Universidad del Aconcagua

Facultad de Ciencias Sociales y Administrativas

Ingeniería en Software

Nicolás Alberto Guevara

2013

Tutor: Rosana Giménez

Sistema de Control y Gestión Tributaria

Mendoza, 25 de Junio de 2013

NOTA:

Resumen técnico

Se trata de un sistema en el que carga la totalidad de los objetos y contribuyentes de un determinado municipio junto con sus características físicas y tributarias (características previamente establecidas que se tienen en cuenta para el cobro de impuestos municipales), para, de esta manera generarles a cada objeto una cuenta corriente en la que se incluye el monto a pagar de impuesto municipal. Además cuenta con la posibilidad de agregar datos adicionales de interés como son los domicilios real y de pago de los objetos y contribuyentes, la posibilidad de generación de planes de pago de las deudas de cada objeto, la generación de apremios para evitar las prescripciones de deuda, la realización de cobros y el control de los mismos.

Además se agrega la posibilidad a los funcionarios de controlar y verificar un historial de cobros y cuentas generadas por medio de una herramienta muy fácil de utilizar por ellos y práctica como es el Data warehouse.

Indice

1. Introducción

1.1. Planteo de la Problemática

Hoy en día la mayoría de los municipios poseen sistemas muy antiguos y no tienen bien normalizados los datos, por este motivo no pueden llevar un control riguroso sobre los contribuyentes y sus respectivas deudas, ni objetos que estos poseen, de esta forma se hace muy difícil poder identificar a los contribuyentes con deuda y trabajar con ellos para poder llegar a cobrar de alguna manera la deuda por algún tipo de tributo.

Otro problema que existe es que hay gran cantidad de información obsoleta, que ocupa mucho espacio en la base y hace que sea complicado trabajar con los datos de los contribuyentes ya que confunden a los operadores, además al no estar normalizada, no hay datos certeros de los contribuyentes y no hay manera de identificar a un contribuyente único en el sistema y esto también hace complicado el trabajo individual sobre un contribuyente particular en términos de seguimiento de bienes(Cantidad de objetos que posee) o de deuda que posea sobre los mismos.

1.2. Objetivos

1.2.1. Principal

Desarrollar un sistema tributario en el que los usuarios puedan cargar a los contribuyentes de un determinado municipio y sus respectivos objetos (Propiedad Raíz, Comercio, Multa, etc.), para poder llevar un control del cobro de los impuestos tributarios y sus correspondientes deudas (Cuenta corriente de los contribuyentes).

1.2.2. Específicos

- Desarrollar un proceso de migración de datos desde el sistema que se encuentra en funcionamiento alojado en Progress. Todos estos procesos se realizarán en conjunto con el personal de cómputos de la municipalidad, el cual cumplirá la función de supervisión de los procesos que se vayan realizando.
- Desarrollar nuevos módulos en el sistema, los cuales se utilizan para facilitar la utilización del sistema por los usuarios y el entendimiento y búsqueda de información para los objetos de cada tributo.
- Desarrollar módulos de control de pagos, ingresos, y reportes con historiales y auditorias del sistema. Los cuales ayudan a un mejor control de las autoridades sobre el funcionamiento y operatividad del sistema.
- Detallar los requerimientos mínimos de equipos para el correcto funcionamiento del sistema. Los requerimientos mínimos del sistemas son: Windows XP, 1GB de RAM, Procesador Pentium 4: 1,8 GHz, 20 GB disco. Las máquinas que se escogieron para ser instaladas son BANGHO CI-2248.

- Desarrollar módulos del Data warehouse en el que podrán acceder las autoridades municipales y consultar diferentes conjuntos de datos descriptos en los ya mencionados módulos.

- Describir la capacitación a los usuarios de los diferentes sectores del establecimiento y establecer la fecha de las mismas.

1.3. Escenario

Secretaría de Hacienda de la Municipalidad de la ciudad de San Martín

Localizada en la intersección de las calles Boulogne Sur Mer y 25 de Mayo, es el lugar donde se lleva a cabo el cobro de todos los impuestos municipales y donde se realizó la mayoría de las entrevistas y estudio del flujo de la información, además existe una delegación Ubicada en la cuidad de Palmira y dos sectores ubicados en otro establecimiento, estos son, dirección de comercio y dirección de obras públicas.

Por lo tanto la estructuración de este movimiento no es sencilla, más allá del aprendizaje de las plataformas (software) con que se trabaja, se toma en cuenta la cantidad de personas que se acercan al municipio día a día para regularizar su situación.

También persiste el inconveniente de la lejanía de los sectores previamente nombrados y el desarrollo de la conectividad con estos, ya que en la actualidad la conexión es vía telefónica únicamente.

Se realizó una aplicación que cubra tanto las necesidades de control dentro del municipio como los tiempos de respuesta para que el contribuyente tenga que esperar lo menos posible en cada trámite.

La información proporcionada por el Sub-Secretario de Hacienda, junto con el estudio de los procesos del sistema actual y del flujo de la información, se volcó en diferentes tablas lo que resultó en la base de datos de la aplicación.

1.4. Alcances y Limitaciones

1.4.1. Limitaciones

El sistema no va a realizar el cobro de la totalidad de los impuestos municipales, ya que hay sectores que no corresponden a la secretaría de Hacienda.

Existen dos delegaciones que debido a su lejanía y la poca rentabilidad de realizar algún tipo de conexión e instalación de equipos no están incluidas dentro del desarrollo del proyecto.

1.4.2. Alcances

El sistema incluye el cobro de impuestos y control de deuda de los sectores de:

- Catastro municipal
- Obras privadas
- Comercio e Industria
- Sellados
- Multas (excepto automotor)
- Apremios
- Cementerio

Ofrece la posibilidad de cargar y modificar un padrón de contribuyentes, domicilios de los contribuyentes y de objetos (comercios, inmuebles, etc.).

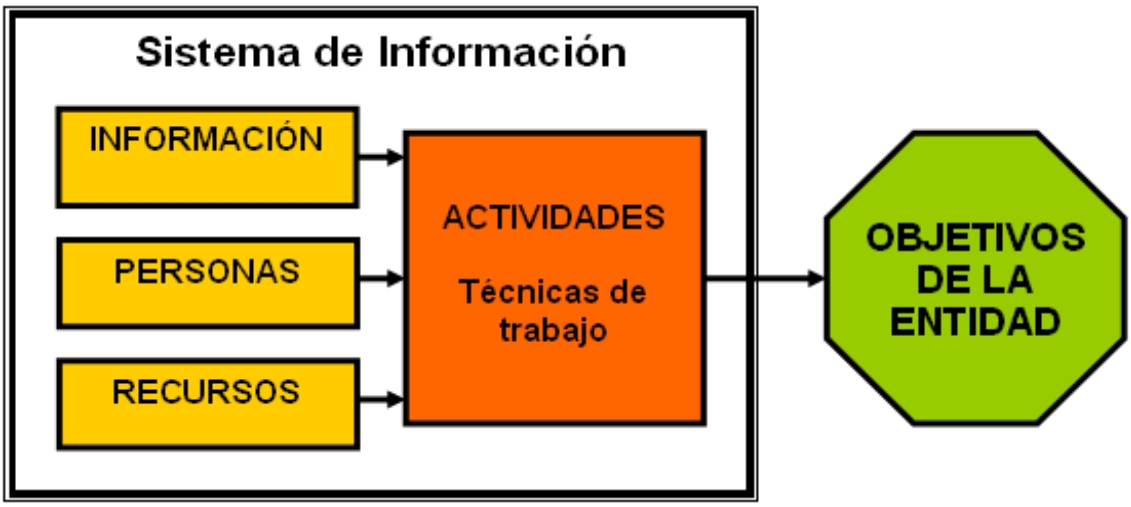
Generación de cuentas corrientes para cada uno de los impuestos incluidos, así como también la creación de planes de pago, impresión y cobro de boletas, cobro y conciliación de cajas.

2. Marco Teórico y Antecedentes

2.1. Sistemas de Información

Un sistema de información (SI) es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad u objetivo. Dichos elementos formarán parte de alguna de las siguientes categorías:

- Personas
- Datos
- Actividades o técnicas de trabajo
- Recursos materiales en general (generalmente recursos informáticos y de comunicación, aunque no necesariamente).



Referencia: http://commons.wikimedia.org/wiki/File:Esquema_sistema_de_informacion.png

Todos estos elementos interactúan para procesar los datos (incluidos los procesos manuales y automáticos) y dan lugar a información más elaborada, que se distribuye de la manera más adecuada posible en una determinada organización, en función de sus objetivos.

Habitualmente el término se usa de manera errónea como sinónimo de *sistema de información informático*, en parte porque en la mayor parte de los casos los recursos materiales de un sistema de información están constituidos casi en su totalidad por sistemas informáticos. Estrictamente hablando, un sistema de información no tiene por qué disponer de dichos recursos (aunque en la práctica esto no suela ocurrir). Se podría decir entonces que los sistemas de información informáticos son una subclase o un subconjunto de los sistemas de información en general.

Referencia:

- Sistemas de información gerencial- Administración de la empresa digital. Pearson Educación- Prentice Hall. Laudon, Jane y Kenneth (2006).
- Análisis y Diseño de Sistemas. Kendall & Kendall

2.2. Sistemas de Bases de Datos

2.2.1. Que son las bases de datos.

Una base de datos es un conjunto de elementos de datos que se describe así mismo, con relaciones entre esos elementos, que presenta una interfaz uniforme de servicio.

También la podemos definir como un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

"Las deficiencias de procesamiento de información antes de las bases de datos comprenden datos codificados, interdependencia entre programas y archivos de datos, repetición de datos e inconsistencias relativas, representación especifica de relaciones entre elementos de datos, falta de coordinación entre programas que usan datos comunes, acceso simultaneo restringido a datos, y métodos de recuperación de errores no uniformes"(James L. Johnson)

Existen programas denominados sistemas gestores de bases de datos, abreviados SGBD, qué 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.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo, en Argentina los datos personales se encuentran protegidos por la Ley 25.326 Protección de los Datos Personales.

2.2.2. Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

Bases de datos jerárquicas

En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Base de datos de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la

información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Bases de datos transaccionales

Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción e industrial, es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales.
Un ejemplo habitual de transacción es el traspaso de una cantidad de dinero entre cuentas bancarias. Normalmente se realiza mediante dos operaciones distintas, una en la que se decrementa el saldo de la cuenta origen y otra en la que incrementamos el saldo de la cuenta destino. Para garantizar la atomicidad del sistema (es decir, para que no aparezca o desaparezca dinero), las dos operaciones deben ser atómicas, es decir, el sistema debe garantizar que, bajo cualquier circunstancia (incluso una caída del sistema), el resultado final es que, o bien se han realizado las dos operaciones, o bien no se ha realizado ninguna.

Bases de datos relacionales

Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y *campos* (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más

fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, *Structured Query Language* o *Lenguaje Estructurado de Consultas*, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años 80 la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Bases de datos multidimensionales

Son bases de datos ideadas para desarrollar aplicaciones muy concretas, como creación de Cubos OLAP. Básicamente no se diferencian demasiado de las bases de datos relacionales (una tabla en una base de datos relacional podría serlo también en una base de datos multidimensional), la diferencia está más bien a nivel conceptual; en las bases de datos multidimensionales los campos o atributos de una tabla pueden ser de dos tipos, o bien representan dimensiones de la tabla, o bien representan métricas que se desean estudiar.

Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.

- Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

SQL: 2003, es el estándar de SQL92 ampliado, soporta los conceptos orientados a objetos y mantiene la compatibilidad con SQL92.

Bases de datos documentales

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

Bases de datos deductivas

Un sistema de base de datos deductiva, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. Las bases de datos deductivas son también llamadas bases de datos lógicas, a raíz de que se basa en lógica matemática. Este tipo de base de datos surge debido a las limitaciones de la Base de Datos Relacional de responder a consultas recursivas y de deducir relaciones indirectas de los datos almacenados en la base de datos.

Ventajas

- Uso de reglas lógicas para expresar las consultas.
- Permite responder consultas recursivas.
- Cuenta con negaciones estratificadas
- Capacidad de obtener nueva información a través de la ya almacenada en la base de datos mediante inferencia.
- Uso de algoritmos de optimización de consultas.

- Soporta objetos y conjuntos complejos.

Desventajas

- Crear procedimientos eficaces de deducción para evitar caer en bucles infinitos.
- Encontrar criterios que decidan la utilización de una ley como regla de deducción.
- Replantear las convenciones habituales de la base de datos.

2.2.3. Tipos de Bases de Datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan.

Según la variabilidad de los datos almacenados

Bases de datos estáticas

Son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones, tomar decisiones y realizar análisis de datos para inteligencia empresarial.

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado, una farmacia, un videoclub o una empresa.

Referencia:

- BASES DE DATOS: Modelos, lenguajes, diseño. JAMES L. JOHNSON

2.2.4. Características de Oracle, Firebird y PostgreSQL.

2.2.4.1. Oracle

Oracle es básicamente un herramienta cliente/servidor para la gestión de base de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que solo se vea en empresas muy grandes y multinacionales, por norma general.

En el desarrollo de paginas Web pasa lo mismo como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server etc.

Oracle como antes lo mencionamos se basa en la tecnología cliente/ servidor, pues bien, para su utilización primero seria necesario la instalación de la herramienta servidor (Oracle8i) y posteriormente podríamos atacar a la base de datos desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas de programación sobre Oracle a partir de esta premisa vamos a desarrollar las principales acepciones de Oracle y sus aplicaciones en las distintas áreas de trabajo.

Es posible lógicamente atacar 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 Developer es una herramienta que nos permite crear formularios en local, es decir, mediante esta herramienta nosotros podemos crear formularios, compilarlos y ejecutarlos, pero si queremos que los otros trabajen sobre este formulario deberemos 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 carpeta.

Este sistema como podemos observar es bastante engorroso y poco fiable pues es bastante normal que las versiones se pierdan y se machaquen con frecuencia. La principal ventaja de esta herramienta es que es bastante intuitiva y dispone de un modo que nos permite componer el formulario, tal y como lo haríamos en Visual Basic o en Visual C, esto es muy de agradecer.

Los problemas anteriores quedan totalmente resueltos con Designe que es una herramienta que se conecta a la base de datos y por tanto creamos los formularios en ella, de esta manera todo el

mundo se conecta mediante Designe a la aplicación que contiene todos los formularios y no hay problemas de diferentes versiones, esto es muy útil y perfecto para evitar machacar el trabajo de otros.

Pero el principal y más notable problema es la falta de un entorno visual para diseñar el formulario, es decir, nos aparece una estructura como de árbol en la cual insertamos un formulario, a la vez dentro de éste insertamos bloques o módulos que son las estructuras que contendrán los elementos del formularios, que pueden estar basados en tablas o no.

Por lo tanto si queremos hacer formularios para practicar o para probar qué es esto de Oracle, recomiendo que usen Developer pues es mucho más fácil e intuitivo al principio.

El software que produce Oracle no sólo soporta datos alfanuméricos ubicados en las tradicionales "filas y columnas" de las bases de datos, sino que también soporta textos sin estructura, imágenes, audio y video. Puede ser usado tanto para el manejo de información personal, como para gigantescas bibliotecas multimedia, y corre en equipos desde la más pequeña laptop hasta la mayor supercomputadora.

Una de las herramientas más populares de esta empresa es "Oracle Power Objects". Fue la primera herramienta en la industria que le permitía a los desarrolladores de software, trabajar con el sistema de arrastrar y soltar los objetos. También les permite desarrollar y correr aplicaciones cliente/servidor que soportan desde cinco a cincuenta usuarios en cualquier plataforma, incluyendo Windows, Macintosh y OS/2 Warp. Para el caso de aplicaciones de gran escala, el programa "Designer/2000" les permite a los equipos de desarrolladores, construir modelos para los más sofisticados sistemas.

Referencias:

- Oracle 9i. Kevin Loney.

- http://docs.oracle.com/cd/E19593-01/E22994/gizfh.html

2.2.4.2. Firebird

Firebird deriva del código fuente de Interbase 6.0 de Borland. Es open source y no hay licencias duales. Tanto para uso comercial como para aplicaciones open source, es totalmente libre. La tecnología de Firebird lleva 20 años funcionando, esto hace que sea un producto muy maduro y estable.

Firebird tiene todas las características y la potencia de un RDBMS. Se pueden manejar bases de datos desde unos pocos KB hasta varios Gigabytes con buen rendimiento y casi sin mantenimiento.

Sus características principales son:

· Soporte completo de Procedimientos Almacenados y Triggers

· Las Transacciones son totalmente ACID compliant

· Integridad referencial

· Arquitectura Multi Generacional

· Muy bajo consumo de recursos

· Completo lenguaje para Procedimientos Almacenados y Triggers (PSQL)

· Soporte para funciones externas (UDFs)

· Poca o ninguna necesidad de DBAs especializados

· Prácticamente no necesita configuración - sólo instalar y empezar a usarla.

· Una gran comunidad y muchas páginas donde conseguir buen soporte gratuito

· Opción a usar la versión embebida - de un solo fichero - ideal para crear CDROM con catálogos, versiones de evaluación o monousuario de aplicaciones

· Docenas de herramientas de terceros, incluyendo herramientas visuales de administración, replicación, etc.

· Escritura segura - recuperación rápida sin necesidad de logs de transacciones

· Muchas formas de acceder a tus bases de datos: nativo/API, driver dbExpress, ODBC, OLEDB, .Net provider, driver JDBC nativo de tipo 4, módulo para Python, PHP, Perl, etc.

· Soporte nativo para los principales sistemas operativos, incluyendo Windows, Linux, Solaris, MacOS.

· Backups incrementales

· Disponible para arquitecturas de 64bits

· Completa implementación de cursores en PSQL

El servidor Firebird viene en tres versiones: SuperServer, Classic y Embedded. Actualmente, Classic es la versión recomendada para máquinas con SMP y algunas otras situaciones específicas.

SuperServer comparte su caché para todas las conexiones y usa un hilo de ejecución para cada conexión. Ésta se suele usar en Windows. Classic inicia un proceso de servidor independiente para cada conexión que se haga.

La versión Embedded es una interesante variación del servidor. Es un servidor Firebird con todas sus características, empaquetado en unos pocos ficheros. El servidor no necesita instalación.

