el parámetro DISTRIBUTED_TRANSACTIONS tiene un valor distinto de 0.

Lock, *LCK*

El proceso de bloqueo está asociado al servidor en paralelo.

1.6 Estructuras Externas

Por estructuras externas se entienden los ficheros que utiliza el servidor de BASE DE DATOS, de los cuales ya se han ido contando algunos aspectos, y otros se han ido intuyendo. Estos ficheros guardan información tanto de los datos almacenados en la BASE DE DATOS como la necesaria para gobernar la propia BASE DE DATOS.

Ficheros de la BASE DE DATOS

En estos ficheros reside la información de la BASE DE DATOS. Solo son modificados por el DBWR. A ellos se vuelcan los bloques sucios de la SGA cuando se hace una validación o cuando sucede un *checkpoints*. Las validaciones de las transacciones no producen un volcado inmediato, sino lo que se conoce por un *commit diferido*. Toda actualización se guarda en los ficheros de *redo log*, y se lleva a la BASE DE DATOS física cuando tenemos una buena cantidad de bloques que justifiquen una operación de E/S. Almacenan los segmentos (datos, índices, *Rollback*) de la BASE DE DATOS. Están divididos en bloques (Bloque Oracle = c * Bloque SO), cada uno de los cuales se corresponde con un *buffer* del *buffer cache* de la SGA. En el bloque de cabecera no se guardan datos de usuario, sino la marca de tiempo del último *checkpoints* realizado sobre el fichero.

Ficheros *redo log*

En ellos se graba toda operación que se efectúe en la BASE DE DATOS y sirven de salvaguarda de la misma. Tiene que haber por lo menos 2, uno de ellos debe estar activo, *online*, y se escribe en ellos de forma cíclica. Existe la posibilidad de almacenar los distintos ficheros de *redo log* en el tiempo mediante el modo ARCHIVER. Así, se puede guardar toda la evolución de la BASE DE DATOS desde un punto dado del tiempo.

Una opción es la utilización de archivos *redo log* multiplexados porque:

- Permite al LGWR escribir simultáneamente la misma información en múltiples archivos *redo log*.
- Se utiliza para protegerse contra fallos en el disco.
- Da una alta disponibilidad a los archivos *redo log* activos u *online*.

Esto se hace definiendo el número de *grupos* y de *miembros* de archivos *redo log* que van a funcionar en paralelo:

- Grupos: funcionan como ficheros *redo log* normales, uno de ellos está activo y el resto espera su turno. Su nombre lleva incorporado una numeración. Deben contener todos los mismos números de miembros.
- Miembros: cada escritura de un registro *redo log* se lleva a cabo en todos los miembros del grupo activo en ese momento. Los miembros deben: Tener el mismo tamaño y el mismo número de secuencia. Deben tener nombres similares y estar en diferentes discos para proteger contra fallos de una manera efectiva.

Cuando se produce algún fallo en los ficheros de *redo log* o en el proceso LGWR es por:

- Si la escritura en un fichero *redo log* falla pero el LGWR puede escribir al menos en uno de los miembros del grupo, lo hace , ignorando el fichero inaccesible y registrando un fallo en un fichero de traza o alerta.
- Si el siguiente grupo no ha sido archivado (modo ARCHIVELOG) antes del cambio de grupo que lo pone activo, ORACLE espera hasta que se produzca el archivado.
- Si fallan todos los miembros de un grupo mientras el LGWR trata de escribir, la instancia se para y necesita recuperación al arrancar.

Se pueden visualizar los nombres y estado de los ficheros de *redo log*:

```
SVRMGR> select group#, status, substr(member,1,60) from v$logfile;
```

También se pueden visualizar estadísticas de los ficheros *redo log*:

```
SVRMGR> select group#, sequence#, bytes, members, archived,
2 status, first_change#, first_time from v$logfile;
```

Ficheros de control

Mantienen la información física de todos los ficheros que forman la BASE DE DATOS, camino incluido; así como el estado actual de la BASE DE DATOS. Son utilizados para mantener la consistencia interna y guiar las operaciones de recuperación. Son imprescindibles para que la BASE DE DATOS se pueda arrancar. Contienen:

- Información de arranque y parada de la BASE DE DATOS.
- Nombres de los archivos de la BASE DE DATOS y *redo log*.
- Información sobre los *checkpoints.*
- Fecha de creación y nombre de la BASE DE DATOS.
- Estado *online* y *offline* de los archivos.

Debe haber múltiples copias en distintos discos, mínimo dos, para protegerlos de los fallos de disco. La lista de los ficheros de control se encuentra en el parámetro CONTROL_FILES, que debe modificarse con la BASE DE DATOS parada.

Se puede componer una sentencia SQL que nos muestre todos los ficheros asociados a una BASE DE DATOS. Este es:

```
SQL> select 'control' tipo, substr(name,1,70) nombre from v$controlfile
2  union all
3  select 'datos' tipo, substr(name,1,70) nombre from v$datafile
4  union all
5  select 'redo log' tipo, substr(name,1,70) nombre from v$logfile
6  /
```

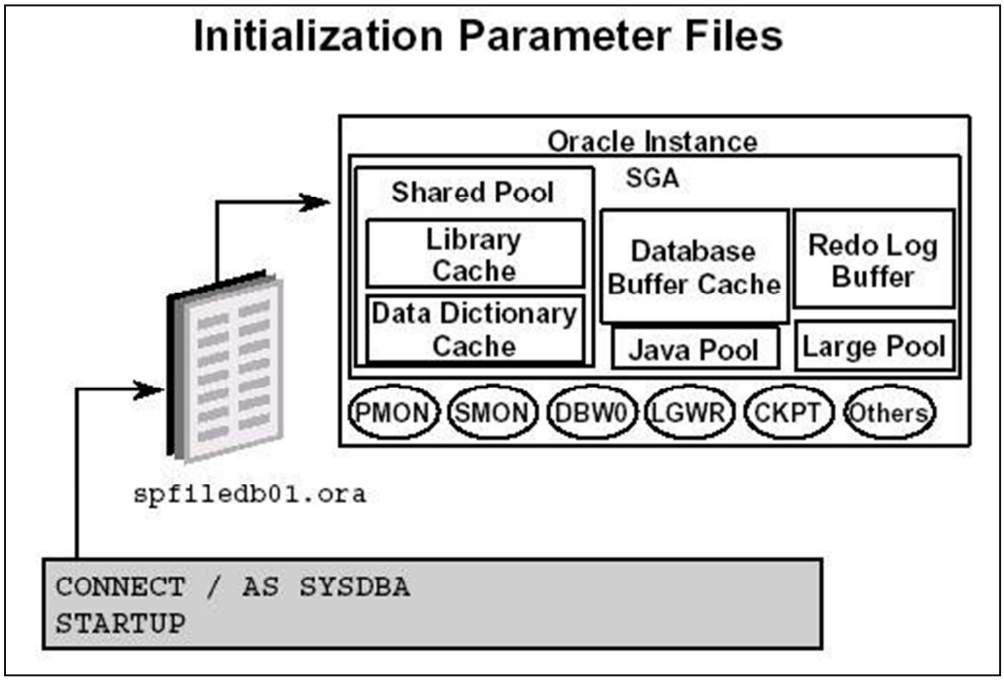
Hasta aquí los tipos de ficheros que se suelen considerar fundamentales en la arquitectura del SGBD de Oracle. Pero existen otros ficheros, que aunque no forman parte de la arquitectura Oracle resultan importantes en el uso del SGBASE DE DATOS.

El Fichero INIT.ORA

Como parte de la distribución software, Oracle provee de un fichero de parámetros de inicialización llamado init.ora. Este fichero contiene los parámetros del sistema Oracle y debe ser utilizado por el DBA para configurar el SGDB y adecuarlo a una determinada explotación. Oracle lee este fichero durante el proceso de arranque para determinar el tamaño de la SGA y encontrar los ficheros de control, entre otros menesteres.

Como el fichero init.ora es fundamental para el arranque de la BASE DE DATOS, debería ser copiado frecuentemente para protegerlo de posibles pérdidas.

Grafico Nº 10: Valores Iníciales de Parámetros



Fuente: http://www.siue.edu/~dbock/cmis565/module3-database_startup.htm

Ficheros de Traza

Oracle crea ficheros de texto llamados de traza para ayudar en la diagnosis de problemas y en el ajuste del SGBASE DE DATOS. Cada proceso del servidor escribe en un fichero de traza asociado cuando es necesario. Los procesos de usuarios también pueden tener asociados ficheros de traza. La situación de estos ficheros de traza del sistema se especifica por el parámetro BACKGROUND_DUMP_DEST, y los de usuario por USER_DUMP_DEST. Oracle crea ficheros de traza automáticamente cuando ocurre algún error.

Un parámetro muy frecuentemente utilizado por los desarrolladores Oracle es el SQL_TRACE, que cuando está puesto a TRUE produce que toda sentencia SQL ejecutada genere información en los ficheros de traza. Este parámetro se puede variar con el siguiente comando:

SQL> alter session set SQL_TRACE=TRUE;

----------Session Altered----------.

El directorio donde se depositan los ficheros de traza debe de examinarse con regularidad para controlar el tamaño de los fichero allí depositados.

2. CONFIGURACION

2.1 El Código Oracle

Cuando el software Oracle se instala en un sistema, se crean Directorios y ficheros, dependientes todos ellos del S.O. Por ejemplo, en el S.O. Unix, todo los Directorios Oracle se encuentran colgando del directorio principal ORACLE_HOME. Todos estos Directorios contienen ficheros ejecutables y *scripts* que son cruciales para el funcionamiento y la administración del SGBD (SISTEMA GESTOR DE BASE DE DATOS), y es lo que se conoce por el código Oracle. Entre el los, una herramienta nos va a ser fundamental en las tareas de administración y puesta en marcha de la BASE DE DATOS: *server manager*, svrmgr. Con ella se convertirán en DBA, y para ejecutarla se deberá ser sus propietarios.

La sentencia es la siguiente:

```
SVRMGR> connect internal
----------------------Connected-----------------------------------------------------------------.
```

Todas las operaciones de administración deben comenzar por conectarse a la BASE DE DATOS.

2.2 Arranque y Parada de la BASE DE DATOS

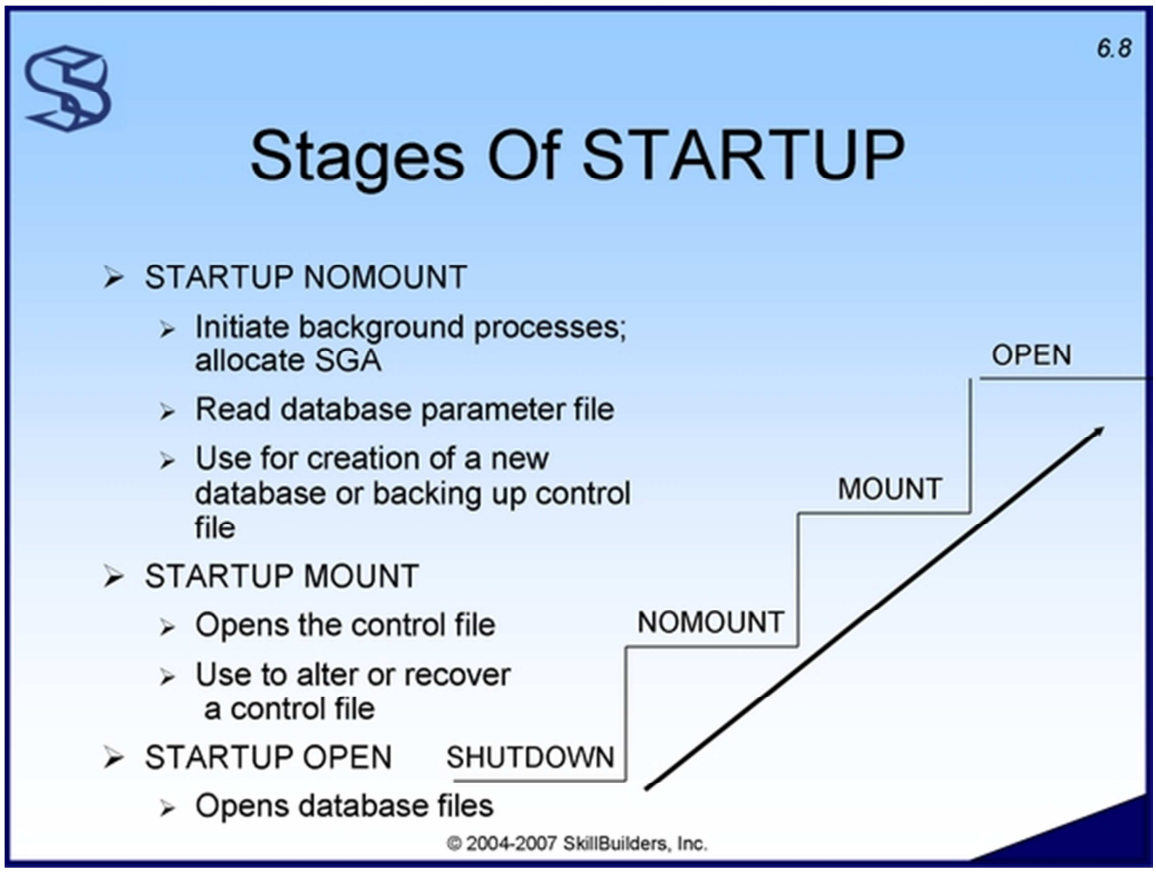
Durante el arranque y parada de la BASE DE DATOS suceden un conjunto de eventos que llevan a la BASE DE DATOS por diferentes estados.

Para que los usuarios puedan acceder a la BASE DE DATOS el DBA necesita abrir la BASE DE DATOS. El siguiente es un ejemplo de apertura de una BASE DE DATOS llamada test.

```
SVRMGR> startup open test
ORACLE instance started.
Total System Global Area 4512688 bytes.
Fixed Size              39732 bytes.
Variable Size         4055164 bytes.
Database Buffers       409600 bytes.
Redo Bufers              8192 bytes.
Database mounted.
Database opened.
```

Cuando se ejecuta el comando startup open la BASE DE DATOS pasa por tres estados (*nomount*, *mount* y *open*) antes de estar disponible. El DBA puede arrancar la BASE DE DATOS hasta uno de los estados con el comando startup: startup nomount, startup mount. A continuación vamos a describir cada uno de los estados por los que pasa la BASE DE DATOS en el proceso de arranque.

Grafico Nº 11: Estados de Inicialización de la Base de Datos de Oracle



Fuente: www.skillbuilders.com

Nomount

```
SVRMGR> startup open test
ORACLE instance started.
Total System Global Area 4512688 bytes.
Fixed Size              39732 bytes.
Variable Size         4055164 bytes.
Database Buffers       409600 bytes.
Redo Bufers              8192 bytes.
```

Oracle lee el fichero init.ora, localiza los ficheros de control, crea e inicializa la SGA, y finalmente arranca todos los procesos Oracle. En este estado la instancia de

BASE DE DATOS está arrancada. Se deberá llevar la BASE DE DATOS al estado *Nomount* cuando se esté creando la BASE DE DATOS o cuando se está restaurando un fichero de control después de haberlo perdido.

Mount

SVRMGR> alter database mount;
Statement processed.

Oracle abre los ficheros de control para localizar los ficheros de datos y los *redo log*, pero no se realizan ninguna comprobación en ellos en este momento. La instancia monta la BASE DE DATOS y la bloquea, verificando que ninguna otra instancia ha montado la misma BASE DE DATOS.
Hay varias razones para querer tener la BASE DE DATOS en el estado *mount*. En general, todas las sentencias SQL del tipo alter database se deben ejecutar en esta etapa. Algunas de las operaciones a realizar cuando la BASE DE DATOS está montada son: efectuar recuperaciones, poner *online*/*offline* un fichero de datos, recolocar los ficheros de datos y *redo log*, crear un nuevo grupo o miembro *redo log*, o borrar un grupo o miembro *redo log* existente.

Open

SVRMGR> alter database open;
Statement processed.

Durante esta etapa, la instancia abre la BASE DE DATOS, bloquea los ficheros de datos, y abre todos los ficheros *redo log*. Si la instancia abre la BASE DE DATOS después de una terminación anormal, o después de una caída, se ejecutará automáticamente el proceso de recuperación utilizando los ficheros *redo log*. Al final de esta etapa la BASE DE DATOS está dispuesta para su uso normal.

Para detener la BASE DE DATOS el comando aplicable es shutdown como se puede ver en el siguiente ejemplo:

SVRMGR> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.

Donde este comando nos permite detener la Base de Datos de varias formas dependiendo de la situación en que nos encontremos o de la forma que el administrador lo requiera por lo que se presenta con tres opciones: *normal*, *immediate* y *abort*.

- ✓ **Shutdown normal**

  Se impide el acceso a la BASE DE DATOS, espera a que todos los usuarios completen todas sus peticiones y se desconecten del servidor. Purga todos los buffers de datos y cachés de *redo log*, actualizando los ficheros de datos y de *redo log*, se eliminan los bloqueos de ficheros, se completan las transacciones en marcha, se actualizan las cabeceras de ficheros, elimina los *threads*, libera los bloqueos de la BASE DE DATOS por parte de la instancia, y sincroniza los ficheros de control y de datos. En resumen, la opción normal cierra la BASE DE DATOS, desmonta la BASE DE DATOS y para la instancia con cuidado y es la opción recomendada para parar la BASE DE DATOS.

- ✓ **Shutdown immediate**

  En ciertas ocasiones puede ser necesario parar la BASE DE DATOS de modo inmediato. Si es así, las sentencias en proceso son terminadas inmediatamente, cualquier transacción no confirmada (*uncommitted*) es vuelta atrás (*rolled back*) y la BASE DE DATOS es parada. La única desventaja de utilizar esta opción es que Oracle no espera a que los usuarios se desconecten. Sin embargo, la BASE DE DATOS será consistenta y no se necesitará recuperación en el siguiente arranque.

- ✓ **Shutdown abort**

  En situaciones de emergencia, y cuando todo lo demás falla, se debe realizar una parada de este tipo. Por ejemplo, cuando un proceso de la instancia muere y la BASE DE DATOS no puede pararse de modo normal o inmediato. Cuando se utiliza la opción *abort* las sentencias SQL son terminadas bruscamente, y las transacciones no confirmadas no son vueltas atrás. Parar la BASE DE DATOS con la opción *abort* requiere recuperación en la siguiente vez que arranque la BASE DE DATOS y esta opción debe ser utilizada sólo cuando no quede más remedio.

2.3 Almacenamiento de Datos

Los datos se almacenan en espacios de tablas, y un espacio de tabla es la entidad lógica que se corresponde con uno o más ficheros físicos. La principal razón de esta organización es el aumento de la flexibilidad a la hora de realizar operaciones con la BASE DE DATOS. En esta sección vamos a dar un repaso a las tareas de administración relacionadas con los espacios de tablas y con los ficheros.

2.3.1 Espacios de Tablas y Ficheros

Los espacios de tablas se utilizan para realizar tareas de gestión de espacio, controlar la disponibilidad de los datos y ejecutar copias de seguridad y recuperaciones parciales.

Gestión de Espacio

El primer espacio de tablas es el SYSTEM. Este espacio de tablas debe estar disponible siempre durante el funcionamiento normal de la BASE DE DATOS porque contiene el diccionario de datos. Después de la creación de la BASE DE DATOS, se recomienda la creación de otros espacios de tablas para que los datos de los usuarios puedan ser separados de los del diccionario de datos. Incluso, si varias aplicaciones se van a ejecutar sobre la misma BASE DE DATOS es recomendable que sus datos estén separados. Para crear un espacio de tablas se puede utilizar el comando create tablespace:

```
SVRMGR> create tablespace nombre_tablespace
2> datafile 'nombre_fichero' size 50M online;
```

En el ejemplo anterior se ha creado un espacio de tablas de 50 Mb. de tamaño. Cada espacio de tabla tiene un conjunto de parámetros de almacenamiento que controla su crecimiento:

- *initial*: tamaño de la extensión inicial (10k).
- *next*: tamaño de la siguiente extensión a asignar (10k).
- *minextents*: número de extensiones asignadas en el momento de la creación del espacio de tablas (1).
- *maxextents*: número máximo de extensiones.
- *pctincrease*: Porcentaje en el que crecerá la siguiente extensión antes de que se asigne, en relación con la última extensión utilizada.
- *optimal*: Tamaño óptimo declarado para este espacio de tablas.
- *pctused*: porcentaje de utilización del bloque por debajo del cual Oracle considera que un bloque puede ser utilizado para insertar filas nuevas en él.
- Si el espacio de tablas necesita más espacio después de su creación se puede puede alterar para añadir uno o más ficheros. Para ello se puede utilizar el comando alter tablespace:

```
SVRMGR> alter tablespace nombre_tablespace
2> add datafile 'nombre_fichero' size 30M;
```

Si se necesitara variar la localización de los ficheros asociados a un espacio de tablas se puede hacer con los comandos alter tablespace (el espacio de tables debe estar *offline*) o alter database (la BASE DE DATOS debe estar montada pero no abierta). Antes de ejecutar los anteriores comandos los ficheros asociados al espacio de tablas deben de haber sido movidos a su nueva localización utilizando los comandos del Sistema Opoerativo oportunos.

Poniendo los *tablespaces offline*

Llevar a un espacio de tablas al estado *offline* significa que se impide el acceso a los datos que almacena. El espacio de tablas SYSTEM nunca puede estar *offline*. Las razones para poner un espacio de tablas *offline* pueden ser varias:

- Un error de escritura en los ficheros que lo soportan.
- El mover los ficheros de sitio, etc.

Después de realizar estas operaciones hay que poner otra vez disponible el espacio de tablas, esto es *on line*

Los espacios de tablas se pueden poner *offline* de tres modos: *normal*, *temporary* e *immediate*. Si no existe ningún error lo recomendable es poner el espacio de tablas *offline* usando el modo normal. Así, se colocará un *checkpoint* en el espacio de tablas antes de ponerlo *offline*.

```
SVRMGR> alter tablespace nombre_tablespace offline normal;
```

Si alguno de los ficheros está corrupto, la opción *normal* fallará y se necesitará el modo *temporary*. La opción *immediate* se utilizará sólo cuando la BASE DE DATOS está en modo ARCHIVELOG, ya que no se produce *checkpoint* alguno.

Poniendo los ficheros *offline*

No es normal poner los ficheros *offline*/*online*. Si un determinado fichero de datos se corrompe, se tendrá que poner *offline*, repararlo y ponerlo *online* de nuevo. Esta operación puede suponer sustituirlo por su copia de seguridad, lo que implicará ejecutar el comando recover datafile antes de poner el fichero *online*.

2.3.2 Segmentos, Extensiones y Bloques

Los datos en la BASE DE DATOS son almacenados físicamente en bloques Oracle: la mínima unidad de espacio físico, y es un múltiplo del bloque del SO (2 Kb usualmente). El tamaño del bloque Oracle se fija por el parámetro DB_BLOCK_SIZE del fichero init.ora. Un tamaño grande de bloque mejora la eficiencia del cache de E/S, pero el tamaño de la SGA aumentará para contener los mismos DB_BLOCK_BUFFERS, lo que significa un problema de memoria.

Una serie de bloques contiguos es una extensión, que es una unidad lógica de almacenamiento. Una serie de extensiones es un segmento. Cuando un objeto es creado, se reserva una extensión en su segmento. Cuando el objeto crezca, necesitará más espacio y se reservarán más extensiones.

Cada segmento tiene un conjunto de parámetros de almacenamiento que controla su crecimiento:

- *initial*: tamaño de la extensión inicial (10k).
- *next*: tamaño de la siguiente extensión a asignar (10k).
- *minextents*: número de extensiones asignadas en el momento de la creación del segmento (1).
- *maxextents*: número máximo de extensiones (99).
- *pctincrease*: Porcentaje en el que crecerá la siguiente extensión antes de que se asigne, en relación con la última extensión utilizada (50).
- *pctfree*: porcentaje de espacio libre para actualizaciones de filas que se reserva dentro de cada bloque asignado al segmento (10).
- *pctused*: porcentaje de utilización del bloque por debajo del cual Oracle considera que un bloque puede ser utilizado para insertar filas nuevas en él.
- *tablespace*: nombre del espacio de tablas donde se creará el segmento.
- Cuando se diseña una BASE DE DATOS se ha de tener mucho cuidado a la hora de dimensionar la BASE DE DATOS y prever el crecimiento de las tablas. A continuación se hacen algunas consideraciones sobre la gestión del espacio para los diferentes segmentos.

