ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

HERRAMIENTA PARA LA EXTRACCIÓN DE INFORMACIÓN DE AUDITORÍA PARA APLICACIONES IMPLEMENTADAS SOBRE BASES DE DATOS ORACLE 9i Y 10g.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

MAYRA DEL PILAR CONCHAMBAY MUZO

GABRIELA FERNANDA VARELA BOLAÑOS

DIRECTOR: ING. PATRICIO MORENO.

Quito, OCTUBRE 2007

DECLARACIÓN

Nosotras, Mayra del Pilar Conchambay Muzo y Gabriela Fernanda Varela Bolaños, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Mayra del Pilar Conchambay Muzo

Gabriela Fernanda Varela Bolaños

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Mayra del Pilar Conchambay Muzo y Gabriela Fernanda Varela Bolaños, bajo mi supervisión.

Ing. Patricio Moreno
DIRECTOR DE PROYECTO

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto comprende el desarrollo de una Herramienta para la extracción de información de Auditoría para aplicaciones implementadas sobre Bases de Datos Oracle 9i y 10g.

El documento está dividido en 5 capítulos. En el capitulo 1, se describe el marco teórico en el cual, se definen los controles de una Auditoría de Base de Datos, se analiza el proceso de activación de una Auditoría Estándar en una Base de Datos Oracle y se selecciona la metodología a utilizar.

En el capitulo 2, se realiza la especificación de los requerimientos, el plan del proyecto y el análisis y diseño del sistema de software de acuerdo a la metodología seleccionada previamente.

En el tercer capítulo, se describen el código fuente más relevante utilizado en el desarrollo del aplicativo. Las pruebas realizadas a la herramienta se describen el capítulo 4, así como la evaluación de la herramienta en un ambiente real, mediante lo cual, se pudo determinar que la herramienta funciona de acuerdo a los requerimientos previamente establecidos.

Finalmente se realizaron conclusiones y recomendaciones, las cuales fueron el fruto de las lecciones aprendidas en el proceso de desarrollo de la herramienta.

INTRODUCCIÓN

La información es un recurso vital para el desarrollo de la organización, puesto que el mayor objetivo de la misma es apoyar a la toma de decisiones de nivel gerencial, logrando con esto un alto nivel competitivo dentro del mercado y obteniendo mayores niveles de capacidad de desarrollo. Una buena gestión de la información permite identificar fortalezas y debilidades con las que cuenta la organización. Para una correcta administración de la información es necesario establecer estrategias y políticas para el uso efectivo y eficiente de la misma, de tal forma que una organización disminuya sus riesgos en la administración global de la misma y pueda enfocarse a un mejor crecimiento y éxito.

Considerando que en toda organización, la Seguridad de la Información ha comenzado a tomar un lugar determinante dentro de la gestión de la Tecnología de la Información (TI), y se ha convertido en un elemento fundamental para toda estrategia de negocio con miras a lograr metas importantes a corto, mediano y largo plazo.

Por ello se ha realizado el presente proyecto de Titulación cuyo propósito es desarrollar una Herramienta para la extracción de Información de Auditoría para aplicaciones implementadas sobre Bases de Datos 9i y 10g (DML Audit Oracle), que permitirá recopilar información de auditoría sobre la actividad de la base de datos en relación a ciertos objetos con el fin de detectar anomalías o errores en el acceso en los datos o con el fin de obtener estadísticas de utilización.

OBJETIVOS

Principal:

El objetivo principal del presente proyecto de Titulación es el desarrollo de una herramienta para la generación asistida de triggers de Auditoría para operaciones DML que se ejecutan sobre las tablas y vistas de la Base de Datos Oracle 9i y 10g.

Específicos

- Contar con una herramienta para la generación asistida de triggers de Auditoría de operaciones DML para aplicativos cuyos esquemas se hayan implementado sobre bases de datos Oracle 9i y 10g.
- Contar con una herramienta para la extracción de información de auditoría sobre las operaciones DML que se ejecutan contra la base de datos Oracle de un aplicativo.

DESCRIPCIÓN DE LOS CAPÍTULOS

En el capítulo uno, Marco Teórico, se revisa conceptos básicos, tales como definiciones de auditoría informática, auditoría de base de datos y controles que se realizan en cada una.
Además se analiza las facilidades de auditoría en una base de datos Oracle 9i y 10g y una descripción de la metodología a utilizar para el desarrollo de la herramienta.

En el segundo capítulo se realiza Análisis y Diseño basados en la metodología Estructurada propuesta para el desarrollo de la herramienta, sin embargo en el primer punto se realiza una revisión de conceptos de desarrollo de aplicaciones con PLSQL, estructuras de datos a utilizar, estructuras funcionales a utilizar y procedimientos externos.

Dentro del Análisis propiamente se realiza la Concepción de la herramienta, la justificación, alcance y plan de proyecto. Además se realiza el proceso de desarrollo de la herramienta, a través de los modelo ambiental, comportamiento, navegabilidad, información y de rendimiento. Se analiza la arquitectura inicial y el entorno de desarrollo.

Dentro del Diseño de la herramienta se realiza el modelo físico y lógico de la base de datos, el diseño lógico de interfaces y se establecen los estándares de programación.

El capítulo tres trata acerca de la construcción de la herramienta donde se detallan las estructuras de datos y funcionales implementadas, así como también se describe el código fuente más relevante utilizado en el desarrollo de la herramienta.

En el cuarto capítulo se realizan las pruebas cuyo principal objetivo es verificar que la herramienta desarrollada cumpla con los requerimientos establecidos inicialmente, y en caso de no hacerlo corregir las falencias encontradas durante el proceso de evaluación de tal forma que se logre alcanzar con el objetivo primordial.
Además se realiza la implantación de DML Audit Oracle, en la empresa Red Partner, concluyendo satisfactoriamente con la misma.

En el capítulo cinco se redactan las conclusiones y recomendaciones las cuales fueron el fruto de las lecciones aprendidas en el proceso de desarrollo de la herramienta.

Finalmente se describe la bibliografía utilizada que sirvió de consulta y apoyo para el desarrollo de DML Audit Oracle.

CAPÍTULO 1 MARCO TEÓRICO

1.1 FUNDAMENTOS DE AUDITORÍA INFORMÁTICA EN BASE DE DATOS.

1.1.1 AUDITORÍA INFORMÁTICA

La Auditoría Informática es la revisión y la evaluación de los controles, sistemas y procedimientos de informática de los equipos de cómputo, a fin de que por medio del señalamiento de procedimientos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones.