Firebird viene con un completo paquete de utilidades de línea de comando que te permiten crear bases de datos, generar estadísticas, ejecutar comandos y scripts SQL, hacer y recuperar copias de seguridad, etc. Si prefieres usar herramientas visuales, hay montones de opciones donde elegir, incluyendo gratuitas.

En Windows, se puede ejecutar Firebird como servicio o como aplicación. El instalador puede crear un icono en el panel de control que permite controlar el servidor (iniciarlo, pararlo, etc.).

Firebird es un SGBD en plataforma cliente/servidor. El servidor acepta peticiones TCP/IP de los clientes, por defecto sobre el puerto 3050 (gds_db). Además puede comunicarse usando IPX. Para que los equipos clientes puedan conectarse al servidor es necesario instalar unas herramientas cliente, generalmente una librería, que en Windows consiste en el fichero gds32.dll/fbclient.dll. Cuando instalamos Firebird en un sistema podemos llegar a una configuración en dos niveles o en n-niveles.

Firebird tiene establecido un sistema de seguridad basado en usuarios y contraseñas. Por defecto, cuando se hace una instalación, se tiene el usuario SYSDBA con contraseña 'masterkey'. Esta información de usuarios se almacena en un fichero de base de datos Firebird: security2.fdb (para versiones de Firebird 2 o superiores).

Firebird aporta este protocolo implementado en forma de una DLL en el caso de WIN que se sitúa en una carpeta accesible del disco duro (normalmente c:\windows\system32) llamada GDS32.DLL, usada para compatibilidad con versiones anteriores o fbclient.dll.

Referencia:

- http://www.firebird.com.mx/descargas/documentos/tema_2-caracteristicas_basicas.pdf

2.2.4.3. PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Algunas de sus principales características son, entre otras:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.....

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (*foreign keys*).
- Disparadores (*triggers*): Un disparador o *trigger* se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:
    - El nombre del disparador o *trigger*
    - El momento en que el disparador debe arrancar
    - El evento del disparador deberá activarse sobre...
    - La tabla donde el disparador se activará
    - La frecuencia de la ejecución
    - La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (*triggers*).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

Referencia:

- http://www.postgresql.org/about/

2.2.5. Pasos en la realización de la base de datos

ELEMENTOS A CONSIDERAR EN EL DISEÑO DE UNA BASE DE DATOS

El Sistema de Información Geográfica (SIG) le permite a la institución realizar una serie de tareas que van desde las operaciones cotidianas hasta la planificación estratégica. Sin embargo, esta flexibilidad intrínsica del sistema sólo se logra cuando este se implementa de forma eficiente. El SIG le permitirá almacenar, analizar y compartir datos con diversos departamentos o unidades en su institución; así como integrar el uso de otras tecnologías como los Sistema de Posicionamiento Global (SPG), el procesamiento digital de imágenes y los sistemas de multimedia. EL diseño e implementación de una base de datos sólida y eficiente es un requisito para lograr un SIG exitoso y saludable. Normalmente la fase de diseño y elaboración de la base de datos geo referenciada puede consumir hasta un 80% del costo total del SIG (ESRI, 1994). La clave para diseñar una buena base de datos es hacer las preguntas correctas a los individuos apropiados de la compañía. Por ejemplo, para iniciar el proceso puede responderse a las siguientes preguntas:

- Cómo puede implementarse el SIG para optimizar las operaciones y procesos que actualmente se realizan? o cambiar la forma en que se logra una meta particular?
- Cuáles datos son de más beneficio para la institución/empresa?
- Cuáles datos pueden compartirse entre departamentos?
- Quién es o deberá ser responsable por su mantenimiento?

Para responder a las preguntas anteriores se requiere de un buen conocimiento de la organización, de sus funciones o servicios que provee y de la tecnología SIG. La implementación de un SIG es similar a cualquier otra actividad que involucre la toma de una decisión. El proceso inicia con la definición o clarificación de las metas de la institución, luego de los procesos a diferentes niveles de detalle mediante los cuales se obtiene la información y finalmente la forma en que las funciones o servicios son implementados. Lo anterior nos lleva a definir:

- Metas
- Manejo de datos
- Requerimientos de uso de datos
- Proceso de colecta de datos

- Implementación de servicios

OBJETIVOS DEL DISEÑO

El diseño es un proceso que involucra tanto a los administradores como a los usuarios y al personal técnico que utilizará el SIG. A lo largo del proceso se definen los objetivos y metas, se estudian las alternativas de diseño y se prepara en un plan de implantación. La meta final del proceso es generar un diseño que asegure flexibilidad, fiabilidad y consistencia en la base de datos. En términos generales, el diseño provee a la compañía con una imagen de donde se encuentran, a donde se dirigen y como llegar al estado deseado. Con forme se avanza en el proceso de diseño se definen los datos requeridos y la estructura de datos geoespaciales que mejor se ajusta a los usos que se darán a la base de datos y al SIG.

Una base de datos bien diseñada debe:

- Cumple con los objetivos para los cuales fue creada y apoya el logro de las metas de la empresa (apoya su desarrollo institucional).
- Contiene sólo los datos necesarios para logra las metas de la empresa. Datos redundantes no son permitidos en una base de datos, excepto cuando forman parte explícita del diseño de la base de datos.
- Los datos están organizados de tal forma que todos los usuarios tienen acceso al mismo set de datos. Esto asegura la integridad de la información que se utiliza en la compañía o empresa.
- El diseño es lo suficientemente flexible como para suplir las necesidades de múltiples usuarios.
- Permite mantener organizados tanto el componente espacial como los atributos de los elementos que utiliza la empresa o institución.

Una base de datos correctamente diseñada ofrece los siguientes beneficios:

- Mayor flexibilidad en la recuperación y análisis de los datos.

- Incrementa la posibilidad de que los usuarios desarrollen aplicaciones utilizando los datos disponibles en la base de datos. Esto se logra cuando la base de datos almacena datos que pueden ser utilizados por diversos usuarios.
- Los costos de captura, almacenamiento y uso es compartido por diversos usuarios o departamentos. Esto racionaliza el uso de recursos en la institución.
- Una base de datos integrada facilita su mantenimiento y por lo tanto asegura la integridad de la información utiliza por los diferentes usuarios (facilita el gestión de las transacciones).
- El diseño es lo suficientemente flexible como para acomodar actualizaciones o modificaciones en el futuro.
- Minimiza los datos redundantes y por lo tanto hace más eficiente la creación, mantenimiento y uso de la base de datos.

El diseño e implementación de la base de datos involucra tres fases o momentos bien definidos en el proceso, a saber:

MODELO CONCEPTUAL: Datos necesarios para lograr los objetivos y metas de la empresa. Los datos serán utilizados para modelar datos geográficos y no geográficos y las relaciones que existen entre ellos.

MODELO LOGICO: Convergencia del modelo de datos geoespaciales a utilizar con los requerimientos de datos por parte de la empresa o compañía.

MODELO FISICO: En esta fase se implementa y ajusta el diseño de la base de datos para optimizar su rendimiento considerando el software, equipo de cómputo y la configuración de la red de la institución.

ELEMENTOS PARA UN DISEÑO EXITOSO

A continuación se brindan algunas sugerencias para lograr un diseño armonioso y con un alto grado de aceptación de la base de datos:

1. Involucre a los usuarios: Recuerde que los usuarios son los que hacen al sistema exitoso. Además, ellos pueden proveer información de los procesos que realizar actualmente, así como sugerir cómo la base de datos podría facilitar su trabajo, su eficiencia y por ende redundar en ahorros económicos para la empresa. Finalmente, cuando el usuario se

involucra en el proceso de diseño desde las fases tempranas se crea una sensación de pertenencia o propiedad del proyecto y por ende los comprometerá a utilizarlo una vez que esté en operación.

2. Focalice su esfuerzo: Aun cuando la meta del proceso de diseño es finalizar con una base de datos que cumpla los objetivos de la empresa; no es necesario crear un diseño detallado de todo el sistema en un solo proyecto. Al inicio, sólo es necesario tener una idea clara del diseño conceptual de la base de datos. Posteriormente se puede ejecutar cada componente por etapas, asignando los recursos necesarios a cada una de las tareas.
3. Forme un equipo de trabajo: Durante el proceso de diseño e implementación de la base de datos se requiere de información muy diversa y del concurso de individuos con múltiples disciplinas. El equipo de trabajo debe estar formado por individuos que conozcan las funciones de la organización que se modela, que posean conocimientos sobe entrevistas, sobre modelado, conocimientos sobre SIG y que además puede comunicarse con los administradores de la compañía.
4. Sea creativo: El proceso de diseño de la base de datos es un excelente momento para identificar y planificar aquellos mecanismos necesarios para optimizar los objetivos y metas de la organización. La tecnología y los medios para capturar, almacenar, visualizar y comunicar información cambian constantemente y por lo tanto el equipo de trabajo debe utilizar toda su creatividad para obtener un producto novedoso y eficiente.
5. Genere productos: Elabore un plan de tal forma que los productos de las diferentes fases estén claramente especificados. Defina quién o quienes son los responsables, establezca fechas de culminación de cada proceso y asigne los recursos requeridos en cada fase. Por ejemplo, un producto de la primera fase de diseño debería ser la matriz de datos y funciones. Los productos son los logros por medio de los cuales se evalúa el avance en el proceso de diseño. Cuanto más pronto se identifiquen los errores o debilidades en el proceso de diseño de la base de datos menor será el costo requerido para sanear la base de datos.
6. No adicione detalle innecesario: Los detalles tienen como objetivo proveer información para que el equipo tome las decisiones correctas en el momento correcto. No trate de definir elementos no requeridos en las primeras fases del diseño. Por ejemplo, no defina coberturas o atributos durante la primera fase del proceso. En esta fase sólo es necesario definir los objetivos que tendrá la base de datos.

7. Documente todos los pasos y decisiones durante el proceso de diseño: EL objetivo básico de documentar cada paso en el proceso de diseño es proveer la información requerida por el equipo de trabajo para tomar las decisiones correctas en el momento apropiado. EL uso de diagramas y tablas facilitará el proceso de comunicación y a la vez permitirá sintetizar en pocos documentos los elementos esenciales del proceso de diseño.
8. Mantenga el foco en sus objetivos y metas: El diseño de la base de datos y de los productos generados deben apuntar a cumplir con los objetivos y metas de la organización. En caso de dudas o confusión, consulte sus notas sobre objetivos y prioridades. Otra alternativa es convocar nuevamente a los usuarios del sistema y jerarcas de la empresa para clarificar dudas y redefinir objetivos y metas.
9. Sea flexible en el diseño: EL diseño esbozado por el equipo de trabajo es una guía y por lo tanto no debe utilizarse como una 'camisa de fuerza'. El diseño debe percibirse como un documento cambiante que debe adecuarse a las necesidades de la organización, a las nuevas tecnologías y a la mejor comprensión del uso de un SIG por parte del personal involucrado en el proyecto.
10. Planifique la implementación del modelo: Recuerde que el modelo es sólo una aproximación a la realidad y que además se requiere de recursos (humanos, tiempo, financieros) para implementarlo. Por ejemplo, debe establecerse prioridades, la forma en que se implementará cada etapa y los datos requeridos por cada aplicación.

Referencia:

- http://www.mapealo.com/Costaricageodigital/Documentos/alfabetizacion/Diseno_bases_datos.pdf

2.3. Lenguaje de Consultas Estructurado (SQL)

Fue originalmente diseñado por IBM, aunque actualmente casi todos los sistemas relacionales, de cualquier proveedor, implementan alguna versión de SQL. Tanto la organización estadounidense de normalización, ANSI, como la internacional, ISO, han desarrollado sucesivas definiciones de un SQL estándar (SQL-92).

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

El SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (LDD), lenguaje de definición de vistas (LDV) y lenguaje de manipulación de datos (LMD). Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas.

Existen dos tipos de sentencias:

- Sentencias de definición de datos que proporcionan las órdenes para definir o modificar esquemas de relación, eliminar relaciones y crear índices. Ej: CREATE y DROP.
- Sentencias de manipulación de datos (Data Manipulation Language) que nos permiten realizar consultas y mantener los datos es decir; insertar, suprimir o modificar los datos. Ej: SELECT, INSERT, UPDATE y DELETE.

Referencia:

- Introducción a las Bases de Datos Relacionales - MENDELZON – ALE - Editorial: PEARSON

2.3.1. Oracle SQL Developer

Oracle SQL Developer es una herramienta gráfica gratuita que mejora la productividad y simplifica las tareas de desarrollo de bases de datos. Con SQL Developer, se puede navegar por los objetos de base de datos, ejecutar sentencias SQL y scripts SQL, editar y depurar PL / SQL, manipular y exportar datos, y ver y crear informes. Puede conectarse a bases de datos Oracle, y puede conectarse a bases de datos de terceros (no-Oracle), ver metadatos y datos y migrar las bases de datos de Oracle.

Las características principales son:

- Compartir fácilmente los detalles de objetos con otros participantes de sus proyectos
- Utilizar informes instantáneos para obtener detalles sobre las sesiones y espacios de tabla de su base de datos, terminar sesiones y cerrar la base de datos
- Aprovechar los servicios copiar y exportar para facilitar el trabajo con múltiples esquemas
- Generar la Documentación de la Base de Datos
- Monitoreo y Administración de sesiones y esquemas con Informes
- Copiar Objetos a un Esquema Nuevo

Debido a que la base de datos con la que se va a trabajar es Oracle, se ha seleccionado esta herramienta como la adecuada para el desarrollo de tareas de base de datos por las características anteriormente detalladas y además debido que al ser desarrollada por la misma empresa posee mayor compatibilidad en cuanto a funcionalidades, mayor estabilidad con la base ya sea en la conexión, como en la velocidad de ejecución de consultas, funciones, procedimientos, etc.

Referencia:

- http://docs.oracle.com/cd/E35137_01/index.htm
- http://www.oracle.com/technetwork/es/articles/sql/o58sql-100505-esa.html

2.3.2. PL/SQL

PL/SQL (Procedural Language/Structured Query Language) es un lenguaje de programación incrustado en Oracle.

PL/SQL soporta todas las consultas, ya que la manipulación de datos que se usa es la misma que en SQL, incluyendo nuevas características

La unidad Básica de PL/SQL es el bloque, todos los programas PL/SQL están compuestos por Bloques, que pueden estar anidados. Normalmente, cada bloque define una unidad lógica de trabajo en el programa, separando así unas tareas de otras.

Los programas o paquetes de PL/SQL se pueden almacenar en la base de datos como otro objeto, y todos los usuarios que estén autorizados tienen acceso a estos paquetes, esta característica lo hace muy útil a la hora de trabajar con diferentes perfiles de usuarios ya que se le da permiso de lectura a determinados usuarios para determinados procesos o paquetes lo que lo hace más seguro ya que se controla por ejemplo la modificación de datos según los permisos de los usuarios de los sectores previamente nombrados en el Sistema Tributario.

```
Declare
/* Sección declarativa -- aquí se construyen las variables PL/SQL, tipos, cursores y subprogramas locales */
v_num_empleados number(2);
begin
/* sección ejecutable -- aquí se incluyen las instrucciones SQL y procedimientos esta es la sección
principal del bloque y la única que es OBLIGATORIA*/
insert into depart values (99, 'Provisional', null);
update emple set dept_no=99
where dept_no=20;
v_num_empleados := SQL%ROWCOUNT;
delete from depart
where dept_no=20;
DBMS_OUPUT.PUT_LINE(v_num_empleados || Empleados ubicados en Provisional);
exception
/* seccion de tratamiento de excepciones -- aqui se incluyen las instrucciones para el tratamiento de las
excepciones*/
WHEN others THEN
ROLLBACK;
RAISE_APPLICATION_ERROR(-20000, 'Error en la aplicacion');
end;
```

Referencia:

- http://www.orafaq.com/tools/allround/developer.htm
- http://www.software-shop.com/in.php?mod=ver_producto&prdID=159

2.3.3. TOAD

Toad Development Suite for Oracle es un conjunto completo de herramientas de desarrollo Oracle utilizadas para el desarrollo y administración de diferentes bases de datos relacionales usando SQL.

Características Principales:

- Automatizar las pruebas de código funcional e identificar la diferencia entre la ejecución de códigos funcionales lograda y la deseada
- Diseñar dependencias y relaciones de PL/SQL, de forma gráfica y dentro de una base de datos a través del Code Road Map
- Automatizar las evaluaciones de rendimiento y de escalabilidad, incluyendo las reproducciones del volumen de trabajo de la base de datos
- Crear programas según estándares de codificación de proyectos predefinidos y aplicar el formato de códigos
- Analizar el código en busca de declaraciones SQL problemáticas y, con un solo clic, encontrar la alternativa SQL más eficiente

Para poder utilizarlo se debe pagar una licencia, a diferencia de Oracle SQL Developer que es gratuito. Además existen seis versiones para diferentes tipos de tareas, a diferencia de Oracle SQL Developer que posee todas las funcionalidades en la misma versión. Demás está decir que hay que pagar para adquirir cada versión, lo cual la convierte en una herramienta con un alto costo.

Referencia:

- http://www.questsoftware.es/toad-for-oracle/
- http://toadworld.com

2.4. Forms

2.4.1. Oracle Forms

Oracle Forms es un producto de software para la creación de pantallas que interactúan con una base de datos Oracle. Cuenta con un IDE que incluye un navegador de la hoja de propiedades de objeto, y editor de código que utiliza PL/SQL o SQL Developer. Fue desarrollado originalmente para ejecutarse en el servidor en sesiones de carácter del modo de terminal. Fue portado a otras plataformas, incluyendo Windows, para funcionar en un servidor de cliente medio ambiente. Las versiones posteriores fueron portadas a Java en el que se ejecuta en un Java EE contenedor y se puede integrar con Java y servicios web.

El objetivo principal de los Forms es crear sistemas de entrada de datos que tienen acceso a una base de datos Oracle.

El motivo de elección de este producto es la alta compatibilidad que posee con la base de datos Oracle ya que es una herramienta del mismo proveedor (Oracle). Además de otorgar una fácil conexión con la base de datos y excelente manejo de bloques de datos.

El único gran punto en contra es una gran falencia a la hora de realizar un buen diseño y el manejo de imágenes en los forms, así como la adaptación a diferentes configuraciones de pantallas.

Referencia:

- http://www.oracle.com/technetwork/developer-tools/forms/new-features-134509.pdf
- http://www.oracle.com/technetwork/developer-tools/forms/forms11gr2newfeatures-497502-en-gb.pdf

2.4.2. Visual Basic .NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic, pero el manejo de las instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas.