Segmentos de Datos

El espacio del diccionario de datos se suele mantener más o menos constante, aunque es crítico que tenga suficiente espacio para crecer en el espacio de tablas SYSTEM. Así, hay que tener cuidado de colocar las tablas de usuario, los índices, segmentos temporales y los segmentos de *rollback* en otros espacios de tablas. Además, es recomendable que el espacio de tablas SYSTEM esté al 50% o 75% de su espacio disponible. Finalmente, asegurarse que los usuarios no tienen privilegios de escritura en el espacio de tablas SYSTEM.

Las tablas crecen proporcionalmente con el número de filas, ya que se puede suponer que la longitud de las filas es constante.

Segmentos de Índice

Los índices crecen en tamaño en mayor proporción que las tablas asociadas si los datos en la tabla son modificados frecuentemente. La gestión del espacio es mejor si se mantienen los índices de tablas grandes en espacios de tablas separados.

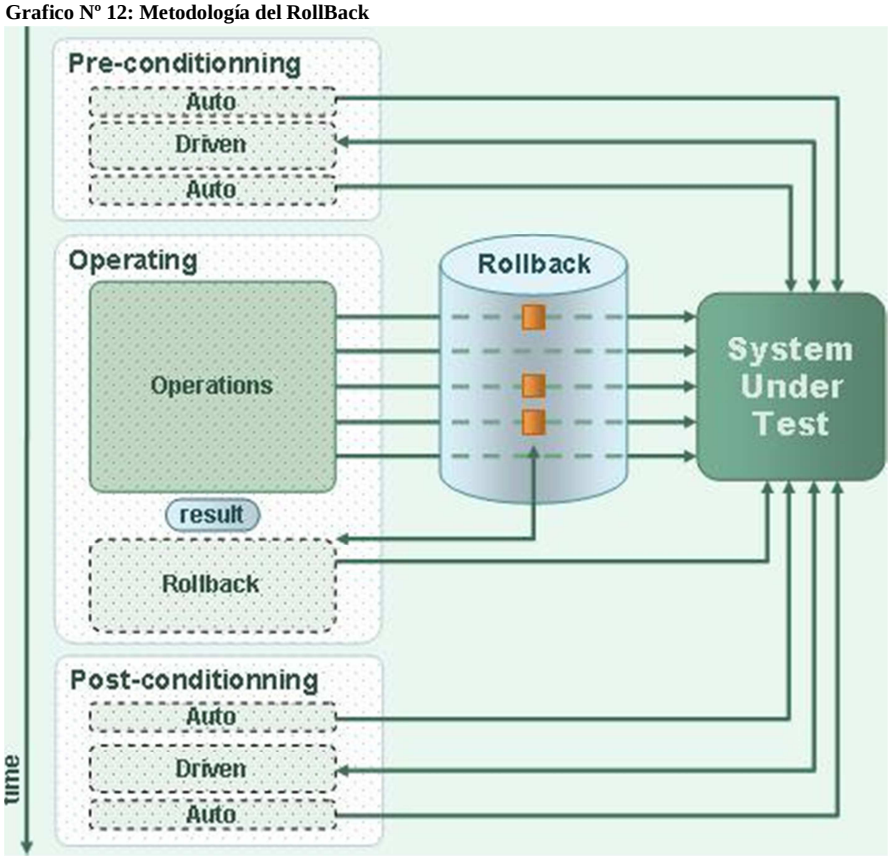
Segmentos de *Rollback*

Los segmentos de *rollback* almacenan la imagen anterior a una modificación de un bloque. La información en el segmento de *rollback* se utiliza para asegurar la consistencia en lectura, el *rollback* (el valor en el segmento de *rollback* se copia en el bloque de datos) y la recuperación.

Es importante comprender cuál es el contenido de un segmento de *rollback*. No almacenan el bloque de datos modificado entero, sólo la imagen previa de la fila o filas modificadas. La información del segmento de *roolback* consiste en varias entradas llamadas *undo*. Por ejemplo, si se inserta una fila en una tabla, el *undo* necesitará sólo el *rowid* de la fila insertada, ya que para volver atrás la inserción sólo hay que realizar un delete. En la operación de actualización, se almacenará el valor antiguo de las columnas modificadas. El segmento de *rollback* asegura que la información *undo* se guardan durante la vida de la transacción.

Un segmento de *rollback* como cualquier otro segmento consiste en una serie de extensiones. Sin embargo, la mayor diferencia entre un segmento de datos y otro *rollback* es que en este último las extensiones se utilizan de manera circular. Así, habrá que tener cuidado a la hora de fijar el tamaño del segmento de *rollback* para que la cabeza no pille a la cola.

Grafico Nº 12: Metodología del RollBack



Fuente: http://www.xqual.com/documentation/tutorial_rollback.html

Segmentos Temporales

Los segmentos temporales se crean cuando se efectúan las siguientes operaciones:

- Create Index
- Select con distinct, order by, union, intersect y minus.
- Uniones no indexadas.
- Ciertas subconsultas correlacionadas.

Si las tablas a ordenar son pequeñas la ordenación se realiza en memoria principal, pero si la tabla es grande se realiza en disco. El parámetro SORT_AREA_SIZE determina el lugar donde se hace la ordenación. Incrementándole se reduce la creación de segmentos temporales.

2.4 Configuración de la BASE DE DATOS

Mientras se diseña la BASE DE DATOS hay que considerar la posible recuperación de una caída, y las prestaciones de la BASE DE DATOS, relacionando todo esto con las necesidades de la implantación y los medios disponibles. La configuración de la BASE DE DATOS está relacionada con los ficheros de control, los ficheros *redo log* activos y los archivados.

2.4.1 Gestionando los Ficheros de Control

Los ficheros de control contienen el esquema de la BASE DE DATOS. Es uno de los más importantes ficheros e imprescindible para el uso normal de la BASE DE DATOS. Así que daremos alguna pista para su gestión.

El parámetro CONTROL_FILES del fichero init.ora contiene la lista de todos los ficheros de control. Cuando se arranca la BS, Oracle lee el fichero init.ora para determinar cuántos ficheros de control se usan en la BASE DE DATOS y dónde están. Durante la fase de montaje, se abren los ficheros de control para leer el esquema de la BASE DE DATOS. Aunque Oracle escribe en todos los ficheros de control, sólo lee el primero listado en el parámetro CONTROL_FILES.

Para protegerlos contra fallos de almacenamiento, se sugiere que al menos existan dos ficheros de control, cada uno en un disco diferente, aunque es buena idea mantener más copias en diferentes discos. Esto es una política de *espejado* que protege frente a fallos en disco. Si un disco falla y se pierden todos los ficheros en él, se puede seguir utilizando los ficheros de control de otros discos. Esto supone una pequeña sobrecarga al sistema, ya que cada vez que se produce un *checkpoint* o cambia el esquema de la BASE DE DATOS, todos los ficheros de control son actualizados.

Cuando se produce un fallo en algún disco y algún fichero de control se pierde hay que parar la BASE DE DATOS con la opción *abort*, copiar el fichero de control que queda en otro disco, editar el fichero init.ora para reflejar este cambio, y volver a levantar la BASE DE DATOS.

Si un fallo ha producido la pérdida de todas las copias de los ficheros de control habrá que recrearlos con el comando create controlfile. Si algunos de los parámetros MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES y MAXINSTANCES varía habrá que utilizar también el comando CREATE CONTROLFILE.

2.4.2 Gestionando los Ficheros *Redo Log* Activos

Oracle proporciona la posibilidad de *espejar* los ficheros *redo log* activos. Mecanismo conocido como ficheros *redo log* multiplexados. Oracle necesita al menos dos grupos de ficheros *redo log*, cada uno con un miembro como mínimo. Oracle efectúa escrituras en paralelo a cada miembro, pero si están en el mismo disco, realmente la escritura se serializa.

Otro aspecto a tener en cuenta es el tamaño de los ficheros *redo log*. Si son muy pequeños, el LGWR deberá cambiar de ficheros demasiado frecuentemente, lo que reduce su rendimiento. Por otro lado, si los ficheros *redo log* son demasiado grandes, se necesitará mucho tiempo en las recuperaciones, ya que se tendrán que recuperar muchas transacciones.

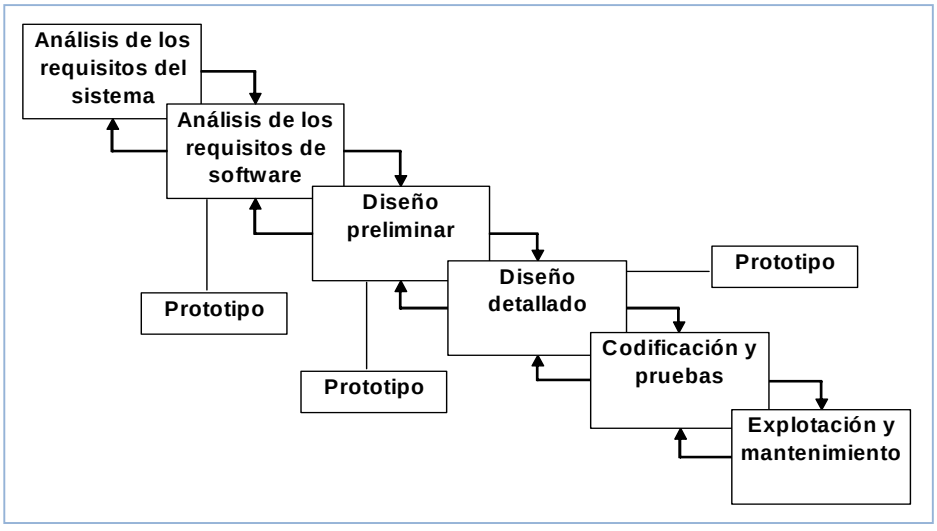
Otro aspecto muy importante es la elección del número correcto de grupos, ya que disponer de demasiados pocos grupos puede acarrear problemas cuando estamos en modos ARCHIVELOG y tenemos una tasa de transacciones muy alta. Esto puede suponer que un grupo que todavía está archivando por el proceso ARCH se convierta en el grupo en el que el LGWR necesite escribir, lo que produciría que la BASE DE DATOS se parara, ya que el LGWR tienen que esperar a que el grupo esté disponible, una vez que su contenido ha sido archivado. Para la mayoría de las implantaciones, tener entre 2 y 10 grupos puede ser suficiente. El número de grupos no puede exceder de MAXLOGFILES, ni el número de miembros puede ser mayor que MAXLOGMEMBERS.

3. METODOLOGIA

El presente documento también redacta cada uno de las etapas realizadas durante la elaboración del proyecto, se escogió la metodología **Prototipo**, debido a que el sistema se va desarrollando en pruebas, una vez identificados todos los requisitos del sistema, se realiza o se construye un diseño preliminar y detallado (Prototipo), el mismo que fue revisado y corregido por un personal entendido en el sistema.

Una de las principales ventajas que nos ofrece la metodología prototipo es reducir el riesgo de construir productos que no satisfagan las necesidades del usuario, aumentando la probabilidad de éxito. Otro punto a favor es que el grupo de desarrollo del sistema puede en cualquier etapa del sistema, regresar a etapas anteriores y modificarlas sin ningún inconveniente.

Grafico Nº 13: Estados y pasos del proyecto



Fuente: http://www.xqual.com/documentation/tuto01k.html

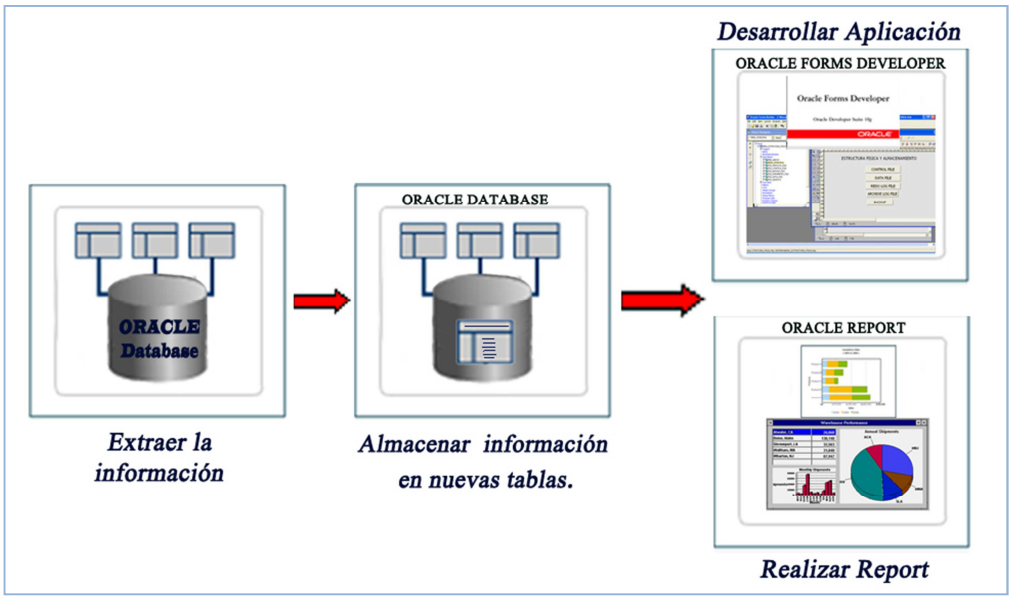
Metodología Prototipo

3.1) Análisis

La información relevante de los valores actuales de los parámetros a tratar se extraen al realizar consultas a múltiples vistas dinámicas proporcionadas por Oracle en su diccionario de datos, esta información será almacenadas en tablas adicionales creadas independientemente por los programadores, las consultas administrativas se realizaran en el Gestor de Bases de Datos Oracle Database (Oracle 10g). El desarrollo de la aplicación se llevara a cabo por medio de la Herramienta Oracle Forms Developer (Oracle 10g), las estadísticas serán reflejadas a través de la Herramienta Oracle Reports (Oracle 10g).

Todo esto se realizará con el propósito de tener la información en un repositorio distinto al de la Base de Datos original, organizado de tal manera que las consultas al mismo sean más rápidas y se pueda obtener mejor información para el análisis que debe realizar el administrador de la base de datos.

Grafico Nº 14: Manejo de Información



Fuente: http://www.xqual.com/documentation/tuto02k.html

3.1.2) Casos de Uso

Los usuarios podrán utilizar el sistema bajo los siguientes conceptos:

- Información general de los componentes de la estructura de la sistem global area (PGA, tamaños UNDO Tablespace, Rendimiento en cuanto a bloqueos).
- Necesidad de consultar actuales e históricos por rango de fechas de los parámetros a tratar.
- Consultar que esquema es el que está continuamente causando bloqueo.
- Consulta de valores óptimos post-instalación de los parámetros antes mencionados.

3.1.3) Diseño del sistema

La solución tecnológica consiste de un modulo de menú principal, el mismo que está estructurado de la siguiente manera.

Las estructuras principales: consulta de valores actuales e históricos, así como sus reportes y también el rendimiento en cuanto a bloqueos también con su determinado reporte.

Dentro del menú “GENERAL”: se despliega un submenú, el que contiene: consulta_parametros_sga, consulta_parametros_undo.

En otra pantalla del menú se tendrá 2 tipos de reporte: tabular y estadístico con los respectivos valores analizados de las tablas historiales.

Se tendrá un botón “Bloqueos” que enlazara al modulo, el cual mostrara como oracle maneja la información del rendimiento en cuanto a bloqueos se refiere.

4. IMPLEMENTACIÓN

Consecuente con lo que se describió anteriormente de la Base de Datos de Oracle, sus características, propiedades, funcionamiento y componentes, se procede a describir lo planteado en el alcance.

No se debe de dejar de puntualizar los parámetros que son cargados al momento de levantar la base a través del INIT.ORA que genera el SPFILE para configuración inicial de la Base de Datos como son:

Grafico Nº 13: Parámetros de Inicialización

Name	Value	Description
dblink_encrypt_login	FALSE	Enforce password for distributed login; always be encrypted
hash_join_enabled	TRUE	Enable/disable hash join
log_parallelism	1	number of log buffer strands
max_rollback_segments	37	Maximum number of rollback segments in SGA cache
mts_circuits	0	Maximum number of circuits
mts_dispatchers	[NULL]	Specifications of dispatchers
mts_listener_address	[NULL]	Address(es) of network listener
mts_max_dispatchers	5	Maximum number of dispatchers
mts_max_servers	20	Maximum number of shared servers
mts_multiple_listeners	FALSE	Are multiple listeners enabled?
mts_servers	0	Number of shared servers to start up
mts_service	devbill	Service supported by dispatchers
mts_sessions	0	Maximum number of shared server sessions
optimizer_max_permutations	2000	Optimizer maximum join permutations per query block
oracle_trace_collection_name	[NULL]	Oracle TRACE default collection name
oracle_trace_collection_path	?/otrace/admin/cdf	Oracle TRACE collection path
oracle_trace_collection_size	5242880	Oracle TRACE collection file maximum size
oracle_trace_enable	FALSE	Oracle TRACE enabled/disabled
oracle_trace_facility_name	oracled	Oracle TRACE default facility name
oracle_trace_facility_path	?/otrace/admin/fdf	Oracle TRACE facility path
partition_view_enabled	FALSE	Enable/disable partitioned views
row_locking	always	Row-locking
serializable	FALSE	Serializable
transaction_auditing	FALSE	Transaction auditing records generated in the redo log
undo_suppress_errors	FALSE	Suppress RBU errors in SMU mode

Fuente: http://www.dba-oracle.com/t_oracle10g_initialization_parms.htm

Se profundiza en los parámetros sobre los cuales se concentra el análisis como son:

4.1 UNDO

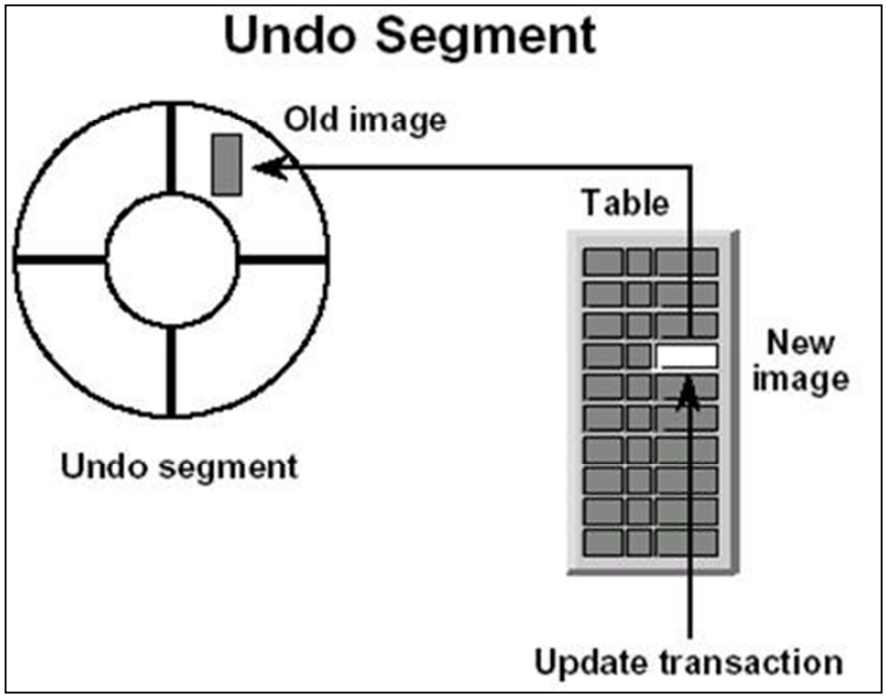
La característica de UNDO se basa principalmente en regresar a un estado anterior por cualquier inconsistencia, fallo, o mala aplicación de las sentencias por parte del usuario.

Es una Unidad lógica de trabajo que contiene una o más sentencias SQL; se trata de una unidad indivisible o atómica. Los efectos de las sentencias de una transacción pueden ser todos validados (aplicados a la base de datos) o retrocedidos. Comienza en la primera sentencia SQL ejecutable y termina cuando es validada o retrocedida, de forma explícita, mediante las sentencias COMMIT o ROLLBACK, o implícitamente, en el caso de sentencias DDL.

El Espacio de "UNDO" es un conjunto de registros que guardan información, relativa a acciones realizadas por una transacción, necesaria para:

- Recuperación de la base de datos.
- Proporcionar consistencia en lectura (imagen de los datos).
- Retroceder transacciones ("rollback").
- Análisis de datos previos usando Oracle Flashback Query.
- Recuperación lógica usando Oracle Flashback.

Grafico Nº 14: Segmentos de UNDO



Fuente: http://www.siue.edu/~dbock/cmis565/module10-undo_data.htm

En caso de producirse una caída del sistema y quedar transacciones activas (sin validación commit- ni retroceso – rollback-), Oracle recupera la información del espacio de "UNDO" y una vez hecho se realiza el "rollback" de dichas transacciones, es así que en la recuperación de base de datos y una vez aplicados los cambios guardados en los ficheros de "redo", el espacio de "UNDO" sirve para deshacer los efectos de transacciones no validadas. Este proceso recibe el nombre de "ROLLING BACK" o "TRANSACTION RECOVERY". Por lo que el retroceso de una transacción:

- Se aplican todos los cambios almacenados en orden inverso hasta llegar al dato original.
- Se libera cualquier bloqueo de datos efectuado por la transacción.
- Finaliza la transacción.

Es pertinente que se señale la importancia de esta característica que maneja la Base de Datos Oracle a través de UNDO ya que me permite como ya se reitero en varias ocasiones RECUPERAR estados anteriores de la Base de Datos, previniéndose de ciertas complicaciones y fallos.

Esto permite dar a conocer la importancia que es que los parámetros relacionas y que manejan o afectan el PERFORMANCE de la Base de Datos Oracle y que directamente influyen en esta característica son los siguientes parámetros:

4.4.1 UNDO_RETENTION

El valor por defecto es 900 segundos (15 minutos), y este se puede modificar para cubrir más tiempo la recuperación de la Base de Datos; por lo consiguiente es importante redefinir el valor del UNDO_RETENTION en relación del tamaño del tablespaces de UNDO para su soporte y lógicamente por el tipo de transacciones por la cual la Base de Datos es requerida, caso contrario se puede determinar valores no concordantes, principalmente entre el UNDO_RETETION y el Tamaño del Tablespaces de UNDO ya que si el segundo no está siendo totalmente usado y la operativiliadad de la base requiere más tiempo de retención para ROLLBACK entonces deberá ser modificado el UNDO_RETENTION por un valor mayor; caso contrario si el nivel de transacciones no requieren tener un alto rango de tiempo para el retroceso que genera el ROLLBACK, entonces el valor que debe ser revisado seria el del tamaño del Tablespaces de UNDO y decrecerlo en el porcentaje que el UNDO_RETENTION lo permita.

4.1.2 UNDO_MANAGEMENT

UNDO_MANAGEMENT específica para UNDO el modo de gestión del espacio que el sistema debe utilizar. Cuando se establece en AUTO la instancia se inicia en el modo de gestión automática de UNDO. Mientras que el modo manual de gestión de UNDO, el Tablespaces de UNDO se asigna externamente como segmentos de rollback.

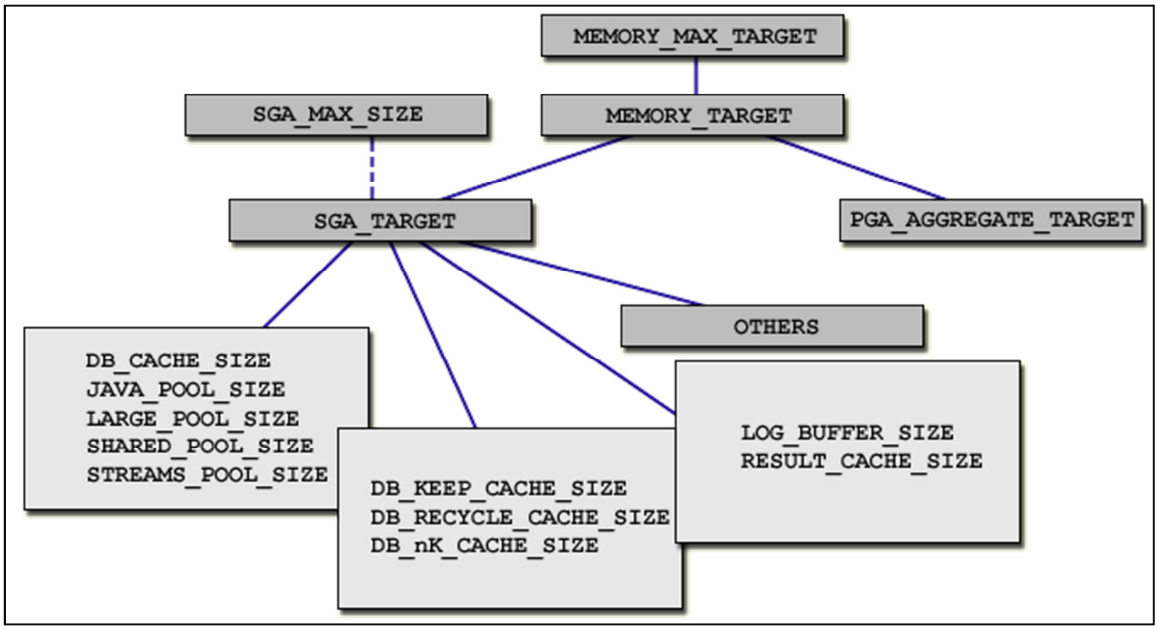
4.1.3 UNDO_BLOCK_PER_SEC

Con este parámetro se mide la cantidad secuencial de bloques que se encuentran en UNDO los cuales pueden trabajar en conjunto, se los referencian más que nada para la toma de medidas y muestreo de la funcionalidad de los otros parámetros ya que individualmente, este UNDO_BLOCK_PER_SEC no se recopila información adecuada y concisa que nos dé una referencia más clara de su finalidad y trabajo como un solo parámetro sino, como en conjunto de ellos.