La Auditoría Informática es el proceso de recoger, agrupar y evaluar evidencias para determinar si un sistema informatizado salvaguarda los archivos, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización y utiliza eficientemente los recursos.

La Auditoría Informática es de vital importancia para el buen desempeño de los sistemas de información, ya que proporciona los controles necesarios para que los sistemas sean confiables y con un buen nivel de seguridad. Además debe evaluar la organización de centros de información, el hardware y software.

1.1.2 AUDITORÍA DE BASE DE DATOS

Es el monitoreo y registro de las acciones realizadas por los usuarios sobre la Base de Datos, esto puede basarse en acciones individuales, tal como ejecutar una sentencia SQL[1] o en una combinación de factores como el nombre, aplicación, el tiempo, etc.

La Auditoría es normalmente usada para:

- Investigar actividades sospechosas

[1] Acrónico de Structured Query Language

Por ejemplo si un usuario no autorizado borra datos de las tablas, el administrador puede auditar todas las operaciones realizadas sobre los objetos de la Base de Datos, y puede determinar información como la hora de conexión del usuario, los objetos modificados, etc.

- Monitorear y almacenar información de las actividades de la Base de Datos.

  Por ejemplo el administrador puede acumular información estadística, sobre las tablas actualizadas últimamente, determinar el número de usuarios que se conectan, conocer los objetos más utilizados, conocer las horas pico de transaccionalidad, etc. Es decir le permitirá obtener información útil para la optimización de la Base de Datos.

Todas las operaciones realizadas por los usuarios de un Sistema de Base de Datos generan pistas de Auditoría que son registradas en archivos de Auditoría. Las pistas de Auditoría pueden ser utilizadas para descubrir qué es lo que ha ocurrido y quién es el responsable por la supuesta modificación de la información.

En caso de sospecha de falla en la seguridad, este archivo puede ser consultado para conocer los daños causados y/o identificar a los responsables de las operaciones irregulares.

Para implementar una Auditoría de la Base de Datos, primero se debe determinar las razones por las cuales se va a realizar una Auditoría y posteriormente solo auditar el menor número de sentencias, usuarios, objetos requeridos para prevenir información innecesaria de Auditoría y optimizar el uso de recursos, por ejemplo al auditar una determinada acción de la Base de Datos, se determina exactamente que tipo de actividad se va a auditar que se requiere rastrear, auditar solo actividades que le interesen, auditar solo el tiempo necesario para generar la información.

1.1.3 CONTROLES QUE REALIZA UNA AUDITORÍA DE BASE DE DATOS

1.1.3.1 Controles Físicos

Para soportar los esfuerzos y los desastres (como son los incendios, inundaciones, desastres naturales, entre otros) se debe contar con un sitio seguro de almacenamiento para los archivos y los documentos que se están utilizando. Además, todos los archivos de respaldo, los programas y otros documentos importantes se deberán almacenar en lugares externos de las instalaciones que tengan un nivel adecuado de seguridad. Los dispositivos para protección de archivos deberán emplearse para evitar un borrado accidental bajo condiciones de temperatura y humedad.

1.1.3.2 Control de Acceso a Datos

Este control se realiza para asegurar que los usuarios tengan el perfil y privilegios asignados por el Administrador, además asegurar que los comandos críticos sobre los objetos de la Base de Datos (create, alter, drop, etc) sean registrados, revisados y evaluados.

1.1.3.3 Control de Concurrencia

Cuando se comparten datos entre usuarios, se deben establecer controles de concurrencia para asegurar la consistencia en la actualización y lectura de la Base de Datos. El control de transacciones concurrentes en una base de datos brinda un eficiente desempeño del Sistema de Base de Datos, puesto que permite controlar la ejecución de transacciones que operan en paralelo, accesando a información compartida y, por lo tanto, interfiriendo potencialmente unas con otras. [3]

---

[3] Control de Concurrencia de Transacciones en un Sistema de Base de Datos, http://www.informatizate.net/articulos

1.1.3.4 Control de Respaldos y Recuperación

Los controles de respaldos y recuperación deben asegurar la vigencia y protección de respaldos de datos, así como la realización de pruebas de integridad y legibilidad desde los respaldos, su documentación y evaluación.

1.1.4 TÉCNICAS DE AUDITORÍA DE BASE DE DATOS

1.1.4.1 Características que apoyan la seguridad en los Sistemas de Bases de Datos

Estas características tienen que ver con la exactitud, consistencia y confiabilidad de la información y con la privacidad y confidencialidad de los datos. Las Bases de Datos tienen dentro de sus características elementos que pueden ser utilizados para garantizar la calidad de la información almacenada y procesada.

1.1.4.2 Claves Primarias

Para un valor clave primaria solo existirá una tupla o registro en la tabla. Esta situación garantiza que no se tendrá información repetida o discordante para un valor de clave y puede ser usada como control, para evitar la inclusión de información inconsistente o repetida en las tablas.

1.1.4.3 Dominio de los atributos

El dominio de un atributo define los valores posibles que puede tomar este atributo. Además de los dominios "naturales", usados como tipos de datos, el administrador del sistema puede generar sus propios dominios definiendo el conjunto de valores permitidos. Esta característica, usada en forma correcta, se convierte en mecanismo de control, restricción y validación de los datos a ingresar. Hay que resaltar que estas restricciones siempre serán evaluadas en forma automática por el DBMS.

1.1.4.4 Reglas de Integridad

Son restricciones que definen los estados de consistencia de la Base de Datos. Se debe implementar en especial para verificaciones en cada actualización,

para evitar que se caiga en estados de inconsistencia. En particular se debe verificar que se implementen correctamente la regla de la Entidad (un atributo primo no puede ser nulo) y reglas de Integridad Referencial, esta ultima garantiza que solo se puedan incluir registros para valores previamente ingresados en otras tablas.

1.1.4.5 Reglas de integridad del negocio

Cada negocio funciona en forma diferente y tiene reglas asociadas a su actividad que pueden ser definidas como restricciones en la Base de Datos. Esto implicaría que cualquier operación que se realice debe respetar estas limitantes. Estas son condiciones que la administración coloca a la operación y como principio en el desarrollo de una aplicación, deben ser respetadas por esta.

1.1.4.6 Vistas

Sirve como mecanismo de compartimentación de la información almacenada, permitiendo presentar a diferentes usuarios parte del universo, según se considere necesario. Según las políticas de seguridad, es usual que una gran parte de los usuarios nunca tengan acceso directamente a las tablas completas, sino que lo hagan a través de las vistas, las cuales, por ser un objeto, son sujetas de otras medidas de seguridad.