La gran mayoría de programadores de VB.NET utilizan el entorno de desarrollo integrado Microsoft Visual Studio en alguna de sus versiones (desde el primer Visual Studio .NET hasta Visual Studio .NET 2010, que es la última versión de Visual Studio para la plataforma .NET), aunque existen otras alternativas, como SharpDevelop (que además es libre).

Al igual que con todos los lenguajes de programación basados en .NET, los programas escritos en VB .NET requieren el framework.NET o Mono para ejecutarse.

Posee más herramientas de opciones visuales que Oracle Forms así como también una alta compatibilidad para desarrollos WEB y alguna aplicación para celulares así como el manejo de imágenes e iconos más vistosos.

Aunque es más complicado de configurar en cuanto a la conexión y no posee la compatibilidad que tiene Oracle Form con la base de datos.

Referencia:

- http://msdn.microsoft.com/es-es/library/bbykd75d(v=vs.100).aspx

2.5. Reports

2.5.1. Oracle Report Developer

Oracle Reports es una herramienta para la elaboración de informes con los datos almacenados en una base de datos Oracle. Oracle Reports consiste en Oracle Reports Developer (un componente de la suite Oracle Developer) y Oracle Reports Server Application Services (un componente del servidor de aplicaciones de Oracle).

Oracle Reports Developer viene con su propio generador de consultas que es bastante útil, pero el cuadro de declaración edición de la consulta es demasiado pequeño para consultas complejas y su fuente no es un tamaño fijo, por lo que no es posible dar formato al código usando sangría. Consultas de gran tamaño son muy difíciles de leer que es más fácil para copiar el texto en un editor de SQL, modificarlo y copiarlo de nuevo. La construcción de la consulta en un editor de SQL (PL/SQL) permite el desarrollo y la optimización de la misma de una manera más fácil, también de esta manera podremos probar el resultado que nos arroja el reporte y de esta manera incluir todos los datos que nos sean necesarios. El informe no se ejecutará más rápido que la consulta que se basa.

Siempre es conveniente ponerle alias de las columnas con el mismo nombre con el prefijo del alias con un identificador de grupo. Si usted tiene una columna "Nombre" de la tabla Padrón, alias como "Pnombre." De esta manera, será más fácil de encontrar en los nombres de las fuentes. Si bien esto no es necesario que un informe simple, es una buena práctica que dará sus frutos en el desarrollo de informes más complejos, con varias consultas.

Referencia:

- http://www.dulcian.com/papers/ODTUG/2004/ODTUG%202004_Balcu2_Oracle%20Reports%20Beginner's%20Guide.html

2.5.2. Crystal Reports

Crystal Reports es una aplicación de inteligencia empresarial utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos (bases de datos).

Varias aplicaciones, como Microsoft Visual Studio, incluyen una versión OEM de Crystal Reports como una herramienta de propósito general de informes/reportes.

Los usuarios al instalar Crystal Reports en un equipo y utilizarlo para seleccionar filas y columnas específicas de una tabla de datos compatibles, pueden organizar los datos en el informe en el formato que necesiten. Una vez que el diseño está completo, el informe se puede guardar como un archivo con extensión rpt. Se puede acceder nuevamente al informe reabriendo el mismo, y poder refrescar los datos. Si la fuente de base de datos se ha actualizado, el informe se refrescará reflejando estas actualizaciones. Características:

- El más completo acceso a datos: Crystal Reports provee más opciones de conectividad a datos que cualquier otra herramienta. Incluye más de 30 drivers para acceso a bases de datos relacionales, fuentes de datos XML y cubos OLAP (Incluyendo sistemas ERP, CRM, Oracle, IBM DB2 y Microsoft SQL Server). También puede acceder a datos personalizados a través de JavaBeans y objetos COM (ADO record sets) para una conectividad más flexible.

- Diseño integral y opciones de formato: La tecnología flexible de diseño de Crystal Reports provee control completo sobre el acceso y la presentación de los datos en los reportes. Posee más de 100 opciones de formato, incluyendo parámetros, mapas, tablas cruzadas, gráficos e hipervínculos, para incrementar el impacto de los reportes. También se incluyen más de 160 formulas, funciones y operadores para un control completo de la presentación de los datos.

- Productividad: El nuevo Repositorio Crystal permite almacenar elementos clave de los reportes, tales como objetos de texto, imágenes, sentencias SQL y funciones personalizadas. Gracias a este repositorio central se pueden reutilizar estos objetos en múltiples reportes. Este almacén centralizado de objetos permite minimizar los esfuerzos de mantenimiento de los reportes y al mismo tiempo ser más productivo en el diseño de reportes nuevos.

Se escogió Oracle Reports ante Crystal Reports por la alta compatibilidad y la fácil configuración y conexión con la base e datos utilizada.

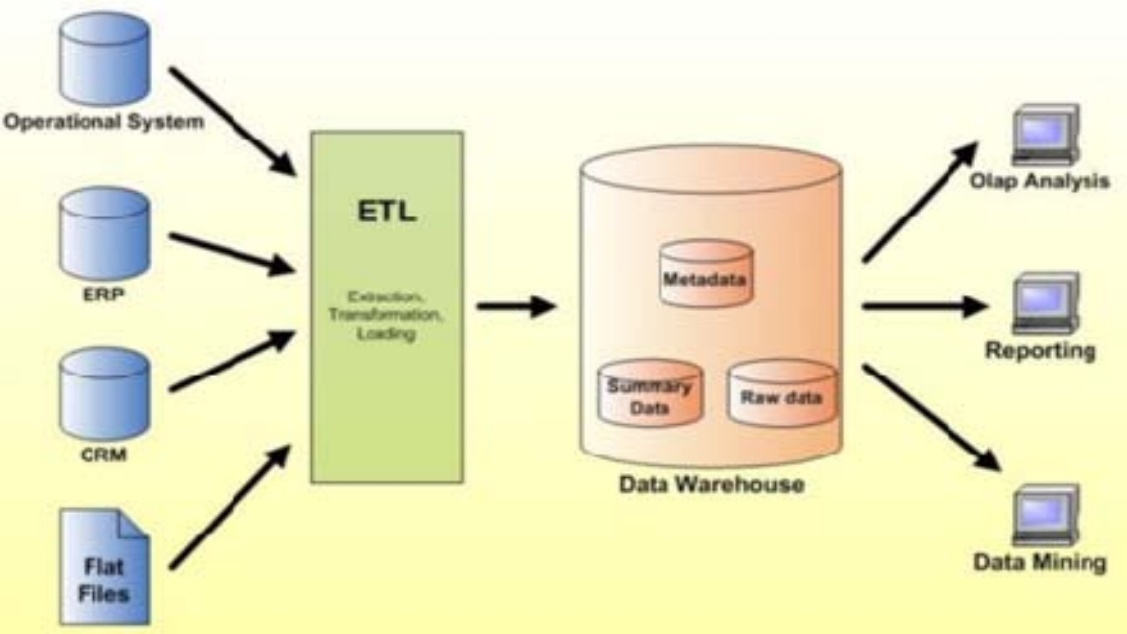
En tanto a las opciones graficas es bastante más completo cristal Reports pero se optó por la compatibilidad y el fácil acceso a los datos.

Referencia:

- http://www.ecured.cu/index.php/Crystal_Reports
- http://www.crystalreportsbook.com/Forum/

2.6. Data Warehouse (Almacén de datos)

Es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata, sobre todo, de un expediente completo de una organización, más allá de la información transaccional y operacional, almacenado en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos (especialmente OLAP, procesamiento analítico en línea). El almacenamiento de los datos no debe usarse con datos de uso actual. Los almacenes de datos contienen a menudo grandes cantidades de información que se subdividen a veces en unidades lógicas más pequeñas dependiendo del subsistema de la entidad del que procedan o para el que sea necesario.



Referencia: http://commons.wikimedia.org/wiki/File:Data_warehouse_overview.JPG

Características Principales:

- Orientado a temas.- Los datos en la base de datos están organizados de manera que todos los elementos de datos relativos al mismo evento u objeto del mundo real queden unidos entre sí.

- Variante en el tiempo.- Los cambios producidos en los datos a lo largo del tiempo quedan registrados para que los informes que se puedan generar reflejen esas variaciones.

- No volátil.- La información no se modifica ni se elimina, una vez almacenado un dato, éste se convierte en información de sólo lectura, y se mantiene para futuras consultas.

- Integrado.- La base de datos contiene los datos de todos los sistemas operacionales de la organización, y dichos datos deben ser consistentes.

Diseño de un almacén de datos

Para construir un Data Warehouse se necesitan herramientas para ayudar a la migración y a la transformación de los datos hacia el almacén. Una vez construido, se requieren medios para manejar grandes volúmenes de información. Se diseña su arquitectura dependiendo de la estructura interna de los datos del almacén y especialmente del tipo de consultas a realizar. Con este criterio los datos deben ser repartidos entre numerosos data marts. Para abordar un proyecto de data warehouse es necesario hacer un estudio de algunos temas generales de la organización o empresa, los cuales se describen a continuación:

- Situación actual de partida.- Cualquier solución propuesta de data warehouse debe estar muy orientada por las necesidades del negocio y debe ser compatible con la arquitectura técnica existente y planeada de la compañía.

- Tipo y características del negocio.- Es indispensable tener el conocimiento exacto sobre el tipo de negocios de la organización y el soporte que representa la información dentro de todo su proceso de toma de decisiones.

- Entorno técnico.- Se debe incluir tanto el aspecto del hardware (mainframes, servidores, redes,...) así como aplicaciones y herramientas.

- Expectativas de los usuarios.- Un proyecto de data warehouse no es únicamente un proyecto tecnológico, es una forma de vida de las organizaciones y como tal, tiene que contar con el apoyo de todos los usuarios y su convencimiento sobre su bondad.

- Etapas de desarrollo.- Con el conocimiento previo, ya se entra en el desarrollo de un modelo conceptual para la construcción del data warehouse.

- Prototipo.- Un prototipo es un esfuerzo designado a simular tanto como sea posible el producto final que será entregado a los usuarios.

- Piloto.- El piloto de un data warehouse es el primero, o cada uno de los primeros resultados generados de forma iterativa que se harán para llegar a la construcción del producto final deseado.

- Prueba del concepto tecnológico.- Es un paso opcional que se puede necesitar para determinar si la arquitectura especificada del data warehouse funcionará finalmente como se espera.

Referencia:

- Ganczarski, Joe. Data Warehouse Implementations: Critical Implementation Factors Study. VDM Verlag, 2009.

2.7. Login

Autenticación o autentificación, en términos de seguridad de redes de datos, se puede considerar uno de los tres pasos fundamentales (AAA). Cada uno de ellos es, de forma ordenada:

- Autenticación: En la seguridad de ordenador, la autenticación es el proceso de intento de verificar la identidad digital del remitente de una comunicación como una petición para conectarse. El remitente siendo autenticado puede ser una persona que usa un ordenador, un ordenador por sí mismo o un programa del ordenador.

- Autorización: Proceso por el cual la red de datos autoriza al usuario identificado a acceder a determinados recursos de la misma.

- Auditoría: Mediante la cual la red o sistemas asociados registran todos y cada uno de los accesos a los recursos que realiza el usuario autorizados o no.

El problema de la autorización a menudo, es idéntico a la de autenticación; muchos protocolos de seguridad extensamente adoptados estándar, regulaciones obligatorias, y hasta estatutos están basados en esta asunción. Sin embargo, el uso más exacto describe la autenticación como el proceso de verificar la identidad de una persona, mientras la autorización es el proceso de verificación que una persona conocida tiene la autoridad para realizar una cierta operación. La autenticación, por lo tanto, debe preceder la autorización.

Métodos de autenticación

- Los métodos de autenticación están en función de lo que utilizan para la verificación y estos se dividen en tres categorías:

- Sistemas basados en algo conocido. Ejemplo, un password (Unix) o passphrase (PGP)

- Sistemas basados en algo poseído. Ejemplo, una tarjeta de identidad, una tarjeta inteligente(smartcard), dispositivo USB tipo epass token, smartcard o dongle criptográfico.

- Sistemas basados en una característica física del usuario o un acto involuntario del mismo: Ejemplo, verificación de voz, de escritura, de huellas, de patrones oculares.

Cualquier sistema de identificación ha de poseer unas determinadas características para ser viable:

- Ha de ser fiable con una probabilidad muy elevada (podemos hablar de tasas de fallo de en los sistemas menos seguros).
- Económicamente factible para la organización (si su precio es superior al valor de lo que se intenta proteger, tenemos un sistema incorrecto).
- Soportar con éxito cierto tipo de ataques.
- Ser aceptable para los usuarios, que serán al fin y al cabo quienes lo utilicen.
- Mecanismo general de autenticación

La mayor parte de los sistemas informáticos y redes mantienen de uno u otro modo una relación de identidades personales (usuarios) asociadas normalmente con un perfil de seguridad, roles y permisos. La autenticación de usuarios permite a estos sistemas asumir con una seguridad razonable que quien se está conectando es quien dice ser para que luego las acciones que se ejecuten en el sistema puedan ser referidas luego a esa identidad y aplicar los mecanismos de autorización y/o auditoría oportunos.

El primer elemento necesario (y suficiente estrictamente hablando) por tanto para la autenticación es la existencia de identidades biunívocamente identificadas con un identificador único. Los identificadores de usuarios pueden tener muchas formas siendo la más común una sucesión de caracteres conocida comúnmente como Login.

El proceso general de autenticación consta de los siguientes pasos:

1. El usuario solicita acceso a un sistema.
2. El sistema solicita al usuario que se autentique.

3. El usuario aporta las credenciales que le identifican y permiten verificar la autenticidad de la identificación.

4. El sistema valida según sus reglas si las credenciales aportadas son suficientes para dar acceso al usuario o no.

Control de acceso

Un ejemplo familiar es el control de acceso. Un sistema informático supuesto para ser utilizado solamente por aquellos autorizados, debe procurar detectar y excluir el desautorizado. El acceso a él por lo tanto es controlado generalmente insistiendo en un procedimiento de la autentificación para establecer con un cierto grado establecido de confianza la identidad del usuario, por lo tanto concediendo esos privilegios como puede ser autorizado a esa identidad. Los ejemplos comunes del control de acceso que implican la autentificación incluyen:

- Retirar de dinero de un cajero automático.
- Control de un computador remoto sin Internet.
- Uso de un sistema Internet banking.

Sin embargo, observar que mucha de la discusión sobre estos asuntos es engañosa porque los términos se utilizan sin la precisión. Parte de esta confusión puede ser debido “al tono de la aplicación de ley” de mucha de la discusión. Ninguna computadora, programa de computadora, o poder del usuario de la computadora “confirman la identidad” de otro partido. No es posible “establece” o “probar” una identidad, cualquiera. Hay ediciones difíciles que están al acecho debajo de qué aparece ser una superficie directa.

Es solamente posible aplicar una o más pruebas que, si están pasadas, se han declarado previamente para ser suficientes proceder. El problema es determinarse qué pruebas son suficientes, y muchos tales son inadecuados. Tienen sido muchos casos de tales pruebas que son spoofed con éxito; tienen por su falta demostrada, ineludible, ser inadecuadas. Mucha gente continúa mirando las pruebas -- y la decisión para mirar éxito en pasar -como aceptable, y para culpar su falta en “sloppiness” o

“incompetencia” de parte alguien. El problema es que la prueba fue supuesta para trabajar en la práctica -- no bajo condiciones ideales de ningún sloppiness o incompetencia-y no. Es la prueba que ha fallado en tales casos. Considerar la caja muy común de un email de la confirmación a el cual deba ser contestado para activar una cuenta en línea de una cierta clase. Puesto que el email se puede arreglar fácilmente para ir a o para venir de direcciones falsas, éste es justo sobre la menos autentificación robusta posible. El éxito en pasar esta prueba significa poco, sin consideración alguna hacia sloppiness o incompetencia.

Autenticación por multifactor

Los factores de la autenticación para los seres humanos se clasifican generalmente en cuatro casos:

· Algo que el usuario es (ejemplo, la huella digital o el patrón retiniano), la secuencia de ADN (hay definiciones clasificadas de cuál es suficiente), el patrón de la voz (otra vez varias definiciones), el reconocimiento de la firma, las señales bio-eléctricas únicas producidas por el cuerpo vivo, u otro identificador biométrico).

· Algo que el usuario tiene (ejemplo, tarjeta de la identificación, símbolo de la seguridad, símbolo del software o teléfono celular)

· Algo que el usuario sabe (ejemplo, una contraseña, una frase o un número de identificación personal (el PIN) del paso).

· Algo que el usuario hace (ejemplo, reconocimiento de voz, firma, o el paso).

· Autenticación mediante dos factores "algo que tengo" la llave + "algo que se" un número de PIN (token criptográfico)

· Autenticación triple factor "algo que tengo" el dispositivo criptográfico + "algo que se" una clave de autenticación tipo PIN (al token criptográfico) + "quién soy" la huella dactilar que me permite autenticarme al dispositivo de forma unívoca.

Una combinación de métodos se utiliza a veces, ejemplo, una tarjeta de banco y un PIN, en este caso se utiliza el término “autenticación del dos-factor”. Históricamente, las huellas digitales se han utilizado como el método más autoritario de autenticación, pero procesos legales recientes en los E.E.U.U. y a otra parte han levantado dudas fundamentales sobre fiabilidad de la huella digital.

Otros métodos biométricos son prometedores (las exploraciones retinianas y de la huella digital son un ejemplo), pero han demostrado ser fácilmente engañados en la práctica. En un contexto de los datos de la computadora, se han desarrollado los métodos criptográficos (véase la autenticación digital de la firma y de la desafío-respuesta) que actualmente no pueden ser engañados si (y solamente si) la clave del autor no se ha comprometido. Que el autor (o cualquier persona con excepción de un atacante) sabe (o no sabe) sobre un compromiso es inaplicable. No se sabe si estos métodos criptográficos basados en la autenticación son probablemente seguros puesto que los progresos matemáticos inesperados pueden hacerlos vulnerables al ataque en futuro. Si eso llega a ocurrir, pondría en cuestión mucho de la autenticación en el pasado. En particular, un contrato digital firmado podría ser cuestionado en cuanto se descubriera un nuevo ataque contra la criptografía subyacente a la firma.