4.2 FUNCTION OF SGA

Ambos parámetros existen en la versión **10g**. Su significado es diferente.

Grafico Nº 15: Parámetros que manejan la SGA



Fuente: http://www.toadworld.com/KNOWLEDGE/KnowledgeXpertforOracle/tabid/648/TopicID/AMM1/Default.aspx

Sga_max_size: Establece el máximo tamaño que puede alojar la **SGA** cuando se levanta la instancia de base de datos. Este parámetro permitirá aumentar el tamaño de la SGA sin necesidad de iniciar la instancia, teniendo en cuenta que el total de la SGA no exceda este parámetro.

Sga_target: Especifica el total de tamaño que dispondrá la SGA cuando la instancia se inicia. Si utilizamos este parámetro no tendremos necesidad de definir los valores para **db_cache_size, shared_pool_size, large_pool_size, java_pool_size** puesto que oracle automáticamente ajusta estos componentes incluyendo **stream_pool_size**.

Normalmente **sga_max_size** y **sga_target** tendrán el mismo valor, pero habrá veces cuando se quiera ajustar para el máximo número de cargas y en este caso podrá ser diferente. En este caso sga_max_size será mayor que sga_target, de este modo podrás alojar dinámicamente el ajuste del parámetro sga_target.

4.2.1 LARGE_POOL_SIZE

Asigna una agrupación grande de modo que Oracle disponga de una agrupación por separado para solicitar asignaciones de memoria grandes y reducir el impacto en el SGA en su conjunto. También debe establecer el parámetro **LARGE_POOL_MIN_ALLOC**.

4.2.2 SHARED_POOL_SIZE

La agrupación compartida reside en el SGA de instancias de Oracle y contiene las áreas de SQL compartidas y el diccionario de datos. El área de SQL compartida contiene la información necesaria para ejecutar sentencias SQL simples. El diccionario de datos contiene los nombres de usuario del servidor de Oracle, los privilegios y las funciones, los nombres y las definiciones de los objetos de base de datos, las restricciones de integridad, las asignaciones de espacio de objetos de la base de datos, la información de auditoría, los procedimientos almacenados y los desencadenantes.

4.2.3 DB_CACHE_SIZE

Este parámetro especifica el tamaño de la agrupación de almacenamientos en búfer predeterminada para los búfers que utilizan el tamaño de bloque que especifica el parámetro **DB_BLOCK_SIZE**.

4.2.4 LOG_BUFFER

Es una aplicación que genera muchos registros y que, comúnmente, tiene entre 3 y 5 MB de tamaño de búfer de registro. Comprueba las estadísticas de reintentos de asignación de búfer de rehacer en la vista V$SYSSTAT para verificar si este valor es alto. Un valor alto indica que debe incrementarse el tamaño del búfer del registro.

4.3 RASTREO

Activar el rastreo de SQL extendido para un programa es fácil si tiene acceso al programa fuente. La primera cosa que tiene que hacer es asegurarse de que su sesión está utilizando los valores apropiados para sus parámetros TIMED_STATISTICS y MAX_DUMP_FILE_SIZE. Parametros los cuales vamos testearlos mediante la cantidad de bloqueos pueden tener un esquema.

4.3.1 TIMED_STATISTICS

La utilidad de trazado puede, opcionalmente, proporcionar información acerca de los tiempos de ejecución. Para que esta información quede almacenada es necesario activar el parámetro *TIMED_STATISTICS*.

Dicho parámetro se puede activar a nivel de base de datos mediante su inclusión en el fichero de parámetros de la base de datos Oracle.

4.3.2 MAX_DUMP_FILE_SIZE

Para poder conocer el MAX_DUMP_FILE_SIZE se debe saber cómo interpretar User trace file (Archivo de Rastreo del Usuario). Una vez que estos trace file se han generado hay que aprender a interpretarlos.

Podemos gestionar el tamaño de estos archivos mediante una serie de parámetros en el INIT.ora Parámetro especificado Tamaño máximo para User Trace que es el MAX_DUMP_FILE_SIZE.

4.4 MANIPULACION Y TRATO DE LOS PARAMETROS

Claro está que al implementar el aplicativo TESTEADOR basado en los parámetros ya mencionados, se lo realiza en base a su manipulación de artificios matemáticos corroborados por los programadores y analista de Base de Datos que son llevados a cabo a través de los scripts realizados en lenguaje de programación PL/SQL.

A continuación se mostraran los Scripts que se usaron para la comparación y trato de estos parámetros y la manera en que se obtuvieron los valores que estadísticamente y por resultado del scripts son considerados como los valores óptimos que requiere la Base de Datos los cuales al final son comparados con la configuración previa de los mismo para determinar su modificación o no.

- Para la determinación del tamaño de UNDO actual

```
SELECT SUM(a.bytes) "UNDO_SIZE"
  FROM v$datafile a,
       v$tablespace b,
       dba_tablespaces c
 WHERE c.contents = 'UNDO'
   AND c.STATUS = 'ONLINE'
   AND b.name = c.tablespace_name
   AND a.ts# = b.ts#;
```

- Para Obtener el valor optimo que debe tener el UNDO_RETENTION

```
SELECT d.undo_size/(1024*1024) "ACTUAL UNDO SIZE [MByte]",
       SUBSTR(e.value,1,25) "UNDO RETENTION [Sec]",
       ROUND((d.undo_size / (to_number(f.value) *
       g.undo_block_per_sec))) "OPTIMAL UNDO RETENTION [Sec]"
  FROM (
       SELECT SUM(a.bytes) undo_size
          FROM v$datafile a,
               v$tablespace b,
               dba_tablespaces c
```

```
         WHERE c.contents = 'UNDO'
           AND c.STATUS = 'ONLINE'
           AND b.name = c.tablespace_name
           AND a.ts# = b.ts#
       ) d,
       v$parameter e,
       v$parameter f,
       (
       SELECT MAX(undoblks/((end_time-begin_time)*3600*24))
              undo_block_per_sec
         FROM v$undostat
       ) g
WHERE e.name = 'undo_retention'
AND f.name = ‘db_block_size’
```

- Script General para la determinación y control de los Tablespaces

```
col Tablespace_Name FOR a15 tru       heading 'Tablespace'
col Total    FOR 999G999G999G999G990  heading 'Total|(Bytes)'
col Max FOR 999G999G999G999G990 heading 'Hueco (Bytes)|Mayor'
col Free     FOR 999G999G999G999G990  heading 'Total|Libre'
col Pct_Free       FOR 990D00         heading '% Libre'
col Max_Table FOR 999G999G999G990 heading 'Max.Ext.|Tabla'
col Max_Index FOR 999G999G999G990 heading 'Max.Ext.|Indice'
col NOK FOR A3 tru heading 'Posibilidad|de Crecimiento'
Col ERLOG New_Value Mylog;
SET ECHO Off TERMOUT Off FEED Off NEWPAGE 0;
SELECT 'C:\Oracle\log\Crecimiento_'||host_name||'_'||
instance_name||'.log' Erlog FROM V$Instance;
SET TERMOUT ON FEED ON NEWPAGE 1;

Spool &Mylog;
SELECT /*+ RULES */ Tablespace_Name,
       Total,
       Max,
       Free,
       (100 * Free / Total) Pct_Free,
       Max_Table,
       Max_Index,
       decode(sign(Max - greatest(nvl(Max_Table,0),
nvl(Max_Index,0))),1, NULL, ' X ') NOK
```

```
FROM  (SELECT Tablespace_Name DT_Ts_Name, max(next_extent +
                ((pct_increase / 100) * next_extent))
Max_Table
            FROM dba_tables
            GROUP BY Tablespace_Name),
      (SELECT Tablespace_Name DI_Ts_Name, max(next_extent +
                ((pct_increase / 100) * next_extent))
Max_Index
            FROM dba_Indexes
            GROUP BY Tablespace_Name),
      (SELECT Tablespace_Name FS_Ts_Name, max(Bytes) Max,
sum(Bytes) Free
            FROM dba_free_space
            GROUP BY tablespace_name),
      (SELECT Tablespace_Name, sum(Bytes) Total
            FROM dba_data_files
            GROUP BY Tablespace_Name)
WHERE  Tablespace_Name = DI_Ts_Name(+) AND Tablespace_Name =
DT_Ts_Name(+)
  AND Tablespace_Name = FS_Ts_Name(+)
ORDER BY Tablespace_Name
```

- Obtención del Crecimiento oportuno de los Tablespaces a manipula

```
compute sum of "Crecimiento (Mb)" label "Total" ON report
break ON report
SELECT to_char(D.CREATION_TIME,'MM') NMES,
      to_char(D.CREATION_TIME, 'RRRR MM') NANYO,
      T.NAME "Tablespace",
      to_char(D.CREATION_TIME, 'RRRR Month') "Mes",
      round(sum(D.BYTES)/1048576) CRECANYMB
FROM V$Datafile D, V$Tablespace T
WHERE  D.TS#=T.TS#
  AND  D.creation_time > SYSDATE-365
GROUP BY to_char(D.CREATION_TIME,'MM'),
to_char(D.CREATION_TIME, 'RRRR MM'),
         T.NAME, to_char(D.CREATION_TIME, 'RRRR Month')
ORDER BY 2
```

- Manejo y Ordenamiento de Bloques para Optimización del Sistema en Base a valores de parámetros relacionados a los mismos bloques en tamaño y tiempo.

```
clear COLUMNS breaks computes

SET LINES 132 pages 5000;
SET serveroutput ON size 40000 feed off echo off head off;

spool C:\Oracle\log\Reorganizar_Bloques_Vacios.log;

SELECT instance_name "Base de datos" FROM v$instance;

declare
  total_blocks number;
  total_bytes number;
  unused_blocks number;
  unused_bytes number;
  last_used_extent_file_id number;
  last_used_extent_block_id number;
  last_used_block number;
  free_pct number;
  total_espacio_liberar number:=0;
begin
  FOR c1 IN (SELECT owner, segment_type, segment_name
             FROM   sys.dba_segments
             WHERE  owner NOT IN ('SYS', 'SYSTEM',
'PUBLIC','OUTLN','DBSNMP')
               AND  segment_type IN ('TABLE','INDEX')
            )
  loop
    dbms_space.unused_space(c1.owner, c1.segment_name,
c1.segment_type,
                 total_blocks, total_bytes, unused_blocks,
                 unused_bytes, last_used_extent_file_id,
                 last_used_extent_block_id, last_used_block);

    free_pct := trunc((1 - (unused_blocks/total_blocks)) *
100);

    IF unused_bytes > 209715200 then
      dbms_output.put((CASE WHEN (c1.segment_type='TABLE')
THEN 'TABLA: "' ELSE 'INDICE: "' END)||'
'||c1.owner||'.'||c1.segment_name||'"');
      dbms_output.put('Liberaria:
'||trunc(unused_bytes/1024/1024)||' (Mb)');
      total_espacio_liberar := total_espacio_liberar +
round(unused_bytes/1048576);
      dbms_output.new_line;
    end IF;
```

```
  end loop;
  dbms_output.new_line;
  dbms_output.new_line;
  dbms_output.new_line;
  dbms_output.put_line('Total de espacio a recuperar:
'||round(total_espacio_liberar/1024)||' (Gb)');
end;
/

SET serveroutput off;

Spool off;
```

- Encontrar el Valor del UNDO_BLOCK_PER_SEC

```
SELECT MAX(undoblks/((end_time-begin_time)*3600*24))
      "UNDO_BLOCK_PER_SEC"
  FROM v$undostat;
```

- Tamaño de los Bloques en General

```
SELECT TO_NUMBER(value) "DB_BLOCK_SIZE [KByte]"
 FROM v$parameter
WHERE name = 'db_block_size';
```

Como se podrá notar la mayoría de los SCRIPTS se basan en Vistas, que es una utilidad que facilita el Oracle para revisar valores y operaciones a nivel de DBA (Administrador de la Base de Datos) de manera interna y son representadas o reconocidas en el lenguaje de SQL por llevar primero la siguiente característica "V$".

4.5 CONTROL Y MANEJO DE BLOQUEOS

Lo más importante no es detectar el bloqueo sino detectar quien lo produce y tratar de que esto no vuelva a ocurrir. Generalmente son deficiencias por parte del desarrollo. Para obtener esta información es sencilla, con sólo chequear la vista v$lock puedes determinar SID (quien bloquea) y LMODE (modo de bloqueo), es conveniente aclarar, que se produzcan bloqueos no está mal. Es negativa cuando otras sesiones esperan más de lo debido. Oracle tiene herramientas como el EM (Enterprise Mánager) que permiten hacer exactamente esto que se describe pero es una buena forma como la direccionamos y generamos informes de ellos que eso no hace el EM (Enterprise Mánager), se programa como un evento en el cual además puedes verificar otras cosas como si se ha generado un trace o dead lock (Matar un Bloqueo) o bien si hay un nuevo error en alerta de la base..

- Determinación de Bloqueos

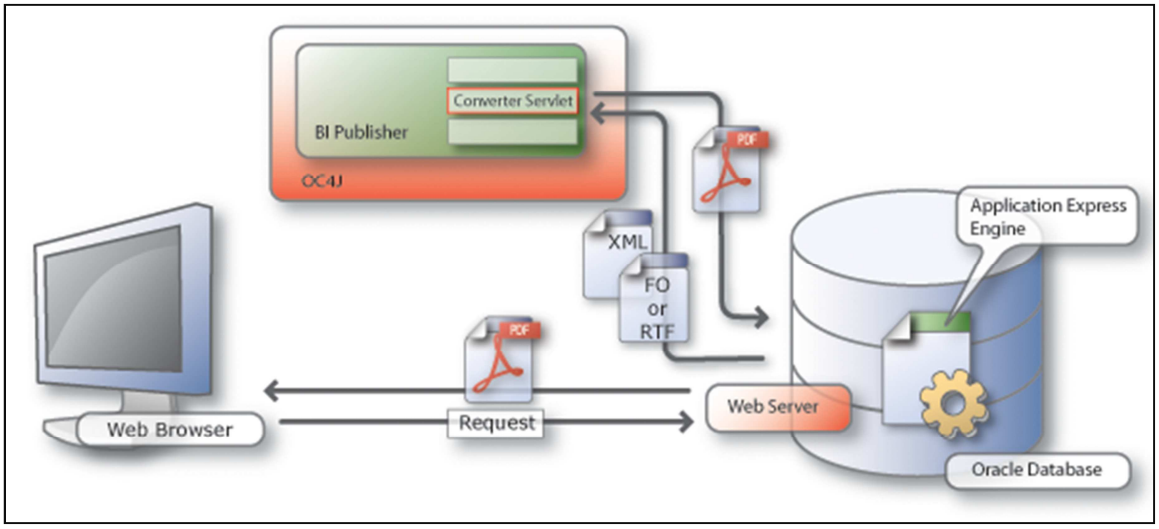
```
dbrito bloqueado >>select * from tbd_registros for update

Scott>>
update tbd_registros set descripcion='vbncvb'
where codigo=5

procedimiento pu_p_revisa

PROCEDURE pu_p_revisa IS
cursor c_session is
select L_LOCKER.SID,
S_LOCKER.USERNAME LOCKER_SCHEMA,
S_LOCKER.OSUSER OS_LOCKER,
S_LOCKER.PROCESS LOCK_PID,
S_WAITER.OSUSER OS_WAITER,
S_WAITER.USERNAME WAITER_SCHEMA,
S_WAITER.PROCESS WAITER_PID,
'TX: '||O.SQL_TEXT SQL_TEXT_WAITER
from
V$LOCK L_WAITER,
V$LOCK L_LOCKER,
V$SESSION S_WAITER,
V$SESSION S_LOCKER,
V$_LOCK L1_WAITER,
V$OPEN_CURSOR O
where S_WAITER.SID = L_WAITER.SID
and L_WAITER.TYPE IN ('TX')
and S_LOCKER.sid = L_LOCKER.sid
and L_LOCKER.ID1 = L_WAITER.ID1
and L_WAITER.REQUEST > 0
and L_LOCKER.LMODE > 0
and L_WAITER.ADDR != L_LOCKER.ADDR
```

```
and L1_WAITER.LADDR = L_WAITER.ADDR
and L1_WAITER.KADDR = L_WAITER.KADDR
and L1_WAITER.SADDR = O.SADDR
and O.HASH_VALUE = S_WAITER.SQL_HASH_VALUE;

LV_OBJETO varchar2(500);
BEGIN
        go_block('BL_T_REVISA');
        clear_block;
        for a in  c_session loop

                SELECT OBJECT_NAME
                INTO LV_OBJETO
                FROM dba_objects, v$lock
                WHERE object_id = id1
                AND type = 'TM'
                AND sid=a.SID
         AND rownum<5;

                        :BL_T_REVISA.SID_QUIEN := a.SID;
                        :BL_T_REVISA.USER_QUIEN := a.LOCKER_SCHEMA;
                        :BL_T_REVISA.USER_MAQUINA := a.OS_LOCKER;
                        :BL_T_REVISA.USER_AQUIEN := a.WAITER_SCHEMA;
                        :BL_T_REVISA.OBJETO := LV_OBJETO;
                        :BL_T_REVISA.SENTENCIA := a.SQL_TEXT_WAITER;
                        next_record;
  --exit when c_session%notfound;
        end loop;

END;


pu_p_mata_sesion


PROCEDURE pu_p_matasesion IS
LN_SERIAL NUMBER;
BEGIN
  SELECT SERIAL#
  into LN_SERIAL
  FROM V$SESSION WHERE SID = :BL_T_REVISA.SID_QUIEN ;

BEGIN
--      FORMS_DDL('alter user '||v_NOMBRE||' identified by '||UPPER(:PASSWORD.NUEVA));

        FORMS_DDL('ALTER SYSTEM KILL SESSION
'''||:BL_T_REVISA.SID_QUIEN||','||LN_SERIAL||'''');
--      ILL SESSION. '''||CRS.SID||','||CRS.SERIAL#||'''';
        null;


        --ALTER SYSTEM KILL SESSION '131,222';--:BL_T_REVISA.SID_QUIEN||','||LN_SERIAL);
END;
END;
```

5. SOFTWARE Y AMBIENTE DEL USUARIO

En esta sección se referenciara y mostrara las ventanas en la cual el usuario del aplicativo TESTEADOR de una Base de Datos Oracle tendrá que manejar en el transcurso de su utilización, donde se le recuerda que estas ventanas fueron desarrolladas a través de la herramienta, Oracle FORMS.

Grafico Nº 16: Solicitud Web a Oracle



Fuente: http://blogs.oracle.com/stevenChan/2009/05/

En todo el Aplicativo tendremos en total un número de 8 Ventanas distintas que serán la interfaces de referencia para el Usuario; Las cuales están divididas en 4 tipos que son:

- 1 Venta denominada “Menú Principal”
- 3 Ventanas que son denominadas “De Trabajo”
- 3 Ventanas que se denominas “Invocación Gráfica”
- 1 Ventana que se denomina “Bloqueos”

Así mismos contamos con 6 Ventanas que no requieren interacción con el Usuario ya que son reportes gráficos y Estadísticos para una mejor compresión del Usuario las mismas que se llevan a cabo a través de la herramienta REPORTS; y se dividen en 2 grupos:

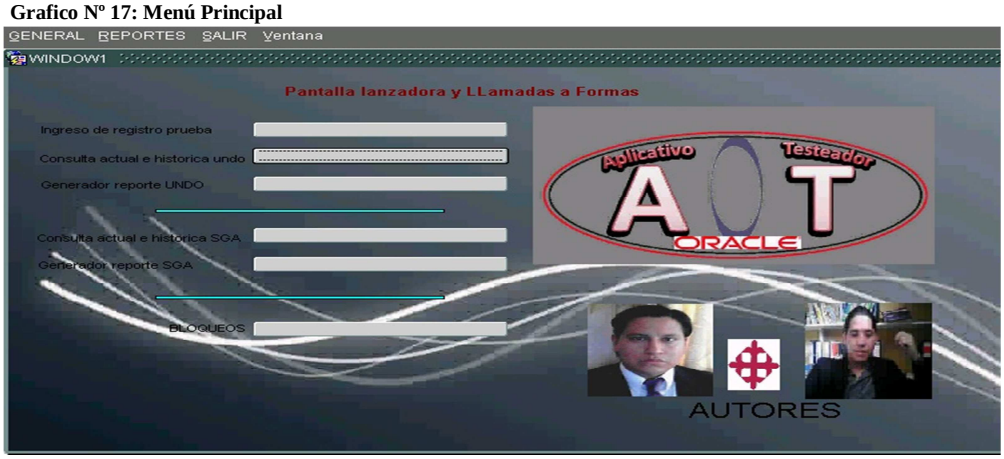
- 3 Reports denominados “Reporte Gráfico”
- 3 Reports Llamados “Reporte Estadístico”

4.1 MENU PRINCIPAL

Esta ventana es la primera por la cual el Usuario se relacionara y como su nombre lo indica presentara el Menú de los tipos de Acciones que se realizara; los cuales son los que se definió en el Alcance de este documento como son: Rendimiento, Control, Seguridad.

Detrás de esta ventana es bueno puntualizar que se cuenta con un script, con sentencias DML para la Base de Datos que no han sido enteramente trabajadas ni cargadas, para así poder ser aprovechada el Aplicativo TESTEADOR en su totalidad, generando comandos INSERT que mantienen COMMIT y ROLLBACK y así alimentar la base de información y darle operativiliadad que alteren los valores de los parámetros y le den manejo para observar su comportamiento.

Grafico Nº 17: Menú Principal

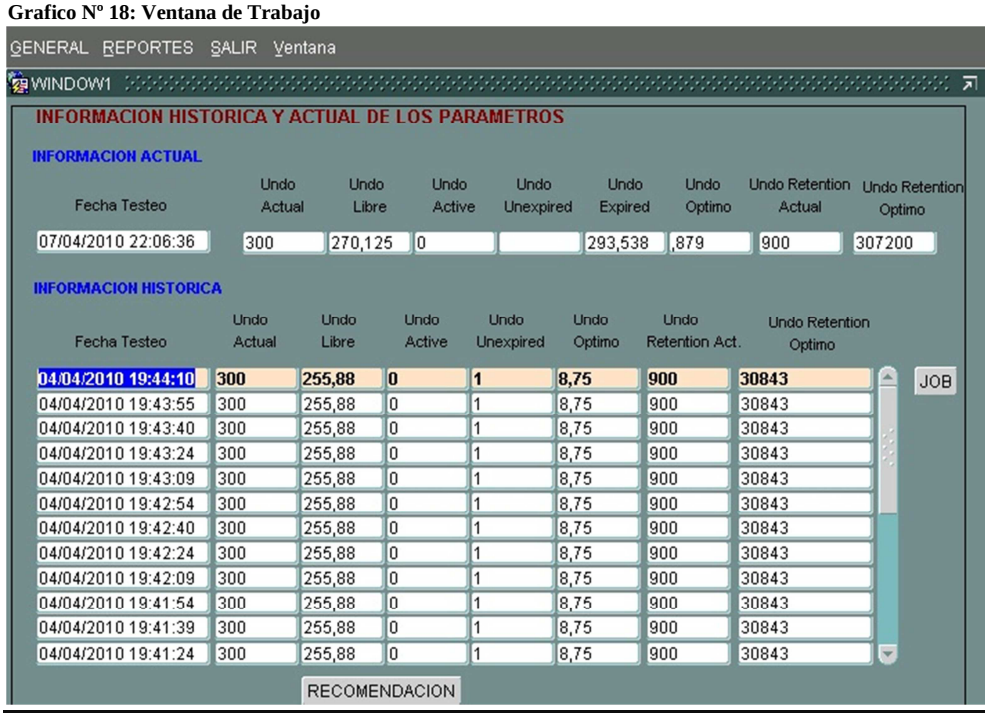


En esta ventana como trabajo oculto (al levantarse genera un script por debajo) alimenta a una tabla que crea el Aplicativo TESTEADOR denominada HISTORICO, con los valores que ya tiene la Base de Datos, ya que después las otras ventanas, reportes y Scripts anteriormente detallados trabajaran con esta tabla para su funcionamiento.