1.1.4.7 Perfiles de usuario y Acceso a objetos de la Base de Datos

Es la asignación de nombres de usuarios, con su respectiva clave de acceso y perfiles asociados. Pueden también ser creados roles que serán concedidos a los usuarios según sus funciones.

1.1.4.8 Criptografía de Datos

Como recurso de seguridad, se puede mezclar o codificar los datos de modo que, al momento de ser almacenados en disco duro o trasmitidos por alguna línea de comunicación, no sean más que bits no legibles para aquellos que los accedan por un medio no oficial. La criptografía es de gran importancia en las

Bases de Datos pues la información esta almacenada por largos periodos de tiempo en medios de fácil acceso, como discos duros.

1.2 FACILIDADES DE AUDITORÍA EN UNA BASE DE DATOS ORACLE 9i Y 10g.

La Base de Datos Oracle tiene un conjunto de opciones que le permiten auditar todas las operaciones que se realizan dentro de ella. Previo a la activación de una Auditoría el administrador de la Base de Datos debe definir lo siguiente:

- Identificar usuarios, sentencias, u objetos a auditar.
- Identificar las sentencias ejecutadas, decidir si se va auditar las actividades ejecutadas satisfactoriamente o los intentos fallidos. Se recomienda no auditar ambos.
- Monitorear el crecimiento del registro de Auditoría (Audit Trail), para evitar problemas de espacio y para protegerlo de accesos no autorizados, los archivos de Auditoría se encuentran en $ORACLE_HOME/rdbms/audit directory.

La auditoria de la Base de Datos Oracle no debe limitarse solamente a la ejecución de comandos de auditoría, se pueden utilizar otras técnicas que son satisfactorias.

1.2.1 CATEGORÍAS DE AUDITORÍA

1.2.1.1 Auditoría Estándar

Monitorea y registra las operaciones realizadas por un determinado usuario.
Permite obtener información acerca de actividades específicas de la Base de Datos. Por ejemplo se puede determinar que operación realizó determinado usuario, que privilegios utilizó para ello, que objeto afectó, etc.

1.2.1.2 Auditoría de la Aplicación o Basada en Valores

La Auditoría de la Base de Datos no almacena los valores modificados de las columnas, pero si almacena los cambios de las columnas. Es implementada

mediante código en las aplicaciones cliente, procedimientos almacenados o triggers.

Se registra las acciones realizadas por el usuario a mayor nivel de detalle. Los disparadores permiten registrar filas completas de datos antes y después del cambio y almacenarlos en una tabla de registro. Uno de los inconvenientes de este método es que no permite registrar las consultas realizadas sobre los objetos de la Base de Datos.

Ejemplo:
Creamos la tabla log_emp para almacenar los datos insertados en la tabla EMP. La estructura de la tabla EMP se encuentra descrita en la **Figura 1.1**.

```
CREATE TABLE emp_audit
(
  old_empno NUMBER(4),
  old_ename VARCHAR2(10),
  old_job VARCHAR2(9),
  new_empno NUMBER(4),
  new_ename VARCHAR2(10),
  new_job VARCHAR2(9),
  changed_by VARCHAR2(8),
  timestamp DATE
);
```

Figura 1.1: Estructura de la Tabla EMP.

Creamos el disparador que captura el valor de los datos al realizar la operación update sobre la tabla gvarela.EMP. En la **Figura 1.2** se describe el código del disparador.

```
CREATE OR REPLACE TRIGGER Log_changes_emp
BEFORE UPDATE ON gvarela.emp
FOR EACH ROW
BEGIN
INSERT INTO emp_audit
```

```
(changed_by, timestamp, old_empno, old_ename, old_job,new_empno,  new_ename,
new_job)
 VALUES

 (USER, SYSDATE, :old.empno, :old.ename, :old.job,:new.empno, :new.ename,
:new.job);
 END Log_changes_EMP
/
```

Figura 1.2:Código Disparador Log_changes_emp

Para visualizar el funcionamiento del disparador, el usuario system se conecta a la Base de Datos y ejecuta la siguiente sentencia:

update scott.emp set empno=45 where empno=6;

```
SQL> select old_empno,new_empno,changed_by from emp_audit;

 OLD_EMPNO  NEW_EMPNO CHANGED_BY
---------- ---------- --------
         4         45 GVARELA
```

Figura 1.3: Resultados de Auditoria

Finalmente, para visualizar los cambios auditados, realizamos una consulta a la tabla Log_changes_emp, como se muestra en la **Figura 1.3**.

1.2.1.3 Fine Grained Audit

Fine-grained audit permite monitorear las sentencias SELECT realizadas sobre los objetos de la Base de Datos. Esta característica esta basada en triggers internos que se disparan cuando una sentencia SQL es analizada. Fine grained audit es administrado por el paquete DBMS_FGA, que permite al administrador de la Base de Datos definir políticas de Auditoría. Los registros que cumplen con dicha política son insertados en el archivo fine-grained audit trail.

El nivel de detalle del monitoreo incluye:

- La sentencia SQL ejecutada.
- El nombre del usuario que ejecutó la sentencia.
- El SCN (System Change Number) de la sentencia.
- La variable BIND de la sentencia.

Para entender mejor el funcionamiento de Fine Grained Audit partiremos del siguiente ejemplo, el esquema MILLER con privilegios RESOURCE y CONNECT es dueño de las tablas EMP y DEPT.

Creamos la política sobre la columna salario de la tabla EMP cuando el identificador del departamento (deptid) sea 10.

```
execute dbms_fga.add_policy( object_schema => 'MILLER',-
                        object_name => 'emp',-
                        policy_name => 'audit_emp_salary',-
                        audit_condition => 'deptno = 10',-
                        audit_column => 'sal',-
                        handler_schema => 'system',-
                        handler_module   => 'log_emp_salary',-
                        enable => TRUE );
```

Figura 1.4: Código de política de Auditoria

Se realizan sentencias SELECT sobre la tabla emp, se conecta el usuario MILLER y ejecuta la siguiente sentencia:

SELECT ename FROM MILLER.emp WHERE deptno = 10;

Se conecta el usuario PIET y ejecuta la siguiente sentencia:

SELECT ename FROM dept pd , emp pe
WHERE pd.deptno=pe.deptno
AND pd.deptno=10 AND sal > 1000;

Para visualizar los registros de auditoria, se realiza la siguiente consulta

```
SELECT to_char(timestamp, 'YYMMDDHH24MI')
AS timestamp, db_user, policy_name, sql_text
FROM dba_fga_audit_trail;
```

Los resultados de la consulta están descritos en la **Figura 1.5**.