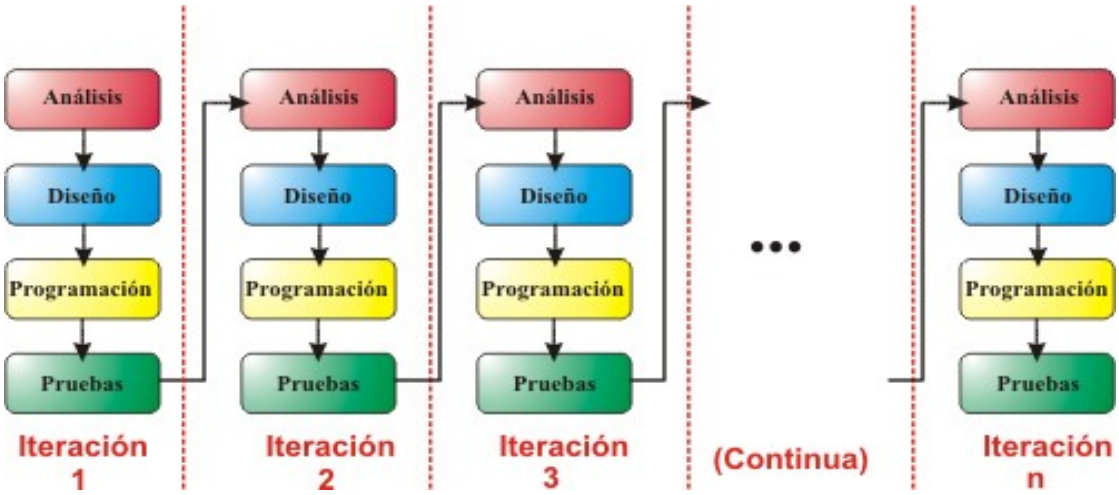
Referencia:

- http://www.rediris.es/cert/doc/unixsec/node14.html

2.8. Desarrollo Iterativo e Incremental

La metodología que va a utilizar para el desarrollo del Sistema Tributario es Desarrollo iterativo e incremental el cual se describe brevemente a continuación:

La idea principal detrás de mejoramiento iterativo es desarrollar un sistema de programas de manera incremental, permitiéndole al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior, incrementando, versiones entregables del sistema. El aprendizaje viene de dos vertientes: el desarrollo del sistema, y su uso (mientras sea posible). Los pasos claves en el proceso son comenzar con una implementación simple de los requerimientos del sistema, e iterativamente mejorar la secuencia evolutiva de versiones hasta que el sistema completo esté implementado. En cada iteración, se realizan cambios en el diseño y se agregan nuevas funcionalidades y capacidades al sistema.



Referencia:http://www.aurumsol.com/espanol/articulos/art1/images/iterativo.jpg

El proceso en sí mismo consiste de:

- Etapa de inicialización: Se crea una versión del sistema. La meta de esta etapa es crear un producto con el que el usuario pueda interactuar, y por ende retroalimentar el proceso. Debe

ofrecer una muestra de los aspectos claves del problema y proveer una solución lo suficientemente simple para ser comprendida e implementada fácilmente.

- Etapa de iteración: Esta etapa involucra el rediseño e implementación de una tarea de la lista de control de proyecto, y el análisis de la versión más reciente del sistema. La meta del diseño e implementación de cualquier iteración es ser simple, directa y modular, para poder soportar el rediseño de la etapa o como una tarea añadida a la lista de control de proyecto. El código puede, en ciertos casos, representar la mayor fuente de documentación del sistema. El análisis de una iteración se basa en la retroalimentación del usuario y en el análisis de las funcionalidades disponibles del programa. Involucra el análisis de la estructura, modularidad, usabilidad, confiabilidad, eficiencia y eficacia (alcanzar las metas).

- Lista de control de proyecto: Contiene un historial de todas las tareas que necesitan ser realizadas. Incluye cosas como nuevas funcionalidades para ser implementadas, y áreas de rediseño de la solución ya existente. Esta lista de control se revisa periódica y constantemente como resultado de la fase de análisis.

Las guías primarias que guían la implementación y el análisis incluyen:

- Cualquier dificultad en el diseño, codificación y prueba de una modificación debería apuntar a la necesidad de rediseñar o recodificar.

- Las modificaciones deben ajustarse fácilmente a los módulos fáciles de encontrar y aislados. Si no es así, entonces se requiere algún grado de rediseño.

- Las modificaciones a las tablas deben ser especialmente fáciles de realizar. Si dicha modificación no ocurre rápidamente, se debe aplicar algo de rediseño.

- Las modificaciones deben ser más fáciles de hacer conforme avanzan las iteraciones. Si no es así, hay un problema primordial usualmente encontrado en un diseño débil o en la proliferación excesiva de parches al sistema.

- Los parches normalmente deben permanecer solo por una o dos iteraciones. Se hacen necesarios para evitar el rediseño durante una fase de implementación.

- La implementación existente debe ser analizada frecuentemente para determinar qué tal se ajusta a las metas del proyecto.
- Las facilidades para analizar el programa deben ser utilizadas cada vez para ayudar en el análisis de implementaciones parciales.
- La opinión del usuario debe ser solicitada y analizada para indicar deficiencias en la implementación referida por él.

Referencia:

- Sistemas de información. Kendall & Kendall
- Patterns of Software Systems Failure And Success. Jones, C. International Thomson Publishing

3. Desarrollo

3.1. Finalidad del Proyecto

La finalidad del proyecto es poder otorgar un mejor manejo de la información a un municipio, proporcionando además de la posibilidad de poder contar con una base de datos mejor organizada y de esta forma poder llevar un control más riguroso sobre la información del sistema tributario. Además cuenta con una interfaz mas amigable para los operadores haciendo mas fácil y rápido el manejo del sistema lo que conlleva a una mejor y mas rápida atención hacia los contribuyentes.

Los puntos fundamentales en el desarrollo del sistema son:

- Instalación de una base de datos estable y segura
- Desarrollo de un sistema de formularios
- Administración y gestión de deudas
- Depuración de los datos de los contribuyentes (Domicilios, Objetos, CUIT, etc.)
- Creación de auditorias

3.2. Estudio de Factibilidad

Para el desarrollo de este proyecto se ha efectuado un modesto estudio de factibilidad, para saber si el esfuerzo y tiempo invertidos darán los resultados esperados.

El estudio fue realizado en base a encuestas cortas a funcionarios del municipio, a operadores del sistema y a los contribuyentes.

Los resultados fueron altamente positivos, ya que actualmente existe un servicio muy limitado en tanto a integridad y calidad de los datos.

Por otro lado, se ha mostrado un prototipo, detallando diferentes funcionalidades nuevas respecto al sistema existente y mejoras de las ya existentes, con una amplia aceptación, sobre todo del lado de las autoridades municipales (funcionarios).

3.3. Inversión Inicial

Una ventaja de este proyecto es que fue iniciado sin necesidad de una inversión económica ya que el sistema funciona por contrato con el municipio que este interesado en aplicarlo, la única inversión seria en el desarrollo de un modelo conceptual que ejemplifique el funcionamiento y justifique su productividad.

El relevamiento del funcionamiento del sistema tributario actual solo requiere de tiempo, o sea que tampoco requiere inversión económica.

En un futuro no muy lejano, suponiendo el éxito de la iniciativa, una inversión económica sería de gran ayuda para la compra de equipos y la contratación de empleados que lleven a cabo el mantenimiento y desarrollo del sistema, aunque en el contrato se especifica que dicha inversión va dentro del presupuesto al municipio.

3.4. Plataforma del Proyecto

Inicialmente se utilizará un servidor que tendrá instalada una base de datos Oracle sobre un sistema operativo Windows Server 2008 R2, luego se instalaran los clientes en la máquinas de los operadores para que vayan probando el sistema mientras va evolucionando una vez desarrollada la versión de prueba, los clientes se instalaran sobre un sistema operativo Windows XP o Windows Seven, dependiendo de las características de hardware de la máquina que se está instalando.

En un futuro será de gran importancia montar un servidor de contingencia el cual debe ser una copia exacta del principal, la instalación de todos los clientes y sus respectivas aplicaciones (Form Developer, Report, PL/SQL, etc.).

3.5. Especificaciones Técnicas

3.5.1. De la Base de Datos

La base de datos elegida entre Oracle, Firebird y PostgreSQL es Oracle 9i. Los fundamentos de esta elección son:

1.-Oracle es el motor de base de datos relacional más usado a nivel mundial.

2.-Puede ejecutarse en todas las plataformas, desde una PC hasta un supercomputador.

3.-Posee gran capacidad de almacenamiento y una velocidad de respuesta muy elevada.

4.-Oracle soporta todas las funciones que se esperan de un servidor "serio": un lenguaje de diseño de bases de datos muy completo (PL/SQL) que permite implementar diseños "activos", con triggers y procedimientos almacenados, con una integridad referencial declarativa bastante potente.

5.-Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.

6.-El software del servidor puede ejecutarse en multitud de sistemas operativos.

7.-Existe incluso una versión personal para Windows 9x, lo cual es un punto a favor para los desarrolladores que se llevan trabajo a casa.

8.-En relación a los Objetos, este sistema ha comenzado a evolucionar en esta dirección, añadiendo tipos de clases, referencias, tablas anidadas, matrices y otras estructuras de datos complejas. Desafortunadamente, la implementación actual de las mismas no ofrece una ventaja clara en eficiencia, como sería de esperar, y sí provocan la incompatibilidad de los diseños que aprovechan las nuevas características con otras bases de datos.

9.-Un aceptable soporte y abundante bibliografía.

3.5.2. Del Servidor

El servidor recomendado es un "HP Proliant Ml370 G5" por su capacidad de procesamiento, almacenamiento y durabilidad de sus componentes, su relación precio/calidad y porque cubre todos los requisitos necesarios para el proyecto.

La elección del servidor es en realidad por preferencia propia con la marca "HP" ya que existen varios equipos de otras empresas con similares precios y prestaciones, por ejemplo: DELL "PowerEdge T420"o IBM "System X3100 M4 Xeon".

Las características del procesador se describen a continuación

Procesador	
Descripción de procesador	(1) Quad-Core Intel® Xeon® Processor E5430 (2.66 GHz, 1333MHz FSB, 80W)
Cantidad de procesadores	1
Velocidad del procesador	2.66 GHz
Tipo de procesador	(1) Quad-Core Intel® Xeon® Processor E5430 (2.66 GHz, 1333MHz FSB, 80W)
"Front Side Bus" del procesador	1333MHz
Memoria	

Memoria estándar	2GB (2 x 1GB); 1 memory card
Características de protección de memoria	Online Spare
Descripción de caché	12MB (2 x 6MB) Level 2 cache
Unidades	
Unidades de disco rígido incluidas	None ship standard
Unidad óptica	48x CD-ROM Drive (diskette optional)
Bahías	
Cantidad de bahías de media altura	2
Conexión de red	
Controlador de red	Embedded NC373i Multifunction Gigabit Sever Adapters
Almacenamiento	
Capacidad de almacenamiento	2.336GB maximum
Controlador de	HP Smart Array P400/256MB Controller (RAID

almacenamiento	0/1/1+0/5)
Conexión de almacenamiento estándar	Hot plug SFF SAS; Hot plug SFF SATA
Características de sistema	
Detalle de administración remota	Integrated Lights-Out 2 with optional upgrade
Software	
Software de administración remota	Integrated Lights-Out 2 with optional upgrade
Chasis	
Chasis del "Form factor"	Tower
Configuración total del "Form factor"	5U
Energía	
Tipo de fuente de alimentación	RPS optional
Expansión	

Ranuras de expansión	8 total - (2) 64-bit/ 100MHz PCI-X and (6) PCI-Express
Dimensiones y peso	
Dimensiones	46.67 x 21.92 x 72.39 cm (Tower)21.92 x 44.45 x 67.31 cm (Rack)
Peso	34.93 kg (Tower)30.84 kg (Rack

Referencia:

- http://www.hp.com/latam/catalogo/co/ml300_smb/sp/458345-001.html

3.6. Cronograma del Desarrollo del Sistema

A continuación se muestra el cronograma estipulado para el desarrollo del sistema tributario.

En la fase 1 se encuentra la tarea de estudio y migración del sistema actual, se realiza durante todo el desarrollo del proyecto, es el ultimo proceso que se ejecuta antes de la implementación. En esta fase se estudian cada uno de los módulos del sistema actual y se diseñan para el nuevo sistema adaptándolo a las funcionalidades nuevas que este pueda llegar a agregar aparte de las existentes.

En la Fase 2 del cronograma están las tareas de entrevistar al experto y los usuarios delegados por cada sector municipal. Así como también la recolección de información. En las entrevistas se tratan temas como:

- Conformidad con el modulo en el sistema actual y que le gustaría agregar o quitar a cada modulo.
- Desarrollo de algún modulo nuevo que no exista en el sistema actual.
- Una vez terminado el modulo del sistema nuevo se le pide a los usuarios que respondan preguntas con respecto a la conformidad del nuevo modulo, en que mejoro con respecto al sistema actual, que no le gusto, que cambios le gustaría que se hagan, que cosas cree que se deben agregar, etc.

En la Fase 3 se encuentra el análisis y el diseño de los formularios, reportes, procesos y codificación.

Una vez estudiados los módulos a desarrollar y analizadas las entrevistas realizadas se procede a la creación y modificación de los módulos analizados en lasa fases anteriores.

Desde la fase 4 se realizan las tareas de pruebas, implementación y capacitación.

Una vez terminado el desarrollo de los módulos se procede a la prueba de estos y en caso de ser satisfactorias las pruebas, se procede a la implementación de los mismos. Las pruebas las realizan, el jefe de proyecto y algunas de las personas entrevistadas en la fase 2.

La fase 5, de documentación, se realiza al finalizar el proyecto.

Al final de proyecto, cuando ya están aprobados todos los módulos por las personas responsables se realiza la documentación de los mismos en la cual se incluye una explicación de cada campo que lo compone y un manual de utilización del mismo.

Debido a que se trabaja con una metodología incremental y evolutiva, todas las fases se ejecutan a lo largo del desarrollo, excepto la fase 5 de documentación que es al final.

La finalización del proyecto se estima en 365 días, es decir, aproximadamente unos 12 meses.

3.7. Metodología para el desarrollo del sistema Tributario.

La metodología que va a utilizar para el desarrollo del sistema tributario es desarrollo iterativo e incremental.

Se eligió esta metodología de desarrollo ya que se cuenta con la posibilidad de tener una comunicación constante con los usuarios, esto hace que a medida que se van desarrollando los diferentes módulos del sistemas, se vayan entregando para que los usuarios lo prueben y así poder obtener una opinión de los mismos y realizar mejoras sobre los módulos ya desarrollados u obtener información sobre los módulos a desarrollar, de esta forma a medida que se va desarrollando el sistema también se van perfeccionando los módulos desarrollados.

A medida que se van entregando los diferentes módulos se realiza una capacitación de cada uno y luego los usuarios lo prueban y se les entrega un formulario en el cual ponen los fallos que estos encuentren o su opinión con respecto a cambios o mejoras que ellos proponen.

En la entrega del próximo modulo, se les piden las encuestas a los usuarios y se estudian y desarrollan las mejoras propuestas por los mismos.

Este proceso se repite por cada módulo hasta lograr la conformidad de la mayoría de los usuarios del mismo.

De esta forma a la hora de la puesta en marcha del sistema, este ya se encuentra casi en su totalidad probado y aprobado por los usuarios y de esta manera no van a surgir mayores inconvenientes de funcionamiento.

3.8. Equipo de Desarrollo

Las personas que componen un grupo o un equipo, como en todos los ámbitos deben cumplir ciertas características y cada uno de ellos dentro del equipo desarrolla un papel distinto.

A continuación se detallará cada componente del equipo dentro del desarrollo y cuál es la función de cada uno:

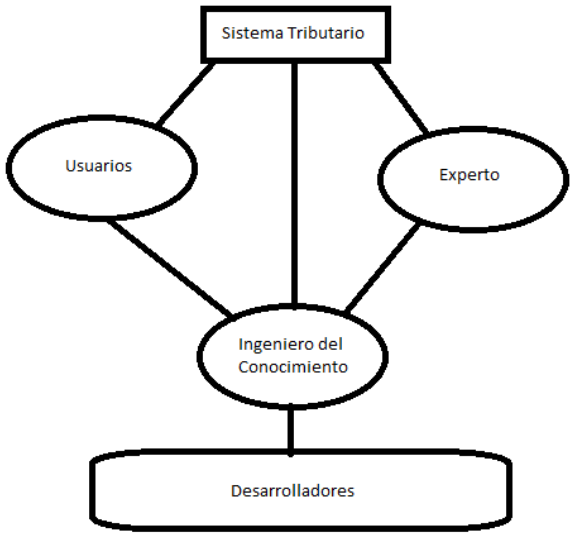
· El experto: La función del experto es la de poner sus conocimientos especializados a disposición del sistema tributario (Gerente de cómputos municipal y sub-secretario de hacienda).

· El ingeniero del conocimiento: Plantea las preguntas al experto, estructura sus conocimientos y los implementa en la base de conocimiento (Gerente de sistemas).

· El desarrollador PL/SQL, Oracle form y Oracle reports, el mismo ingeniero del conocimiento puede cumplir alguna de estas funciones (DBA, programadores y diseñadores).

· El usuario: aporta sus deseos y sus ideas, determinando especialmente los cambios y mejoras con respecto al sistema existente, así como también su opinión con respecto al sistema en desarrollo (Encargados de cada sector municipal por ej: Catastro, Cementerio, Apremios, etc.).

El esquema de representación del equipo de desarrollo es el siguiente:



En el desarrollo del sistema tributario, el ingeniero del conocimiento y el experto trabajan muy unidos. El primer paso consiste en elaborar los problemas que deben ser resueltos por el sistema. Precisamente en la primera fase de un proyecto es de vital importancia determinar correctamente el ámbito estrechamente delimitado de trabajo. Aquí se incluye ya el usuario posterior, o un representante del grupo de usuarios (Los representantes se escogen por sector). Para la aceptación, y para el éxito del sistema, es de vital y suma importancia tener en cuenta los deseos y las ideas del usuario.

