INDICE

INTRODUCCION Pag

CAPITULO I PUENTE JDBC/ODBC PARA CONSULTA

EN BASE DE DATOS

- TECNOLOGIAS EN INTERNET 4

1.1.1Sun Microsystem 4

1.1.2 Microsoft 5

- SERVIDORES WEB 5

1.2.1Concepto 5

1.2.2Principales Servidores Web 7

- CGI 11

1.3.1 Lenguajes 12

- Servlets 12

1.4.1Ejecución de Servlets 13

- API 14
- ASP 14

1.6.1 Ventajas respecto a la programación CGI 16

1.6.2 Requisitos 17

CAPITULO II ARQUITECTURA CLIENTE/SERVIDOR

- DEFINICION 17

2.2 ELMENTOS PRINCIPALES 18

2.3EVOLUCION DE LA ARQUITECTURA CLIENTE/SERVIDOR 19

2.3.1La era de la computadora central 19

2.3.2La era de las computadoras dedicadas 19

2.3.3La era de la conexión libre 20

2.3.4La era del computo a través de redes 20

• ESTILOS DEL MODELO CLIENTE SERVIDOR 21

2.4.1Presentación distribuida 21

2.4.2Presentación remota 22

2.4.3 Lógica distribuida 23

2.4.4 Administración de datos remota 23

2.4.5 Base de datos distribuida 24

2.5 Características de los sistemas cliente servidor 25

2.6 Tipos de servidores 25

2.6.1 Servidores de archivos 25

2.6.2 Servidores de bases de datos 25

2.6.3 Servidores de transacciones 26
2.6.4 Servidores de Groupware 26

2.6.5 Servidores de objetos 26

2.6.6 Servidores web 26

2.7 Definición de middleware 26

2.7.1 Tipos de Middleware 27

2.8 Funciones de un programa servidor 27

2.9 Necesidades de un servidor que debe suplir 28

un sistema operativo

2.10 LOS SOCKETS 29

2.11 IP 32

CAPITULO III INTERFACES ENTRE JDBC/OBDC

3.1JDBC 32

3.1.1Funciones 35

3.1.2 Características 35

3.1.3 Tipos de Drivers JDBC 37

3.1.4 Acceso de la API JDBC a las BD 40

3.2 OBDC 41

3.2.1Características 44

3.2.2 Conexión a un recurso ODBC 45

3.2.3 Campo de aplicación y alcance 45

3.3Conectividad de Bases de Datos de Java (JDBC) 47

3.4JDBC vs. ODBC 47

3.5JDBC vs. ODBC 48

3.6 Preparación del entorno de desarrollo 50

3.6.1 El primer programa con JDBC 57

3.6.2Consultas en JDBC 60

CAPITULO IV MANEJADORES DE BASE DE DATOS

4.1 ORACLE 67

4.2 MICROSOFT SQL SERVER

4.2.1 Características 69

4.2.2Funciones 70

4.3 SYBASE SQL 71

4.3.1 Soluciones IPS de Sybase para la Integración de Datos 72

de la Empresa

4.4 INFORMIX 73

4.5 DB2 73

4.5.1 Escalabilidad 74

4.6 DBBASE 76

4.7ACCESS 76

4.7.1Características 76

4.7.2 Crear una Base de Datos 76

CAPITULO I PUENTE JDBC/OBDC PARA CONSULTAS EN BASES

DE DATOS

- **TECNOLOGIAS EN INTERNET**
- **Sun Microsystem**

Sun Microsystem provee a sus clientes las diferentes líneas de productos de plataformas informáticas y servicios asociados.
Los productos de Sun son ampliamente reconocidos en el mercado, y la empresa en si misma es considerada líder en plataformas para soluciones sobre internet.

Los sistemas de computación de red que ofrece Sun abarca desde estaciones de trabajo hasta servidores empresariales de alto rendimiento para aplicaciones de gran escala y misión critica. Pasando también por productos de almacenamiento masivo que ofrecen un alto rendimiento y versatilidad que aseguran una alta disponibilidad y una amplia línea de respaldo. Brindando de esa manera una opción de respaldo automático confiable y sin interrupciones.

Algunas de las características importantes de los productos son:

- Plataformas de tecnología abierta, confiable y escalable.
- Utilizar el mismo ambiente operativo a lo largo de toda su familia de sistemas de computación.
- Crecimiento tanto en funcionalidad como en capacidad, en forma gradual e ilimitada.
- Integrar múltiples plataformas.

- **Microsoft**

Desde su comienzo en 1975, la misión de Microsoft ha sido la de crear software para las PC personales, las cuáles facultan y enriquecen a las personas en su trabajo, en la escuela y en el hogar. La visión inicial de una PC en cada escritorio y en cada hogar está asociada hoy en día al serio compromiso con las tecnologías relacionadas al Internet que incrementan el poder y el alcance de la PC y el de sus usuarios.
Los productos de Microsoft ofrecen un conjunto integral de soluciones para crear una infraestructura de negocios flexible e integrada. Desde la mensajería y la colaboración hasta la administración de base de datos y desde el comercio electrónico.

- **Servidores Web**

1.2.1 Concepto

El servidor de Paginas es la parte primordial de cualquier sitio de Internet, ya que es el encargado de generar y enviar la información a los usuarios finales.

En un sentido muy estricto un "Web Server" no es lo mismo que "Application Server", pero ultimamente estos dos términos se prestan a una gran confusión . Cuando se crearon los primeros Servidores de paginas ("Web Server") como Apache, éste solo era encargado de enviar los datos al usuario final, pero cualquier otra información que requiriera de algun tipo de personalización era realizada por un interpretador que ejecutaba un Script, generalmente en Perl el cual es un lenguaje que permite manipular de una manera muy flexible cualquier archivo de texto que exista en un sistema(1).