TIMESTAMP	DB_USER	POLICY_NAME	SQL_TEXT
201301822	MILLER	AUDIT_EMP_SALARY	SELECT salary FROM miller.emp
201311143	PIET	AUDIT_EMP_SALARY	SELECT ename FROM dept pd , emp pe WHERE pd.deptno=pe.deptno AND pd.deptno=10 AND sal>1000

Figura 1.5: Resultados ejemplo auditoria FGA

1.2.1.4 Oracle Vault

Es una opción de la Base de Datos 10g release 2 Enterprise Edition que permite proteger los datos del acceso no autorizado de los usuarios con privilegios de administrador y refuerza la estructura de la Base de Datos para evitar cambios no autorizados.

1.2.2 HABILITANDO Y DESHABILITANDO UNA AUDITORÍA ESTÁNDAR

Una vez decidido que auditar, se utiliza el parámetro de inicialización AUDIT_TRAIL, para habilitar la Auditoría para una instancia. Este parámetro indica si el Audit Trail está escribiendo a la tabla de la Base de Datos o un archivo del sistema operativo[4] .

[4] Oracle Database 10g: Administration Workshop I, Oracle Corporation, cap 18

Este parámetro puede tomar uno de los valores mostrados en la **Tabla 1.1**.

DB	Habilita la generación de entradas en la tabla AUD$ del esquema SYS, excepto los registros que siempre son escritos en el Audit Trail del sistema operativo.
XML	Publica todos los elementos del nodo de registro de Auditoría excepto Sql_Text y Sql_Bind en el archivo XML audit del sistema operativo.
DB,EXTENDED	Habilita la generación de entradas en la tabla AUD$ del esquema SYS, incluye también las columnas SQL bind and SQL text de tipo CLOB, estas columnas son pobladas solo si este parámetro es especificado.
XML,EXTENDED	Publica todos los elementos del nodo de registro de Auditoría incluidos Sql_Text y Sql_Bind en el archivo XML audit del sistema operativo, estas columnas son pobladas solo si este parámetro es especificado.
OS	Habilita la Auditoría de la Base de Datos para que todos los registros de Auditoría se almacenen directamente en el Audit Trail del sistema operativo, si el sistema operativo lo permite.
NONE	Deshabilita la Auditoría, este es el valor por defecto.

Tabla 1.1: Opciones de Auditoría Estándar

Los cambios hechos sobre los objetos auditados no requieren reiniciar la Base de Datos, solo se debe reiniciar cuando se hace habilita o deshabilita toda la Auditoría.

Los registros que contiene la información auditada pueden almacenarse en una tabla del diccionario de datos denominada AUD$, o en un archivo en el sistema operativo que se denomina operating system Audit Trail.

1.2.3 DATABASE AUDIT TRAIL

Database Audit Trail consiste una tabla denominada SYS.AUD$ perteneciente al esquema SYS, la información proviene de varias vistas predefinidas tal como DBA_AUDIT_TRAIL. Los registros de Auditoría pueden contener diferentes tipos de información, dependiendo de los tipos de las opciones de Auditoría previamente configuradas.

El Audit Trail no almacena los datos embebidos en las sentencias SQL, por ejemplo si se requiere conocer los valores de las filas previo a un update y los nuevos valores de las filas después de ejecutado el update el archivo Audit Trail no almacena dichos valores, sin embargo este tipo especial de Auditoría se la puede implementar mediante el método Fine-Grained Auditing.

1.2.4 OPERATING SYSTEM AUDIT TRAIL

El archivo Audit Trail del sistema operativo puede incluir la siguiente información:

- Registros de Auditoría generados por el sistema operativo.
- Registros de Auditoría generados por la Base de Datos.
- Acciones que son siempre auditadas en la Base de Datos.
- Registros de Auditoría para usuarios administradores.

Los registros de Auditoría escritos en el sistema operativo pueden contener información encriptada, y pueden ser desencriptados utilizando las tablas del diccionario de datos y los mensajes de error se describen en la **Tabla 1.2**.

Información Encriptada	**Como desencriptar**
Action code	Describe el intento o la operación realizada, utilizando una lista de códigos listados en la tabla AUDIT_ACTIONS del diccionario de datos.
Privileges used	Describe los privilegios del sistema utilizados en la operación, usando el listado de la tabla SYSTEM_PRIVILEGE_MAP del diccionario de datos.
Completion code	Describe el resultado de la operación intentada, utilizando el listado de mensajes de Error de la Base de Datos Oracle. Las operaciones exitosas retornan 0 y las fallidas retornan el código de error correspondiente a la razón de la operación fallida.

Tabla 1.2: Mensajes de Error

Si el archivo de inicialización de la Base de Datos init.ora especifica AUDIT_TRAIL=XML, los registros de Auditoría son escritos en el sistema

operativo como archivos XML. La vista dinámica V$XML_AUDIT_TRAIL permite a los DBAs disponer de dicha información mediante sentencias SQL.

El parámetro de inicialización AUDIT_FILE_DEST especifica el directorio en el que el Audit Trail del sistema operativo y los archivos XML se almacenan cuando se habilita la Auditoría mediante AUDIT_TRAIL=OS o AUDIT_TRAIL=XML.

Los registros de Auditoría para el usuario SYS se almacenan en el ese directorio si se especifica el parámetro AUDIT_SYS_OPERATIONS.

Para cambiar de directorio se realiza un ALTER SYSTEM SET AUDIT_FILE_DEST = <dir> DEFERRED, el cambio tomará efecto en las siguientes sesiones.

1.2.5 SYSLOG AUDIT TRAIL

Una potencial vulnerabilidad del archivo Audit Trail del sistema operativo es el privilegio del usuario, tal como DBA que puede modificar o borrar los registros de Auditoría, por ello para minimizar este riesgo se puede utilizar el archivo syslog Audit Trail el cual permite registrar la información en un archivo del sistema operativo, cuya ubicación está determinado por un proceso en background denominado syslog.

Esta vulnerabilidad no está disponible en Windows porque los registros de Auditoría no pueden ser modificados directamente. En los Sistemas operativos Windows los registros de Auditoría son almacenados y monitoreados a través de Event Viewer.