4.2 VENTANA DE TRABAJO

Esta ventana tiene la misma estructura entre las 3 que se determina a través del Menú Principal, por lo que esta es la iteración siguiente de la selección que se lleve a cabo en la primera ventana del aplicativo, en ella se describe y se aprecia los parámetros que conlleva cada selección, donde también se refleja el valor que tiene como configuración en la parte superior y los primeros resultados del test que se formulan de los valores que se revisan en la parte inferior de la ventana, en el multiregistro en el cual se carga los valores que se cambian en los parámetros allí detallados por cada 5 segundos o 10 segundos dependiendo de la recurrencia de este dato, donde si es de impacto alto este modifica el resumen que está en la parte superior, el usuario podrá apreciar el ingreso o mejor dicho la generación de cada registro en el multiregistro por cada sección de tiempo al hacer Click en el Button (Botón) que se encuentra en la parte superior derecha de la Ventana de Trabajo.

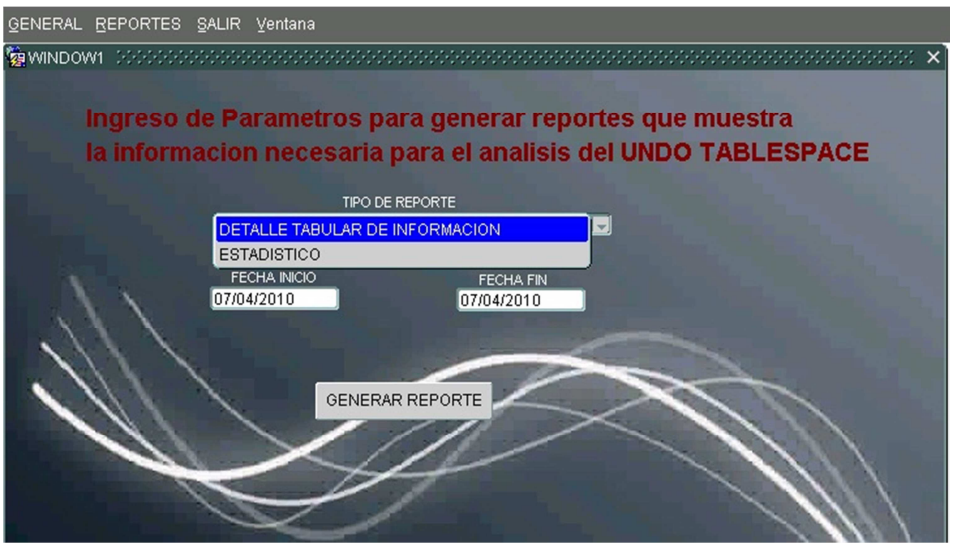
Grafico Nº 18: Ventana de Trabajo



4.3 INVOCACIÓN GRÁFICA

Esta ventana es de control, selección y de invocación de los Reportes que se generan y cada una enlaza cada ventana de trabajo y los reportes que contienen la función de entre los 3 tipos de Test que realiza el Aplicativo (Rendimiento, Control y Seguridad).

Grafico Nº 19: Invocación Gráfica



4.4 REPORTE GRÁFICO

En el se genera una Ventana con invocación REPORT para mostrar gráficos de tipo Vector (como electrocardiograma en la rama Médica), para tener un panorama claro del comportamiento que ha tenido la Base de Datos en el tipo de Test que realizo, cualquiera de los tres que requiera, al igual que un grafico de pastel para la Generalidad.

Grafico Nº 20: Reporte Gráfico Vector

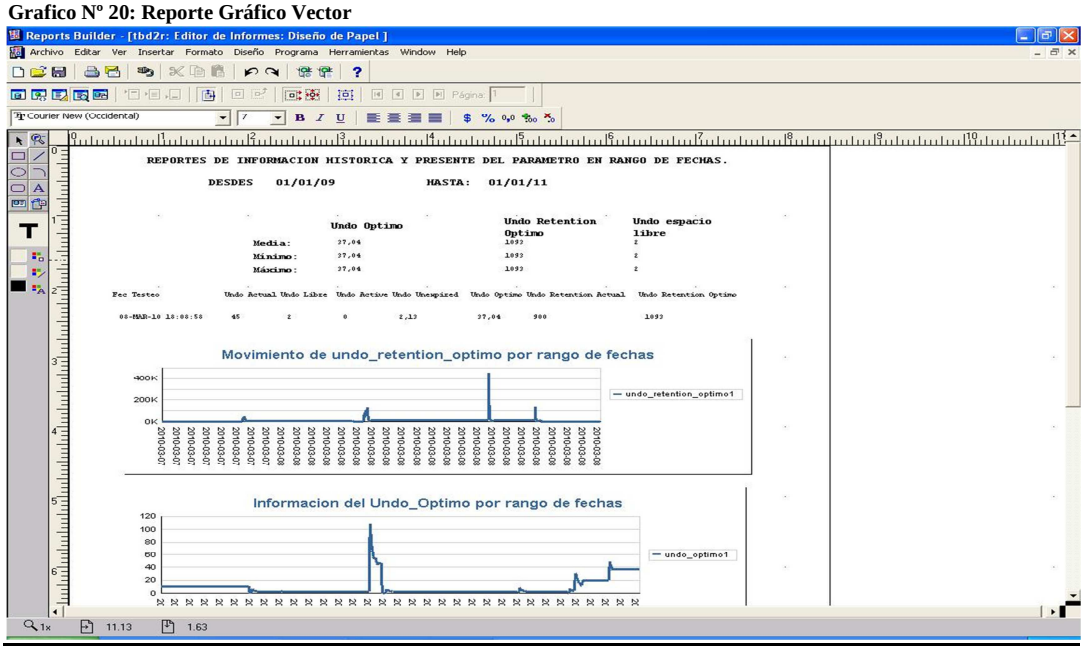
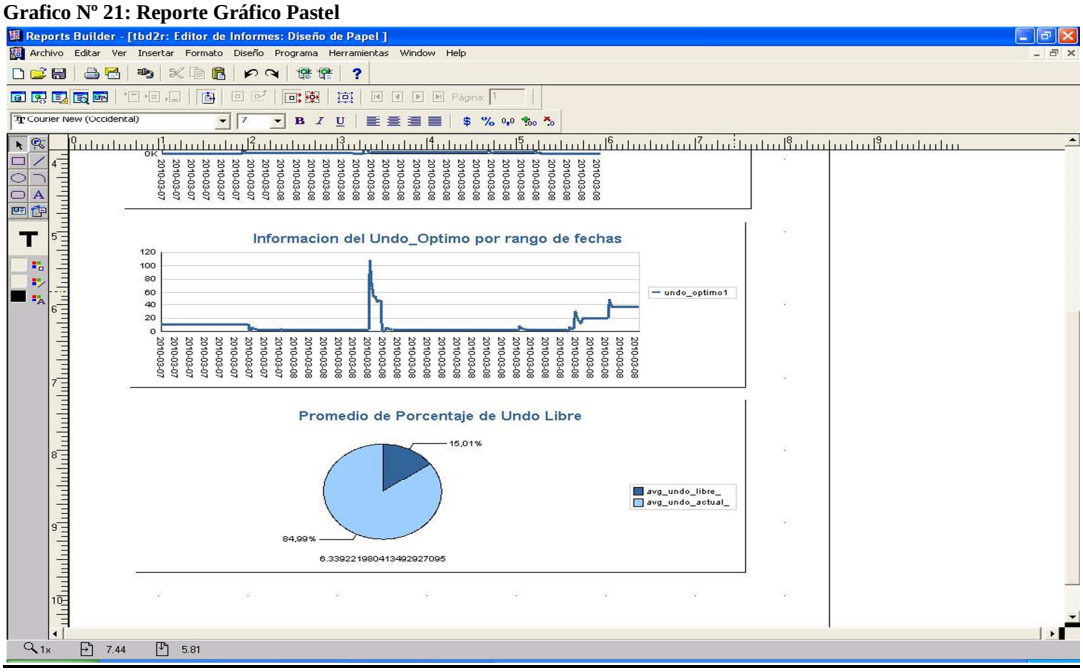


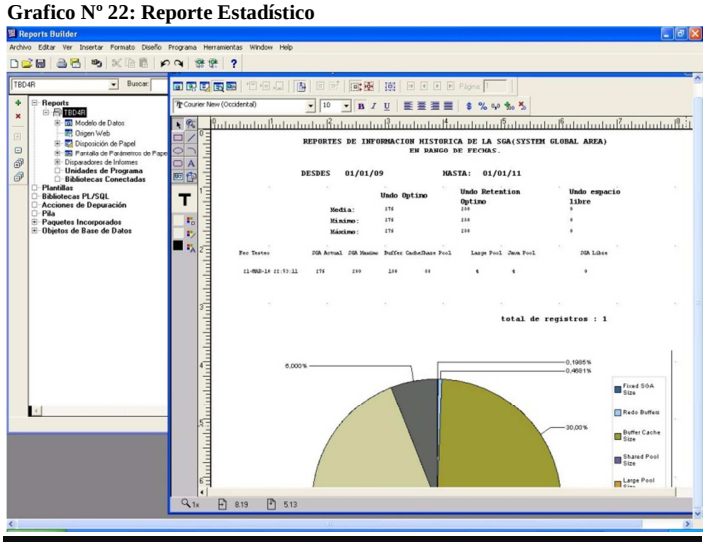
Grafico Nº 21: Reporte Gráfico Pastel



4.5 REPORTE ESTADISTICO

Este al igual que el anterior es generado por Report y muestra de manera histórica lo detallados de los valores a un nivel mas especifico y desmenuzado como valores tabulares para un representación más estadística de ellos.

Grafico Nº 22: Reporte Estadístico



4.6 BLOQUEO

En esta se demuestra tubularmente los bloqueos que se generan, donde cada vez que se refresca a través del botón se alimentara de bloqueos que se generen o de los que son liberados, y emite reporte de los mismo como pastel.

Se puede ver que esta enlazado también la herramienta de reportes, el cual se puede enviar como parámetros las fechas de inicio y de fin.

Grafico Nº 23: Bloqueo

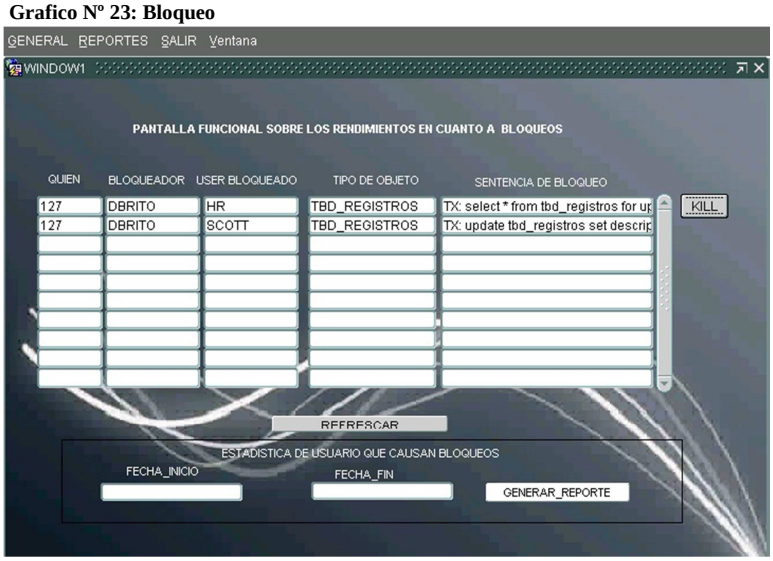
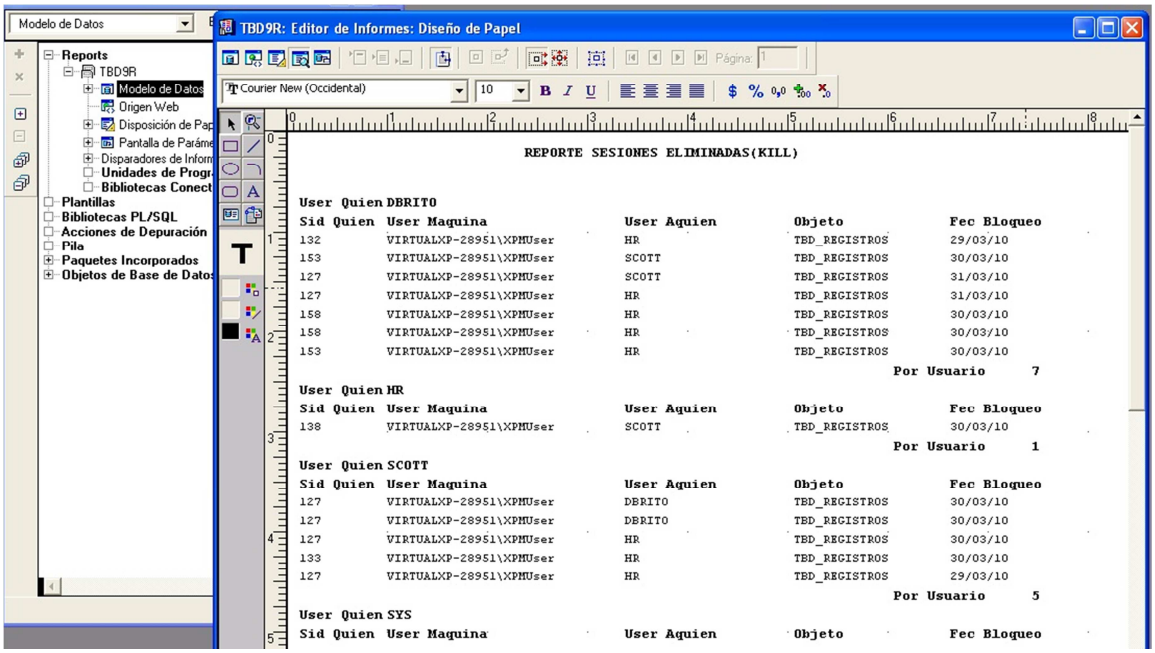


Grafico Nº 24: Reporte. Bloqueo


Modelo de Datos
Reports
TBD9R
Modelo de Datos
Origen Web
Plantillas
Bibliotecas PL/SQL
Acciones de Depuración
Pila
Paquetes Incorporados
Objetos de Base de Datos
TBD9R: Editor de Informes: Diseño de Papel
Courier New (Occidental)
10
REPORTE SESIONES ELIMINADAS(KILL)
User Quien DBRITO
Sid Quien User Maquina User Aquien Objeto Fec Bloqueo
132 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 29/03/10
153 VIRTUALXP-28951\XPMUser SCOTT TBD_REGISTROS 30/03/10
127 VIRTUALXP-28951\XPMUser SCOTT TBD_REGISTROS 31/03/10
127 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 31/03/10
158 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 30/03/10
158 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 30/03/10
153 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 30/03/10
Por Usuario 7
User Quien HR
Sid Quien User Maquina User Aquien Objeto Fec Bloqueo
138 VIRTUALXP-28951\XPMUser SCOTT TBD_REGISTROS 30/03/10
Por Usuario 1
User Quien SCOTT
Sid Quien User Maquina User Aquien Objeto Fec Bloqueo
127 VIRTUALXP-28951\XPMUser DBRITO TBD_REGISTROS 30/03/10
127 VIRTUALXP-28951\XPMUser DBRITO TBD_REGISTROS 30/03/10
127 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 30/03/10
133 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 30/03/10
127 VIRTUALXP-28951\XPMUser HR TBD_REGISTROS 29/03/10
Por Usuario 5
User Quien SYS
Sid Quien User Maquina User Aquien Objeto Fec Bloqueo

6. ESTUDIO DE COSTO

6.1 Costo y Sueldos

Para poder llevar a cabo el proyecto se debe de tener en cuenta los diferentes factores económicos que se deben incurrir para el desarrollo del mismo.
Primeramente más que nada y de importancia esta la adquisición de la licencia de la Base de Datos sobre la cual se Trabaja como es Oracle, la que se detalla a continuación.

Licenciamiento Oracle……..$ 30000,^{00}

Para la parte de elaboración; como es el diseño, programación, implementación, puesta en marcha y las pruebas respectivas, se es necesario valorar el recurso humano (sueldos), como se ve en la tabla siguiente.

sueldos	Mensuales	T. de Trab	N de Req	Total
Diseño	4000	15 Días	2	$ 4.000,00
Programación	3600	1 Mes	2	$ 7.200,00
Implementación	2000	1 Semana	3	$ 1.500,00
Pruebas	2000	1 Mes	1	$ 2.000,00
		Gasto Salarial		$ 14.700,00

T. de Trab: Tiempo de Duración de la Fase
N de Req: Numero de Requerimientos (RRHH)

Por último el hardware (equipos), que se necesitaran para lo antes detallado como el caso de un servidor para la instalación de la Base de Datos, de portátiles para acceder a ellas tanto como administradores y/o usuarios, y de los componentes necesarios para elaborar una red de comunicación entre lo ya detallado.

Hardware	Cant.	Precio U	Precio T
Servidor	1	$ 2.500,00	$ 2.500,00
Laptop	2	$ 1.500,00	$ 3.000,00
Red Inalámbrica	1	$ 500,00	$ 500,00
	Gasto de Equipos		$ 6.000,00

5.2 Relación Costo Beneficio

Costo:

Los costos del aplicativo se detallaros en el ítem anterior y además de este costo económico que se genera se debe acotar el costo de tiempo que genera la espera del mismo que es de 3 meses más, el colchón de precaución de u mes que da de un total de 4 meses.

Beneficio:

1. Los costos de este proyecto son mínimos comparado a la reducción de falos que se pueden ocasionar en la base de datos y problemas de referencias de los bloques de datos.

2. Se mantiene un informe estadístico del comportamiento de la Base de Datos en sus parámetros. La ventaja es mantener informes estadísticos, los cuales ofrecen la oportunidad de hacer o diseñar planes estratégicos para atender situaciones que pueden ocasionar pérdida de datos o del performance de la Base de Datos.

3. El sistema nos generara valores los parámetros que permitan optimizar la base de datos en 100% sin tomar en cuenta la robustez del Hardware sobre la que se trabaja.

CONCLUSIÓN

Como conclusión podemos llegar que el aplicativo Testeador servirá para toda base de Datos Oracle que necesite ser optimizada por la alta demanda que ella maneja, por lo que se potencializa con la Base de Datos que ya tienen un tiempo de vida alto, con relación a la operatividad, basado principalmente en los parámetros que referencian: Rendimiento, Control, Seguridad.

Todos estos relacionados con los parámetros de la característica de UNDO, de los parámetros que manejan y rigen la SGA.

Y de control como son los bloqueos por parte de los usuarios.

GLOSARIO DE TERMINOS

DEFOULT: Valores, procedimientos o pasos que vienen predefinidos por la instalación para llevarse a cabo sin consultar al usuario de la misma es aplicada comúnmente en las configuraciones recomendadas y no personalizadas.

CLIENTE/SERVIDOR: Arquitectura que consiste en la comunicación de un terminal por medio de peticiones (cliente), a otra que mantiene los servicios (servidor).

ESCALABILIDAD: Es la propiedad deseable de un sistema, red, procedimiento, base de datos, que indica la habilidad de extender su margen de operaciones sin perder calidad, es decir hacerse más grande sin perder las virtudes de los servicios ofrecidos

MULTIPLATAFORMA: Base de Datos que puede funcionar en diversas plataformas o sistemas operativos

PARCHES: Correcciones o cambios que se les aplican para mejoras o nuevas implementaciones de versiones anteriores o de una misma con incorporaciones.

COMPILAR: Es el proceso por el cual se traducen las instrucciones escritas en un determinado lenguaje de programación al lenguaje de máquina.

COMANDOS: es una instrucción u orden que el usuario proporciona a un sistema informático en este caso Base de Dato.

PERFORMANCE: Funcionamiento, Desempeño que mantiene la Base de Datos.

OFFLINE: Que se encuentra fuera de línea esto quiere decir que no está operando o está apagado, no eléctricamente sino que los servicios no los tiene disponible.

CLAVE PRIMARIA: Es el código que identifica un registro de cada tabla este es único para cada tabla.

CICLICO: Que se repite en un orden inmutable y que se caracteriza por el retorno a la posición inicial.

INSTANCIA: Es lo que se sube o se baja de la Base de Datos prácticamente lo que se carga para que esta pueda operar.

ARBOL B+: Representa una colección de datos ordenados de manera que se permite una inserción y borrado eficientes de elementos. Es un índice, multinivel, dinámico, con un límite máximo y mínimo en el número de claves por nodo.

CLUSTER: Es un conjunto contiguo de sectores que componen la unidad más pequeña de almacenamiento de un disco. Los archivos se almacenan en uno o varios clústeres, dependiendo de su tamaño de unidad de asignación.

ALGORITMO LRU: Es un algoritmo de reemplazo de páginas en memoria y se basa en la terminología de la sus sigla LRU (Last Recent Use), "La Ultima Recientemente Usada".

ASINCRONO: Tipo de comunicación donde cada byte se transmite del emisor al receptor de modo independiente. Hace referencia al suceso que no tiene lugar en total correspondencia temporal con otro suceso.

SERVIDOR: Cualquier ordenador de una red que proporciona servicios a otras estaciones de la red.

DIAGNOSIS: Acto de identificar el mal de algo o daño de un proceso o su distorsión o falla.

CABECERA: Parte de un Bloque, archivo, tabla, etc. Que mantiene información del mismo tanto de su ubicación, características y configuración.

SENTENCIA: Texto de lenguaje que describe una acción que se debe ejecutar.

KERNEL: El núcleo, parte esencial de la Base de Datos que provee los servicios más básicos.

SCRIPTS: (traducción literal es guión) o archivo de órdenes o archivo de procesamiento por lotes, es un programa usualmente simplel núcleo.

BIBLIOGRAFÍA

http://www.ajpdsoft.com/

http://www.blogs.oracle.com

http://www.dba-oracle.com

http://www.download.oracle.net

http://www.educiencia.org

http://www.forosdelweb.com

http://www.napolifirewall.com

http://www.oracle.com

http://www.oraclecuresebooks.com

http://www.wikilearning..com

- ***Información sobre Oracle Forms Developer 10g***

http://www.monografias.com/trabajos25/oracle/oracle.shtml?monosearch

- ***Información sobre Oracle Reports 10g***

http://tecnoxml.wikidot.com/oracle&usg=__LuUqO1ztmV-5QyqR4_lyiEaowUs=&h=337&w=600&sz=33&hl=es&start=24&um=1&itbs=1&tbnid=PWWs7TSiNv2S1M:&tbnh=76&tbnw=135&prev=/images%3Fq%3Dvistas%2Bdinamicas%2Boracle%26start%3D20%26um%3D1%26hl%3Des%26sa%3DN%26ndsp%3D20%26tbs%3Disch:1

ANEXO A.

Manual de Instalación A.O.T Herramientas

Objetivos:

El presente manual de instalación contiene todos los pasos necesarios para instalar la base de datos Oracle la cual se necesitara para la implementación del sistema A.O.T (Aplicativo Oracle Testeador).

El sistema de instalación tiene varios pasos a considerar.

Para poder implementar el aplicativo A.OT se necesitaran los siguientes programas instalados:

- Sistema Gestor de Bases de Datos : Oracle Database 10g.
- Oracle Forms Developer10g
- Oracle Reports.

Uso de los programas descritos anteriormente

Oracle Database 10g es el motor de Base de Datos donde se almacena los datos en forma estructurada y su diccionario de datos.

Con **Oracle Forms** Developer10g se desarrollo la solución tecnológica y se creó los formulario que invocaran a los reports.

Con **Oracle Reports10g** se realizo el diseño de los reportes.

Proceso de Instalación.