Sin embargo conforme las demandas de los Servidores de paginas "Web Server" incrementaron fue necesario eficientizar este proceso, ya que el llamar un interpretador para que ejecutara otro programa (en el caso Perl) ponia una demanda muy fuerte sobre el "Host" que mantenia el Servidor de Paginas "Web Server"

Hoy en día, se pudiera decir que TODOS los "Servidores de Paginas" ya son "Servidores de Aplicaciones" ya que se les ha desarrollado alguna funcionalidad

especial que les permite realizar, valga la redundancia, aplicaciones de Servidor .

Y para agregar a la confusión en la Industria hoy dia ya abundan los "Java Application Servers" que son otra cosa bastante diferente y malamente ya son designados "Application Servers" o "Servidores de Aplicaciones" a solas.

Dependiendo de la funcionalidad se trae consigo complejidad al sistema, ya sea en la forma de requerimientos del sistema (memoria , procesadores), carga administrativa (configuración, tiempo de desarrollo) o alguna otra.

En el diagrama anterior se puede observar que nuestro navegador solicita información al servidor de paginas, esta solicitud inicial se lleva acabo mediante HTTP sin embargo una vez que esta solicitud llega al Servidor ésta puede tomar varias acciones.

En el esquema las lineas punteadas (−− −−) indican el "Host" (computadora fisica) mientras las lineas solidas son un proceso/programa dentro del mismo "Host", bajo esta hipotesis se describiran los "Servidores de Paginas".

1.2.2 Principales Servidores Web

Entre los principales Servidores Web podemos encontrar los siguientes:

- Apache 1.1.3
- Nestcape FastTrack 2.0,Enterprise 2.0,Enterprise 3.0
- Microsoft IIS
- WebLogic Tengah
- Lotus Domino Go Web Server
- IBM Interner Conecction Server
- Java Web Server

A continuación se dará una explicación de algunos de los Servidores Web antes mencionados.

1.2.2.1 Apache

Apache es uno de los Servidores de paginas más utilizados, posiblemente porque ofrece instalaciones sencillas para sitios pequeños y si se requiere es posible expandirlo hasta el nivel de los mejores productos comerciales. Si se utiliza para un sitio pequeño que solo contenga archivos en HTML, esto es, no requiera de aplicaciones de servidor su funcionallidad es excelente, pero que sucede cuando se requiere una *aplicación de Servidor* ? La *aplicación de servidor* implica lo siguiente:

Cuando el servidor de paginas (Apache) recibe la requisición de "x" pagina éste reconoce cuando debe enviar un documento estatico (HTML) o ejecutar algún tipo de

aplicación, en el diagrama se puede observar que la solicitud de "x" pagina invoca (llama) un programa en Perl y este a su vez solicita información a una base de datos, por lo tanto para llevar acabo esta operación debieron iniciarse 2 procesos nuevos,(quizás esto no sea de gran importancia para un sitio de 100 visitas diarias, pero que sucedería con uno de 2 visitas por segundo ?

Si no se tienen los suficientes recursos en cuanto a memoria y procesadores se refiere, seguramente caerá el

servidor de paginas o bien se queme el "Host" por la demanda excesiva. Apache tiene tanto tiempo de desarrollo que han sido desarrolladas diferentes soluciones para evitar estas ineficiencias, algunas:

- Es capaz de utilizar otros interpretadores y lenguajes como "Tcl", "PhP","Python" .
- Puede conectarse directamente a una Base de datos .
- Entre otras

Cabe mencionar que muchos sitios de alto tráfico aún permanecen bajo este tipo de Arquitectura, en ocasiones si se tienen los recursos suficientes continua siendo costeable esta metodología a migrar a otro tipo de desarrollo, sin embargo siempre es conveniente conocer otras alternativas.

1.2.2.2 AOLServer

AOLServer fue diseñado conociendo varias deficiencias que existían en el modelo inicial utilizado por Apache.
AOLServer desde sus versiones iniciales fue desarrollado con "Threading" en mente, esto es, compartir la memoria del Proceso general en varios sub−procesos o Threads"(2), esto no solo eficientiza las conexiones al servidor de paginas sino

también reduce la carga sobre el mismo.

Otra ventaja de AOLServer es el ofrecimiento de ADP ("Aol Dynamic Pages") que son muy similares a las ASP's (Active Server Pages) de Microsoft o JSP (Java Server Pages) por Java, la diferencia estriba que ADP's utilizan un API diseñado especialmente para accesar los elementos del servidor, pero su funcionamiento es igual al de ADP y JSP: mezclar elementos de HTML con elementos de programación para generar contenido dinámico.

1.2.2.3 IIS (Information Server)

IIS es el servidor de paginas desarrollado por Microsoft para Windows NT/2000, a diferencia de los dos servidores de paginas mencionados anteriormente, IIS solo puede operar en plataformas Windows. El punto más favorable de este servidor es ASP's que facilitan el desarrollo de aplicaciones, sin embargo existen alternativas como ADP's de Aolserver y JSP's para Java.

Cabe mencionar que hay un servidor similar que es Zope el cual es un servidor de paginas"(Aplicaciones) Open Source que utiliza Python como su Scripting Language y es capaz de accesar un gran numero de Bases de Datos .

- **CGI**

CGI es una norma para establecer comunicación entre un servidor web y un programa, de tal modo que este último pueda interactuar con internet(3). También se usa la palabra CGI para referirse al programa mismo, aunque lo correcto debería ser script. De todos modos en este tutorial nos interesa aprender a escribir estos programas por lo cúal usaremos la palabra indistintamente, como es costumbre en castellano.