Para especificar el nivel del archivo SYSLOG se especifica el parámetro AUDIT_SYSLOG_LEVEL en el archivo de inicialización init.ora a facility y a priority, con el formato AUDIT_SYSLOG_LEVEL=facility.priority, donde facility describe que parte del sistema operativo.

1.2.6 OPCIONES DE AUDITORÍA

1.2.6.1 Auditoría de las sentencias SQL

Son aquellas ejecutadas sobre un objeto sin especificar un esquema en el cual operan. Las sentencias pueden caer en las siguientes categorías:

Sentencias DDL[5]

Por ejemplo AUDIT TABLE audita todas las sentencias CREATE TABLE, DROP TABLE.

Sentencias DML[6]

Por ejemplo AUDIT SELECT TABLE audita todas las sentencias SELECT sin tener en cuenta la tabla o la vista.

1.2.6.2 Auditoría de privilegios

La Auditoría de privilegios esta más enfocado a la Auditoría de sentencias, audita las sentencias que utilizan un privilegio del sistema, por ejemplo AUDIT SELECT ANY TABLE audita todas las sentencias de los usuarios con el privilegio AUDIT SELECT ANY TABLE.

Si se determina una Auditoría por sentencias y a su vez una Auditoría de privilegios, solo un registro de Auditoría es generado, por ejemplo si la sentencia a auditar es TABLE y el privilegio del sistema es CREATE TABLE, ambos son auditados, pero solo un registro es generado cuando la tabla es creada.

La Auditoría de privilegios audita tipos específicos de sentencias, por ejemplo en la Auditoría de sentencias la cláusula TABLE audita sentencias CREATE TABLE, ALTER TABLE, y DROP TABLE, sin embargo al auditar por privilegios, CREATE TABLE se audita solamente sentencias CREATE TABLE, porque solo una sentencia CREATE TABLE requiere un privilegio CREATE TABLE

---

[5] DDL Lenguaje de Definición de Datos
[6] DML Lenguaje de Manipulación de Datos

1.2.6.3 Auditoría de un objeto de un esquema

Audita específicas clases de sentencias SQL sobre un particular objeto de un esquema determinado de la BD, tal como AUDIT SELECT ON HR.EMPLOYEES, este tipo de Auditoría se aplica a todos los usuarios de la Base de Datos.

Estas opciones de Auditoría aceptan las siguientes condiciones:

- WHENEVER SUCCESSFUL / WHENEVER NOT SUCCESSFUL
- BY SESSION / BY ACCESS

Whenever successful

Cuando una determinada opción de Auditoría incluye la cláusula WHENEVER SUCCESSFUL, audita solo las operaciones exitosas de la sentencia ejecutada.

Whenever not successful

Cuando una determinada opción de Auditoría incluye la cláusula WHENEVER NOT SUCCESSFUL, audita solo las operaciones no exitosas de la sentencia ejecutada.

Auditar una sentencia no exitosa provee un reporte si la sentencia SQL es valida pero falla, por falta de privilegios o el esquema al cual esta referenciado el objeto no existe.

Cláusula by access

Mediante la cláusula BY ACCESS cada ejecución de una sentencia auditable genera un registro en el archivo de Auditoría.

Por ejemplo al auditar la sentencia SELECT TABLE.

El usuario JWARD se conecta a la Base de Datos y ejecuta cinco sentencias SELECT a la tabla departamentos y se desconecta de la Base de Datos.

El usuario SWILLIAMS se conecta a la Base de Datos y ejecuta tres sentencias SELECT a la tabla departamentos y se desconecta de la Base de Datos.

En este caso el archivo Audit Trail contiene ocho registros con ocho sentencias SELECT.

Cláusula by session

Mediante la cláusula BY SESSION se genera un registro por cada sesión, por cada usuario y por cada objeto.

Por ejemplo al auditar la sentencia SELECT TABLE.

El usuario JWARD se conecta a la Base de Datos y ejecuta cinco sentencias SELECT a la tabla departamentos y se desconecta de la Base de Datos.

El usuario SWILLIAMS se conecta a la Base de Datos y ejecuta tres sentencias SELECT a la tabla empleados y se desconecta de la Base de Datos.

En este caso el archivo Audit Trail contiene dos registros de auditoría, cada uno con ocho sentencias SELECT por cada sesión.

La información de Auditoría se encuentra en las vistas listadas en la Tabla 1.3

VISTA	DESCRIPCIÓN
ALL_DEF_AUDIT_OPTS	Opciones de Auditoría por defecto
DBA_STMT_AUDIT_OPTS	Opciones de auditoria de sentencias SQL
DBA_PRIV_AUDIT_OPTS	Opciones de Auditoría de privilegios
DBA_OBJ_AUDIT_OPTS	Opciones de Auditoría de Objetos
DBA_AUDIT_TRAIL	Audita todas las entradas del Audit Trail
DBA_AUDIT_OBJECT	Audita todas las entradas sobre objetos de esquemas
DBA_AUDIT_SESSION	Almacena la información de las sesiones.
DBA_AUDIT_STATEMENT	Audita todas las entradas sobre conexiones y desconexiones.

Tabla 1.3: Vistas de Auditoría

1.3 METODOLOGÍA A UTILIZAR

1.3.1 SELECCIÓN DE LA METODOLOGÍA

La Ingeniería de Software es relativamente joven con respecto a sus similares en otros campos del conocimiento; sin embargo, se ha desarrollado en forma vertiginosa correspondiendo a los avances de la tecnología electrónica y de las ciencias de la computación.

En el mismo sentido, el desarrollo de software ha debido ajustarse a las herramientas tecnológicas disponibles y principalmente a la capacidad de abstracción de los lenguajes de programación, lo cual se refleja en los distintos enfoques de las metodologías de desarrollo que han aparecido desde finales de los 60 hasta la actualidad.

Estas consideraciones y la naturaleza del problema a ser resuelto, junto con los recursos y plazos existentes, brindaron los lineamientos para escoger una metodología para el presente proyecto.

1.3.1.1 Respecto a la tecnología:

El proyecto planteado implicaba trabajar sobre un gestor de bases de datos Oracle, y explotar su funcionalidad, lo cual significaba utilizar lenguajes de programación acordes, como PL/SQL y posiblemente C.

Tanto PL/SQL como C, son estructurados, por lo cual un análisis y modelado orientado a objetos, habría resultado menos apropiado respecto a la tecnología que un enfoque estructurado, más coherente con procedimientos y funciones.