Una vez delimitado el dominio, nos pondremos a engrosar nuestro sistema con los conocimientos del experto. El experto debe comprobar constantemente si su conocimiento ha sido transmitido de la forma más conveniente. El ingeniero del conocimiento es responsable de una implementación correcta, pero no de la exactitud del conocimiento. La responsabilidad de esta exactitud recae en el experto.

Para la aceptación y éxito, es de vital y suma importancia tener en cuenta los deseos y las ideas de los usuarios.

A ser posible, el experto deberá tener comprensión para los problemas que depara el procesamiento de datos. Ello facilitará mucho el trabajo. Además, no debe ignorarse nunca al usuario durante el desarrollo, para que al final se disponga de un sistema que le sea de máxima utilidad.

La estricta separación entre usuario, experto e ingeniero del conocimiento no deberá estar siempre presente. Pueden surgir situaciones en las que el experto puede ser también el usuario. Este es el caso, cuando exista un tema muy complejo cuyas relaciones e interacciones deben ser determinadas una y otra vez con un gran consumo de tiempo. De esta forma el experto puede ahorrarse trabajos repetitivos.

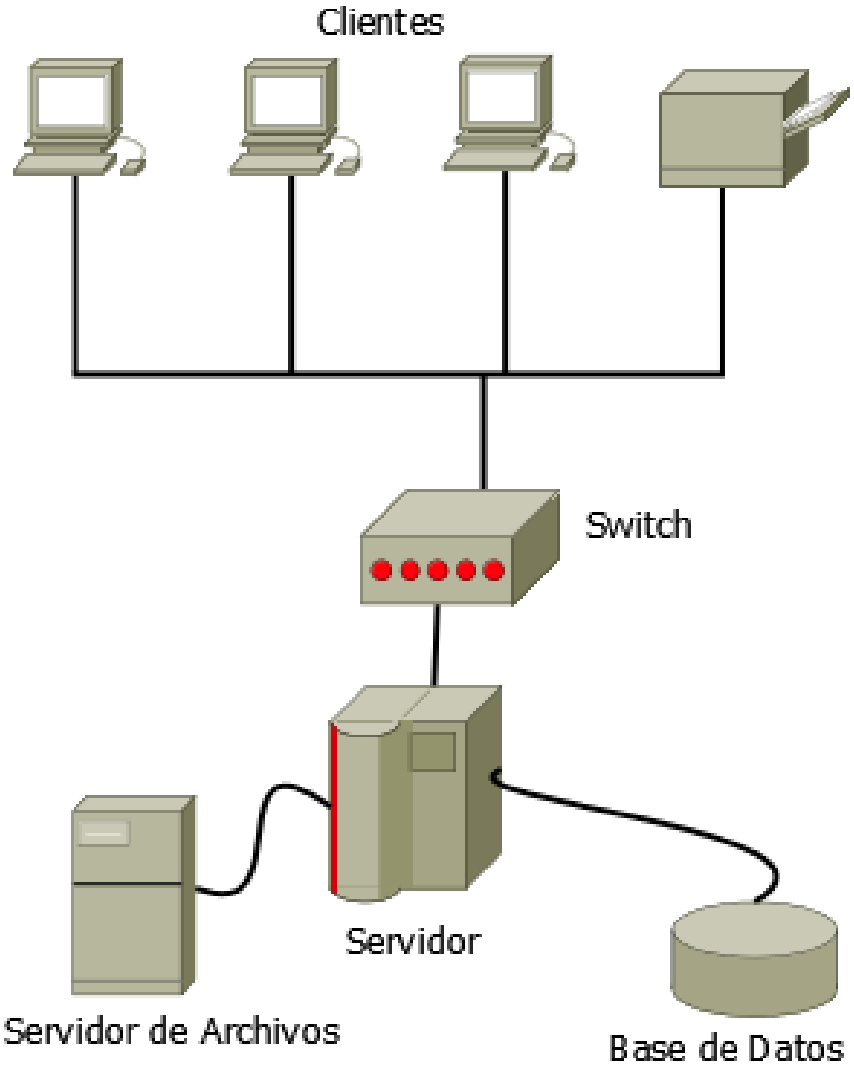
La separación entre experto e ingeniero del conocimiento permanece, por regla general inalterada.

A su vez el ingeniero del conocimiento de saber expresar las necesidades e información otorgada por el experto y los usuarios al equipo de desarrollo para un correcto desarrollo de la aplicación.

Es indispensable que el ingeniero del conocimiento conozca las herramientas de desarrollo, entonces podrá supervisar la una correcta aplicación de los requisitos obtenidos.

3.9. Arquitectura del Sistema Tributario.

En el siguiente diagrama podemos ver que hay un swicht que segmenta las redes, por un lado podemos encontrar la red de usuarios donde se encuentran las impresoras y estaciones de trabajo. Por otro lado vamos a encontrar la red donde se encuentra el servidor que posee alojados la base de datos y el servidor de archivos.



3.10. Pasos En La Creación De La Base De Datos

Puntos a tener en cuanta a la hora de crear una base de datos:

- Decidir nombre único para la instancia, nombre de base de datos, tamaño del bloque Oracle, set de caracteres, número máximo de archivos de datos, y número máximo de archivos de redolog
- Decidir la estructura de almacenamiento físico de la base de datos (File system).
- Copiar y editar el archivo de parámetros (ini.ora) que permita inicializar la instancia Oracle.
- Configurar las variables apropiadas del sistema operativo(ORACLE_SID), otras variables tales como ORACLE_HOME,ORACLE_BASE deben estar previamente definidas
- Crear el archivo de password Invocar el SQLPLUS y conectarse a la base de datos como sysdba.
- Iniciar la instancia en estado NOMOUNT. En este estado se crea una base de datos nueva.
- Crear la base de datos. Con ayuda de los asistentes se puede definir: los esquemas, los tablespace, tablas, índices y secuencias

Creación de la base de datos paso a paso:

Desde el punto de vista físico, una base de datos es, para Oracle, un conjunto de ficheros, a saber:

- Datafiles, ficheros de datos, definidos en la creación de la base de datos.
- Log files, ficheros de log, definidos también en la creación de la base de datos.
- Init.ora, fichero de texto que contiene los parámetros de configuración de la base de datos.
- Control files, ficheros de control, definidos en el init.ora
- Password file, fichero con la password del BDA y los operadores (todos los demás usuarios están definidos en tablas).

Así para crear una base de datos, una vez instalado Oracle, debemos seguir los siguientes pasos:

1) Definir **ORACLE_SID**

```
ORACLE_HOME = E:\Oracle\Product\10.0.0
ORACLE_SID = GESTION
```

2) Crear el fichero **INIT.ORA**

```
C:\>ORACLE_HOME\database\initGESTION.ora

control_files =
(/path/to/control1.ctl,/path/to/control2.ctl,/path/to/control3.ctl)
undo_management = AUTO
undo_tablespace = UNDOTBS1
db_name = GESTION
db_block_size = 8192
sga_max_size = 1073741824
sga_target = 1073741824
```

3) Definir fichero de **passwords**

```
$ORACLE_HOME\bin\orapwd file=ORACLE_HOME\database\pwdGESTION.ora password=oracle
entries=10
```

Podemos generar los pasos 2) y 3) con una sola instrucción:

```
oradim -new -sid GESTION -intpwd  -maxusers 20 -startmode auto
 -pfile E:\Oracle\Product\10.0.0\Database\initGESTION.ora
```

4) Arrancar la **instancia**

```
C:\>sqlplus  / as sysdba
sql> startup nomount
```

5) Crea la **base de datos** con el nombre(o SID) GESTION y el char set WE8ISO8859P1

```
CREATE DATABASE GESTION
LOGFILE 'E:\OraData\GESTION\LOG1GESTION.ORA' SIZE 2M,
```

```
'E:\OraData\GESTION\LOG2GESTION.ORA' SIZE 2M,
'E:\OraData\GESTION\LOG3GESTION.ORA' SIZE 2M,
'E:\OraData\GESTION\LOG4GESTION.ORA' SIZE 2M,
'E:\OraData\GESTION\LOG5GESTION.ORA' SIZE 2M
EXTENT MANAGEMENT LOCAL
MAXDATAFILES 100
DATAFILE 'E:\OraData\GESTION\SYS1GESTION.ORA' SIZE 50 M
DEFAULT TEMPORARY TABLESPACE temp TEMPFILE 'E:\OraData\GESTION\TEMP.ORA' SIZE 50
M
UNDO TABLESPACE undo DATAFILE 'E:\OraData\GESTION\UNDO.ORA' SIZE 50 M
NOARCHIVELOG
CHARACTER SET WE8ISO8859P1;
```

6) Ejecutar sql de creación: catalog.sql y catproc.sql

Sintaxis completa:

```
   CREATE DATABASE nombreDB opciones
```

Donde las opciones:

```
   DATAFILE filespec AUTOEXTEND OFF
   DATAFILE filespec AUTOEXTEND ON [NEXT int K | M] [MAXSIZE int K | M]
   MAXDATAFILES int
   EXTENT MANAGEMENT LOCAL
   DEFAULT TEMPORARY TABLESPACE tablespace [TEMPFILE filespec]
                                        [EXTENT MANAGEMENT LOCAL] [UNIFORM [SIZE
int K | M]]
   UNDO TABLESPACE tablespace [DATAFILE filespec]
   LOGFILE [GROUP int] filespec
   MAXLOGFILES int
   MAXLOGMEMBERS int
   MAXLOGHISTORY int
   MAXINSTANCES int
   ARCHIVELOG | NOARCHIVELOG
   CONTROLFILE REUSE
   CHARACTER SET charset
   NATIONAL CHARACTER SET charset
   SET TIMEZONE = 'time_zone_region'
```

```
SET TIMEZONE = '{+|-} hh:mm'
FORCE LOGGING
USER SYS IDENTIFIED BY password
USER SYSTEM IDENTIFIED BY password
```

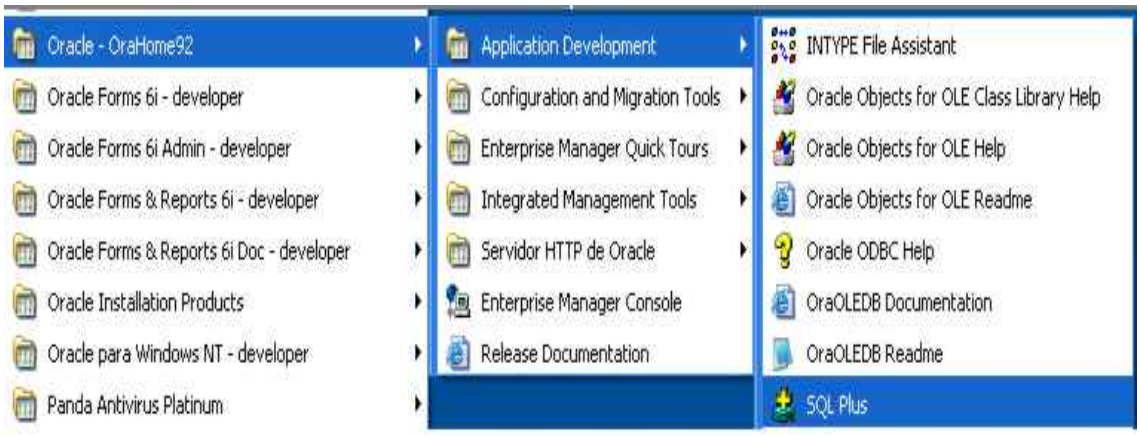
Se puede poner más de un DATAFILE o LOGFILE separando los nombres de fichero con comas DATAFILE filespec1, filespec2, filespec3

Si no se especifican claves, Oracle establece "change_on_install" para SYS y "manager" para SYSTEM.

Despues de crear la base de datos podemos cambiar entre los modos ARCHIVELOG, NOARCHIVELOG con la sentencia ALTER DATABASE.

3.11. Pasos para la creación de Usuarios

Vamos a crear un usuario sin privilegios para poder trabajar sin complicaciones en nuestra base de datos, para ello utilizaremos la herramienta SQL Plus, que la podremos encontrar en el grupo de programas Oracle-OraHome92 en Application Development->SQL Plus

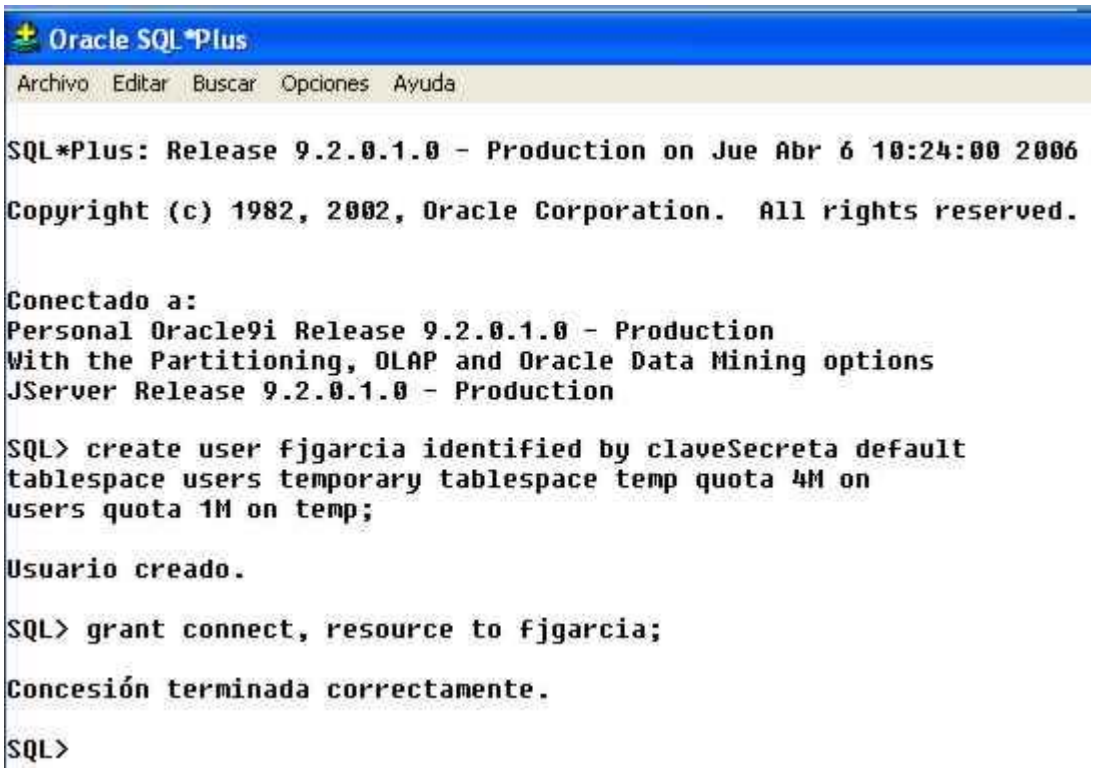


Nos pedirá que nos identifiquemos, entramos como el superusuario system y con la contraseña que dimos durante la configuración. Pulsamos el botón Aceptar.

Una vez identificados en SQL Plus vamos a crear un usuario, se recomienda que la cuenta de usuario tenga el mismo nombre que la que tenéis en la BD de la Escuela y trabajar sobre ella. Esto se puede hacer, con las siguientes sentencias:

```
create user <operador> identified by <password> default
tablespace users temporary tablespace temp quota 4M on
users quota 1M on temp;
grant connect, resource to <x9999999>;
```

Por ejemplo:



Si todo sale bien nos dirá primero que el usuario ha sido creado y que la concesión ha terminado correctamente.

3.12. Creación de la Conexión con la Base de Datos

Para acceder al servidor Oracle lo vamos a hacer añadiendo un nombre de servicio (o alias), usando para ello la aplicación Oracle Net8 Easy Config, que la podremos encontrar en el grupo de programas Oracle para Windows NT-Developer

Los pasos que hay que dar son:

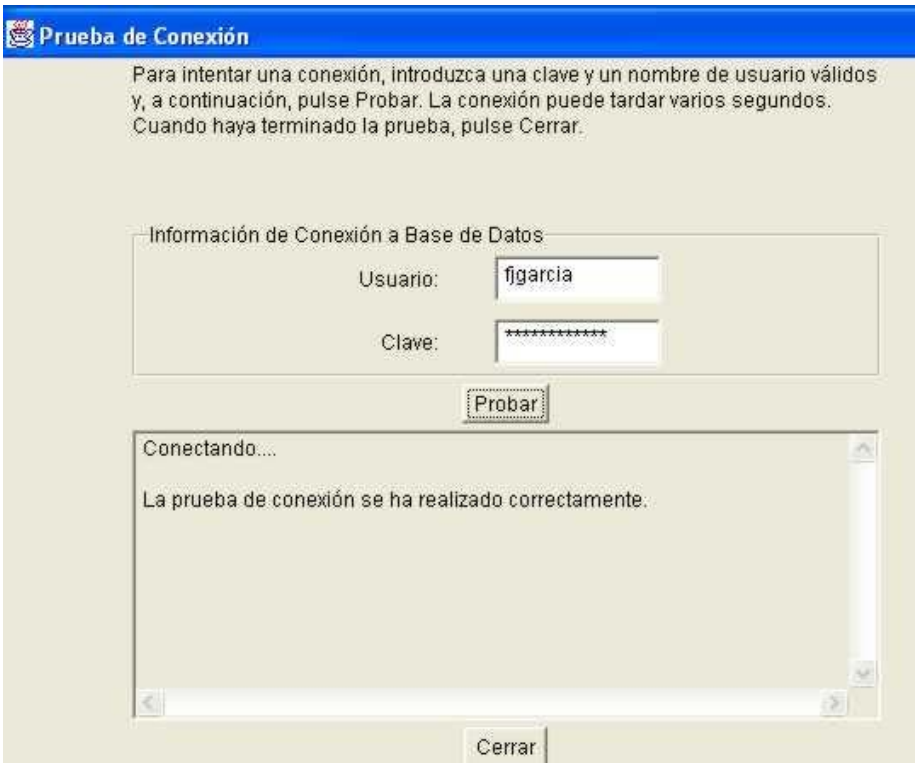
- Añadir un nuevo servicio, por ejemplo, lo puedes llamar UGR y pulsas siguiente

- El protocolo que va a usar es TCP/IP (Protocolo Internet).