Un CGI (Common Gateway Interface) es un programa que se ejecuta en tiempo real en un Web Server en respuesta a una solicitud de un Browser. Cuando esto sucede el Web Server ejecuta un proceso hijo que lo recibirá los datos que envía el usuario (en caso de que los haya), pone a disposición del mismo algunos datos en forma de variables de ambiente y captura la salida del programa para enviarlo como respuesta al Browser.

El propósito de los CGI´s es proveer "inteligencia" e interactividad a un sitio web por ejemplo encontrar un sitio en Yahoo utilizando solo los links que se proveen puede ser una labor frustrante, sin embargo usar el

formulario y solicitar una búsqueda personalizada suele frustrarnos (un poco) menos, ya que un CGI nos provee de una respuesta hecha a la medida (eso dice la teoría) de nuestra consulta.

- **Lenguajes**

Un CGI se puede programar en cualquier lenguaje soportado por el sistema operativo del servidor, uno de ellos podría ser C el podrá ser fácilmente portado a casi cualquier sistema operativo.

Si se escribe CGIs para IIS (Windows NT), se debe asegurar de compilar en modo 32 bits, de otro modo no funcionarán.

Otro lenguaje popular para escribir CGI´s es el Perl, el cuál es interpretado y al igual que el C es altamente portable entre sistemas operativos.

- **Servlets**

Un servlet de forma intuitiva se puede definir como un programa independiente de plataforma que aporta la misma funcionalidad a la programación en el lado del servidor que tradicionalmente han realizado la interfaz CGI(4).Con respecto a esta tecnología aporta numerosas ventajas que citaremos a continuación:

  - Independencia de la plataforma. (La tan anhelada premisa del write once run everywhere aun no totalmente conseguida).Esto proporciona un menor esfuerzo de codificación con respecto a soluciones dependientes del servidor web y de la plataforma como ISAPI o NSAPI.
  - Ejecución en paralelo de múltiples peticiones por una sola instancia del servlet. Tradicionalmente en los programas CGI se ejecuta un proceso distinto para cada petición lo que conlleva una gradual degradación del rendimiento y una necesidad de recursos muy elevada. En un servlet todas las peticiones se atienden en el mismo proceso por distintos hilos y una vez que se ha cargado el servlet este permanece en memoria hasta que se reinicie el servidor o hasta que se le diga lo contrario con lo cual las subsiguientes peticiones son mas rápidas al encontrarse el programa ya cargado en memoria.
  - Un servlet puede ejecutarse(incido en esto puede no es necesario) en una sandbox o recinto de seguridad parecido al modelo que se sigue con los applets. Debido a esto pueden colocarse servlets en servidores dedicados a hosting sin que la empresa tema por la integridad del servidor y la seguridad de las aplicaciones. Históricamente el rechazo al uso de los servlets se ha debido a la injustificada leyenda de la falta de velocidad de ejecución del lenguaje Java. Sin embargo si observamos las lenguajes de script que tradicionalmente se han usado para escribir aplicaciones CGI´s (Perl, PHP..) nos encontramos con que también son interpretados lo que unido a la necesidad de lanzar un proceso por petición provocan un rendimiento considerablemente menor.

- **Ejecución de Servlets**

En la actualidad la mayoría de servidores web tanto comerciales como de licencia libre tienen la capacidad de ejecutar servlets a través de plug−ins o módulos. Señalaremos unos cuantos:

  - Apache 1.1.3
  - Nestcape FastTrack 2.0,Enterprise 2.0, Enterprise 3.0
  - Microsoft IIS
  - WebLogic Tengah
  - Lotus Domino Go Web Server
  - IBM Interner Conecction Server
  - Java Web Server

- **API**

API, Interfaz de programación de aplicación(5) (Application Programming Interfaz) es un grupo de clases e interfaces que determina una serie de comportamientos que son iguales a todas las plataformas y que establecen las bases de datos para todos los applets y aplicaciones de Java. Aunque un programa dependa de las estrategias del programador, estas clases e interfaces son las herramientas que usan todos los programadores al crear sus aplicaciones Java.

- **ASP**

Haciendo historia se recordara que en un principio sólo existían páginas web escritas en HTML, por la naturaleza de este lenguaje, totalmente estáticas.

A continuación llegó el CGI (Common Gateway Interface), que eran aplicaciones externas que se ejecutaban en el servidor(6).
Esta aplicación recibía peticiones como si se tratase de una página HTML normal, y en función de los parámetros que tenía previamente establecidos y los valores que se le hubiesen suministrado la máquina cliente, devolvía como resultado una página HTML.
Con esto se lograba que las páginas de internet dejaran de ser estáticas para convertirse en dinámicas, es decir, nuestra aplicación CGI devolvía una página distinta en función del usuario que la estuviera consultando, el día, la hora. Un ejemplo típico de este tipo de tecnologías lo representan los buscadores de internet. Simplemente constan de una aplicación que se conecta a una base de datos para realizar la consulta que le hayamos solicitado, y esta aplicación nos devuelve como resultado una nueva página web que incluye los resultados de dicha consulta. Sin este tipo de tecnología, el concepto de buscador jamás existiría, o por lo menos, no sería viable su puesta en marcha.

El lenguaje VBScript para servidor o ASP actúa de manera parecida a una aplicación CGI, pero la mejora que introduce radica en que la aplicación ya no es un programa ejecutable, sino un sencillo código Script que nos va a permitir controlar todo lo que ocurre en el servidor y construir páginas dinámicas.

La principal ventaja que aporta la programación ASP es la sencillez y la comodidad, pero aunque sigue existiendo un programa, su funcionamiento es totalmente transparente para el programador, ahora es el servidor quien se ocupa del trabajo, nuestra misión únicamente consiste en escribir rutinas en Script.