1.3.1.2 Respecto al problema

El problema a ser resuelto consistía en crear una aplicación capaz de auditar operaciones DML en Bases de Datos Oracle. Por lo cual, el programa no sería complejo por tener muchas funcionalidades, ni por tener que manejar diversos

tipos de usuarios o un esquema de datos elaborado; el aplicativo sería complejo en cuanto a su programación dentro del gestor de bases de datos.

Esta situación brindaba la pauta de que una metodología como la propuesta por el Proceso Unificado (basada en la arquitectura y en casos de uso) demandaría establecer modelos que posiblemente no se requirieran para analizar y diseñar nuestro software. En cambio, un enfoque estructurado, como el propuesto por Yourdon, sería capaz de representar todo el sistema mediante modelos más sencillos y mediante especificaciones de los procesos que se convertirían en procedimientos y funciones a posteriori.

1.3.1.3 Respecto a los recursos y plazos

Es conocido que una de las fortalezas del Proceso Unificado es la realización de iteraciones e incrementos, y la producción de artefactos que evolucionan constantemente. No obstante, esto requiere de un importante esfuerzo, que a su vez se traduce en tiempo y por ende en costos.

Además, teniendo en cuenta que la complejidad del desarrollo no se encontraba en analizar y modelar un sistema “rico en clases y objetos”, sino en determinar mejores formas de programar procedimientos y funciones que trabajaran en el motor de bases de datos Oracle, habría sido poco producente emplear un gran esfuerzo en cumplir cabalmente el Proceso Unificado. La elección de una metodología estructurada permitiría un mayor énfasis en el análisis y diseño de funciones, disparadores, procedimientos, disminuyendo el esfuerzo para construir el programa, y a su vez los recursos y tiempo necesarios.

La metodología Orientada a Objetos permite abstraer el nivel de complejidad del modelo de Datos, al tener un modelo de datos sencillo, no sería lo más recomendado utilizar dicha metodología.
Se requiere mayor cantidad de recursos (tiempo, recurso Humano) para aplicar la metodología Orientada a Objetos, el presente proyecto se enfoca a explotar

las funcionalidades del motor de Base de Datos por ello una metodología estructurada se adapta mejor al enfoque del problema y la solución.
Un atributo importante dentro del paradigma orientado a objetos es la reutilización, la herramienta manejará archivos planos por ello el nivel de reutilización es bajo.

1.3.2 ETAPAS DE LA METODOLOGÍA ESTRUCTURADA

1.3.2.1 Concepción del proyecto

Comprende la identificación del problema, la cual tiene por objetivo lograr un entendimiento claro de las necesidades de la organización y del ambiente en que operará la herramienta a implantar. Además, introduce las actividades de estimación de recursos y planificación de plazos y responsabilidades.

1.3.2.2 Análisis del Sistema

El análisis utiliza una combinación de texto y diagramas para representar los requisitos de datos, funciones y comportamientos.
Dentro de la estructura del modelo de análisis tenemos los siguientes elementos:

- El diccionario de datos que es un almacén que contiene definiciones de todos los objetos de datos consumidos y producidos por el software.
- El diagrama de Flujo de datos indica como se transforman los datos a medida que se desarrolla el aplicativo, además permite representar las funciones y subfunciones que transforman el flujo de datos.
- El diagrama de estados indica como se comporta el sistema frente a los eventos externos.

Los productos en esta fase son:

- Modelo Ambiental
- Modelo de Comportamiento
- Modelo de Información
- Modelo de Rendimiento

1.3.2.3 Diseño

Se dedica a la creación de una jerarquía apropiada de módulos de programas y de interfaces para implantar las especificaciones creadas en la etapa de análisis.

Durante el diseño se desarrollan, se revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes de software. El diseño da como resultados representaciones del software capaces de orientar claramente su codificación.

La modularidad y el concepto de abstracción permiten reutilizar y simplificar los componentes del software.

Los entregables en la fase de Diseño son:

- Modelo Físico de la base de datos
- Estándares de programación
- Diseño preliminar de interfaces

1.3.2.4 Construcción

Incluye la codificación y la integración de módulos en un esqueleto progresivamente más completo del sistema final.
Los productos en esta etapa son:

- Código Fuente
- Plan de Pruebas
- Resultados de las Pruebas
- Manuales de Usuario

1.3.2.5 Pruebas

Esta etapa tiene por objetivo verificar la funcionalidad y confiabilidad del programa y descubrir la mayor cantidad de errores (tal vez sea mejor decir: “y determinar posibles deficiencias programáticas y de diseño”), para ello se debe planificar y ejecutar una serie de pasos: pruebas de unidad, integración, validación y del sistema. Las pruebas de unidad y de integración se centran en la verificación funcional de cada módulo y en la incorporación de los módulos en una estructura de programa. La prueba de validación demuestra el seguimiento de los requisitos de software y la prueba del sistema valida el software una vez que se ha incorporado en producción.

1.3.2.6 Instalación

Actividad final, pone en funcionamiento el sistema.

CAPITULO 2.- ANÁLISIS Y DISEÑO DE LA HERRAMIENTA

En el presente capítulo se define el proyecto, es decir, se identifica el problema, se determinan los objetivos a cumplirse y el alcance respectivo; además, se documenta la planificación de recursos y esfuerzo.

Así mismo, se especifican los requerimientos, se los analiza y en base a ello se elaboran los modelos: esencial, de comportamiento, de información y de rendimiento. Finalmente, se define el entorno de desarrollo, se diseña el modelo físico de la base de datos, se diseñan las interfaces a nivel lógico, y se especifican los estándares de programación a utilizarse en la siguiente parte del desarrollo.

2.1 ANÁLISIS

2.1.1 CONCEPTOS GENERALES

2.1.1.1 Concepto de desarrollo de aplicaciones con PL/SQL

*2.1.1.1.1 SQL *PLUS[7]*

SQL *Plus es una herramienta que permite establecer conexión con el servidor de la Base de Datos Oracle, para poder trabajar con la información de la misma. Utilizando SQL*Plus se puede ingresar, revisar, guardar, recuperar y correr comandos SQL, bloques PL/SQL y SQL*Plus, ingresar comandos SQL*Plus para configurar su ambiente, realizar cálculos y presentar los resultados de una consulta, interactuar con el usuario final, subir o bajar una Base de Datos, realizar una conexión a Base de Datos, definir variables, capturar errores, listar definiciones para una tabla y administrar la Base de Datos.