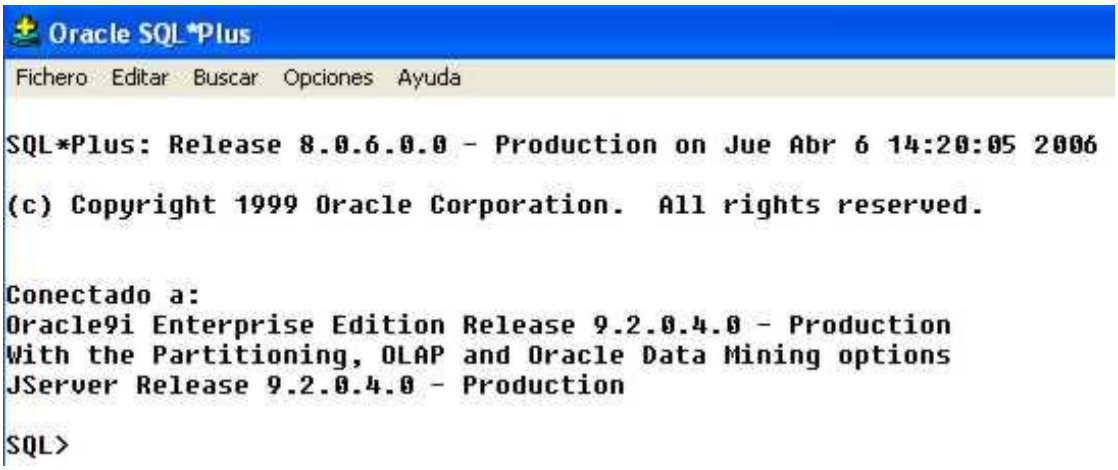
- El nombre del host será oracle0.ugr.es. Sería conveniente comprobar antes que podéis acceder a dicho servidor, para ello desde MS-Dos (símbolo del sistema) poned "ping oracle0.ugr.es", si no responde es probable que se haya caído el servidor. El número de puerto es 1521(el valor por defecto, no cambiar)

- El identificador del sistema (SID de la Base de datos) es PRACTBD

- Ya sólo nos hace falta probar que la conexión funciona perfectamente, para ello usaremos nuestro usuario y contraseña que tenemos en la base de datos de la ETSII



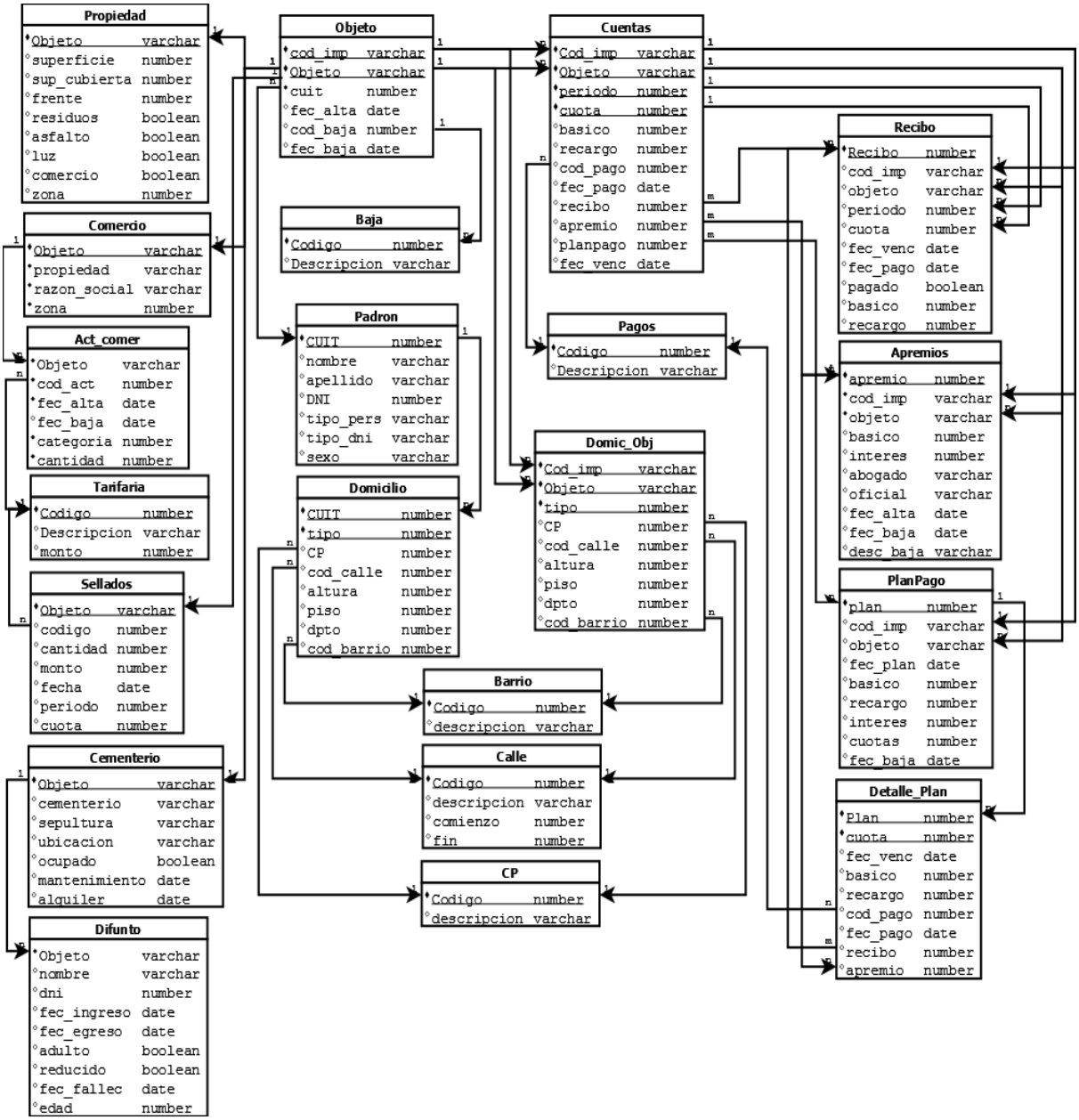
De esta forma para conectarse a la base de datos (con las distintas herramientas de Developer), te hará falta el usuario de la escuela, su password y en la cadena de conexión pones el alias creado (en este ejemplo, UGR). Por ejemplo, si utilizamos la versión de SQL Plus que viene con Developer y que podrás encontrar el grupo de programas Oracle para Windows NT-Developer tendrás que poner dicha cadena de conexión, por ejemplo:



Puede que algún cortafuegos (firewall) nos de problemas al usar el puerto 1521, si ese es el caso se debe permitir la utilización de ese puerto en el cortafuegos.

3.13. Funcionamiento interno del Sistema Tributario

3.13.1. Estructura de la Base de Datos



3.13.1.1. Descripción de las Tablas

Objeto

En la tabla "objeto" se encuentran almacenados la totalidad de objetos cargados en el sistema tributario, identificado cada uno por la combinación de campos "cod_imp" y "objeto". Donde el código de impuesto indica el tipo de impuesto al que pertenece cada objeto

- Cod_imp: Representa el impuesto al que hace referencia el registro actual. (Ejemplo: '01' Propiedad; '04' Comercio; '05' Sellados; '06' Plan de Pagos; '07' Cementerio; '09' Apremios).
- Objeto: Es el número de identificación único con el que se hace referencia a cada objeto. Existe iguales números de objetos para diferentes impuestos.
- CUIT: Es el número de identificación del propietario del objeto con el cual fue almacenado en el sistema.
- Fec_alta: Fecha en la cual paso a formar parte del sistema el objeto y a partir de la cual debe empezar a tributar.
- Cod_baja: Código que representa los motivos de la baja del objeto.
- Fec_baja: Fecha en la que se dio de baja el objeto en el sistema y a partir de la cual deja de tributar.

Padrón

En la tabla "padrón" se encuentran almacenados la totalidad de los contribuyentes (Personas, empresas, sociedades, etc.) cargados en el sistema tributario, identificado cada uno por el campo "CUIT".

- CUIT: Número que se utiliza para identificar de manera única a una persona física o jurídica.
- Nombre: Nombre de la persona o empresa (contribuyente).

- Apellido: Apellido de la persona (contribuyente).
- DNI: Numero de documento o identificador de una persona.
- Tipo_dni: Tipo de documento, puede ser: DNI, Libreta Cívica, Cedula Extranjera, etc.
- Tipo_pers: Se utiliza para saber si es una persona física, jurídica, fallecido, extranjero (hay algunos fallecidos que no poseen DNI y se los guarda con un número especial).

Propiedad

Esta tabla es la que contiene almacenadas las características de todas las propiedades del sistema. El campo identificador es "objeto" que hace referencia al numero de propiedad único en la tabla, el cual hace referencia al de la tabla "objeto".

- Objeto: Es el identificador único de una propiedad, hace referencia al campo objeto de la tabla objeto.
- Superficie: Cantidad de metros cuadrados totales de la parcela de la propiedad.
- Sup_cubierta: Cantidad de metros cuadrados totales cubiertos (construidos) de la propiedad.
- Frente: Metros lineales del frente de la propiedad. Cantidad de metros de la propiedad que abarcan contra la calle.
- Residuos: Almacena un valor lógico (si/no) que hace referencia a que si posee servicios de recolección de residuos o no.
- Asfalto: Almacena un valor lógico (si/no) que hace referencia a que si posee calle asfaltada en el frente.
- Luz: Almacena un valor lógico (si/no) que hace referencia a que si posee servicio de alumbrado público.
- Comercio: Almacena un valor lógico (si/no) que hace referencia a que si posee algún comercio activo ubicado en la propiedad.

- Zona: Se almacena un valor numérico que hace referencia a la zona donde se encuentra la propiedad, la zona es en base al tipo de urbanización en la que se encuentra y la cercanía a la capital del departamento. por ejemplo: cuidad = 1, zona muy rural = 6.

Comercio

Esta tabla es la que contiene almacenadas las características principales de todos los comercios del sistema. El campo identificador es "objeto" que hace referencia al numero de comercio único en la tabla, el cual hace referencia al de la tabla "objeto".

- Objeto: Identificador único del comercio en el sistema, hace referencia a la tabla objeto.
- Propiedad: Numero de la propiedad donde se encuentra instalado físicamente el comercio.
- Razon_social: Nombre de fantasía del comercio.

- Zona: Se almacena un valor numérico que hace referencia a la zona donde se encuentra la propiedad, la zona es en base al tipo de urbanización en la que se encuentra y la cercanía a la capital del departamento. por ejemplo: cuidad = 1, zona muy rural = 6.

Act_comer

Esta tabla es la que contiene almacenadas las actividades que desempeña cada comercio, junto con las características de cada una de ellas. Los campos identificadores son "objeto - cod_acti – fec_baja" que hace referencia al numero de comercio, código de actividad de la tarifaria y fecha de baja de la actividad o la validación que sea nulo en caso de que este activa la actividad.

- Objeto: Numero de comercio al que hacen referencia las actividades.
- Cod_acti: Código o identificador único de las actividades de los comercios. Estos códigos se encuentran previamente establecidos en una ordenanza tarifaria con sus respectivos montos por unidad y categoría.
- Fec_alta: Fecha en la cual comenzó a ejercer la actividad el comercio.

- Fec_baja: Fecha en la que dejo de ejercer la actividad el comercio.
- Categoría: Categoría de la actividad que realiza el comercio, esta puede ser fija o ir desde 1 a 3 dependiendo del tamaño de producción de la actividad.
- Cantidad: Numero de productos o cantidad de producción de una actividad. En el caso en que la actividad no necesite definir una cantidad el valor es 1.

Sellados

Esta tabla es la que contiene almacenados los detalles de los sellados cargados a cada contribuyente, junto con las características de cada uno de ellos. Los campos identificadores son "objeto - código - periodo - cuota" el campo objeto hace referencia al numero de objeto asignado para la cuenta de un contribuyente en sellados, el campo código hace referencia al ítem cargado en el sellado y periodo y cuota hacen referencia al año y en numero de sellados cargados para el periodo de carga del sellado.

- Objeto: Número de identificación del sellado, el cual se utiliza para poder generar e identificar los registros en la tabla cuentas, plan, apremio, etc.
- Código: Identificador único de los ítems de la tarifaria. Estos códigos se encuentran previamente establecidos en una ordenanza tarifaria con sus respectivos montos por unidad.
- Cantidad: Cantidad de unidades solicitadas por cada ítem.
- Fecha: Fecha en la cual fue cargado el sellado.
- Periodo: Es el año al que pertenece el sellado creado.
- Cuota: Numero de registro generado para un objeto de sellado en un mismo periodo.

Tarifaria

Esta tabla es la que contiene almacenados la totalidad de los ítems con sus respectivas descripciones y montos que se cobran en el sistema tributario, de esta tabla se sacan todos los valores de los liquidadores de todos los impuestos, así como también se obtienen los ítems de sellados que existen en el sistema y las actividades de comercio. El campo identificador es “código”.

- Código: Identificador único por cada ítem de la tarifaria.
- Descripción: Descripción del ítem al que hace referencia el código.
- Monto: Valor monetario por unidad del ítem.

Cementerio

Esta tabla es la que contiene almacenadas las características principales de todas las sepulturas del sistema. El campo identificador es “objeto” que hace referencia al numero de sepultura único en el sistema, el cual hace referencia al de la tabla “objeto”.

- Objeto: Identificador único de la sepultura en el sistema.
- Cementerio: Cementerio en el que se encuentra físicamente la sepultura (Ramblón, Palmira, Buen Orden).
- Sepultura: Tipo de construcción de sepultura (nicho, fosa, mausoleo).
- Ubicación: Ubicación en la que se encuentra la sepultura dentro del cementerio.
- Ocupado: Almacena si esta o no ocupada la sepultura.
- Mantenimiento: Fecha de vencimiento del mantenimiento de la sepultura(es un sellado que se cobra de forma anual). Identificador de validez del mantenimiento de la sepultura.
- Alquiler: Fecha de vencimiento del alquiler de la sepultura (vence cada 2 años). Identificador de validez del alquiler de la sepultura.

Difunto

Esta tabla es la que contiene almacenados los datos de todos los difuntos que se encuentran en alguna sepultura en el sistema. El campo identificador es “objeto” que hace referencia al numero de sepultura de la tabla “cementerio”. Cada sepultura puede contener de 1 a 6 difuntos.

- Objeto: Sepultura donde se encuentra el difunto.
- Nombre: Nombre y Apellido del difunto.
- DNI: Número de identificación única del difunto.
- Fec_ingreso: Fecha en la que se colocó al difunto en la sepultura.
- Fec_egreso: Fecha en la que se quitó al difunto de la sepultura.
- Adulto: Indica si es adulto o pequeño el difunto.
- Reducido: Indica si fue reducido el difunto.
- Fec_fallec: Fecha de fallecimiento del difunto.
- Edad: Edad que tenía el difunto a la hora de su muerte.

Domicilio

Esta tabla es la que contiene almacenados los domicilios de todos los contribuyentes del sistema. Los campos identificadores son “CUIT - tipo” que hace referencia al CUIT de cada contribuyente y al tipo de domicilio único en la tabla por contribuyente.

- CUIT: Numero único por cada contribuyente en el sistema.
- Tipo: Indica tipo de domicilio. Ejemplo: Real, postal, etc.
- CP: Código postal.
- Cod_calle: Código de calle, hace referencia a una calle única cargada en el sistema.
- Altura: El número de puerta del domicilio en cuestión.

- Piso: Indica el piso en el que se encuentra el domicilio ene l caso de que posea más de 1.
- Depto: Departamento del domicilio.
- Cod_barrio: Código de barrio, hace referencia a un barrio cargado de manera única en el sistema.

Domic_obj

Esta tabla es la que contiene almacenados los domicilios de todos los objetos del sistema. Los campos identificadores son "cod_imp-objeto - tipo" que hace referencia al numero identificador de cada objeto, al impuesto al que pertenece el objeto y al tipo de domicilio único en la tabla por objeto.

- Cod_imp: Indica al tipo de objeto al que esta haciendo referencia, dependiendo del impuesto al que pertenece. Ejemplo: comercio, propiedad, etc.
- Objeto: Identificador único del objeto, hace referencia a la tabla objeto.
- Tipo: Indica tipo de domicilio. Ejemplo: Real, postal, etc.
- CP: Código postal.
- Cod_calle: Código de calle, hace referencia a una calle única cargada en el sistema.
- Altura: El número de puerta del domicilio en cuestión.
- Piso: Indica el piso en el que se encuentra el domicilio ene l caso de que posea más de 1.
- Depto: Departamento del domicilio.
- Cod_barrio: Código de barrio, hace referencia a un barrio cargado de manera única en el sistema.

Barrio

Esta tabla contiene almacenados la totalidad de barrios del sistema con su respectivo código y descripción. El campo identificador es “código”.

- Código: Identificador único por barrio.
- Descripción: Nombre del barrio incluyendo descripciones adicionales. Ejemplo: número de etapas, etc.

Calle

Esta tabla contiene almacenadas la totalidad de calles del sistema con su respectivo código y descripción, además posee 2 campos que indican la numeración valida para la calle. El campo identificador es “código”.

- Código: Identificador único de cada calle cargada en el sistema.
- Descripción: Nombre de la calle incluyendo descripciones adicionales.
- Comienzo: Numero de puerta mínimo permitido para la calle.
- Fin: Numero de puerta máximo permitido para la calle.

CP

Esta tabla contiene almacenados la totalidad de códigos postales del sistema con su respectivo código y descripción. El campo identificador es “código”.

- Código: identificador único del código postal.
- Descripción: Nombre del código postal incluyendo descripciones adicionales.

Cuentas

Esta tabla contiene almacenadas la totalidad de cuentas generadas del sistema perteneciente a los diferentes objetos de cada impuesto. Los campos identificadores son “objeto - cod_imp - periodo - cuota”.

- Cod_imp: Representa el impuesto al que hace referencia a el registro actual.
- Objeto: Es el número de identificación único con el que se hace referencia a cada objeto.
- Periodo: Es el año al que pertenece la cuenta generada.
- Cuota: Numero de registro generado para una cuenta en un determinado periodo.
- Básico: Monto neto generado para la cuenta.
- Recargo: Monto generado por un pago de cuenta realizado posteriormente a la fecha de vencimiento "fec_venc".
- Fec_venc: Fecha limite de pago para la cual se puede pagar sin ningún tipo de recargos.
- Cod_pago: Código que se utiliza para indicar el estado en el que se encuentra un registro en la cuenta. Ejemplo: 1: Impago - 2: Pago - 11: Plan de pago - 12: Apremio - 9: Dado de baja.
- Fec_pago: Fecha en la que fue pagado el registro de la cuenta.
- Recibo: Numero de recibo con el cual fue pagado un registro de la cuenta.
- Apremio: Numero de apremio en el cual se encuentra incluido el registro de la cuenta.
- Planpago: Numero de plan de pagos en el cual se encuentra incluido el registro de la cuenta.