La funcionalidad que podemos dar a nuestras páginas es impresionante, no sólo por el simple hecho de poder implementar nuevas funciones como buscadores, foros, sino cosas más sencillas, como por ejemplo cambiar los hipervínculos de una página si en nuestra empresa tenemos en varios servidores nuestras páginas web (algo típico en servidores de intranet), introducir propaganda que cambie cada vez que se visite la página.

Pero no nos confundamos, disponemos de unas rutinas en el servidor que permiten realizar páginas dinámicas en función de los parámetros que le hayamos introducido a la rutina, y esta devuelve una página web normal y corriente, que es la que recibe el usuario. Es decir, la página HTML final que le llega al usuario es totalmente estática (a menos que insertemos JavaScript, applets o VBScript).
Por lo tanto, a la hora de programar en ASP no hay que olvidar que todas las operaciones que se van a realizar, se van a efectuar SIEMPRE en el servidor y nunca en la máquina cliente.

En resumen, una página ASP es un archivo dentro de nuestro servidor de internet, compuesto por una mezcla de HTML y Scripts (que se ejecutarán en el servidor). Estos Scripts procesarán los datos provenientes del cliente y generará una página HMTL como resultado que posteriormente se enviará a dicho cliente.

1.6.1 Ventajas respecto a la programación CGI:

La programación CGI se realizaba en lenguajes como C++ o Perl, que en la mayoría de los casos, había que compilar y linkar. Presentando además la dificultad añadida de aprender dichos lenguajes, y no siempre

estaban bien resueltos problemas como el acceso a bases de datos, el control del sistema.

ASP soluciona estos problemas. El aprendizaje de este lenguaje es bastante simple (muy similar a Visual Basic), se interpreta (con lo que se ahorra el tiempo de compilación y linkado, pero se pierde a la hora de ejecutarlo), la corrección de los errores es más simple, ofrece un control muy bueno sobre el sistema y el acceso a bases de datos es compatible con ADO y ODBC (garantizándonos así el acceso a bases de datos como Oracle, SQL Server, Access).

1.6.2 Requisitos

Estos requerimientos son muy distintos a los del usuario final. Se necesita el sistema operativo Windows NT 4.0 o Windows 2000 server con el servidor de internet Microsoft Internet IIS 3.0.

CAPITULO II ARQUITECTURA CLIENTE/SERVIDOR

- **DEFINICION**

Con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones:

- Cualquier combinación de sistemas que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde esta ubicada.
- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.
- Es un procesamiento un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.
- El termino cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o mas programas: una aplicación y un servicio soportante.

- **Elementos principales**

Los elementos principales de la arquitectura cliente servidor son justamente el elemento llamado cliente y el otro elemento llamado servidor (7). Por ejemplo dentro de un ambiente multimedia, el elemento cliente seria el dispositivo que (7)puede observar el vídeo, cuadros y texto, o reproduce el audio distribuido por el elemento servidor. Por otro lado el cliente también puede ser una computadora personal o una televisión inteligente que posea la capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audios, fotografías digitales y texto y los distribuye bajo demanda de ser una maquina que cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brinda éstos al cliente.

2.3 Evolución de la arquitectura cliente servidor

2.3.1La era de la computadora central

Desde sus inicios el modelo de administración de datos a través de computadoras se basamento en el uso de terminales remotas, que se conectaban de manera directa a una computadora central(8). Dicha computadora central se encargaba de prestar servicios caracterizados por que cada servicio se prestaba solo a un grupo exclusivo de usuarios. El personal de la llamada área de sistemas se encargaba de consolidar o integrar la información cuando las necesidades de los usuarios lo exigían.

2.3.2 La era de las computadoras dedicadas

Esta es la era en la que cada servicio empleaba su propia computadora que permitía que los usuarios de ese servicio se conectaran directamente. Esto es consecuencia de la aparición de computadoras pequeñas, de fácil uso, mas baratas y mas poderosas de las convencionales.
Este modelo de funcionamiento posee en la actualidad importantes inconvenientes. El primero que, conforme crece el numero de usuarios que requieren acceso a los datos administrados por cada sistema, se presenta la necesidad hacer uso de computadoras cada vez mas poderosas en sus sistemas

de entrada/salida; los nuevos costos producidos son consecuencia del aumento de numero de usuarios a quienes se les tiene que otorgar licencias para poder hacer uso legal de los programas contenidos en los sistemas de información. El segundo inconveniente es que estas computadoras son incapaces de comunicarse entre si y por tanto la información compartida es nula. Estos inconvenientes que se observan hoy son consecuencia directa de la tecnología que se empleo para dar origen a este esquema y que hoy es obsoleto.

- **La era de la conexión libre**

Hace mas de 10 años que la computadoras escritorio aparecieron de manera masiva. Esto permitió que parte apreciable de la carga de trabajo de computo tanto en el ámbito de cálculo como en el ámbito de la presentación se lleven a cabo desde el escritorio del usuario. En muchos de los casos el usuario obtiene la información que necesita de alguna computadora de servicio. Estas computadoras de escritorio se conectan a las computadoras de servicio empleando software que permite la emulación de algún tipo de terminal. En otros de los casos se les transfiere la información haciendo uso de recursos magnéticos o por transcripción.

2.3.4La era del computo a través de redes

Esta es la era que esta basada en el concepto de redes de computadoras, en la que la información reside en una o varias computadoras, los usuarios de esta información hacen uso de computadoras para laborar y todas ellas se encuentran conectadas entre si. Esto brinda la posibilidad de que todos los usuarios puedan acceder a la información de todas las computadoras y a la vez que los diversos sistemas intercambien información.