---

[7] http://download-east.oracle.com/docs/html/A88816_01/toc.htm SQL*Plus User's Guide and Reference Release 9.0.1 Abril 2000

*2.1.1.1.2 Acceso a SQL *PLUS*

Se puede utilizar SQL *Plus desde línea de comandos e interfaz gráfica.

Mediante línea de comandos

Se debe abrir una ventana de comandos y luego escribir el texto sqlplus, obteniendo la siguiente pantalla:

*Figura 2.1: Ventana Comandos para acceder a SQL *Plus*

Posteriormente se debe colocar el nombre de usuario y contraseña para acceder a la base de datos, tras indicar esta información se realizar la conexión con Oracle mediante SQL*Plus, y aparecerá el símbolo: SQL>

Posteriormente se puede introducir comandos SQL. El símbolo SQL puede cambiar a un símbolo con números 1, 2, 3, etc.; en ese caso se nos indica que la instrucción no ha terminado y la línea en la que estamos.

Otra posibilidad de conexión consiste en llamar al programa SQL*Plus indicando contraseña y Base de Datos a conectar. El formato es:

slplus usuario/contraseña@nombreServicioBaseDeDatos

Ejemplo:

slplus dmluser/dmluser@bdd2

Mediante interfaz gráfica

Oracle incorpora un programa gráfico para Windows para utilizar SQL*Plus. Se puede llamar a dicho programa desde las herramientas instaladas en el menú de programas de Windows, o desde la línea de programas escribiendo sqlplusw, obteniendo la siguiente pantalla:

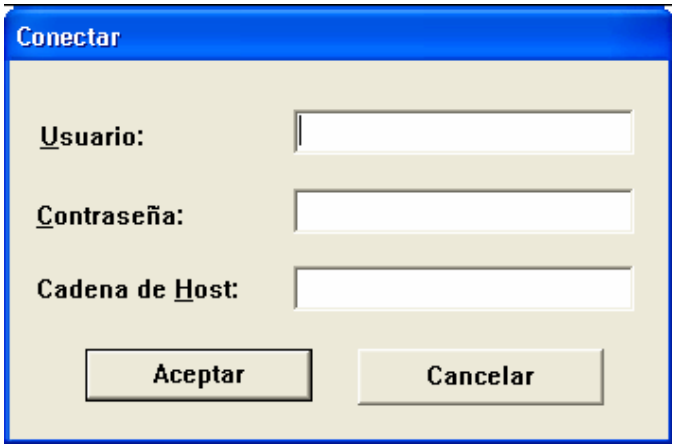

*Figura 2.1: Interfaz gráfica para acceder a SQL *Plus*

Donde se coloca el nombre de usuario y contraseña. La cadena de Host que permite realizar una conexión con una base de datos remota.
A continuación se da clic en OK y la pantalla de SQL*Plus es desplegada.

También se puede llamar a este ambiente desde la línea de comandos utilizando la sintaxis comentada anteriormente. En este caso:
slplusw usuario/contraseña@nombreServicioBaseDeDatos

*2.1.1.1.3 Arquitectura de SQL *PLUS*

La arquitectura de SQL *PLUS, es cliente – servidor y se detalla a continuación:

1.- Cliente, esta compuesta de una interfaz de usuario de línea de comandos y del componente Oracle net.
2.- Base de Datos Oracle 9i y el componente Oracle net.

El componente Oracle net, permite la comunicación entre el cliente de SQL*Plus y la Base de Datos.

Y se muestra en la **Figura 2.3**:

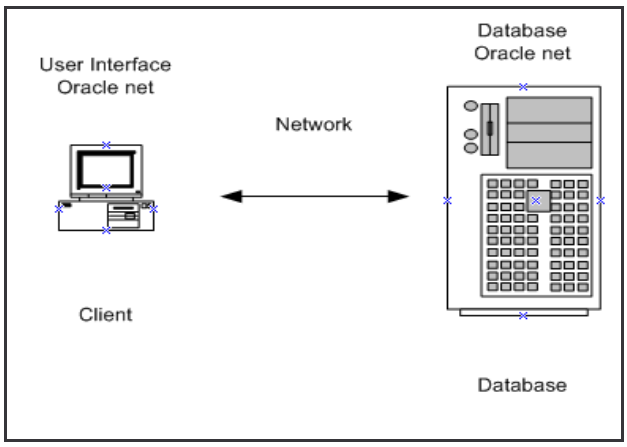

*Figura 2.3: Arquitectura de SQL *Plus*

2.1.1.2 iSQL *PLUS[8]

iSQL*Plus permite el uso del browser para conectarse a Oracle 9i y efectuar las mismas acciones que se hacen en SQL * Plus, ya que este es un componente del producto SQL*Plus. Utilizando iSQL*Plus , se puede acceder y copiar datos entre base de datos, escribir, editar, correr y guardar comandos SQL*Plus, SQL y bloques PL/SQL, calcular e imprimir resultados, listar definiciones para cualquier tabla y realizar administración de la base de datos, si el usuario que ha accedido a la Base de Datos tenga privilegios de SYSDBA.

*2.1.1.2.1 Acceso a iSQL *PLUS*

Utilizar iSQL*Plus es indicar una dirección web en un navegador, esa dirección es la de la página iSQL*Plus de acceso a la Base de Datos.

---

[8] http://download-east.oracle.com/docs/html/A88826_01/toc.htm iSQL*Plus User's Guide and Reference Release 9.0.1 Junio 2001

En la página de acceso se debe introducir el nombre de usuario, contraseña y nombre de la Base de Datos con la que se requiere conectarse. Y aparece la siguiente pantalla:

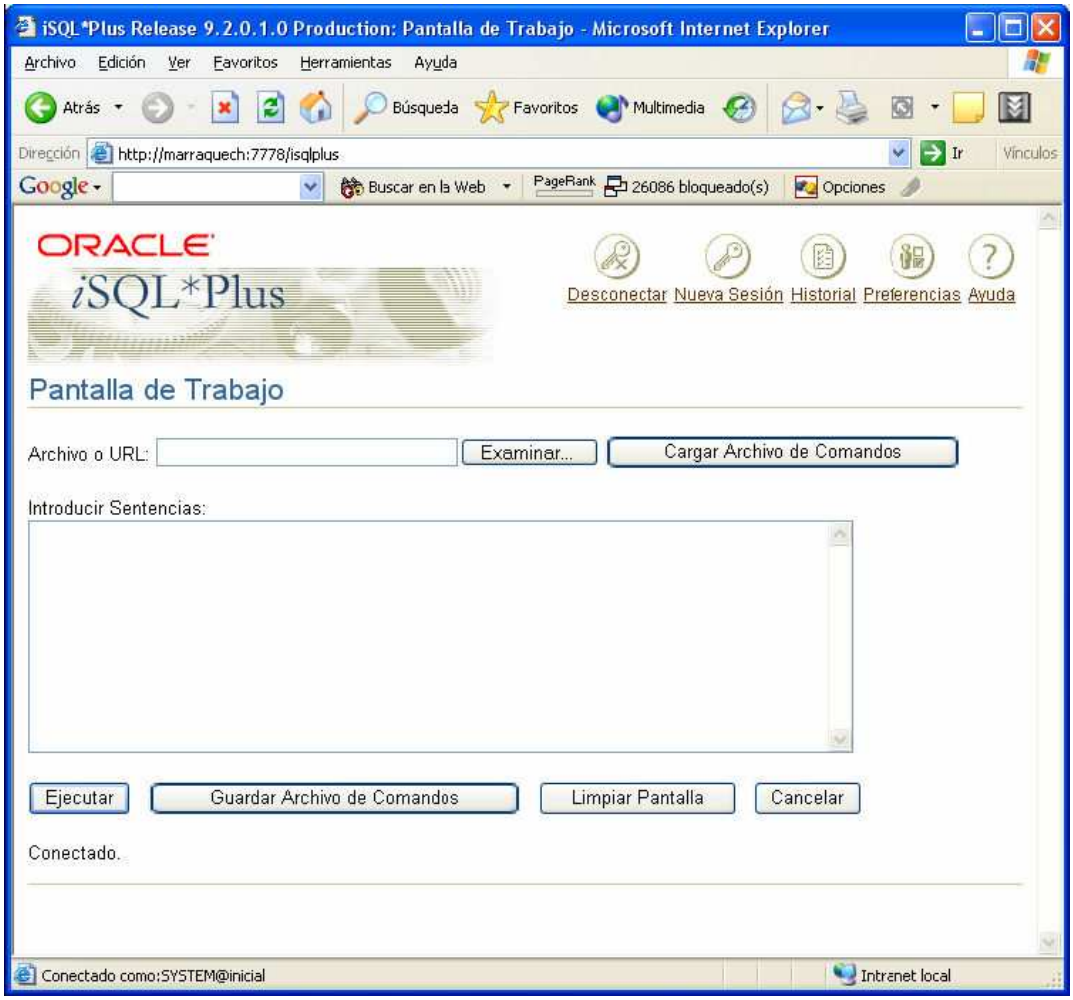

*Figura 2.4: Interfaz gráfica para acceder a iSQL *Plus*

En esa pantalla en el apartado Introducir Sentencias, se escribe la sentencia que se desea consultar o enviar. El botón Ejecutar hace que se valide y se envíe a Oracle.

Se pueden almacenar sentencias SQL usando el botón Examinar y cargar sentencias previamente guardadas mediante Cargar archivos de comandos.

*2.1.1.2.2 Arquitectura de iSQL *PLUS*

La arquitectura de iSQL *PLUS, se muestra en la **Figura 2.5**:

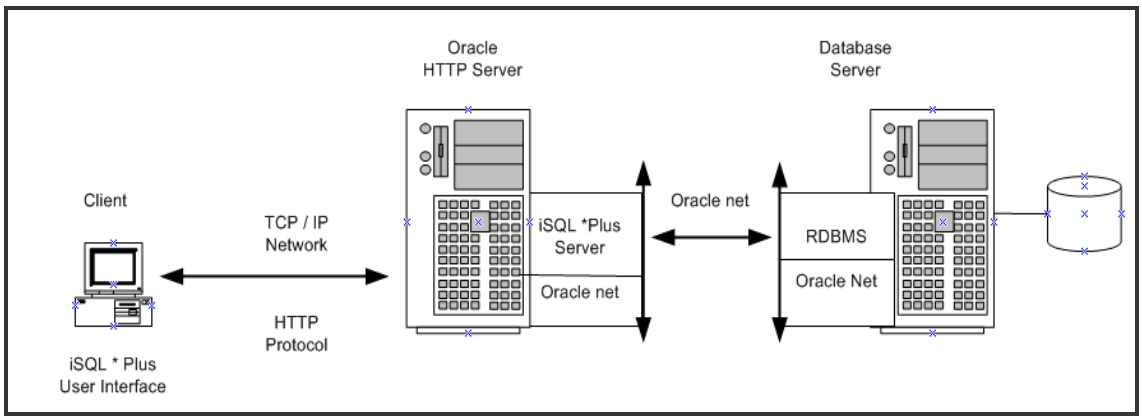

*Figura 2.5: Arquitectura de iSQL *Plus*

La arquitectura de iSQL *PLUS, se base en tres capas que se detallan a continuación:

1.- Capa Cliente, por esta se tiene acceso a la interfaz de usuario de iSQL* Plus utilizando un browser, el mismo que se encuentra conectado a Internet o una intranet. El acceso a iSQL*Plus, es a través del URL de Oracle http Server.

2.- Capa Intermedia, la misma que coordina las interacciones y recursos entre la capa cliente y base de datos

Esta capa está compuesta por: iSQL*Plus Server que permite la comunicación y autenticación entre la interfaz de iSQL*Plus y la Base de Datos, Oracle Net, permite la comunicación entre el cliente de SQL*Plus y la Base de Datos y Oracle HTTP Server, que es un servidor web, cuyo detalle se encuentra en la siguiente sección.

3.- Capa de Base de datos en donde se encuentra la información a ser accesada.

2.1.1.3 HTTP SERVER

Oracle HTTP Server es el componente web del servidor de Base de Datos y servidor de Aplicaciones Oracle, provee una infraestructura clave para atender a una petición realizada a través del protocolo http.
Se basa en la infraestructura de Apache, su arquitectura se muestra en la **Figura 2.6.**

El cliente a través de un navegador realiza una petición al HTTP Server, el listener escucha la petición del usuario y redirecciona al respectivo módulo de acuerdo a los parámetros de la petición, cada módulo se encarga de procesar la petición y enviar respuesta al cliente, dependiendo del tipo de petición el modulo respectivo interactúa con la Base de Datos previamente a responder a la petición del usuario.

Cliente

Figura 1.11: Arquitectura del HTTP Server

*Figura 2.6: Arquitectura de iSQL *Plus*

2.1.1.3.1 Componentes del HTTP Server

La descripción de cada uno de los componentes de HTTP Server se describe en la **Tabla 2.1**.