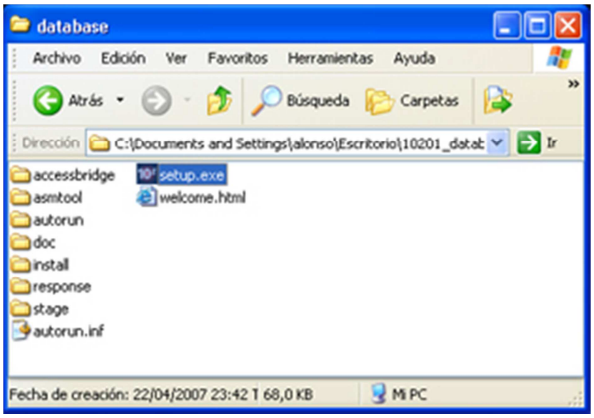
Manual de instalación de Oracle Database 10g.

A continuación la instalación y configuración de las opciones básicas de Oracle 10g en Windowx XP (válido para cualquier versión de Windows: Windows 2000, Windows 2003, etc).

1.- Se debe hacer una verificación de los requerimientos del sistema.

2.- Deshabilitar el firewall.

3.- Buscar en el dvd con el paquete de Oracle 10g carpeta que contiene el instalador de la base de datos, dar un clic en el icono ***SETUP.exe***



2.- Aparece la pantalla de método de Instalación, escoger la opción de ***Instalación Básica,*** Nombre de la Base de Datos Global: nombre con el que se identificará la base de datos, máximo 8 caracteres, Contraseña de Base de Datos: contraseña que se asignará a los usuarios SYS, SYSMAN y DBSNMP.

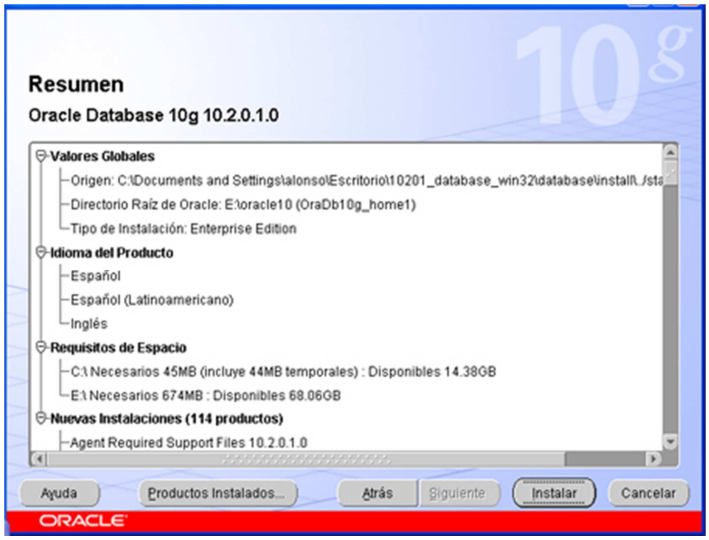
Seleccione este método de instalación porque la instalación base de datos es rapida. Este método necesita una intervención mínima del usuario. Instala el software y opcionalmente, crea una base de datos de uso general con el esquema SAMPLE y el tablespace EXAMPLE, con la información especificada en la pantalla inicial.

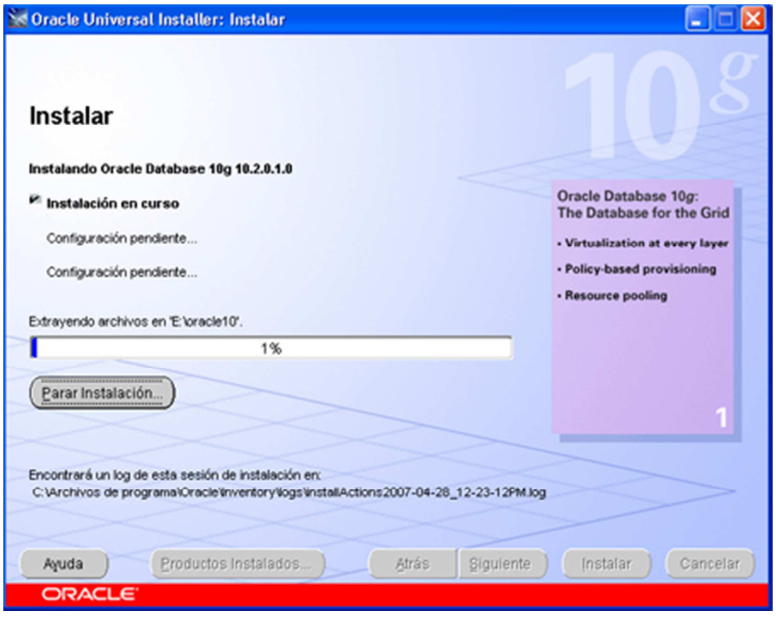
Tras rellenar estos datos pulsaremos "Siguiente" para continuar con la instalación

3.- Aparece una barra de progreso indicando que se está preparando para instalar:

4.- El asistente de instalación verificará si el entorno cumple todos los requisitos mínimos para instalar y configurar los productos seleccionados. Pulsar "***Siguiente***" para continuar:

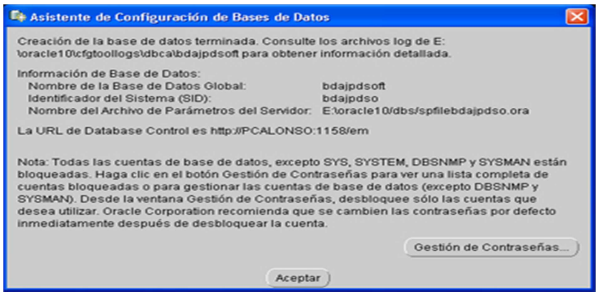
5.- Se presenta una ventana indicando los productos Oracle Database 10g 10.2.0.1.0 que se instalarán. Pulsaremos "***Instalar***" para iniciar la instalación:



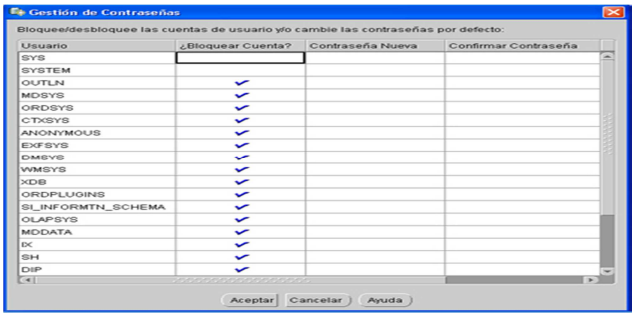


6.- Creación de la base de datos

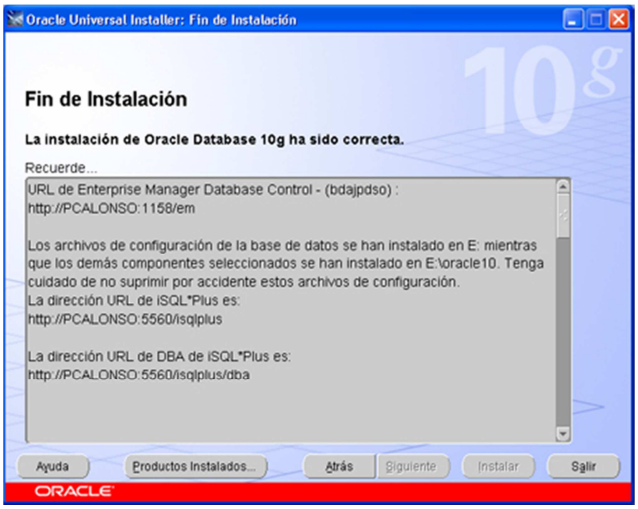
7.- Una vez creada la base de datos, se procede a gestionar las contraseñas de cada usuario (recomendable).



Se puede desbloquear usuarios, para ello pulsaremos en "Gestión de Contraseñas":



Al finalizar la instalación, el asistente mostrará una serie de datos, como e l URL del Enterprise manager y del ISQL*Plus. Pulsar en "Salir" para terminar la instalación:



Pulsar en "***Sí***" para cerrar el asistente de instalación de Oracle Database 10g: Al final de la instalación se ejecuta automáticamente el internet Explorer y despliega una página de

configuración de Oracle llamada Enterprice Manager, donde se nos confirma que la instalación se ha realizado con éxito.

Introducir el usuario y la contraseña para el acceso, en nuestro caso utilizaremos el usuario "system". Pulsaremos el botón "Conectar":

1.2.2. Manual de instalación de Oracle Forms Developer10g.

Para comenzar debemos tener en cuenta de que odas las ventanas deben esta cerradas ya que la mayor memoria se encuentre disponible, para la correcta y eficaz instalación.

Debemos asegurarnos en tener un gran espacio en disco, como mínimo 256 MB.

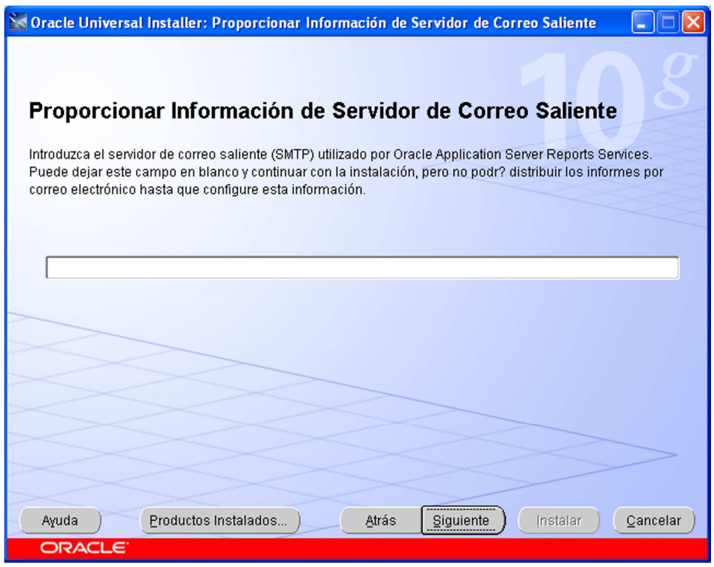
1.- especificar el nombre del ORACLE_HOME, para los productos que sería:

C:\oracle\dev\10.1.2.0.1.

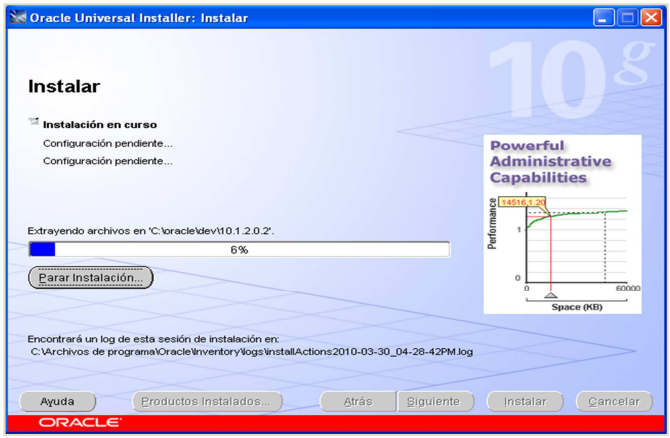
2.- Al terminar la carga seguimos con los tipos de instalación. En la cual nos dirigimos al completo ya que en futuro podríamos alguna herramienta y ya la tendríamos en nuestra maquina.

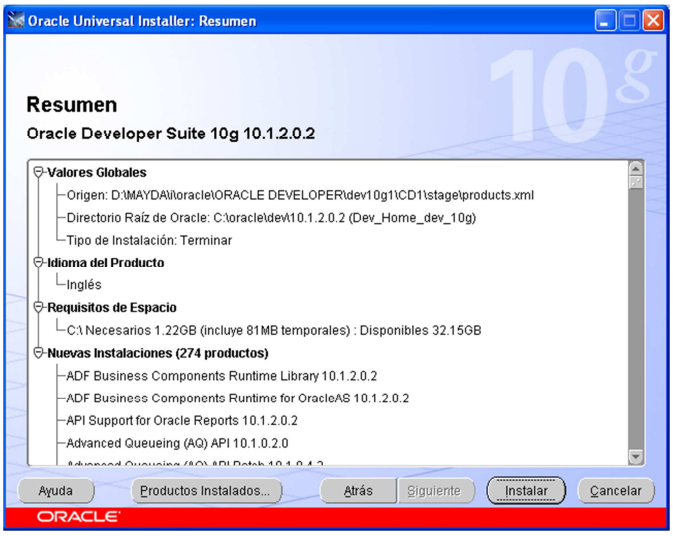
3-. En cuanto al idioma hacemos la elección por el ingles, siendo este mucho más completo.

4.- Cuando el instalador pida el tipo de instalación se quiere hacer, se debe escoger la instalación completa. En una pantalla adicional pedita la dirección sMTP de donde debemos enviar los reportes que Oracle genera, aquí no ponemos nada y damos click en siguiente.

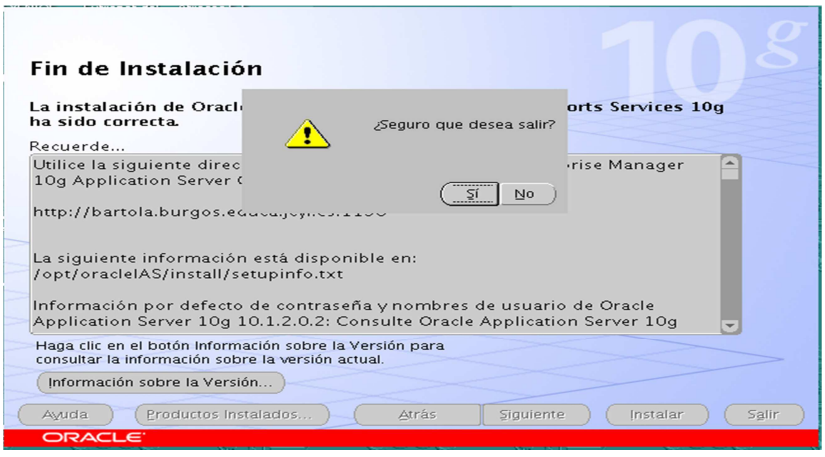


5.- El instalador del Developer, mostrara una ventana con el resumen de lo que se va ha instalar





6.- Se debe esperar hasta que este proceso termine, al cabo de este tiempo el instalador nos dirá que la instalación está completa y no ha existido ningún error. Damos clic en finalizar. Fin de la Aplicación.



Pasos posteriores a la instalación:

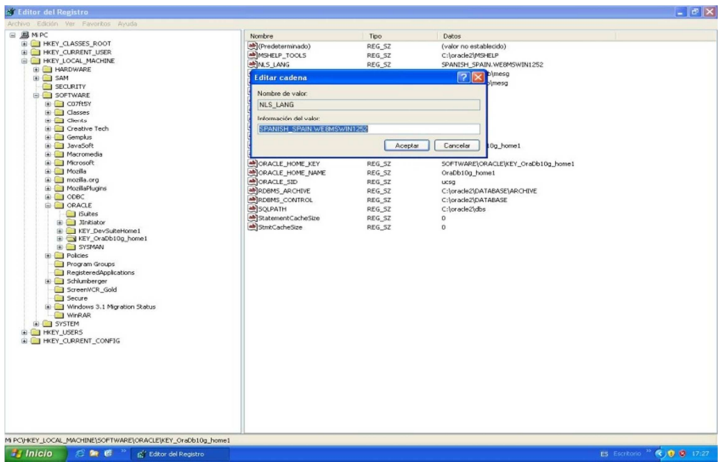
La maquina tiene que estar con estas consideraciones:

1.) Ambiente del regedit

Mi Pc\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_Devsuitehome1.

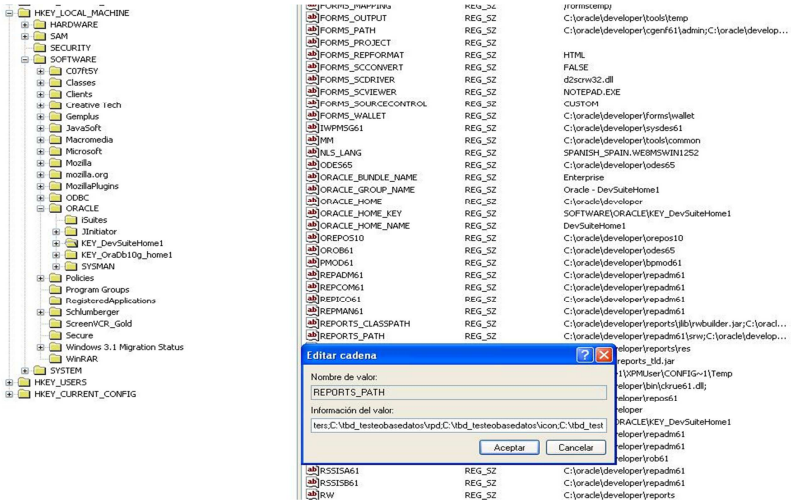
En el feature NLS_LANG..

SPANISH_SPAIN.WE8MSWIN1252



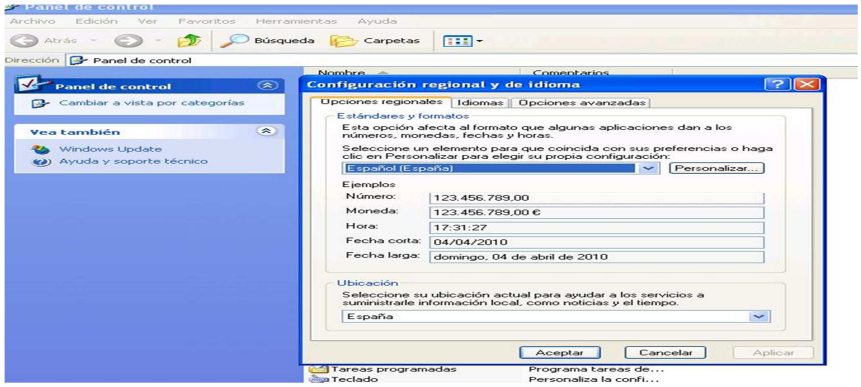
2.) Feature REPORTS_PATH.

C:\oracle\developer\repadm61\srw;C:\oracle\developer\cgenr61\admin\crretc;C:\oracle\developer\cgenr61\admin;C:\oracle\developer\reports\templates;C:\oracle\developer\reports\samples\demo;C:\oracle\developer\reports\integ;C:\oracle\developer\reports\printers;C:\tbd_testeobasedatos\rpd;C:\tbd_testeobasedatos\icon;C:\tbd_testeobasedatos\mnu



3.) Panel de Control. Windows

En las opciones generales de Idioma de Windows seleccionamos Español(España).



Una vez que la plataforma esta configurada, se darà paso al manual técnico

ANEXO B.

Manual Técnico A.O.T

Objetivos

Este manual técnico contendrá todos las unidades de programas(procedure-functions-packages) del aplicativo A.O.T.(Aplicativo Oracle Testeador), es decir, el código fuente puro, que servirá como referencia en caso de futuros avances o implementaciones para este software. Asimismo se detallara la funcionalidad de cada objeto creado y la ubicación donde se encuentra este objeto en el software antes descrito.

Para empezar con la descripción de objetos, nombraremos primero los de tipo tabla; tanto las tablas que se han creado para el aplicativo, como tablas que hemos utilizado del diccionario de datos de oracle.

Alcance

Este manual tiene el pròposito de explicar de manera general el concepto de los diferentes objetos del software implementado, los scripts que se muestran en este manual han sido ejecutados en

Oracle Database 10g, los formularios en Forms Developer y los reportes estadísticos mediante Oracle Reports 10g.

Este manual tiene también la finalidad de ayudar a identificar todas las características y funcionalidades de AOT para el fácil manejo de la persona u operador(DBA) que lo vaya a utilizar. Se detallara más adelante características técnicas tanto de hardware como software, versiones, etc… para el correcto funcionamiento-eficacia-eficiencia de este aplicativo.

Dirigido a que personas

Este manual Tecnico del AOT fue pensado para personas que tienen un nivel de conocimiento mediano o alto con el SISTEMA GESTOR DE BASE DE DATOS ORACLE VERSION 10G, más no para usuarios que se quieran iniciar en el estudio de este sistema de almacenamiento.

Lo que se debe conocer.-

El Producto Oracle Developer Suite 10g trae consigo las herramientas siguiente: Forms Developer10g y ReportsBuilder 10G. Basadas para el desarrollo y reporte de aplicaciones tradicionales cliente-servidor o para arquitecturas de tres capas.

Antes que nada, se indica que este aplicativo esta comprendido por ocho modulos(formularios) denominados de la siguiente forma: tbd0f..mmb al tbd8f..mmb. Estos modulos se los puede compilar mediante la herramientas Forms Developer.

Para el caso de reports, los reportes se encuentran denominados de la siguiente forma: tbd1r.rpt, tbd2r, tbd3r, tbd4r, tbd9r. Estos serán compilados mediante la herramienta Reports Builder.

La herramienta Forms Developer se enlazara con el reports mediante procedimientos creados y descritos posteriormente en este manual.

Especificaciones Tecnicas.

Para AOT(Aplicativo Oracle Testeador) se necesitara los siguientes características en cuanto a software y hardware se refiere:

Requerimientos Mínimos:

- Licencia Oracle versión 10G con su Base de Datos
- PL/SQL DEVELOPER versión 4.0 o superior (Opcional).
- Sistema operativo -> Windows 2000, Windows XP (necesariamente).
- Paquete de Producto ORACLE DEVELOPER SUITE 10G
  - Forms Developer 10G.
  - Report Builder 10G.

- Navegadores: Explorer 7.0 o superior, en su defecto:
  - Mozzila firefox(preferiblemente 3.0.3) las actuales se necesitaría bajar un parche adicional al explorador

Las características de hardware en cuanto a memoria se refiere será de 512 Mb en RAM o superior, el disco duro se encuentran determinado por los requerimientos mínimos del navegador seleccionado por el usuario. Cabe señalar, que este sistema no requiere de ningúna herramienta de las antes señalada.

Volviendo a recalcar que este aplicativo esta realizado bajo las herramientas antes mencionadas, lo cual indica su completa integridad y relación con productos Oracle. Es decir, cualquier versión posterior a la utilizada(10g), este aplicativo podrá ser completamente utilizado sin ningún problema de compatibilidad en versiones.

A continuación se detallara nombre y versión.
Gestor de Base de Datos:

Oracle Database – versión 10g
Herramienta de Desarrollo: PAQUETE ORA DEVELOPER SUITE 10G
Oracle Forms – versión 10g
Herramienta de Reportería:
Oracle – versión 10g

Este aplicativo trabajara de manera frecuente con las tablas propias del diccionario de oracle; por tal razón, se detallara una breve síntesis o esquema conceptual sobre este diccionario.

El diccionario de datos es una parte fundamental en toda base de datos Oracle. Está formado por tablas, vistas y paquetes a los que se puede acceder para obtener información. Las tablas se crean automáticamente durante la instalación y permiten saber:

- Estructura lógica y física de la BD.
- Los usuarios de la BD.
- Restricciones de integridad sobre las tablas de la DB
- Espacio asociado a cada objeto en la DB y la cantidad que se está utilizando por los distintos objetos creados por los usuarios de la DB.

Entre otras cosas, las más importantes las antes mencionadas. Sin embargo, este aplicativo no se responsabiliza por cualquier variación de valor en su entorno de diccionario de datos. Ya que AOT(Aplicativo Oracle Testeador) para mostrar los valores óptimos en cada uno de sus parámetros a testear solamente hace “select’s” o consultas a este diccionario màs no modificaciones o eliminaciónes.

Como se ha dicho en este manual anteriormente, una vez visto los valores optimos del parámetro, el usuario(DBA) o persona que utiliza este aplicativo podrà cambiar ese valor de parámetro mediante la sentencia alter.

Se indica también que como AOT generará reportes, se deberá realizar como especie de artificio el levantamiento del SERVIDOR DE REPORTES, servidor que se instala de forma obligatoria en la instalación del PRODUCTO: ORACLE DEVELOPER SUITE 10G. Más adelante(NUMERAL 7) se conocerá tal artificio para que no se genere ningún error al momento de cargar el aplicativo.