2.3.5 La era de la arquitectura cliente servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores(9)estos últimos responden a la demanda del cliente que la produjo. Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es la Internet.
Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

- **Estilos del modelo cliente servidor**

2.4.1Presentación distribuida

- Se distribuye la interfaz entre el cliente y la plataforma servidora.
- La aplicación y los datos están ambos en el servidor.
- Similar a la arquitectura tradicional de un Host y Terminales.
- El Pc se aprovecha solo para mejorar la interfaz gráfica del usuario.

2.4.1.1 Ventajas

- Revitaliza los sistemas antiguos.
- Bajo costo de desarrollo.
- No hay cambios en los sistemas existentes.

2.4.1.2 Desventajas

- El sistema sigue en el Host.
- No se aprovecha la GUI y/o LAN.
- La interfaz del usuario se mantiene en muchas plataformas.

2.4.2Presentación remota

- La interfaz para el usuario esta completamente en el cliente.
- La aplicación y los datos están en el servidor.

2.4.2.1 Ventajas

- La interfaz del usuario aprovecha bien la GUI y la LAN.
- La aplicación aprovecha el Host.
- Adecuado para algunos tipos de aplicaciones de apoyo a la toma de decisiones.

2.4.2.2 Desventajas

- La aplicaciones pueden ser complejas de desarrollar.
- Los programas de la aplicación siguen en el Host.
- El alto volumen de trafico en la red puede hacer difícil la operación de aplicaciones muy pesadas.
- **Lógica distribuida**
- La interfaz esta en el cliente.
- La base de datos esta en el servidor.
- La lógica de la aplicación esta distribuida entre el cliente y el servidor.

2.4.3.1 Ventajas

- ♦ Arquitectura mas corriente que puede manejar todo tipo de aplicaciones.
- ♦ Los programas del sistema pueden distribuirse al nodo mas apropiado.
- ♦ Pueden utilizarse con sistemas existentes.

2.4.3.2 Desventajas

- ♦ Es difícil de diseñar.
- ♦ Difícil prueba y mantenimiento si los programas del cliente y el servidor están hechos en distintos lenguajes de programación.
- ♦ No son manejados por la GUI 4GL.

2.4.4 Administración de datos remota

- En el cliente residen tanto la interfaz como los procesos de la aplicación.
- Las bases de datos están en el servidor.
- Es lo que comúnmente imaginamos como aplicación cliente servidor

2.4.4.1 Ventajas

- ♦ Configuración típica de la herramienta GUI 4GL.
- ♦ Muy adecuada para las aplicaciones de apoyo a las decisiones del usuario final.

- ♦ Fácil de desarrollar ya que los programas de aplicación no están distribuidos.
- ♦ Se descargan los programas del Host.

2.4.4.2 Desventajas

- ♦ No maneja aplicaciones pesadas eficientemente.
- ♦ La totalidad de los datos viaja por la red, ya que no hay procesamiento que realice el Host.
- ♦ **Base de datos distribuida**
- ♦ La interfaz, los procesos de la aplicación, y , parte de los datos de la base de datos están en cliente.
- ♦ El resto de los datos están en el servidor.

1.4.5.1 Ventajas

- ◊ Configuración soportada por herramientas GUI 4GL.
- ◊ Adecuada para las aplicaciones de apoyo al usuario final.
- ◊ Apoya acceso a datos almacenados en ambientes heterogéneos.
- ◊ Ubicación de los datos es transparente para la aplicación.

1.4.5.2 Desventajas

- ◊ No maneja aplicaciones grandes eficientemente.
- ◊ El acceso a la base de datos distribuida es dependiente del proveedor del software administrador de bases de datos.

2.5 Características de los sistemas cliente servidor

- ◊ Servicio
- ◊ Recursos compartidos
- ◊ Protocolos Asimétricos
- ◊ Transparencia de ubicación
- ◊ Mezcla e igualdad
- ◊ Intercambio basados en mensajes
- ◊ Encapsulamiento de servicios
- ◊ Facilidad de escalabilidad
- ◊ Integridad

2.6 Tipos de servidores

2.6.1 Servidores de archivos

Servidor donde se almacena archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hojas de calculo, etc.

2.6.2 Servidores de bases de datos

Servidor donde se almacenan las bases de datos, tablas, índices. Es uno de los servidores que más carga tiene .

2.6.3 Servidores de transacciones

Servidor que cumple o procesa todas las transacciones. Valida primero y recién genera un pedido al servidor de bases de datos.

2.6.4 Servidores de Groupware

Servidor utilizado para el seguimiento de operaciones dentro de la red.

2.6.5 Servidores de objetos

Contienen objetos que deben estar fuera del servidor de base de datos. Estos objetos pueden ser videos, imágenes, objetos multimedia en general.

2.6.6 Servidores web

Se usan como una forma inteligente para comunicación entre empresas a través de Internet. Este servidor permite transacciones con el acondicionamiento de un browser especifico.

2.7 Definición de middleware

Es un termino que abarca a todo el software distribuido necesario para el soporte de interacciones entre Clientes y Servidores(10).
Es el enlace que permite que un cliente obtenga un servicio de un servidor. (10)
Este se inicia en el modulo de API de la parte del cliente que se emplea para invocar un servicio real; esto pertenece a los dominios del servidor. Tampoco a la interfaz del usuario ni la a la lógica de la aplicación en los dominios del cliente.

2.7.1 Tipos de Middleware

Existen dos tipos de middleware:

- Middleware general
  Este tipo permite la impresión de documentos remotos, manejos de transacciones, autenticación de usuarios, etc.
- Middleware de servicios específicos
  Generalmente trabajan orientados a mensajes. Trabaja uno sola transacción a la vez.

2.8 Funciones de un programa servidor

- Espera las solicitudes de los clientes.
- Ejecuta muchas solicitudes al mismo tiempo.
- Atiende primero a los clientes VIP.
- Emprende y opera actividades de tareas en segundo plano.
- Se mantiene activa en forma permanente.