Recibos

Esta tabla contiene almacenados la totalidad de los recibos generados en el sistema perteneciente a las diferentes cuentas de los objetos. Los campos identificadores son "recibo - objeto - cod_imp - periodo - cuota".

- Recibo: Numero que se utiliza para identificar un recibo impreso desde el sistema.
- Cod_imp: Representa el impuesto al que hace referencia el registro de la cuenta incluida en el recibo.
- Objeto: Es el número de identificación único con el que se hace referencia a cada objeto.

- Periodo: Es el año al que pertenece la cuenta incluida en el recibo.
- Cuota: Numero de registro generado para una cuenta en un determinado periodo incluido en el recibo.
- Básico: Monto neto generado para la cuenta incluida en el recibo.
- Recargo: Monto generado por un pago de cuenta realizado posteriormente a la fecha de vencimiento "fec_venc" calculado hasta la fecha de pago.
- Fec_venc: Fecha limite de pago para la cual se puede pagar sin ningún tipo de recargos.
- Fec_pago: Fecha en la que fue pagado el recibo.
- Pagado: Indicador de estado de pago del recibo.

Apremio

Esta tabla contiene almacenados la totalidad de los apremios generados en el sistema para las cuentas de los objetos. El campo identificador es "apremio".

- Apremio: Numero que se utiliza para identificar de manera única cada apremio.
- Cod_imp: Representa el impuesto al que hace referencia la cuenta incluida en el apremio.
- Objeto: Es el número de identificación que hace referencia al objeto para el cual fue generado el apremio.
- Básico: Sumatoria de los básicos de los registros de la cuenta incluidos en el apremio.
- Interés: Sumatoria de el interés de todos los registros de la cuenta calculados a la fecha de la generación del apremio.
- Abogado: Nombre del Abogado al que le fue asignado el apremio.
- Oficial: Nombre del oficial de justicia al que fue asignado el apremio.
- Fec_alta: Fecha de generación del apremio.

- Fec_baja: Fecha de baja del apremio.
- Desc_baja: Descripción del motivo de la baja del apremio.

PlanPago

Esta tabla contiene almacenados la totalidad de los planes de pago generados en el sistema para las cuentas de los objetos junto con las características principales de cada plan. El campo identificador es "plan".

- Plan: Numero que se utiliza para identificar de manera única cada plan.
- Cod_imp: Representa el impuesto al que hace referencia la cuenta incluida en el plan.
- Objeto: Es el número de identificación que hace referencia al objeto para el cual fue generado el plan.
- Fec_plan: Fecha de generación del plan de pagos.
- Básico: Sumatoria de los básicos de los registros de la cuenta incluidos en el plan.
- Recargo: Sumatoria de el interés de todos los registros de la cuenta calculados a la fecha de la generación del plan.
- Interés: Interés de financiación generado por la creación del plan, depende de la cantidad de cuotas con la que se cree el plan.
- Cuotas: Cantidad de cuotas con las que fue generado el plan.
- Fec_baja: Fecha de baja del plan.

Detalle_plan

Esta tabla contiene almacenado un detalle con la totalidad de los registros de la cuenta de los planes de pago. Los campos identificadores son "plan – cuota - apremio".

- Plan: Numero que se utiliza para identificar de manera única cada plan.

- Cuota: numero que identifica de manera única a cada registro de la cuenta de un determinado plan.
- Fec_venc: Fecha de vencimiento de la cuota del plan.
- Básico: Monto neto generado para la cuota incluida en el plan.
- Recargo: Monto generado por un pago de una determinada cuota realizado posteriormente a la fecha de vencimiento “fec_venc”.
- Cod_pago: Código que se utiliza para indicar el estado en el que se encuentra un registro en la cuenta del plan. Ejemplo: 1: Impago - 2: Pago - 9: Dado de baja.
- Fec_pago: Fecha en la que fue pagado el registro de la cuenta del plan.
- Recibo: Numero de recibo con el cual fue pagado un registro de la cuenta.
- Apremio: Numero de apremio incluido en el plan.

Baja

Esta tabla contiene almacenados la totalidad de motivos de baja del sistema con su respectivo código y descripción. El campo identificador es “código”.

- Código: identificador único de bajas.
- Descripción: detalle del motivo de una baja incluyendo descripciones adicionales.

Pagos

Esta tabla contiene almacenados la totalidad de estados de pagos del sistema con su respectivo código y descripción. El campo identificador es “código”.

- Código: identificador único de pagos.
- Descripción: descripción del estado de pago incluyendo descripciones adicionales.

3.13.2. Procedimientos y Funciones

El hecho de desarrollar procedimiento almacenado en la base de datos tiene varias ventajas, las cuales se detallan a continuación:

- Seguridad: debido a que la base de datos posee mayor seguridad que la carpeta donde se encuentran los archivos publicados.

- Integridad: Muchos formularios a veces utilizan un mismo procedimiento, de esta manera solucionamos este inconveniente ya que solo desarrollamos una vez el procedimiento y luego los forms solamente llaman al mismo que se encuentra en la base, de esta manera también solucionamos trabajos repetitivos a la hora de hacer alguna modificación en el procedimiento y evitamos errores de inconsistencia entre los mismos.

- Modificación: Como ya se nombro en el punto anterior esta metodología de trabajo también facilita la modificación de los procedimientos, ya que solo se modifica uno y los cambios repercuten en todos los forms que lo utilicen.

- Velocidad de procesamiento: Es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes. Los tiempos de ejecución de los mismos son más rápidos lo cual aumenta el tiempo de respuesta.

Usos típicos para procedimientos almacenados incluyen la validación de datos siendo integrados a la estructura de base de datos (los procedimientos almacenados utilizados para este propósito a menudo son llamados disparadores; triggers en inglés), o encapsular un proceso grande y complejo. El último ejemplo generalmente ejecutará más rápido como un procedimiento almacenado que de haber sido implementado como, por ejemplo, un programa corriendo en el sistema cliente y comunicándose con la base de datos mediante el envío de consultas SQL y recibiendo sus resultados.

Los procedimientos pueden ser ventajosos: Cuando una base de datos es manipulada desde muchos programas externos. Al incluir la lógica de la aplicación en la base de datos utilizando procedimientos almacenados, la necesidad de embeber la misma lógica en todos los programas que acceden a los datos es reducida. Esto puede simplificar la creación y, particularmente, el mantenimiento de los programas involucrados.

Llamada a procedimientos desde Form Builder

Una característica muy importante que posee Oracle Form Builder es la facilidad con la que podemos llamar a procedimientos y funciones que se encuentran alojados en el servidor (Base de datos).

Parea realizar la llamada a un procedimiento de la base solamente debemos ingresar a la hora de llamarlo primero el nombre “ ” luego el nombre del procedimiento o función y listo, con eso basta, el Form llamara al procedimiento que se ejecutara en la Base de Datos.

A continuación se detalla un ejemplo de una función:

1) Llamada a la función desde el Form:

```
Domicilio := Tributario.domicilio(objeto, cod_imp, tipo, descripción);
```

2) Declaración de la función en la Base de Datos:

```
create or replace function tributario.domicilio(vobjeto in varchar2,
                                                vcod_imp in varchar2,
                                                vtipo in number) return varchar2
;
```

3) Cuerpo de la función

```
create or replace function tributario.domicilio(vobjeto in varchar2,
                                                vcod_imp in varchar2,
                                                vtipo in number) return varchar2
vdescripcion varchar2(100);
begin
  select d.descripcion || ' - ' || d.puerta
         into vdescripcion
     from tributario.domicilio d
       where d.cod_imp = vcod_imp
       and d.objeto = vobjeto
       and d.tipo = vtipo;

return(vdescripcion);
exception
  when others then
    return(sqlerrm);
end;
```

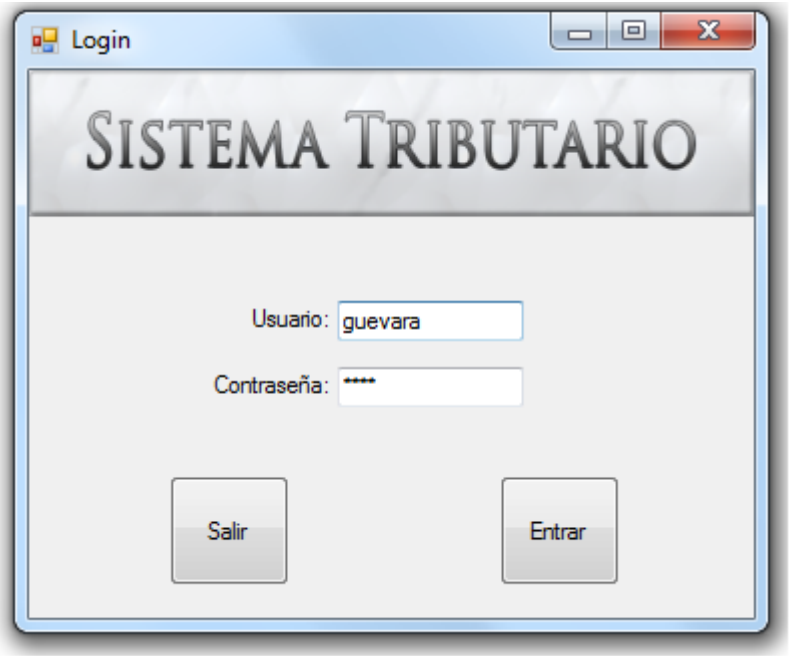
3.13.3. Formularios más importantes del Sistema Tributario

3.13.3.1. Login de Usuarios

En nuestra aplicación se quiere resaltar la importancia que tiene el inicio de sesión de los usuarios y por lo tanto la autenticación de los mismos; ya que todo usuario que accede al sistema puede realizar transacciones de cobranzas a contribuyentes; por eso se puso principal hincapié en el establecimiento de las contraseñas y en la seguridad de las mismas. Cada usuario esta registrado en la base en la tabla de usuarios, donde se almacena junto con cada uno la contraseña correspondiente ha dicho usuario.

En cuanto a los niveles de acceso se trata de restringir los accesos por formularios, basándose en roles otorgados a los diferentes usuarios, y al momento de abrir cada formulario, verificar si posee esos roles, también se pueden deshabilitar opciones del menú directamente por medio de los roles.

En el siguiente form se pueden observar dos campos, los cuales son: usuario, en el cual se debe ingresar el nombre de usuario otorgado por el administrador del sistema; y contraseña, en el cual el usuario ingresa su clave de ingreso la cual fue elegida por él.



3.13.3.2. Contribuyentes

El formulario de contribuyentes nos permite consultar, cargar, modificar y eliminar los sujetos cargados en el sistema.

En este formulario podemos ver una tabla superior en la que podemos ver la totalidad de los contribuyentes cargados en el sistema con sus respectivos datos principales que caracterizan a cada sujeto. Estos son: CUIT, Nombre, Apellido, tipo DNI, DNI, sexo, tipo persona.

En la parte inferior podemos ver los domicilios real y postal de cada contribuyente, el cual se va actualizando a medida que seleccionamos un registro en la tabla superior.

3.13.3.3. Propiedad

En este formulario se pueden consultar todas las propiedades del sistema ingresando el número de propiedad en el campo “propiedad”. También se puede dar de alta una propiedad nueva, modificar una existente o eliminar.

En la parte superior se pueden ver los datos principales de la propiedad, como CUIT y nombre del propietario, fechas de inicio y fecha de baja.

En el centro se encuentran las características de la propiedad que son las que se tienen en cuenta a la hora de generar las cuentas corrientes. Estas son: residuos, luz, asfalto, comercio, superficie, superficie cubierta, frente, zona.

En la parte inferior se encuentran tres pestañas las cuales contienen cada una la siguiente información:

1- La primera pestaña contiene los datos de los domicilios postal y real de la propiedad.

2- La segunda pestaña contiene información de los domicilios postal y real del propietario del inmueble, el cual puede diferir con el real y postal de la propiedad, por ejemplo en el caso en que un propietario posea más de 1 propiedad.

3- La tercera pestaña contiene un cuadro donde se encuentra la totalidad de los registros que componen de la cuenta corriente de la propiedad desde que se dio de alta en el sistema hasta la actualidad o su respectiva fecha de baja.

3.13.3.4. Comercio

En este formulario se pueden consultar todos los comercios del sistema ingresando el número de comercio en el campo “comercio”. También se puede dar de alta un comercio nuevo, modificar uno existente o eliminar.

En la parte superior se pueden ver los datos principales del comercio, como CUIT y nombre del propietario, fecha de alta y fecha de baja.

En el centro se encuentran los datos adicionales del comercio, estos son el nombre del comercio, propiedad en la que se encuentra físicamente el comercio y la zona tributaria del mismo.

En la parte inferior se encuentran cuatro pestañas las cuales contienen cada una la siguiente información:

1- La primera pestaña contiene los datos de los domicilios postal y real del comercio.

2- La segunda pestaña contiene información de los domicilios postal y real del propietario del comercio, el cual puede diferir del domicilio del comercio ya que este puede alquilar un local y tener más de un comercio, al momento de enviar boletos por correo, se tiene en cuenta el domicilio postal del propietario.

3- La tercera pestaña contiene un cuadro donde se encuentra la totalidad de los registros que componen de la cuenta corriente del comercio desde que se dio de alta el comercio hasta la actualidad o su respectiva fecha de baja.

4- La cuarta pestaña contiene un cuadro donde se encuentra la totalidad de las actividades que se desempeñan o se han desarrollado en el comercio. Para el caso de las actividades que ya no desempeña el comercio esta cargada la fecha de baja de la actividad.

De esta tabla se sacan los datos para la liquidación de la cuenta corriente de cada comercio. Se suman los montos de las actividades activas (sin fecha de baja) y la suma total da como resultado el importe a pagar por cada comercio.

3.13.3.5. Sellados

En este formulario se pueden consultar todos los sellados cargados a los contribuyentes del sistema ingresando el número de CUIT de los mismos en el campo “CUIT”. Para este caso el campo objeto en realidad no se utiliza para búsqueda, solo hace referencia a un grupo de sellados que pertenecen a un contribuyente determinado, pero el objeto no existe físicamente. El campo clave para este caso particular es el CUIT, o sea que va a haber un CUIT por objeto.

En la parte superior se pueden ver los datos principales del sellado, como propietario, fecha de alta y fecha de baja.

En la parte inferior se encuentran cuatro pestañas las cuales contienen cada una la siguiente información:

1- La primera pestaña contiene información de los domicilios postal y real del propietario del sellado.

2- La segunda pestaña contiene información detallada de los sellados cargados para el contribuyente, como la descripción del sellado, cantidad, código de la tarifaria al que hace referencia, fecha de generación, monto, etc.

3- La tercera pestaña contiene un cuadro donde se encuentra la totalidad de los registros que componen de la cuenta corriente de los sellados creados para un determinado contribuyente. A medida que se van generando sellados se van agregando en dicha cuenta corriente, para este caso no se genera ninguna cuenta fija, solo se generan sellados cuando el contribuyente los solicite.

3.13.3.6. Cementerio

En este formulario se pueden consultar todas las sepulturas del sistema ingresando el número en el campo “sepultura”. También se puede dar de alta una sepultura nueva, modificar una existente o eliminarla.

En la parte superior se pueden ver los datos principales de la sepultura, como CUIT y nombre del propietario, fecha de alta y fecha de baja.

En el centro se encuentran las características principales de la sepultura, estas son el nombre del cementerio donde se encuentra la misma, debido a que existen tres cementerios (Ramblón, Palmira, Buen Orden). La ubicación de la misma en el cementerio. Fechas de renovación de alquiler y mantenimiento anual. Y por ultimo una marca que indica si la sepultura esta ocupada.

En la parte inferior se encuentran cuatro pestañas las cuales contienen cada una la siguiente información:

1- La primera pestaña contiene los datos del domicilio postal de la sepultura y una tabla con los datos de los difuntos que se encuentren alojados en la misma.

2- La segunda pestaña contiene información de los domicilios postal y real del propietario de la sepultura.

3- La tercera pestaña contiene un cuadro donde se encuentra la totalidad de los registros que componen de la cuenta corriente de las sepulturas.

3.13.3.7. Apremios

En este formulario se pueden consultar todos los apremios del sistema ingresando el número en el campo “apremio”. También se puede dar de alta una boleta de deuda nueva, modificar una existente o eliminarla.

En la parte superior se pueden ver los datos principales del apremio, como propietario del objeto al que hace referencia, fechas de alta y baja.

En el centro se encuentran las características principales del apremio, estas son el objeto e impuesto al que hace referencia el apremio, abogado y oficial de justicia que están llevando a cabo la causa, y por ultimo los montos de las obligaciones que fueron incluidas a la hora de generar el apremio.

En la parte inferior se encuentran cuatro pestañas las cuales contienen cada una la siguiente información:

1- La primera pestaña contiene información de los domicilios postal y real del propietario del apremio.

2- La segunda pestaña contiene información detallada de los registros de la cuenta corriente del objeto que se incluyo en la creación del apremio.

3.13.3.8. Planes de Pago

En este formulario se puede crear o simular un plan de pagos a un determinado objeto, de un determinado impuesto a partir de la deuda que posea al día de la generación del plan, para generar un plan se debe escoger el impuesto del objeto al que se desea generarle dicho plan, luego se ingresa el numero de objeto en el campo objeto, excepto para el caso de sellados en el que se ingresa el CUIT.

En la parte inferior hay una tabla que muestra todos los registros adeudados para el objeto consultado, además posee un campo "X" que es de selección para que el operador pueda seleccionar los registros que desea incluir en el plan a generar.

3.13.3.9. Liquidación

En este formulario se puede consultar la deuda que posee un determinado objeto correspondiente a un Impuesto previamente seleccionado.

En la parte superior se puede ver el propietario del objeto, y la fecha sobre la cual se desea calcular la deuda (Normalmente es el mismo día que se realiza la consulta).

En la parte inferior se puede ver un cuadro que contiene todos los registros de la cuenta corriente que se encuentran impagos y al final un campo de selección para que se pueda seleccionar las cuotas que se vana imprimir para el pago.