A continuación se tendrá la creación de tablas(creadas para el aplicativo):

Ubicación: Creadas en el esquema SYS

Es recomendable eliminar las tablas, para evitar inconvenientes en el momento de insertar datos en las mismas.

```
Drop table tbd_hist_testeo

Drop table tbd_hist_sga

Drop table tbd_bloqueo

Drop table tbd_hist_testeo

Drop table tbd_registros


create table tbd_hist_testeo(
cod_testeo                  number(9)     constraint tbd_hist_testeo_n1 not null,
fec_testeo                  date          constraint tbd_hist_testeo_n3 not null,
undo_actual          number(9,2)    constraint tbd_hist_testeo_n6 not null,
undo_libre                  number(9,2)   constraint tbd_hist_testeo_n7 not null,
undo_active                 number(9,2)   constraint tbd_hist_testeo_n8 not null,
undo_unexpired              number(9,2)   constraint tbd_hist_testeo_n9 not null,
undo_expired                number(9,2)   constraint tbd_hist_testeo_n10 not null,
undo_optimo                 number(9,2)   constraint tbd_hist_testeo_n11 not null,
undo_retention_actual number(9,2)    constraint tbd_hist_testeo_n12 not null,
```

```
undo_retention_optimo        number(9,2)    constraint tbd_hist_testeo_n13 not null,
fec_ingreso                  date           constraint tbd_hist_testeo_n14 not null,
usuario_ingreso              varchar2(20)   constraint tbd_hist_testeo_n15 not null
);


create table tbd_hist_sga(

fec_testeo                   date           constraint tbd_hist_sga_n1 not null,
sga_actual                   number(9,2)    constraint tbd_hist_sga_n2 not null,
sga_max_size                 number(9,2)    constraint tbd_hist_sga_n3 not null,
buffer_cache                 number(9,2)    constraint tbd_hist_sga_n4 not null,
Shared_pool_size             number(9,2)    constraint tbd_hist_sga_n6 not null,
large_pool                   number(9,2)    constraint tbd_hist_sga_n7 not null,
java_pool                    number(9,2)    constraint tbd_hist_sga_n8 not null,
sga_libre                    number(9,2)    constraint tbd_hist_sga_n9 not null,
fec_ingreso                  date           constraint tbd_hist_testeo_n14 not null,
usuario_ingreso              varchar2(20)   constraint tbd_hist_testeo_n15 not null
);

create table tbd_bloqueo(
SID_QUIEN            number(5),
USER_QUIEN    varchar2(15),
USER_MAQUINA         varchar2(30),
USER_AQUIEN  varchar2(30),
OBJETO               varchar2(15),
```

FEC_BLOQUEO DATE

)

create table tbd_registros

(

CODIGO NUMBER(10),

DESCRIPCION VARCHAR2(500)

)

Tablas utilizadas del diccionario de datos que sirvieron de información para el desarrollo de este aplicativo

V$SGAINFO;

v$sgastat;

dba_undo_extents;

dba_free_space;

v$undostat;

dba_tablespaces;

v$datafile;

v$lock

Modulo TBD2F.MMB

Código Fuente

Procedimiento PU_P_ALERTA

Enviara un mensaje en tiempo de ejecución, describiendo la cantidad de tamaño en el tablespace, en este caso de UNDO, y que debe poseer como “optimo” con la transacción que se esta realizando. (inserts de 10000 registros)

Program Units: PU_P_ALERTA

```
PROCEDURE pu_p_alerta IS
limite number:=100;
mensaje varchar2(300);
ALERTA NUMBER;
CURSOR c_espacio_libre
IS
SELECT tablespace_name,
ROUND(sum(bytes)/1024/1024,0) FREESPACE
FROM dba_free_space
WHERE tablespace_name IN ( select TABLESPACE_NAME from dba_tablespaces
where contents='UNDO')
group by tablespace_name;
CURSOR c_espacio_total
IS
select tablespace_name,
round(sum(BYTES/1024/1024),0) TOTALSPACE
FROM dba_data_files b
WHERE tablespace_name IN ( select TABLESPACE_NAME from dba_tablespaces
where contents='UNDO')
GROUP BY b.tablespace_name;
```

```
c_nombre VARCHAR2(20);
c_libre NUMBER(10);
c_total NUMBER(10);
v_bbdd VARCHAR(20);
BEGIN
SELECT name into v_bbdd from v$database;
-- ABRIMOS EL CURSOR Y NOS POSICIONAMOS EN LA PRIMERA LINEA
OPEN c_espacio_libre;
OPEN c_espacio_total;
FETCH c_espacio_libre INTO c_nombre,c_libre;
FETCH c_espacio_total INTO c_nombre,c_total;
-- EN CASO DE QUE EXISTA RESULTADO REALIZAMOS LAS COMPROBACIONES DE ESPACIO
WHILE c_espacio_libre%found
LOOP
-- comprobacion del tablespace ES MENOR DE limite MEGAS
IF (c_libre * 100) / c_total < limite THEN

mensaje := 'alerta de espacio '||v_bbdd||
'. El tablespace con nombre: ' || c_nombre || ' tiene un espacio libre optimo.' ||chr(10)||
'El tamaño libre restante es de: ' || c_libre || ' Megas';

SET_ALERT_PROPERTY('AL_MENSAJE',ALERT_MESSAGE_TEXT,MENSAJE);
  ALERTA := SHOW_ALERT('AL_MENSAJE');
END IF;
FETCH c_espacio_libre INTO c_nombre, c_libre;
FETCH c_espacio_total INTO c_nombre,c_total;
```

```
END LOOP;
CLOSE c_espacio_libre;
CLOSE c_espacio_total;

END;
```

Procedimiento mediante el cual se traerá toda la data actual en cuanto al parametro undo_retention se refiere, así también se calculara el tamaño optimo del undo_retention, segun la transacción que se realizara.

```
PROCEDURE pu_p_recupera_data_act IS
LN_UNDO_ACTUAL                          NUMBER(20,3):=0;
LN_UNDO_RETENTION_ACTUAL                NUMBER(20,3):=0;
LN_UNDO_RETENTION_OPTIMO                NUMBER(20,3):=0;
LN_UNDO_OPTIMO                          NUMBER(20,3):=0;
LN_UNDO_LIBRE                           NUMBER(20,3):=0;
LD_FEC_TESTEO                           DATE;
LN_UNDO_ACTIVE                          NUMBER(20,3):=0;
LN_UNDO_UNEXPIRED                       NUMBER(20,3):=0;
LN_UNDO_EXPIRED                         NUMBER(20,3):=0;
```

```
BEGIN

 BEGIN

        SELECT d.undo_size / (1024 * 1024) "ACTUAL UNDO SIZE [MByte]",

        Substr(e.VALUE,1,25) "ACTUAL UNDO RETENTION [Sec]",

        Round((d.undo_size / (To_number(f.VALUE) * g.undo_block_per_sec))) "UNDO
RETENTION OPTIMO [Sec]"

INTO LN_UNDO_ACTUAL,LN_UNDO_RETENTION_ACTUAL,LN_UNDO_RETENTION_OPTIMO

FROM   (SELECT Sum(a.bytes) undo_size

FROM   v$datafile a,

       v$tablespace b,

        dba_tablespaces c

WHERE  c.contents = 'UNDO'

AND c.status = 'ONLINE'

AND b.name = c.tablespace_name

AND a.ts# = b.ts#) d,

      v$parameter e,

      v$parameter f,

     (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec

      FROM   v$undostat) g

      WHERE  e.name = 'undo_retention'

     AND f.name = 'db_block_size';

 END;


BEGIN
```

```
SELECT   (To_number(e.VALUE) * To_number(f.VALUE) * g.undo_block_per_sec) / (1024 * 1024)
"Tamaño Optimo UNDO [MByte]"

INTO LN_UNDO_OPTIMO

FROM   (SELECT Sum(a.bytes) undo_size

          FROM   v$datafile a,

          v$tablespace b,

          dba_tablespaces c

         WHERE  c.contents = 'UNDO'

          AND c.status = 'ONLINE'

          AND b.NAME = c.tablespace_name

          AND a.ts# = b.ts#) d,

         v$parameter e,

         v$parameter f,

        (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec

         FROM   v$undostat) g

         WHERE  e.NAME = 'undo_retention'

         AND f.NAME = 'db_block_size'   ;

 END;


BEGIN

SELECT         Sum(dfs.bytes)

           / 1024

           / 1024 AS "Espacio Libre DataFile [MB]"

           INTO LN_UNDO_LIBRE

FROM          dba_free_space dfs, dba_tablespaces dts

WHERE dfs.tablespace_name = dts.tablespace_name
```

```
AND dts.contents = 'UNDO'
AND dts.status = 'ONLINE';
END;

 BEGIN
        SELECT SYSDATE AS fecha,
    active.active,
    unexpired.unexpired,
    expired.expired
INTO LD_FEC_TESTEO,LN_UNDO_ACTIVE,LN_UNDO_UNEXPIRED, LN_UNDO_EXPIRED
FROM   (SELECT Sum(bytes / 1024 / 1024) AS unexpired
      FROM   dba_undo_extents
      WHERE  status = 'UNEXPIRED') unexpired,
     (SELECT Sum(bytes / 1024 / 104) AS expired
      FROM   dba_undo_extents tr
      WHERE  status = 'EXPIRED') expired,
     (SELECT CASE
          WHEN Count(status) = 0
          THEN 0
          ELSE Sum(bytes / 1024 / 1024)
        END AS active
      FROM   dba_undo_extents
      WHERE  status = 'ACTIVE') active;
        END;

        BEGIN
```

```
:BL_T_HIST_TESTEO.FEC_TESTEO          :=LD_FEC_TESTEO;
:BL_T_HIST_TESTEO.UNDO_ACTUAL      :=LN_UNDO_ACTUAL;
:BL_T_HIST_TESTEO.UNDO_LIBRE:=LN_UNDO_LIBRE;
:BL_T_HIST_TESTEO.UNDO_ACTIVE:=LN_UNDO_ACTIVE;
:BL_T_HIST_TESTEO.UNDO_UNEXPIRED:=LN_UNDO_UNEXPIRED;
:BL_T_HIST_TESTEO.UNDO_EXPIRED:=LN_UNDO_EXPIRED;
:BL_T_HIST_TESTEO.UNDO_OPTIMO:=LN_UNDO_OPTIMO;
:BL_T_HIST_TESTEO.UNDO_RETENTION_ACTUAL:=LN_UNDO_RETENTION_ACTUAL;
:BL_T_HIST_TESTEO.UNDO_RETENTION_OPTIMO:=LN_UNDO_RETENTION_OPTIMO;


        END;
END;
```

Mediante un job se guardara cada 10 seg. la información en una tabla creada para este propósito, asimismo este procedimiento recuperara esa data y la mostrara en la pantalla de consulta del aplicativo.

```
PROCEDURE pu_p_recupera_data_hist IS
LN_UNDO_ACTUAL                          NUMBER(20,3):=0;
LN_UNDO_RETENTION_ACTUAL                NUMBER(20,3):=0;
LN_UNDO_RETENTION_OPTIMO                NUMBER(20,3):=0;
```

```
LN_UNDO_OPTIMO                                  NUMBER(20,3):=0;
LN_UNDO_LIBRE                                       NUMBER(20,3):=0;
LD_FEC_TESTEO                                       DATE;
LN_UNDO_ACTIVE                                  NUMBER(20,3):=0;
LN_UNDO_UNEXPIRED                               NUMBER(20,3):=0;
LN_UNDO_EXPIRED                                 NUMBER(20,3):=0;
BEGIN
 BEGIN
 SELECT d.undo_size / (1024 * 1024) "ACTUAL UNDO SIZE [MByte]",
          Substr(e.VALUE,1,25) "ACTUAL UNDO RETENTION [Sec]",
            Round((d.undo_size / (To_number(f.VALUE) * g.undo_block_per_sec))) "UNDO
RETENTION OPTIMO [Sec]"
INTO LN_UNDO_ACTUAL,LN_UNDO_RETENTION_ACTUAL,LN_UNDO_RETENTION_OPTIMO
FROM   (SELECT Sum(a.bytes) undo_size
         FROM   v$datafile a,
           v$tablespace b,
           dba_tablespaces c
           WHERE  c.contents = 'UNDO'
           AND c.status = 'ONLINE'
           AND b.name = c.tablespace_name
           AND a.ts# = b.ts#) d,
             v$parameter e,
             v$parameter f,
            (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec
             FROM   v$undostat) g
             WHERE  e.name = 'undo_retention'
```

```
                AND f.name = 'db_block_size';
 END;

BEGIN
 SELECT   (To_number(e.VALUE) * To_number(f.VALUE) * g.undo_block_per_sec) / (1024 * 1024)
"Tamaño Optimo UNDO [MByte]"
INTO LN_UNDO_OPTIMO
FROM   (SELECT Sum(a.bytes) undo_size
           FROM   v$datafile a,
            v$tablespace b,
           dba_tablespaces c
          WHERE  c.contents = 'UNDO'
          AND c.status = 'ONLINE'
          AND b.NAME = c.tablespace_name
          AND a.ts# = b.ts#) d,
     v$parameter e,
     v$parameter f,
     (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec
      FROM   v$undostat) g
     WHERE  e.NAME = 'undo_retention'
     AND f.NAME = 'db_block_size'        ;

END;

BEGIN
SELECT         Sum(dfs.bytes)
```

```
            / 1024
            / 1024 AS "Espacio Libre DataFile [MB]"
            INTO LN_UNDO_LIBRE
FROM     dba_free_space dfs, dba_tablespaces dts
WHERE dfs.tablespace_name = dts.tablespace_name
AND dts.contents = 'UNDO'
AND dts.status = 'ONLINE';
END;

 BEGIN
SELECT SYSDATE AS fecha,
    active.active,
    unexpired.unexpired,
    expired.expired
    INTO
LD_FEC_TESTEO,LN_UNDO_ACTIVE,LN_UNDO_UNEXPIRED, LN_UNDO_EXPIRED
FROM   (SELECT Sum(bytes / 1024 / 1024) AS unexpired
          FROM   dba_undo_extents
         WHERE  status = 'UNEXPIRED') unexpired,
          (SELECT Sum(bytes / 1024 / 104) AS expired
           FROM   dba_undo_extents tr
          WHERE  status = 'EXPIRED') expired,
          (SELECT CASE
           WHEN Count(status) = 0
           THEN 0
           ELSE Sum(bytes / 1024 / 1024)
```

```
            END AS active
FROM   dba_undo_extents
WHERE  status = 'ACTIVE') active;
END;


          BEGIN
                INSERT INTO TBD_HIST_TESTEO (
                      COD_TESTEO,
                      FEC_TESTEO,
                      UNDO_ACTUAL,
                      UNDO_LIBRE,
                      UNDO_ACTIVE,
                      UNDO_UNEXPIRED,
                      UNDO_EXPIRED,
                      UNDO_OPTIMO,
                      UNDO_RETENTION_ACTUAL,
                      UNDO_RETENTION_OPTIMO,
                      FEC_INGRESO,
                      USUARIO_INGRESO )
                      VALUES
                      (
                      1 ,
                      LD_FEC_TESTEO,
                      LN_UNDO_ACTUAL,
                      LN_UNDO_LIBRE,
                      LN_UNDO_ACTIVE  ,
```

```
                    LN_UNDO_UNEXPIRED,
                    LN_UNDO_EXPIRED  ,
                    LN_UNDO_OPTIMO  ,
                    LN_UNDO_RETENTION_ACTUAL,
                    LN_UNDO_RETENTION_OPTIMO,
                    SYSDATE  ,USER );
        END;
        COMMIT;
END;
```

Modulo TBD3F.MMB

Codigo Fuente

Procedimiento que llamara al reporte, a este Procedimiento se le enviara un parametro, el cual indicara el numero de reporte a levantarse.

```
PROCEDURE pu_p_genera_reporte(PN_REPORTE IN number) IS
 repid                  REPORT_OBJECT;
 v_rep                  varchar2(100);
 rep_status varchar2(20);
 List_id          ParamList;
 pl_name                VARCHAR2(10) := 'misp';
```

```
begin

        List_id := Get_Parameter_List('input_params');
        IF NOT Id_Null(List_id) THEN
                Destroy_Parameter_List(List_id);
        END IF;
    List_id := Create_Parameter_List('input_params');

    if PN_REPORTE=1 then
        repid:= find_report_object('TBD1R');
    end if;

    if PN_REPORTE=2 then
        repid:= find_report_object('TBD2R');
    end if;

SET_REPORT_OBJECT_PROPERTY(repid, REPORT_OTHER,'paramform=no');
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_EXECUTION_MODE,BATCH);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_COMM_MODE,SYNCHRONOUS);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESTYPE,CACHE);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESFORMAT,'html');
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_SERVER,'rep_my_computer');

Add_Parameter(List_id,  'PD_FECHA_INICIO',TEXT_PARAMETER,:TBD_TESTEOS.FECHA_INICIO);
```

```
        Add_Parameter(List_id,  'PD_FECHA_FIN',TEXT_PARAMETER,:TBD_TESTEOS.FECHA_FIN);
 Add_Parameter(List_id,  'PN_COD_TESTEO',TEXT_PARAMETER,1);



 v_rep:= RUN_REPORT_OBJECT(repid,List_id);
 rep_status:= report_object_status(v_rep);

 WHILE rep_status in ('RUNNING','OPENING_REPORT','ENQUEUED') LOOP
        rep_status:= report_object_status(v_rep);
 end loop;

 if rep_status = 'FINISHED' then
        WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_my_computer','_blank');
        --WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_virtualxp-28951','_blank');
 else
        message ('error en la ejecucion');
 end if;

end;
```

El objetivo de este procedimiento es testear la base de datos, en este caso, con una inserción de 100000 registros, se insertara en la tabla TBD_REGISTROS, ubicada en el esquema SYS.

```
PROCEDURE pu_p_inserta_registros IS
 DESCR VARCHAR2(100);
```

```
BEGIN
 MESSAGE('SE EMPIEZA A  REALIZAR LA INSERCION DE 10000 REGISTROS EN TDB_REGISTROS');
 FOR I IN 1..100000 LOOP
  DESCR := 'DESCRIPCION '||to_char(I);
  INSERT INTO TBD_REGISTROS(codigo,descripcion) VALUES(I, DESCR);
END LOOP;


MESSAGE('SE TERMINO DE REALIZAR LA INSERCION DE 10000 REGISTROS EN TDB_REGISTROS');


END;
```

PROCEDIMIENTOS SGA (SYSTEM GLOBAL AREA)

Recupera los valores actuales de todos los componentes que conforma la SGA.

```
PROCEDURE pu_p_recupera_data_act IS
LN_SGA_ACTUAL                                   NUMBER(20,3):=0;
LN_SGA_MAX_SIZE                                 NUMBER(20,3):=0;
LN_BUFFER_CACHE                             NUMBER(20,3):=0;
LN_SHARED_POOL_SIZE                             NUMBER(20,3):=0;
LN_LARGE_POOL                                   NUMBER(20,3):=0;
LN_JAVA_POOL                                    NUMBER(20,3):=0;
LN_SGA_LIBRE                                NUMBER(20,3):=0;
```

```
BEGIN

        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_ACTUAL
                FROM V_$SGAINFO x
                where x.name like  'Fixed SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_MAX_SIZE
                FROM V_$SGAINFO x
                where x.name like 'Maximum SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_BUFFER_CACHE
                FROM V_$SGAINFO x
                where x.name like 'Buffer Cache Size';
        end;

        begin
                SELECT x.BYTES/1024/1024
                into LN_SHARED_POOL_SIZE
                FROM V_$SGAINFO x
                where x.name like 'Shared Pool Size';
```

```
        end;


        begin
                SELECT x.BYTES/1024/1024
                into LN_LARGE_POOL
                FROM V_$SGAINFO x
                where x.name like 'Large Pool Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_JAVA_POOL
                FROM V_$SGAINFO x
                where x.name like 'Java Pool Size';
        end;
        begin
                /*
                SELECT x.BYTES/1024/1024
                into LN_SGA_LIBRE
                FROM V_$SGAINFO x
                where x.name like 'Free SGA Memory Available';
                */
                select sum(bytes)/1024/1024 " SGA LIBRE "
                into LN_SGA_LIBRE
                from v$sgastat where name='free memory';
end;
```

```
LN_SGA_ACTUAL:=LN_BUFFER_CACHE+LN_SHARED_POOL_SIZE+LN_LARGE_POOL+LN_JAVA_POO
L;
        BEGIN
:BL_T_HIST_SGA.FEC_TESTEO    :=SYSDATE;
:BL_T_HIST_SGA.SGA_ACTUAL  :=LN_SGA_ACTUAL;
:BL_T_HIST_SGA.SGA_MAX_SIZE:=LN_SGA_MAX_SIZE;
:BL_T_HIST_SGA.BUFFER_CACHE:=LN_BUFFER_CACHE;
:BL_T_HIST_SGA.SHARED_POOL_SIZE:=LN_SHARED_POOL_SIZE;
:BL_T_HIST_SGA.LARGE_POOL:=LN_LARGE_POOL;
:BL_T_HIST_SGA.JAVA_POOL:=LN_JAVA_POOL;
:BL_T_HIST_SGA.SGA_LIBRE:=LN_SGA_LIBRE;




        END;
END;
```

Se realizara un histórico de datos con el SGA, con este procedimiento se recuperara la data y se insertara en la tabla TBD_HIST_SGA.

```
PROCEDURE pu_p_recupera_data_hist IS
LN_SGA_ACTUAL                                    NUMBER(20,3):=0;
LN_SGA_MAX_SIZE                                  NUMBER(20,3):=0;
```

```
LN_BUFFER_CACHE                         NUMBER(20,3):=0;
LN_SHARED_POOL_SIZE                     NUMBER(20,3):=0;
LN_LARGE_POOL                           NUMBER(20,3):=0;
LN_JAVA_POOL                            NUMBER(20,3):=0;
LN_SGA_LIBRE                            NUMBER(20,3):=0;

BEGIN

        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_ACTUAL
                FROM V_$SGAINFO x
                where x.name like  'Fixed SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_MAX_SIZE
                FROM V_$SGAINFO x
                where x.name like 'Maximum SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_BUFFER_CACHE
                FROM V_$SGAINFO x
                where x.name like 'Buffer Cache Size';
        end;
```

```
begin
        SELECT x.BYTES/1024/1024
        into LN_SHARED_POOL_SIZE
        FROM V_$SGAINFO x
        where x.name like 'Shared Pool Size';
end;
begin
        SELECT x.BYTES/1024/1024
        into LN_LARGE_POOL
        FROM V_$SGAINFO x
        where x.name like 'Large Pool Size';
end;
begin
        SELECT x.BYTES/1024/1024
        into LN_JAVA_POOL
        FROM V_$SGAINFO x
        where x.name like 'Java Pool Size';
end;
begin
/*      SELECT x.BYTES/1024/1024
        into LN_SGA_LIBRE
        FROM V_$SGAINFO x
        where x.name like 'Free SGA Memory Available';
        */
        select sum(bytes)/1024/1024 " SGA LIBRE "
        into LN_SGA_LIBRE
```

```
                from v$sgastat where name='free memory';
 end;
LN_SGA_ACTUAL:=LN_BUFFER_CACHE+LN_SHARED_POOL_SIZE+LN_LARGE_POOL+LN_JAVA_POO
L;
        BEGIN
                INSERT INTO TBD_HIST_SGA (
                        FEC_TESTEO,
                        SGA_ACTUAL,
                        SGA_MAX_SIZE,
                        BUFFER_CACHE,
                        SHARED_POOL_SIZE,
                        LARGE_POOL,
                        JAVA_POOL,
                        SGA_LIBRE,
                        FEC_INGRESO,
                        USUARIO_INGRESO )
                        VALUES
                        (
                        SYSDATE,
                        LN_SGA_ACTUAL,
                        LN_SGA_MAX_SIZE,
                        LN_BUFFER_CACHE,
                        LN_SHARED_POOL_SIZE,
                        LN_LARGE_POOL,
                        LN_JAVA_POOL,
                        LN_SGA_LIBRE,
```

```
                    SYSDATE  ,USER );
        END;
        COMMIT;
END;
```