2.9 Necesidades de un servidor que debe suplir un sistema operativo

- Servicios básicos:
  - ◊ Preferencia de tareas.
  - ◊ Prioridad de tareas.
  - ◊ Semáforos.
  - ◊ Comunicaciones entre procesos (ipc).

- ◊ Comunicaciones entre procesos locales/remotos.
- ◊ Hilos.
- ◊ Protección entre tareas.
- ◊ Sistemas de archivos de alto desempeño para usuarios múltiples.
- ◊ Administración eficiente de memoria.

♦ Servicios complementarios:

- ◊ Comunicación ubicuas.
- ◊ Extensiones del sistema operativa en red.
- ◊ Objetos binarios grandes.
- ◊ Servicios de autenticación y autorización.
- ◊ Administración de sistemas.
- ◊ Servicios de bases de datos y transacciones.
- ◊ Servicios de Internet.
- ◊ Servicios orientado a objetos.

2.10 LOS SOCKETS.

Los *sockets* no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable (emita o reciba información) con otro proceso incluso estando estos procesos en distintas máquinas(11)(11). Esta característica de interconectividad entre máquinas hace que el concepto de socket nos sirva de gran utilidad. Esta interfaz de comunicaciones es una de las distribuciones de Berkeley al sistema UNIX, implementándose las utilidades de interconectividad de este Sistema Operativo (*rlogin, telnet, ftp, ...*) usando sockets.
Un socket es al sistema de comunicación entre ordenadores lo que un buzón o un

(11)

teléfono es al sistema de comunicación entre personas: un punto de comunicación entre dos agentes (procesos o personas respectivamente) por el cual se puede emitir o recibir información. La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar ficheros. Debido a esta característica, se podrá realizar redirecciones de los archivos

de E/S estándar (descriptores 0,1 y 2) a los sockets y así combinar entre ellos aplicaciones de la red. Todo nuevo proceso creado heredará, por tanto, los descriptores de sockets de su padre.
La comunicación entre procesos a través de sockets se basa en la filosofía CLIENTE−SERVIDOR: un proceso en esta comunicación actuará de proceso servidor creando un socket cuyo nombre conocerá el proceso cliente, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho socket nombrado.

El proceso crea un socket sin nombre cuyo valor de vuelta es un descriptor sobre el que se leerá o escribirá, permitiéndose una comunicación bidireccional, característica propia de los sockets y que los diferencia de los pipes, o canales de comunicación unidireccional entre procesos de una misma máquina. El mecanismo de comunicación vía sockets tiene los siguientes pasos:

1) El proceso servidor crea un socket con nombre y espera la conexión.
2) El proceso cliente crea un socket sin nombre.
3) El proceso cliente realiza una petición de conexión al socket

servidor.
4) El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.
Es muy común en este tipo de comunicación lanzar un proceso hijo, una vez realizada la conexión, que se ocupe del intercambio de información con el proceso cliente mientras el proceso padre servidor sigue aceptando conexiones. Para eliminar esta característica se cerrará el descriptor del socket servidor con nombre en cuanto realice una conexión con un proceso socket cliente.

−> Todo socket viene definido por dos características fundamentales:

− El tipo del socket, que indica la naturaleza del mismo, el tipo de comunicación que puede generarse entre los sockets.
− El dominio del socket especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.

2.10.1 Tipos de sockets

Define las propiedades de las comunicaciones en las que se ve envuelto un socket, esto es, el tipo de comunicación que se puede dar entre cliente y servidor. Estas pueden ser:

◊ Fiabilidad de transmisión.
Mantenimiento del orden de los datos.
No duplicación de los datos.
El "Modo Conectado" en la comunicación.
◊ Envío de mensajes urgentes.
Los tipos disponibles son los siguientes:
* Tipo SOCK_DGRAM: sockets para comunicaciones en modo no conectado, con envío de datagramas de tamaño limitado (tipo telegrama). En dominios Internet como la que nos ocupa el protocolo del nivel de transporte sobre el que se basa es el UDP.
* Tipo SOCK_STREAM: para comunicaciones fiables en modo conectado, de dos vías y con tamaño variable de los mensajes de datos. Por debajo, en dominios Internet, subyace el protocolo TCP.
* Tipo SOCK_RAW: permite el acceso a protocolos de más bajo nivel como el IP (nivel de red)
* Tipo SOCK_SEQPACKET: tiene las características del SOCK_STREAM pero además el tamaño de los mensajes es fijo.

2.11 IP

Una dirección IP es un código numérico que identifica a un ordenador específico en Internet(12). Las direcciones de Internet son asignadas por un organismo llamado InterNIC. El registro incluye un nombre (whitehouse.gov), nombre de dominio, y un número (198.137.240.100), dirección o número IP.

(12)

CAPITULO III INTERFACES ENTRE JDBC/OBDC

3.1JDBC

Con la ayuda de JDBC, la habilidad de JAVA para integrarse con DBMS comerciales y su naturaleza orientada al manejo de la Red, es posible crear un ambiente ideal tipo cliente−servidor.

Al navegar en el World Wide Web, es fácil darse cuenta de que existe ya mucha información. Muchas compañias estan usando bases de datos relacionales para manejar la información en sus sitios del Web. Por ejemplo, la mayoría de las máquinas de búsqueda usan este tipo de base de datos. Las bases de datos relacionales son ideales para el almacenamiento de grandes cantidades de información, la cual puede ser accedida por muchos usuarios.