3.13.3.10. Cobros

En este formulario se cargan los cobros realizados por las cajas ingresando el número de recibo en el campo “recibo”. Una vez ingresado el número de recibo se pueden observar las características del mismo, por ejemplo objeto, obligaciones incluidas, etc.

En la parte superior se pueden ver las características principales que conforman el recibo, estas son, objeto, el impuesto del objeto, fecha de pago y el CUIT y nombre del propietario.

En la parte inferior se puede ver un cuadro que contiene todos los registros de la cuenta corriente que se encuentran incluidos en el recibo, los cuales conforman el detalle y monto a pagar de la cuenta corriente del objeto en cuestión.

3.13.3.11. Cajas

En este formulario se cargan los cobros realizados por las cajas para una fecha determinada, ingresando la fecha de cobro de la caja

En la parte solamente se encuentra un campo “fecha” en el cual se ingresa la fecha de carga de la caja.

En la parte inferior se puede ver un cuadro que contiene un detalle de todos los recibos cobrados por la caja para la fecha ingresada en la parte superior.

3.13.4. Data Warehouse

Para la realización de la extracción de información a ser almacenada del DW se ha tenido en cuenta que los usuarios que lo van a utilizar son únicamente las autoridades de mas alto rango en el municipio (intendente, director de rentas, secretarios) y el tipo de datos que es de su interés a través del tiempo. Luego de una serie de entrevistas se llego a la conclusión que solo se va a extraer información de la recaudación diaria y por objeto a lo largo del tiempo.

Teniendo en cuenta este resultado que desarrollara la extracción de las tablas “objeto”, “cuentas”, “recibo”.

Una vez realizada la extracción se procede a la creación de las tablas que almacenaran la información obtenida de dicha extracción y de esta manera tener una foto de como se encontraba la recaudación y cuentas en un determinado momento.

Las extracciones se realizaran mensualmente dándoles la posibilidad a las autoridades de tener un control mensual del estado de cuentas y cobros del sistema.

El DW contara con dos módulos:

- Un modulo de cuentas, el cual contendrá información del estado de las cuentas en el momento de la extracción.
- Un modulo de cobros, el cual contendrá información de los cobros realizados hasta el momento de la extracción.

4. CONCLUSIONES Y RESULTADOS

4.1. Conclusiones del Sistema Tributario

La gestión de la información, busca optimizar la integración de los datos y de las aplicaciones con el objeto de brindar interoperabilidad en distintas unidades pertenecientes a una misma organización, en este caso una Municipalidad de la cuidad de Gral. San Martín.

A fin de brindar una solución fiable y acorde a las posibilidades que generan los sistemas de información en los entes estatales, se analizaron los procesos; lográndose generar interfaces de comunicación sencillas, homólogas y generalizadas; generar pasarelas entre sistemas y bases de datos; proporcionar información completa y establecer medios de acceso a través de los servicios de Internet, de forma de garantizar la actualización permanente del sistema y proporcionar accesos a los usuarios.

Como línea futura de trabajo, se puede integrar a dicho sistema de información desarrollado, una integración con las dependencias del mismo en cuanto a sistemas y bases de datos se tratase; con el objetivo de tener todo centralizado; lo que llevaría a mejoras en cuanto al rendimiento, control, cobro de impuestos, etc.

En cuanto a esto la posibilidad de continuación se ve proyectada para un futuro cercano, de esta manera se podría, una vez realizado el rediseño del mismo, incorporar otras opciones de plataformas o lenguajes dentro de la aplicación.

4.2. Resultados de la Aplicación

Los resultados obtenidos durante todo el proceso de diseño y programación de la aplicación se consideran muy buenos en cuanto a que es posible continuar lo realizado hasta el momento para que en un futuro no muy lejano pueda llegar a ser utilizado por la Municipalidad de Gral. San Martín.

El desarrollo de una aplicación de ámbito significativo comienza normalmente con la detección de un problema y la búsqueda de la solución deseada, que se implementa mediante el uso del software. En consecuencia, la aplicación busca una solución a la falta de información centralizada, a la hora de realizar el cobro de los diferentes impuestos; así como también mejorar las falencias del sistema actual. La aplicación puede definirse, a grandes rasgos, como un sistema cobrador de impuestos que se adecuo a este problema específico.

Se puede asegurar también que para desarrollar un sistema de dicha magnitud hay que tener en cuenta cientos si no miles de mecanismos, ya que la información a tratar es muy importante, y una falla en el mismo llevaría a muchas pérdidas en cuanto a dinero y días de trabajo para el Municipio.

Con respecto a los objetivos planteados al comienzo de la tesis se llegaron a las siguientes conclusiones:

- El proceso de migración de datos desde el sistema alojado en Progress se realizó exitosamente. La aprobación de dicho proceso fue a cargo del Gerente de Sistemas del municipio, quien fue el encargado de cotejar los datos almacenados en Progres y los datos almacenados en Oracle.

- El desarrollo de los módulos del sistema fue aceptado de manera exitosa por los operadores. Ellos elogiaron que dichos módulos son intuitivos y de fácil manejo. Además destacaron la mayor información que el mismo le proporciona en cada pantalla.

- En los puestos de trabajo cuyos equipos no cumplían con los requerimientos mínimos se procedió al cambio de los mismos.

- Con respecto al desarrollo de los módulos del Data warehouse no se logró la implementación en la totalidad de los módulos. Obteniendo el desarrollo del módulo de cuentas pero no así el de cobros. Debido a inconsistencias en los datos almacenados en Progress.

- Las capacitaciones se realizaron en forma y fecha, pero la totalidad de los operadores no asistieron. Generando como consecuencia directa que tuvieran dificultades al momento de el manejo del sistema tributario. Cabe destacar que los que sí asistieron comprendieron muy rápidamente el manejo del sistema.

4.3. Recomendaciones

Con base en las pruebas realizadas se recomienda sin cuestionamientos a Oracle para esta aplicación especifica, puesto que este último se plantea como alternativa para aquellas personas o entidades a los que les surja la necesidad de utilizar un sistema de gestión de bases de datos con características como un alto desempeño, confiabilidad, integridad, y disponibilidad de la información, esta recomendación está sustentada en el diseño y ejecución de la base de datos, tomando como apoyo adicional documentación bibliografía, la cual constituyen el soporte para la validación de esta recomendación, Oracle obtuvo un puntaje altísimo de recomendación basada en opiniones del experto, en los criterios de desempeño , tolerancia a fallas/recuperación y carga del sistema, empatando con Oracle en el criterio de integridad.

El lenguaje utilizado fue SQL y Oracle Forms Builder

SQL: el cual brindó muchas herramientas para la realización de dicha aplicación; fue elegido por ser de conocimiento del desarrollador y por ser un potente lenguaje de programación así como también estar bastante estandarizado.

Oracle Forms Builder: Escogido no por ser el más óptimo, sino por la facilidad de desarrollo y conexión con la base de datos, por ejemplo en las opciones de bloques de datos, así como en las llamadas a las funciones de la base de datos. No podría asegurar que este lenguaje sea el correcto para el desarrollo de esta aplicación pero se adapto lo suficiente como para dar solución al problema planteado.

Del resultado de las pruebas en la aplicación se puede confirmar que Oracle es rápido y eficiente, es una solución para empresas que buscan una vía rápida de despliegue de bases de datos, dando garantías en la calidad del soporte, Oracle está diseñado para servicios informativos de gran envergadura y es el más utilizado en muchos ambientes.

Se espera que este documento sea de apoyo para la toma de decisiones, en la optimización de servicios y productos informáticos, teniendo claro que cada necesidad es diferente y por ende el cumplimiento de las expectativas no siempre requiere los mismos recursos.

5. Anexos

5.1. Pasos en la Instalación

5.1.1. Instalación de la Base de Datos

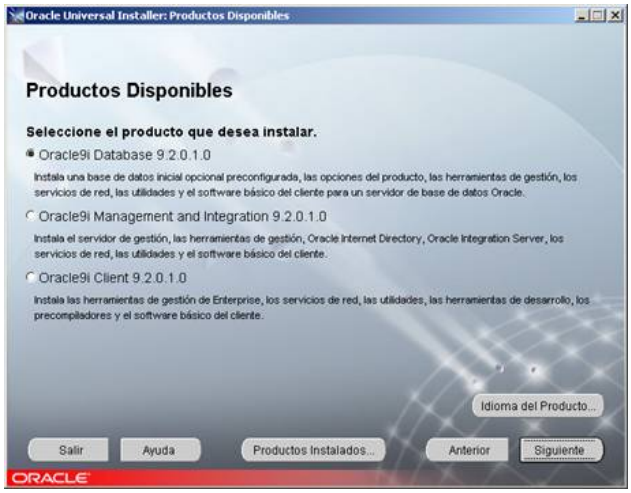
INSTALACIÓN DE ORACLE 9I

Instalación De Oracle 9i En Windows

Introduciremos el CD 1 de la instalación, los tres CDs que componen el programa de instalación se pueden descargar gratuitamente (siempre que no sea con fines lucrativos) desde la propia web de Oracle: www.oracle.com

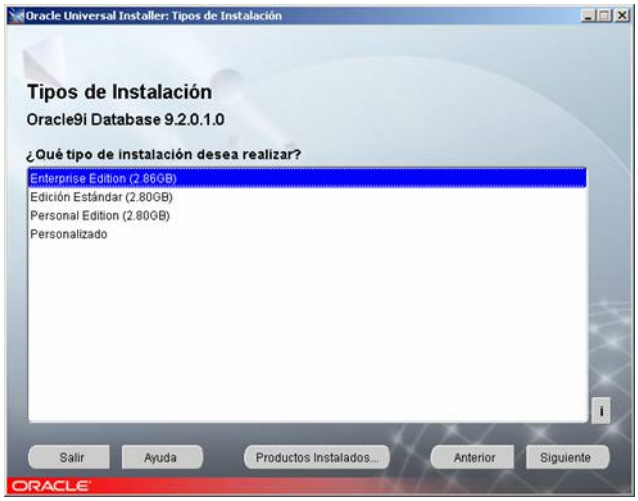
Tras seleccionar la ruta de los ficheros de Oracle pulsaremos en siguiente.

- A continuación seleccionaremos el tipo de instalación que deseemos:



En nuestro caso, puesto que crearemos la base de datos en el servidor donde estamos realizando la instalación seleccionaremos Oracle9i Database 9.2.0.1.0. Si ya disponemos de un servidor de Oracle con las correspondientes bases de datos a las que queramos acceder será suficiente con seleccionar la opción Oracle9i Client 9.2.0.1.0 , en este caso la instalación es bastante más simple, pues sólo es necesario especificar la IP o el nombre de red del servidor de Oracle así como el nombre (sid) de la base de datos a la que queramos acceder.

- Seleccionaremos el tipo de instalación que queramos realizar:



En nuestro caso, seleccionaremos Enterprise Edition, si queremos realizar una instalación más avanzada (Especificando manualmente las opciones a instalar) seleccionaremos personalizado, en este caso aparecería una ventana como esta:

- Seleccionaremos la siguiente opción dependiendo del uso que le queramos dar a nuestra base de datos, normalmente es para uso general. Si no queremos crear una base de datos en el proceso de instalación (Se puede crear en otro momento) seleccionaremos sólo software:

- Seleccionaremos el puerto para Oracle MTS Service, normalmente se suele seleccionar el puerto por defecto 2030. Este parámetro es muy importante pues, si decidimos cambiar el puerto por defecto, cuando queramos que un cliente se conecte al servidor deberemos especificar el puerto que hayamos seleccionado en este punto de la instalación:

- Especificaremos el nombre de la base de datos (Con un máximo de 8 caracteres):

Nota: el SID es el identificador interno que utilizará Oracle para referenciar a nuestra base de datos, se puede elegir uno diferente al del nombre de la base de datos, aunque se suele utilizar el mismo.

- En este punto de la instalación seleccionaremos la ubicación de los archivos de la base de datos que la instalación creará. Oracle recomienda que la ubicación de los archivos de la base de datos esté en un disco físico distinto al de los archivos de la instalación (Software de Oracle).

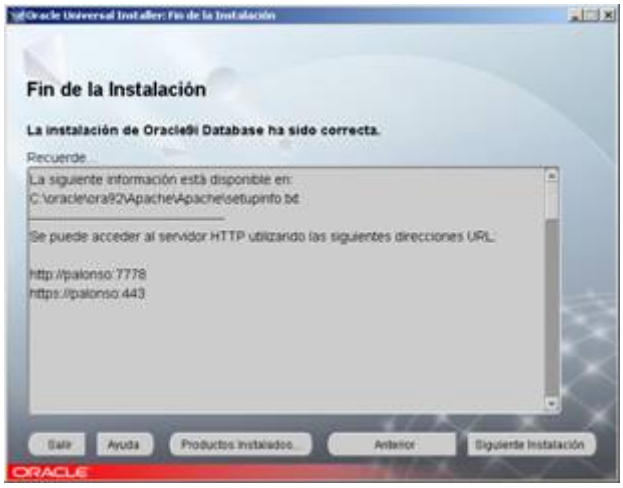
También recomienda que los archivos de Redo Log estén multiplexados (Varias copias, esto se configura en la consola de administración de Oracle) y en diferentes discos físicos. Lógicamente es lo recomendable por Oracle y sólo se configura así cuando se trata de una base de datos que tendrá múltiples accesos concurrentes (Al mismo tiempo) y con un volumen de datos importante, pues el desembolso económico en hardware para la correcta instalación de Oracle puede ser importante. En nuestro caso, instalaremos los archivos de la base de datos en un segundo disco duro instalado exclusivamente para Oracle. A pesar de todo no hay ningún problema por instalar la base de datos en el mismo disco duro que el software de Oracle. Si nuestra organización dispone, por ejemplo, de unos 20 usuarios conectados a Oracle no habría una pérdida del rendimiento por instalarla en el mismo disco duro. Por supuesto esto es orientativo pues dependerá también del volumen de datos que necesite cada usuario así como de otros factores (rpm del disco duro, características del servidor de Oracle (Procesadores, memoria RAM, …), velocidad de la red local, tipo de conexión que realice el software de nuestra empresa que acceda a Oracle (El acceso nativo sin necesidad de utilizar controladores ODBC incrementa considerablemente el rendimiento frente a accesos mediante controladores ODBC que no dejan de ser una pasarela entre el software y Oracle):

- Seleccionaremos el juego de caracteres que vayamos a utilizar, si dejamos el juego de caracteres por defecto Oracle utilizará el juego de caracteres que tengamos configurado en nuestro sistema operativo, es la opción recomendada. Este parámetro sólo sería importante en el caso en que queramos exportar nuestra base de datos a otro servidor de Oracle, en este caso será importante que el juego de caracteres del servidor que recibirá los datos coincida con el que los exportó. Si no coinciden podrían aparecer erróneamente caracteres como "á,¬" alojados en nuestra base de datos:

- Como último paso de la preinstalación nos aparecerá una ventana con el software que se va a instalar, tras comprobar que es correcto pulsaremos en Instalar:

- Introduciremos los CDs de instalación conforme los vaya pidiendo el programa:

- Tras la instalación de Oracle y la creación de la base de datos aparecerá una ventana indicando que el proceso de instalación ha finalizado:



5.1.2. Instalación de Oracle Form

Nos descargarnos Oracle Forms Developer/Services 6i Release 2 for Windows 98/NT/2000/XP desde la página web de Oracle. Téngase en cuenta que para ello tendremos que aceptar las condiciones del contrato y estar dados de alta previamente en la página web de Oracle (Para los que quieran la versión de Linux, tendrán pueden descargarla desde Oracle Forms Developer/Services 6i Release 2 for Linux. O experimentar con las últimas versiones en Oracle Application Server Forms and Reports.).

Una vez descargado, tendremos un fichero comprimido llamado 6i_rel2_xp.zip (si no le hemos cambiado el nombre al descargarlo). Lo descomprimimos y ejecutamos Setup.exe. Seguimos los siguientes pasos:

- Lo primero que nos aparece es una ventana para que introduzcamos algunas definiciones de instalación, cogemos los mismos valores que en la imagen y le damos a ok.

- Nos pregunta que herramienta instalar, seleccionamos Oracle Forms Developer y le damos la botón aceptar/ok.

- Cuando nos pregunta por el tipo de instalación seleccionamos Typical

- Cuando nos pregunta si queremos instalar Forms Server, seleccionamos No

- Le damos a Aceptar un par de veces y ya hemos instalado Oracle Forms Developer.

5.1.3. Instalación de Oracle Reports

Usando el mismo fichero Setup.exe obtenido a partir de descomprimir el fichero 6i_rel2_xp.zip también podremos instalar Oracle Reports Developer

- Lo primero que nos aparece es una ventana para que introduzcamos algunas definiciones de Instalación de Oracle, cogemos los mismos valores que en la imagen y le damos a Ok.

- Nos pregunta que herramienta instalar, seleccionamos "Oracle Reports Developer" y le damos la botón Aceptar.

- Cuando nos pregunta por el tipo de instalación seleccionamos Typical

- Cuando nos pregunta si queremos instalar Reports Server, seleccionamos No

- Le damos a Aceptar un par de veces y ya hemos instalado Oracle Reports Developer.

6. Referencia

6.1. WEB

- http://flanagan.ugr.es
- http://www.monografias.com
- http://es.wikipedia.org
- http://www.hp.com
- http://www.crystalreportsbook.com
- http://www.ecured.cu
- http://www.microsoft.com
- http://www.solovb.net/
- http://www.oracle.com
- http://www.orafaq.com
- http://www.postgresql.org/
- http://www.firebird.com.mx

6.2. Libros

- GANCZARSKI Joe: Data Warehouse Implementations: Critical Implementation Factors Study - VDM Verlag - 2009.
- JOHNSON L. James: BASES DE DATOS: Modelos, lenguajes, diseño - Oxford - 2000.
- KENDALL & KENDALL: Análisis y Diseño de Sistemas - VI Edición - Pearson - México - 2005.
- KEVIN Loney: Oracle 9i - McGraw-Hill - 2002.
- LAUDON Jane & KENNETH: Sistemas de información gerencial- Administración de la empresa digital - Pearson Educación - Prentice Hall - 2006.
- MENDELZON & ALE: Introducción a las Bases de Datos Relacionales - Pearson Educación - Prentice Hall - 2000.