REPORTE_SGA.- procedimiento mediante el cual se generara, el enlace de forms con el reports.

```
PROCEDURE pu_p_genera_reporte(PN_REPORTE IN number) IS
 repid                  REPORT_OBJECT;
 v_rep                  varchar2(100);
 rep_status varchar2(20);
 List_id        ParamList;
 pl_name                VARCHAR2(10) := 'misp';


begin

                List_id := Get_Parameter_List('input_params');
                IF NOT Id_Null(List_id) THEN
                        Destroy_Parameter_List(List_id);
                END IF;
        List_id := Create_Parameter_List('input_params');
```

```
        if PN_REPORTE=1 then
                repid:= find_report_object('TBD3R');
        end if;

        if PN_REPORTE=2 then
                repid:= find_report_object('TBD4R');
        end if;

SET_REPORT_OBJECT_PROPERTY(repid, REPORT_OTHER,'paramform=no');
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_EXECUTION_MODE,BATCH);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_COMM_MODE,SYNCHRONOUS);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESTYPE,CACHE);
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESFORMAT,'html');
SET_REPORT_OBJECT_PROPERTY(repid, REPORT_SERVER,'rep_my_computer');

Add_Parameter(List_id,  'PD_FECHA_INICIO',TEXT_PARAMETER,:TBD_TESTEOS.FECHA_INICIO);
        Add_Parameter(List_id,  'PD_FECHA_FIN',TEXT_PARAMETER,:TBD_TESTEOS.FECHA_FIN);
Add_Parameter(List_id,  'PN_COD_TESTEO',TEXT_PARAMETER,1);


v_rep:= RUN_REPORT_OBJECT(repid,List_id);
rep_status:= report_object_status(v_rep);

WHILE rep_status in ('RUNNING','OPENING_REPORT','ENQUEUED') LOOP
        rep_status:= report_object_status(v_rep);
end loop;
```

```
 if rep_status = 'FINISHED' then

        WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_my_computer','_blank');

        --WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_virtualxp-28951','_blank');

 else

        message ('error en la ejecucion');

 end if;


end;
```

BLOQUEOS

El procedimiento pu_p_matasesion eliminara el esquema que está causando los bloqueos.

```
PROCEDURE pu_p_matasesion IS

LN_SERIAL NUMBER;

ALERTA NUMBER;

BEGIN

 SELECT SERIAL#

 into LN_SERIAL

 FROM V$SESSION WHERE SID = :BL_T_REVISA.SID_QUIEN ;


BEGIN
```

```
FORMS_DDL('ALTER SYSTEM KILL SESSION '''||:BL_T_REVISA.SID_QUIEN||','||LN_SERIAL||'''');

END;
pu_p_inserta();

END;
```

Este procedimiento consultara en el diccionario de datos los cambios efectuados en las tablas con la eliminación del esquema antes descrito

```
PROCEDURE pu_p_revisa IS
cursor c_session is
select L_LOCKER.SID,
S_LOCKER.USERNAME LOCKER_SCHEMA,
S_LOCKER.OSUSER OS_LOCKER,
S_LOCKER.PROCESS LOCK_PID,
S_WAITER.OSUSER OS_WAITER,
S_WAITER.USERNAME WAITER_SCHEMA,
S_WAITER.PROCESS WAITER_PID,
'TX: '||O.SQL_TEXT SQL_TEXT_WAITER
from
V$LOCK L_WAITER,
V$LOCK L_LOCKER,
V$SESSION S_WAITER,
```

```
V$SESSION S_LOCKER,
V$_LOCK L1_WAITER,
V$OPEN_CURSOR O
where S_WAITER.SID = L_WAITER.SID
and L_WAITER.TYPE IN ('TX')
and S_LOCKER.sid = L_LOCKER.sid
and L_LOCKER.ID1 = L_WAITER.ID1
and L_WAITER.REQUEST > 0
and L_LOCKER.LMODE > 0
and L_WAITER.ADDR != L_LOCKER.ADDR
and L1_WAITER.LADDR = L_WAITER.ADDR
and L1_WAITER.KADDR = L_WAITER.KADDR
and L1_WAITER.SADDR = O.SADDR
and O.HASH_VALUE = S_WAITER.SQL_HASH_VALUE;

LV_OBJETO varchar2(500);
BEGIN
        go_block('BL_T_REVISA');
        clear_block;
        for a in  c_session loop

                SELECT OBJECT_NAME
                INTO LV_OBJETO
                FROM dba_objects, v$lock
                WHERE object_id = id1
                AND type = 'TM'
```

```
            AND sid=a.SID
        AND rownum<5;

                    :BL_T_REVISA.SID_QUIEN := a.SID;
                    :BL_T_REVISA.USER_QUIEN := a.LOCKER_SCHEMA;
                    :BL_T_REVISA.USER_MAQUINA := a.OS_LOCKER;
                    :BL_T_REVISA.USER_AQUIEN := a.WAITER_SCHEMA;
                    :BL_T_REVISA.OBJETO := LV_OBJETO;
                    :BL_T_REVISA.SENTENCIA := a.SQL_TEXT_WAITER;
                    next_record;
    end loop;

END;
```

Este Procedimiento inserta los datos consultados en el Procedimiento anterior en la table tbd_bloqueos, datos que son necesarios para el reporteados, hacienda un análisis màs fino sobre lo sucedido en cuanto a bloqueo y eliminación de esquemas nos referimos.

```
PROCEDURE pu_p_inserta IS
ALERTA NUMBER;
BEGIN
  insert into tbd_bloqueo(
        SID_QUIEN,
        USER_QUIEN,
```

```
          USER_MAQUINA,
         USER_AQUIEN,
        OBJETO,
        FEC_BLOQUEO
          )
          values
          (:BL_T_REVISA.SID_QUIEN,
           :BL_T_REVISA.USER_QUIEN,
           :BL_T_REVISA.USER_MAQUINA,
           :BL_T_REVISA.USER_AQUIEN,
           :BL_T_REVISA.OBJETO,
           SYSDATE
          );
          SET_ALERT_PROPERTY('AL_OK',ALERT_MESSAGE_TEXT,'REGISTRO GUARDADO');
 ALERTA := SHOW_ALERT('AL_OK');

 COMMIT;
END;
```

REPORTE_BLOQUEOS

```
PROCEDURE pu_p_genera_reporte IS
 repid                  REPORT_OBJECT;
 v_rep                  varchar2(100);
 rep_status varchar2(20);
```

```
List_id             ParamList;
pl_name                     VARCHAR2(10) := 'misp';


begin


                    List_id := Get_Parameter_List('input_params');
                    IF NOT Id_Null(List_id) THEN
                              Destroy_Parameter_List(List_id);
                    END IF;
          List_id := Create_Parameter_List('input_params');


          repid:= find_report_object('TBD9R');
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_OTHER,'paramform=no');
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_EXECUTION_MODE,BATCH);
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_COMM_MODE,SYNCHRONOUS);
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESTYPE,CACHE);
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESFORMAT,'html');
 SET_REPORT_OBJECT_PROPERTY(repid, REPORT_SERVER,'rep_my_computer');


 Add_Parameter(List_id,  'PD_FECHA_INICIO',TEXT_PARAMETER,:BL_T_REVISA.TEXT_ITEM56);
          Add_Parameter(List_id,  'PD_FECHA_FIN',TEXT_PARAMETER,:BL_T_REVISA.TEXT_ITEM57);
-- Add_Parameter(List_id,  'PN_COD_TESTEO',TEXT_PARAMETER,1);
```

```
 v_rep:= RUN_REPORT_OBJECT(repid,List_id);

 rep_status:= report_object_status(v_rep);


 WHILE rep_status in ('RUNNING','OPENING_REPORT','ENQUEUED') LOOP

        rep_status:= report_object_status(v_rep);

 end loop;


 if rep_status = 'FINISHED' then

        WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_my_computer','_blank');

        --WEB.SHOW_DOCUMENT('/reports/rwservlet/getjobid'||substr(v_rep,instr(v_rep,'_',-
1)+1)||'?'||'server=rep_virtualxp-28951','_blank');

 else

        message ('error en la ejecucion');

 end if;


end;
```

JOBS CREADOS

Los jobs de Oracle son como concepto general una herramienta para la planificación y/o programación de tareas dentro de la base de datos; la tarea que va a realizar el job en nuestro aplicativo es la de insertar cada 10 segundos los valores de las tablas del diccionario de datos

testeado por la inserción de registros. Visualizar su cambio dinamico de valores e insertarlos en la tabla correspondiente.

```
create or replace package tbd_k_testeobasededatos is

 -- Author  : Ing.Diego Brito I.
 -- Created : 07/03/2010 16:07:29
 -- Purpose :
PROCEDURE pu_p_inicio_lanza_job;
PROCEDURE pu_p_recupera_data_hist_undo;
PROCEDURE pu_p_recupera_data_hist_sga;
end  tbd_k_testeobasededatos;

create or replace package body tbd_k_testeobasededatos is

procedure pu_p_inicio_lanza_job is
 begin
    dbms_job.isubmit(104, 'begin
tbd_k_testeobasededatos.pu_p_recupera_data_hist_undo;end;',SYSDATE,'SYSDATE +
0.003/24',FALSE);
    dbms_job.isubmit(105, 'begin tbd_k_testeobasededatos.pu_p_recupera_data_hist_sga;end;'
,SYSDATE,'SYSDATE + 0.003/24',FALSE);
   commit;

 end;
```

```
PROCEDURE pu_p_recupera_data_hist_undo IS

LN_UNDO_ACTUAL           NUMBER(20,3):=0;

LN_UNDO_RETENTION_ACTUAL   NUMBER(20,3):=0;

LN_UNDO_RETENTION_OPTIMO   NUMBER(20,3):=0;

LN_UNDO_OPTIMO           NUMBER(20,3):=0;

LN_UNDO_LIBRE            NUMBER(20,3):=0;

LD_FEC_TESTEO            DATE;

LN_UNDO_ACTIVE           NUMBER(20,3):=0;

LN_UNDO_UNEXPIRED        NUMBER(20,3):=0;

LN_UNDO_EXPIRED          NUMBER(20,3):=0;


BEGIN

 BEGIN

        SELECT d.undo_size / (1024 * 1024) "ACTUAL UNDO SIZE [MByte]",

           Substr(e.VALUE,1,25) "ACTUAL UNDO RETENTION [Sec]",

           Round((d.undo_size / (To_number(f.VALUE) * g.undo_block_per_sec))) "UNDO
RETENTION OPTIMO [Sec]"

           INTO
LN_UNDO_ACTUAL,LN_UNDO_RETENTION_ACTUAL,LN_UNDO_RETENTION_OPTIMO

       FROM   (SELECT Sum(a.bytes) undo_size

            FROM   v$datafile a,

                v$tablespace b,

                dba_tablespaces c

            WHERE  c.contents = 'UNDO'

                AND c.status = 'ONLINE'
```

```
            AND b.name = c.tablespace_name
            AND a.ts# = b.ts#) d,
        v$parameter e,
        v$parameter f,
        (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec
         FROM   v$undostat) g
   WHERE  e.name = 'undo_retention'
        AND f.name = 'db_block_size';
END;
BEGIN

SELECT  (To_number(e.VALUE) * To_number(f.VALUE) * g.undo_block_per_sec) / (1024 * 1024)
"Tamaño Optimo UNDO [MByte]"
INTO LN_UNDO_OPTIMO
FROM   (SELECT Sum(a.bytes) undo_size
    FROM   v$datafile a,
        v$tablespace b,
        dba_tablespaces c
    WHERE  c.contents = 'UNDO'
        AND c.status = 'ONLINE'
        AND b.NAME = c.tablespace_name
        AND a.ts# = b.ts#) d,
    v$parameter e,
    v$parameter f,
    (SELECT Max(undoblks / ((end_time - begin_time) * 3600 * 24)) undo_block_per_sec
     FROM   v$undostat) g
```

```
WHERE  e.NAME = 'undo_retention'
    AND f.NAME = 'db_block_size'  ;

 END;

 BEGIN
SELECT        Sum(dfs.bytes)
           / 1024
           / 1024 AS "Espacio Libre DataFile [MB]"
           INTO LN_UNDO_LIBRE
FROM          dba_free_space dfs, dba_tablespaces dts
           WHERE dfs.tablespace_name = dts.tablespace_name
             AND dts.contents = 'UNDO'
             AND dts.status = 'ONLINE';
 END;

 BEGIN
  SELECT SYSDATE AS fecha,
    active.active,
    unexpired.unexpired,
    expired.expired
    INTO LD_FEC_TESTEO,LN_UNDO_ACTIVE,LN_UNDO_UNEXPIRED, LN_UNDO_EXPIRED
FROM   (SELECT Sum(bytes / 1024 / 1024) AS unexpired
     FROM   dba_undo_extents
     WHERE  status = 'UNEXPIRED') unexpired,
    (SELECT Sum(bytes / 1024 / 104) AS expired
```

```
   FROM   dba_undo_extents tr
   WHERE  status = 'EXPIRED') expired,
  (SELECT CASE
          WHEN Count(status) = 0
          THEN 0
          ELSE Sum(bytes / 1024 / 1024)
        END AS active
   FROM   dba_undo_extents
   WHERE  status = 'ACTIVE') active;
END;


BEGIN
  INSERT INTO TBD_HIST_TESTEO (
  COD_TESTEO,
  FEC_TESTEO,
  UNDO_ACTUAL,
  UNDO_LIBRE,
  UNDO_ACTIVE,
  UNDO_UNEXPIRED,
  UNDO_EXPIRED,
  UNDO_OPTIMO,
  UNDO_RETENTION_ACTUAL,
  UNDO_RETENTION_OPTIMO,
  FEC_INGRESO,
  USUARIO_INGRESO )
  VALUES
```

```
    (
    1 ,
    LD_FEC_TESTEO,
    LN_UNDO_ACTUAL,
    LN_UNDO_LIBRE,
    LN_UNDO_ACTIVE  ,
    LN_UNDO_UNEXPIRED,
    LN_UNDO_EXPIRED  ,
    LN_UNDO_OPTIMO  ,
    LN_UNDO_RETENTION_ACTUAL,
    LN_UNDO_RETENTION_OPTIMO,
    SYSDATE  ,USER );
        END;
END;



PROCEDURE pu_p_recupera_data_hist_sga IS
LN_SGA_ACTUAL                                   NUMBER(20,3):=0;
LN_SGA_MAX_SIZE                               NUMBER(20,3):=0;
LN_BUFFER_CACHE                               NUMBER(20,3):=0;
LN_SHARED_POOL_SIZE                         NUMBER(20,3):=0;
LN_LARGE_POOL                                   NUMBER(20,3):=0;
LN_JAVA_POOL                                    NUMBER(20,3):=0;
LN_SGA_LIBRE                                  NUMBER(20,3):=0;
```

```
BEGIN

        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_ACTUAL
                FROM V_$SGAINFO x
                where x.name like  'Fixed SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_SGA_MAX_SIZE
                FROM V_$SGAINFO x
                where x.name like 'Maximum SGA Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_BUFFER_CACHE
                FROM V_$SGAINFO x
                where x.name like 'Buffer Cache Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_SHARED_POOL_SIZE
                FROM V_$SGAINFO x
                where x.name like 'Shared Pool Size';
        end;
```

```
        begin
                SELECT x.BYTES/1024/1024
                into LN_LARGE_POOL
                FROM V_$SGAINFO x
                where x.name like 'Large Pool Size';
        end;
        begin
                SELECT x.BYTES/1024/1024
                into LN_JAVA_POOL
                FROM V_$SGAINFO x
                where x.name like 'Java Pool Size';
        end;
        begin

                select sum(bytes)/1024/1024 " SGA LIBRE "
                into LN_SGA_LIBRE
                from v$sgastat where name='free memory';
 end;
LN_SGA_ACTUAL:=LN_BUFFER_CACHE+LN_SHARED_POOL_SIZE+LN_LARGE_POOL+LN_JAVA_POO
L;
        BEGIN
                INSERT INTO TBD_HIST_SGA (
                        FEC_TESTEO,
                        SGA_ACTUAL,
                        SGA_MAX_SIZE,
                        BUFFER_CACHE,
```

```
                    SHARED_POOL_SIZE,
                    LARGE_POOL,
                    JAVA_POOL,
                    SGA_LIBRE,
                    FEC_INGRESO,
                    USUARIO_INGRESO )
                    VALUES
                    (
                    SYSDATE,
                    LN_SGA_ACTUAL,
                    LN_SGA_MAX_SIZE,
                    LN_BUFFER_CACHE,
                    LN_SHARED_POOL_SIZE,
                    LN_LARGE_POOL,
                    LN_JAVA_POOL,
                    LN_SGA_LIBRE,
                    SYSDATE  ,USER );
        END;

END;


end  tbd_k_testeobasededatos;
```

Consultas a los datos de las tablas del sistema

```
select * from V$SGAINFO;

select * from v$sgastat;

select * from dba_undo_extents;

select * from dba_free_space;

select * from v$undostat;

select * from dba_tablespaces;

select * from v$datafile;
```

Se quiere demostrar en este manual que todas las funcionalidades del aplicativo AOT, se encuentra configuradas para el correcto funcionamiento en cualquier base de datos oracle 10g o posterior, como lo mencionamos anteriormente.

ANEXO C.

Manual de Usuario A.O.T

1 Manual de Usuario

1.1 Objetivo

El presente manual tiene como objetivo describir el uso de las salidas mediante consultas administrativas e informes estadísticos que realiza el aplicativo una vez que ha sido ejecutado o utilizado. Este manual es de uso administrativo, o para aquellos usuarios que necesitan información correspondiente al uso de la solución tecnológica.

1.2 Alcance

El manual que se presenta a continuación explica el modo de usar ambos resultados, los reportes que se encuentran centralizados para que el usuario los pueda revisar, las consultas administrativas.

1.3 Implementación del Sistema

Para poder utilizar el Sistema "AOT", se debe instalar la base de datos (Oracle Database 10g), PL/SQL Developer y el explorador Mozilla 3.0.9.

El manual que se presenta a continuación explica el modo de usar ambos resultados, los reportes que se encuentran centralizados para que el usuario los pueda revisar, las consultas administrativas.

El presente manual también tiene como objetivo fundamental, explicar las funcionalidades del aplicativo A.O.T, a nivel de usuario.

Antes de empezar a usar el sistema, el usuario, necesitara crear varios sinónimos en el **ESQUEMA QUE VAYA A CREAR;** para que el aplicativo funcione a plenitud en la maquina-servidor o Terminal donde esta posicionado la base de datos.

Estos sinónimos referenciaran algunas tablas propias del diccionario de datos las cuales se tendrán acceso mediante roles que se detallaran posteriormente.

Para el caso de este manual de uso se llevara de ejemplo un usuario "DBRITO" el cual se lo considerara un DBA o usuario que va a operar el aplicativo. Se indica también en este manual que si el usuario es una persona que tiene la base de datos instalada localmente, deberá crear un usuario y añadir el rol de "DBA" a dicho usuario.

Cabe indicar que en oracle la afectación entre mayúsculas y minúsculas en cuanto a sentencias DDL(create-alter-drop) no es muy importante, por tal motivo todos los comandos en este manual se realizaràn con minúsculas.

ESQUEMA SYS:

```
SQL>create user dbrito identified by dbrito;
SQL>grant dba to dbrito;
SQL>GRANT CONNECT, RESOURCE TO dbrito;
```

Una vez creado el usuario DBRITO procedemos en este mismo esquema ("SYS") a crear las tablas que necesitara el aplicativo para el correcto testeo de la base de datos. Estas tablas detallan la información histórica de los parámetros a tratar.

(Igualmente el scritp de ejecución para la creación de tablas **script_tablas_tbd.sql** y demás ejecutables se añadirá a la entrega de este software).

Una vez creada las tablas en SYS procedemos a otorgarle privilegios de insert, select, update a DBRITO

sys>>grant insert,select,update on sys.tbd_registros to DBRITO;
sys>>grant insert,select,update on sys.tbd_hist_testeo to DBRITO;
sys>>grant insert,select,update on sys.tbd_hist_sga to DBRITO;
sys>>grant insert,select,update on sys.tbd_bloqueo to DBRITO;

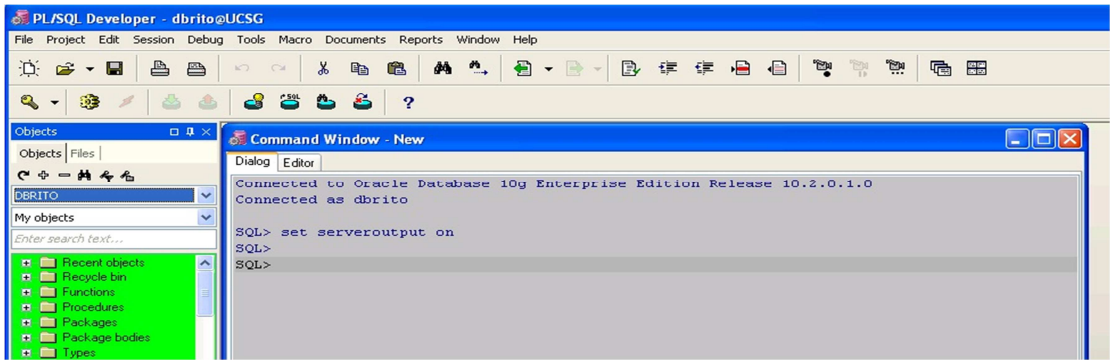
Se cerrara el esquema SYS ya que no se lo va a necesitar más en el futuro.

Esquema DBRITO

Con la finalización de la creación de tablas mencionado anteriormente.

A continuación se dará paso a crear los sinónimos para el usuario DBRITO, de la siguiente forma:

Una vez en el esquema “DBRITO” y abriendo una ventana SQL-WINDOW. Escribimos lo siguiente y ejecutamos(F8).
create synonym tbd_registros for sys.tbd_registros;
create synonym tbd_hist_testeo for sys.tbd_hist_testeo;
create synonym tbd_hist_sga for sys.tbd_hist_sga
create synonym tbd_bloqueo for sys.tbd_bloqueo;



El aplicativo fuente tendrá 8 modulos(0 al 7) de forms llamados: TBD0F.mmb, TBD1F.mmb TBD2F.mmb, TBD3F.mmb, TBD4F.mmb, TBD5F.mmb, TBD6F.mmb, TBD7F.mmb, TBD8F.mmb.

Añadiendo a la explicaciòn este manual, que el primer modulo llamado TBD0F, es la pantalla principal y la pantalla llamadora a los demás formas o módulos.

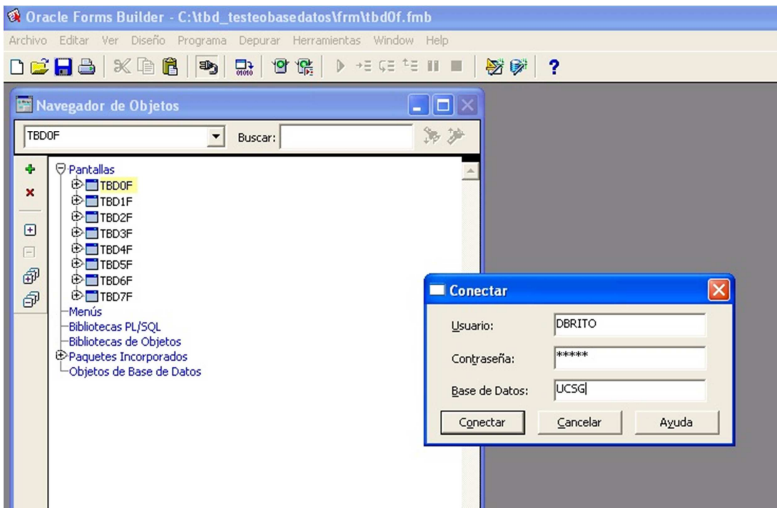
La herramienta de forms builder nos permite directamente interactuar con la base de datos pero obliga a conectarnos con usuario y clave. Por tal motivo los pasos anteriores(creación de usuarios-creacion de tablas-creacion de sinomimos) son esenciales para el correcto funcionamiento del AOT. (claro esta que si se tiene un usuario con rol de dba no habrá necesidad de crear ningún usuario).

Para que se siga con el ejemplo de uso de este manual se conectara de la siguiente forma, una vez

USER: DBRITO

CONTRASEÑA: DBRITO

BASE DE DATOS: UCSG ->>nombre de la base de datos.



Se ha mencionado anteriormente que el sistema cuenta con 8 módulos de formularios. Sin embargo, se hace énfasis que ejecutando el primer formulario (TBD0F) se podrá acceder al menú principal del sistema, desde ahí se podrá controla toda las llamadas a formas(ventanas)del AOT.