Hoy en día, la mayoría de los gestores de bases de datos relacionales tienen soporte para la utilización de interfaces HTML. Para algunas aplicaciones, las páginas HTML favorecen al interfaz, pero, en aplicaciones más complejas se presentan ciertas limitaciones que no permiten generar un buen trabajo. En estos casos, es conveniente la utilización de lenguajes de programación como JAVA, que permitan elaborar aplicaciones para generar el interfaz con la base de datos.

La verdad es que todo lo que se pueda hacer en C++ se puede hacer con JAVA. Y mejor aún, al combinar JAVA con JDBC, se presentan nuevas espectativas para comunicarse con las bases de datos a través de un esquema similar al de las aplicaciones en C y C++. Muchos usuarios siguen confundidos en cuanto a la naturaleza de JAVA y piensan que es solo útil para animaciones y applets sencillos, para el Web.

Vendedores de productos comerciales para bases de datos como Oracle, IBM, Sybase, SAS y Borland han puesto su atención en la nueva metodología de integración JAVA−DBMS. Una parte de este interés es impulsado por la posibilidad de obtener aplicaciones con una mejor imagen. Pero en la mayoría de los casos, estas compañías lo ven como una nueva oportunidad para que los programadores puedan tomar ventaja de esta tecnología en un futuro próximo.

Antes que nada definiremos lo que es JDBC. Es una API de JAVA para permitir ejecutar instrucciones SQL(13 (Structured Query Language:Lenguaje estructurado de consultas), que es un lenguaje de alto nivel para crear, manipular, examinar y gestionar bases de datos relacionales

Podemos decir en tres frases lo que hace JDBC:

- Establece una conexión con una BD, que puede ser remota o no.
- Envia sentencias SQL a la BD.
- Procesa los resultados obtenidos de la BD.

JDBC es un API incluido dentro del lenguaje Java para el acceso a bases de datos. Consiste en un conjunto de clases e interfaces escritos en Java que ofrecen un completo API para la programación de bases de datos, por lo tanto es la una solución

100% Java que permite el acceso a bases de datos, la primera aparición de JDBC (JDBC 1.0) se encuentra dentro del paquete java.sql que ha fue incorporado en la versión del JDK 1.1.x (Java Development Kit) correspondiente a la versión 1.1 del lenguaje Java, JDBC 2.0 sigue estando en el mismo paquete pero en las versiones JDK 1.2 y JDK 1.3 que se corresponden con la versión 2 del lenguaje Java, o también denominada plataforma Java 2 (Java 2 Platform). JDBC es un especificación formada por una colección de interfaces y clases abstractas, que deben implementar todos los fabricantes de drivers que quieran realizar una implementación de su driver 100% Java y compatible con JDBC (JDBC−compliant driver)

Debido a que JDBC está escrito completamente en Java también posee la ventaja de ser independiente de la plataforma. No será necesario escribir un programa para cada tipo de base de datos, una misma aplicación escrita utilizando JDBC podrá manejar bases de datos Oracle, Sybase, o SQL Server. Además podrá ejecutarse en cualquier sistema que posea una Máquina Virtual de Java, es decir, serán aplicaciones completamente independientes de la plataforma.

3.1.1Funciones

Básicamente el API JDBC hace posible la realización de las siguientes tareas:

Establecer una conexión con una base de datos.

Enviar sentencias SQL.

Manipular los datos.

Procesar los resultados de la ejecución de las sentencias.

3.1.2 Características

JDBC es independiente de la plataforma al estar escrito en Java. JDBC es una API de bajo nivel ya que hace llamadas SQL directas(14), Sun desea que JDBC pueda ser llamado desde otra API de más alto nivel que pueda simplificar la labor del programador, aunque la utilización de JDBC es sencilla y potente. Se tiene noticia de que ya existen diversos proyectos en marcha que intentan crear estas APIs de alto nivel. Aquí el término API hace referencia a un conjunto de clases e interfaces. Una forma de ver las características de JDBC es enfrentarlo con otro API que permita también el acceso a bases de datos, uno de los más usados y extendidos es el API de Microsoft ODBC (Open DataBase Connectivity).

ODBC permite la conexión a casi todo tipo de bases de datos en casi todas las plataformas, por lo tanto ¿porqué no podemos simplemente usar ODBC desde Java?. El hecho es que se puede utilizar ODBC desde Java, pero a través de JDBC con lo que se denomina el puente JDBC−ODBC (JDBC−ODBC Bridge, desarrollado por Sun e Intersolv), que se tratará más adelante. En este momento la pregunta se transforma en ¿para qué necesitamos entonces JDBC?, hay varias respuestas para esta pregunta:

1. Usar ODBC directamente desde Java no es apropiado ya que usa un interfaz en C, y las llamadas desde Java a código nativo de C pueden ocasionar diversos problemas de seguridad y en la portabilidad de las aplicaciones.

2. Una traducción literal del API de ODBC escrito en C no es adecuado, ya que,

Java no utiliza punteros y sin embargo ODBC hace un uso bastante frecuente de ellos. Se puede considerar que JDBC es una traducción de ODBC a un interfaz de programación orientada a objetos que es natural para los programadores de Java.

3. ODBC es más complicado de aprender, mezcla características sencillas con avanzadas y tiene opciones complejas incluso para las consultas más sencillas.

4. JDBC es necesario para disponer de una solución Java pura (100% pure Java). Cuando se utiliza ODBC el gestor de drivers y los drivers deben ser instalados manualmente en cada máquina cliente. Mientras que JDBC está escrito completamente en Java y su código se

instala automáticamente.

3.1.3 Tipos de Drivers JDBC

Los drivers nos permiten conectarnos con una base de datos determinada(15). Existen cuatro tipos de drivers JDBC, cada tipo presenta una filosofía de trabajo diferente, a continuación se pasa a comentar cada uno de los drivers:

1. **El puente JDBC−ODBC**: Este driver conecta Java con el ODBC de Microsoft vía métodos nativos. Este driver fue desarrollado por JavaSoft, y forma parte del JDK para Windows.