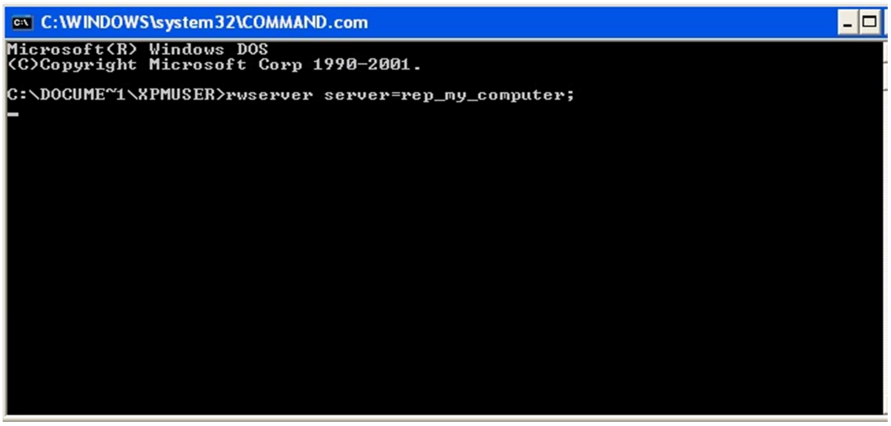
Este sistema únicamente tendrá un nivel de acceso que es el propio de oracle ya que los operadores de este software serán personas que sean DBA o en su defecto administradores locales que tengan su base de datos en su propia maquina.

Una vez conectado el usuario, preferiblemente, este manual recomienda teclear SHIFT+CTRL+K y consecutivamente CTRL+T. Para la correcta compilación del modulo, se recalca que esto es opcional, pero valido para cualquier validación.

Una vez adentro, este manual cambiara la forma de denominaciones, a los módulos, los llamaremos “formas” para posteriores explicaciones.

Antes de ejecutar la forma principal(TBD0F) se recuerda que el aplicativo generara reportes, con el objetivo de visualizar e indicar detalladamente por periodo de fecha las novedades y cambios dinámicos en los parámetros que vamos a testear.

Por eso se recalca, antes de usar el aplicativo se tiene que levantar el Servidor de Reportes de Oracle de la siguiente forma: (**artificio**).



Command Windows:

- rwserver server=rep_my_computer;

Automáticamente, se levantará el servidor de reportes, se lo minimizara para el uso de este aplicativo, una vez terminado de utilizarlo se procederá a apagarlo.

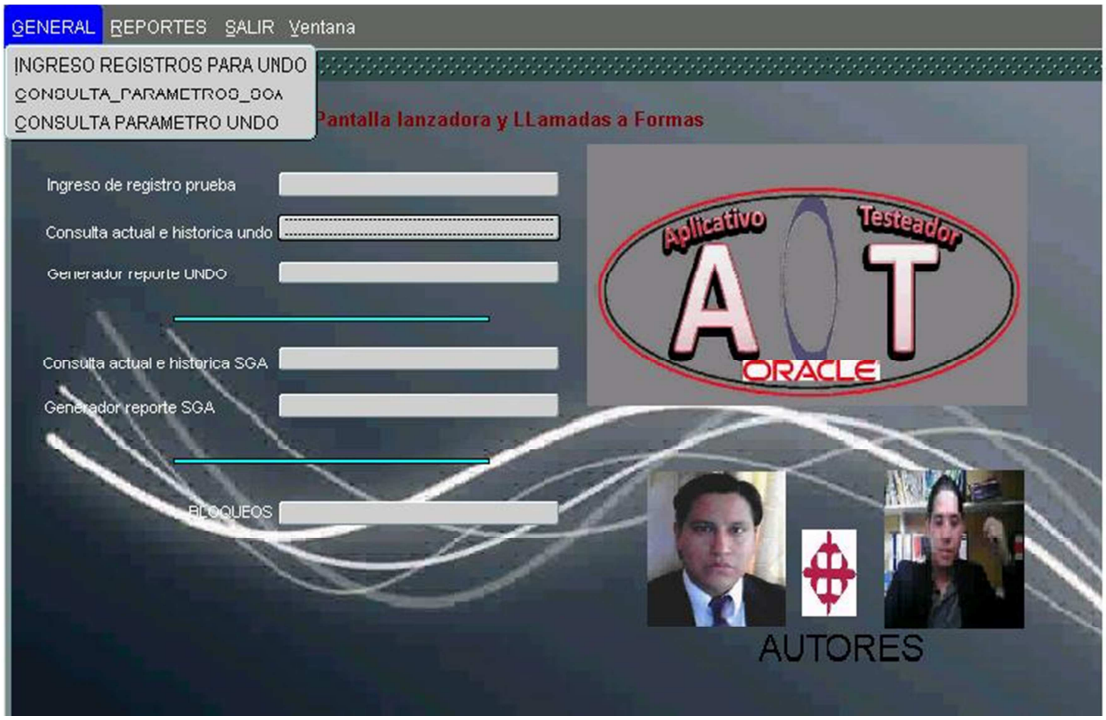
Solamente se dara click en el botón "Cerrado" que aparece arriba y a continuación en el command se escribirá EXIT para salir.

Funcionalidad del sistema

Menù principal

Parametro de Undo y SGA(System Global Area)

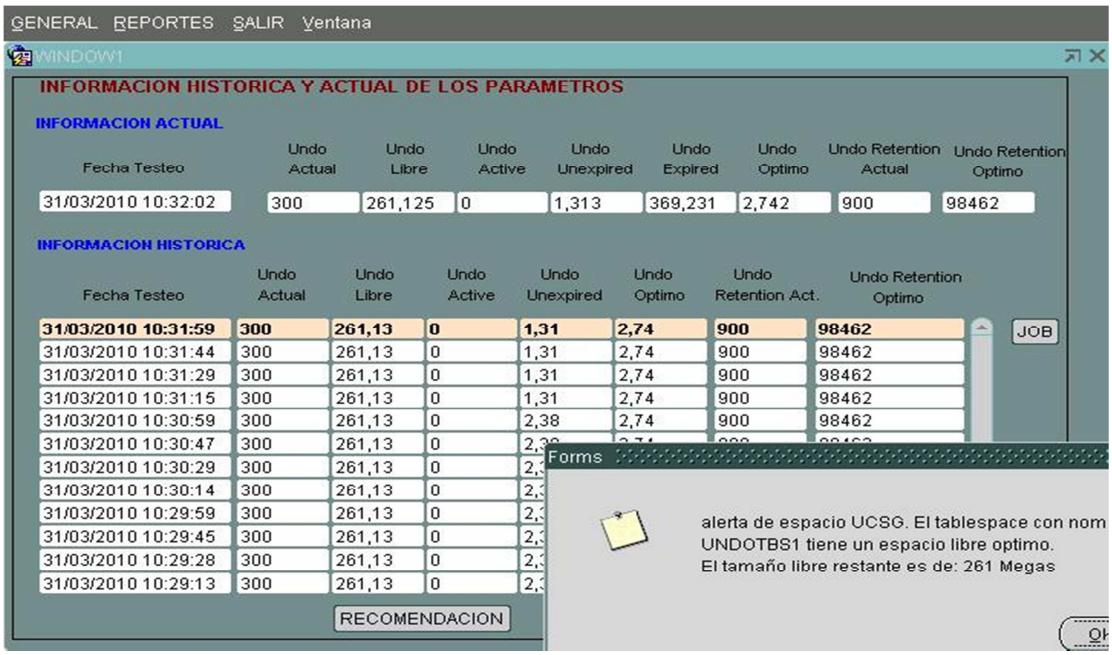
Undo y SGA.



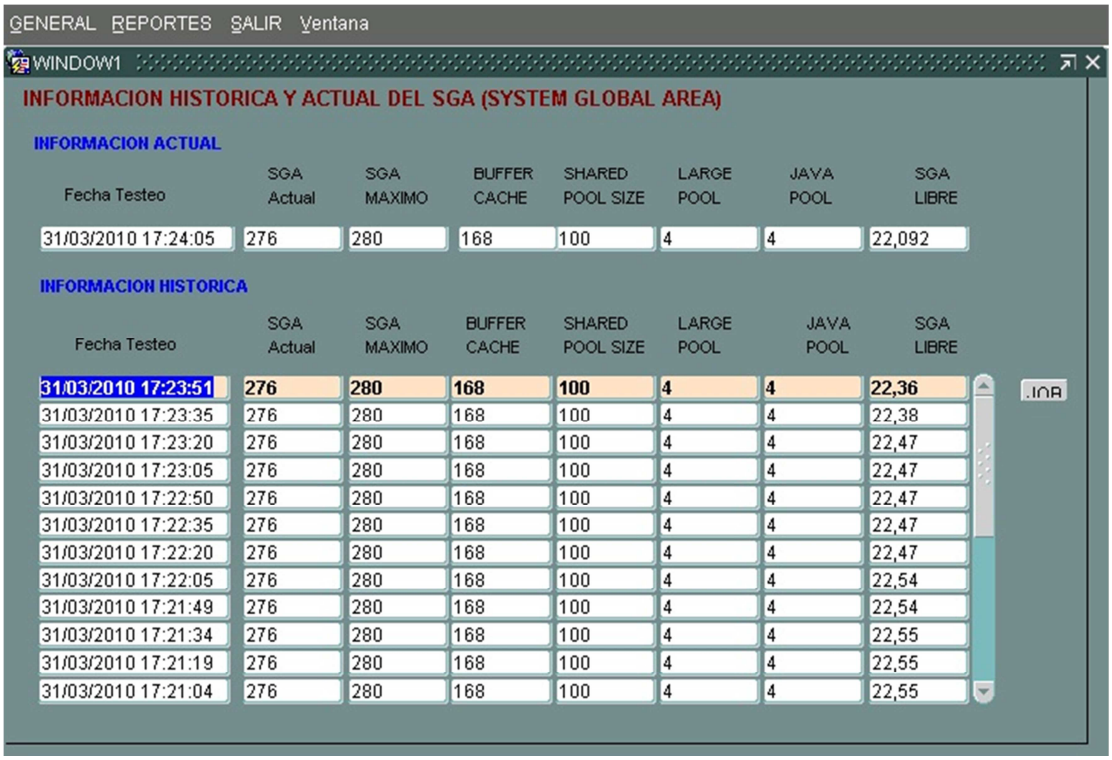
El sub-menú INGRESO REGISTRO PARA UNDO, opción con la cual el usuario se trasladara a otra ventana con el objetivo de realizar un testeo a la base de datos. En este caso la prueba o testeo será de inserts a la tabla TBD_REGISTROS(creada inicialmente en este manual) con 10000 registros haciéndo commit a la tabla. Este transacción traerá como consecuencia la variación de los parámetros de undo como los parámetros de la SGA(system global area), dando el aplicativo la facilidad de ver cuando son los valores optimos de cada parámetro, asi como su actual valor con la transacción antes descrita.

Con esta transacción de registros variaran los valores del parámetro undo_retention el cual el administrador tendrá la potestad de cambiarlos si ve la necesidad con la transacción ejectuada.

Para visualizar estos cambios en valores del undo_retention, se hara doble-click a la pestaña denominada: consulta_parametros_undo. Con la cual se tendrá la siguiente ventana:



Para este caso como se ve en el ejemplo tenemos un valor actual de 300 mb de tamaño actual de undo y undo libre de 261,25 equivalentes a un 15% de tablespace utilizado, también observamos que el **undo_retention_optimo** es de 98462 que seria 27 minutos. Valores con los cuales el operador podrá analizar y cambiarlos mediante la sentencia alter.

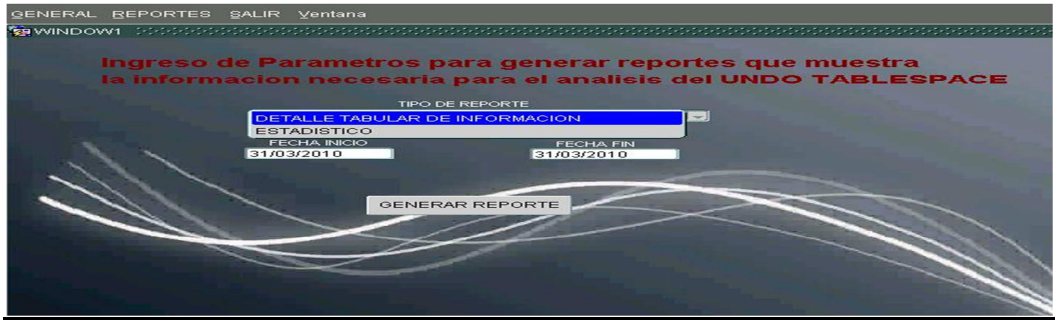


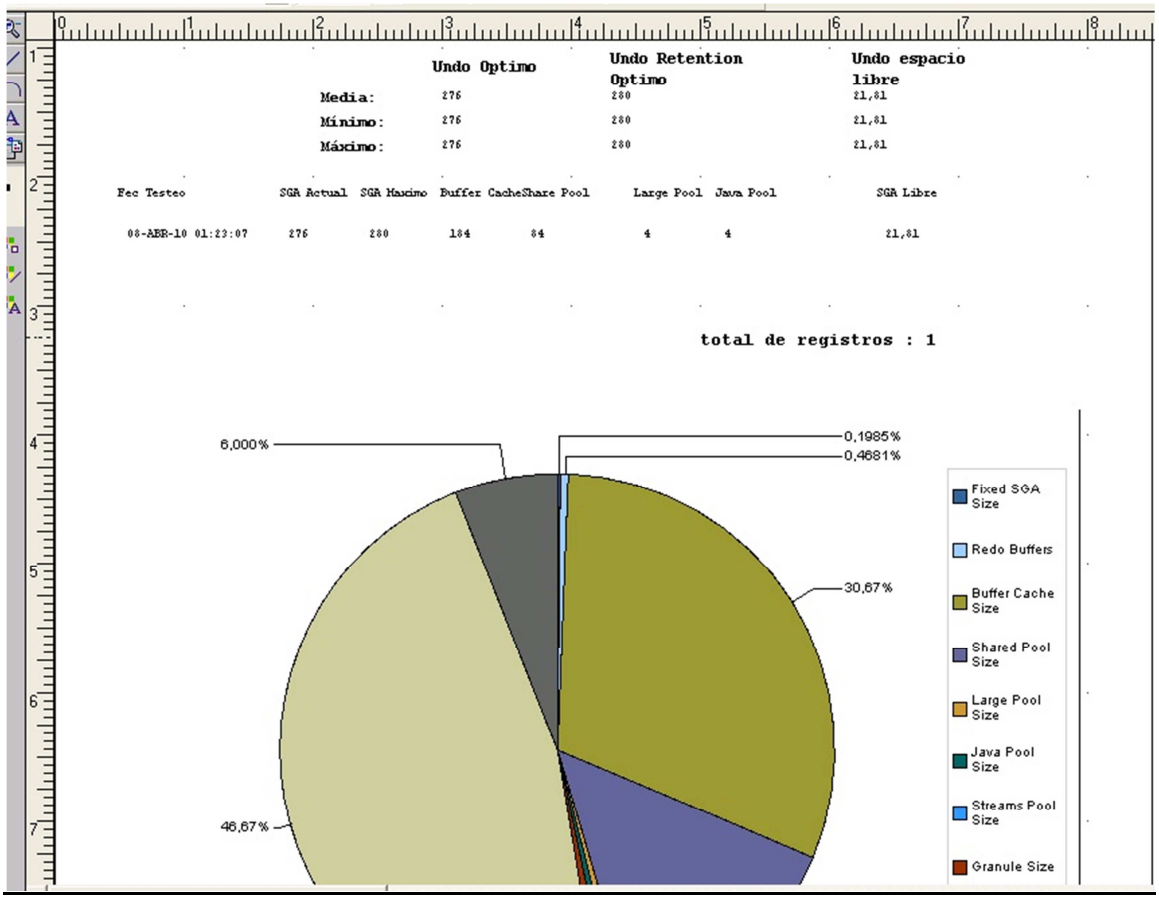
Reportes UNDO y SGA.-

Con el botón o la pestaña denominada “Generador Reporte Undo” se presenta la siguiente pantalla, esta describirá los diferente tipos de reportes que están implementados en este aplicativo. El primero “DETALLE TABULAR DE INFORMACION” el cual describirá de manera visual y por fecha los valores de los parámetros a tratar, en este caso el de undo.

El segundo “ESTADISTICO”, el cual presentara de forma analítica y con pasteles de medición la historia de datos de estos valores del parámetro de UNDO.

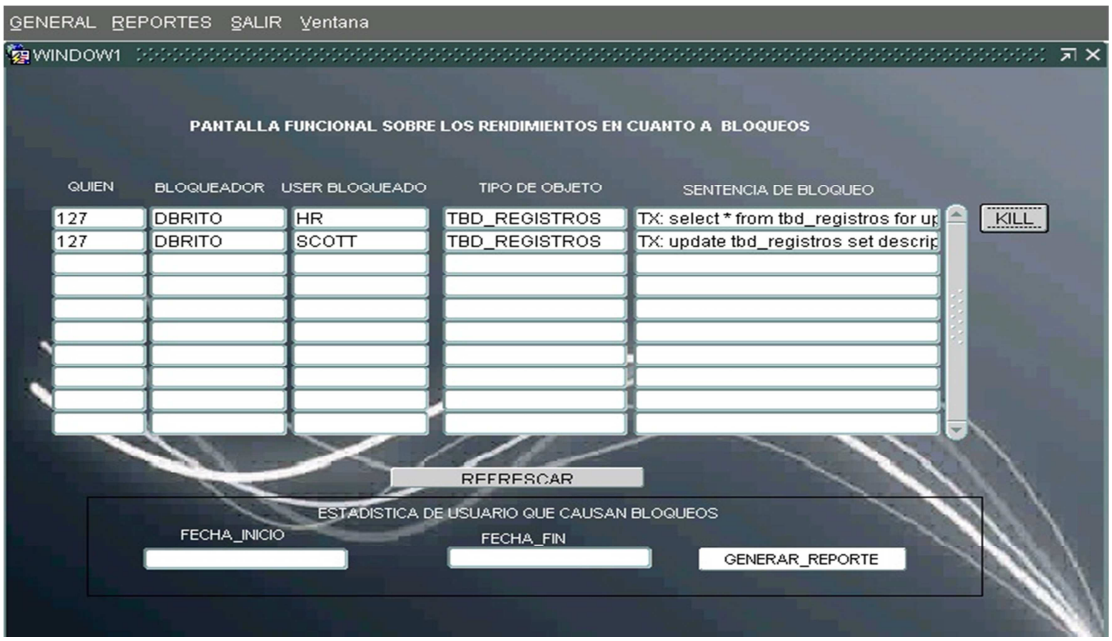
Mismas características que se proveen con la pestaña “Generado Reporte Sga”



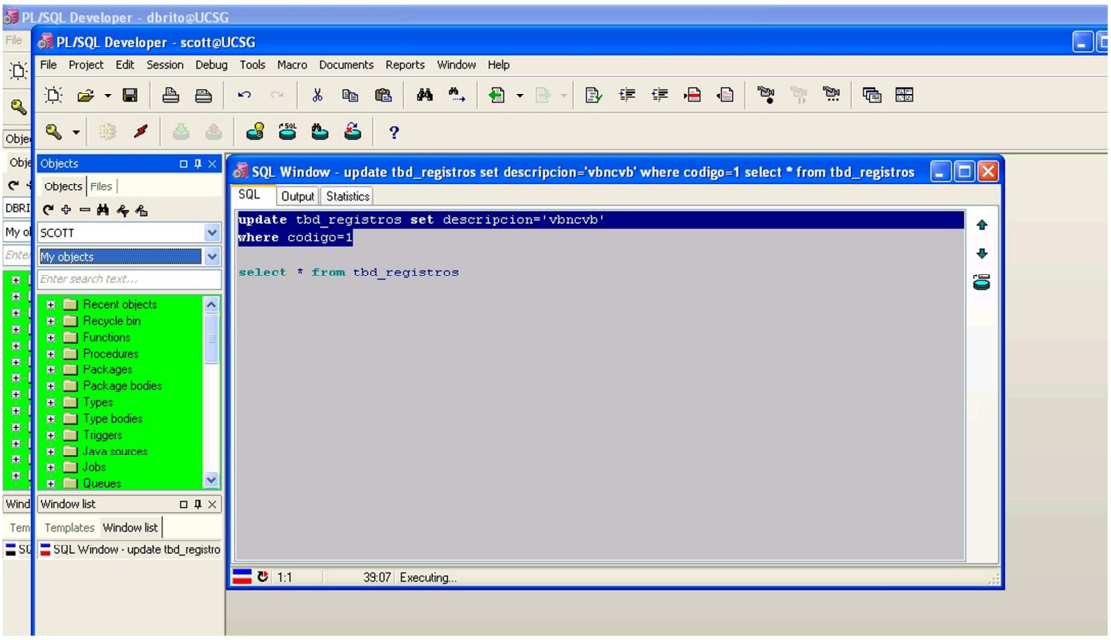


RENDIMIENTO EN BLOQUEOS.

Con la pestaña denominada “Bloqueos” accederemos a la forma siguiente



Anteriormente a para ejemplo se ha bloqueado intencionalmente a varios usuario mostrados abajo. (Visualiza cuadros)

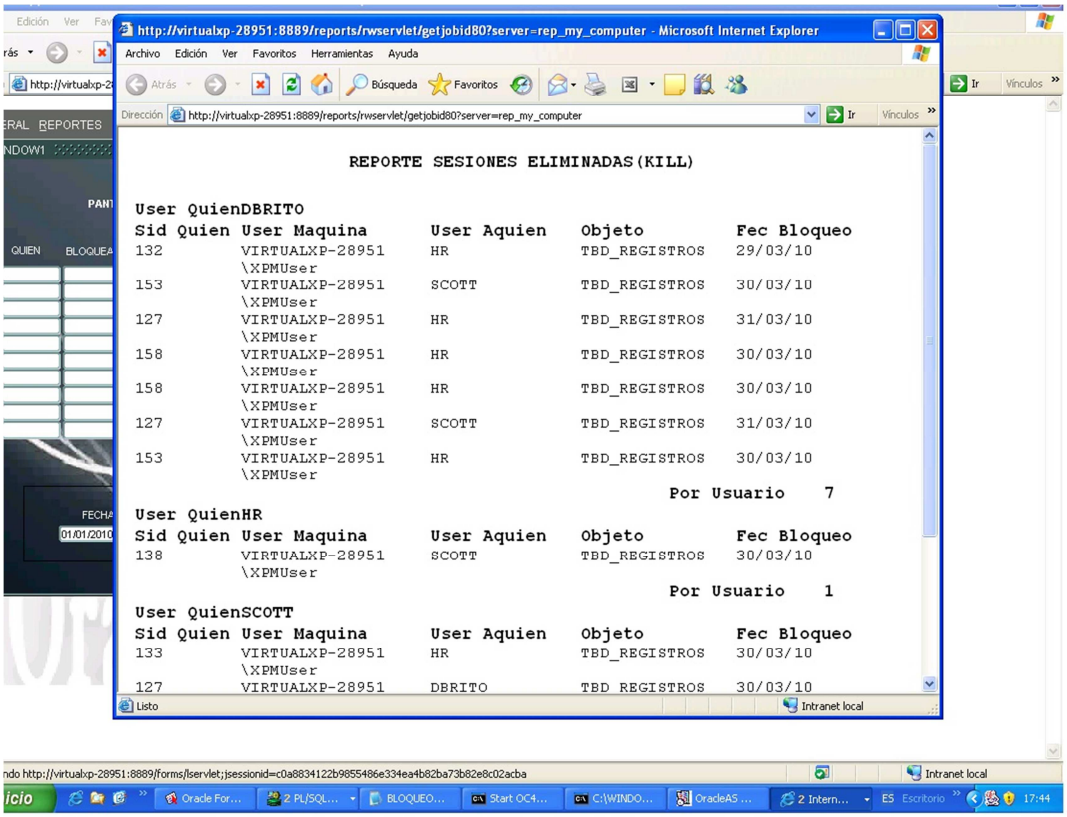


Como se observa en los graficos anteriores se muestra que el bloquedor en este ejemplo usuario: DBRITO, bloque al otro usuario SCOTT y lo deja en estado WAITING, en la pantallama vemos como SCOTT tiene demasiado minutos y sigue en “executing” para realizar su proceso.

En la forma se visualiza un botón “KILL” el cual al oprimirse podrá “eliminar ” la sesión del BLOQUEADOR y consecuentemente el usuario que está bloqueado se liberara de dicho bloqueo.

Se hace referencia, que estos registros se almacenan en la tabla TBD_BLOQUEO, para la función de reporte, el cual va a permitir realizar un análisis mas fino, acerca de que usuario tiene mas problemas en cuanto a bloqueo se refiere.

Esto se logra haciendo doble_click(insertando fecha_inicio-fecha_fin) en el botón GENERAR_REPORTE.



Se quiere demostrar en este manual que todas las funcionalidades del aplicativo AOT, se encuentra configuradas para el correcto funcionamiento en cualquier base de datos oracle 10g o posterior, como lo mencionamos anteriormente.