Ventajas: Dada la gran popularidad de ODBC, casi todas las bases de datos cuentan actualmente con un driver ODBC. Por esta razón, la conectividad con Java se puede hacer inmediatamente sin hacer ningún desarrollo adicional.
Desventas: El uso de métodos nativos implica que esta conectividad no se puede usar en applets. Además la conexión se vuelve más lenta por usar ODBC.

2. Un driver mixto Java − API Nativo: Este driver convierte las llamadas JDBC en llamados a métodos nátivos del cliente de la base de datos. Este driver típicamente es desarrollado por el mismo fabricante de la base de datos.
Ventajas: Casi todos los motores tienen un cliente ya probado y depurado, que hace que el desarrollo de estos driver pueda ser implementado fácilmente por todos los proveedores.
Desventajas: Tampoco se puede usar en applets debido al llamado a métodos nativos.

- **Un driver Gateway**: Este driver traduce los llamados JDBC en un protocolo independiente del motor, que luego es traducido a en protocolo específico de la base de datos en un servidor intermedio. Este servidor (gateway) puede conectar los clientes Java con muchos proveedores de bases de datos diferentes usando un sólo driver en el cliente.
  Ventajas: Esta es la alternativa más flexible para conectarse con diferentes motores de bases de datos simultáneamente. Además, como el driver está totalmente desarrollado en Java, puede funcionar en applets y aplicaciones.
  Desventajas: El uso de una máquina intermedia hace necesariamente más lenta la transferencia de datos que usando drivers tipo 2 ó 4.
  Nota: La versión Cliente/Servidor de JBuilder trae un driver de este tipo llamado "Borland DataGateway", que brinda conexión nativa de muy alto desempeño con los siguientes motores de bases de datos: Oracle, InterBase, Sysbase, Informix, DB2, MS SQL, Access, FoxPro, dBase y Páradox. Este driver también soporta conexiones ODBC.

4Un driver JDBC 100% Java Puro con Protocolo Nativo: Este driver convierte todas las llamadas JDBC a un protocolo de red específico al manejador de bases de datos. Esta es la solución ideal para casi todas las situaciones.
Ventajas: La eliminación de intermediarios, hace que el rendimiento accesando la base de datos sea óptimo. Al ser un driver sólo Java, puede funcionar tanto en applets como en aplicaciones.
Desventajas: Es el tipo de driver más difícil de implementar por los proveedores de bases de datos, porque involucra desarrollo en el cliente y en el servidor. Muchos proveedores no ofrecen aún esta forma de conectividad.

La forma más apropiada de accesar bases de datos en Java es vía JDBC. Para usarlo debemos conseguir un driver JDBC apropiado para el motor. Existen cuatro tipos de drivers JDBC. Los mejores son los tipos 3 (Gateway) y 4 (driver 100% Java con Protocolo Nativo).

3.1.4 Acceso de la API JDBC a las BD

La API JDBC soporta dos modelos distintos de acceso a las BD:

- ◊ Modelo de dos capas.
- ◊ Modelo de tres capas.

3.1.4.1 Modelo de dos capas

En este modelo la aplicación JAVA o el Applet, se conectan directamente con la BD(16). Esto significa que el driver JDBC especifico para conectarse con la BD estara instalado en el sistema local.La BD puede estar en otra maquina y se accede a ella mediante red.Esta configuración tambien se llama Cliente/Servidor.El programa cliente envia instrucciones SQL a la BD, y esta las procesa y envia los resultados de vuelta al usuario.

3.2.4.2 Modelo de tres capas

En este modelo, las instrucciones son enviadas a una capa intermedia que se encarga de enviar las sentencias SQL a la BD.El manejador de BD procesa las sentencias y retorna los resultados a la capa intermedia que se encarga de enviarlos al usuario.

3.2 OBDC(17)

Se escribe una aplicación para acceder a las tablas de una DB de Access, ¿qué ocurrirá si después se quiere que la misma aplicación, y sin reescribir nada, utilice tablas de SQL Server u otra DB cualquiera? La respuesta es sencilla: no funcionará. Nuestra aplicación, diseñada para un motor concreto, no sabrá dialogar con el otro. Evidentemente, si todas las DB funcionaran igual, no tendríamos este problema.... aunque eso no es probable que ocurra nunca.

Pero si hubiera un elemento que por un lado sea siempre igual, y por el otro sea capaz de dialogar con una DB concreta, solo tendríamos que ir cambiando este elemento, y nuestra aplicación siempre funcionaría sin importar lo que hay al otro lado... algo así como ir cambiando las boquillas de una manguera. A esas piezas intercambiables las llamaremos orígenes de datos de ODBC.

OBDC, Open Data Base Conectivity o lo que es lo mismo, conectividad abierta de bases de datos es un intermediario entre bases de datos y aplicaciones, cuya tarea es sostener una conversación de preguntas y respuestas entre dos "sujetos" que no hablan el mismo idioma y que gestionan sus recursos de forma diferente(17). ODBC es entonces un armatoste que alberga controladores. El armatoste sirve para gestionar los controladores, y los controladores son los que saben "hablar" con las bases de datos.

En ODBC no se tiene que hacer gran cosa, es una simple tarea, se llama crear un origen de datos, otros le denominan fuente en vez de origen. Un origen o fuente de datos consiste en el nombre, el controlador y la base de datos. Por ejemplo, si un usuario quiere tener acceso a una base de datos de Access, digamos que se llama Negocio.mdb, desde una hoja de cálculo de Excel para consultar su volumen de ventas por país, este usuario crea un nuevo origen de datos en ODBC llamado Volumen_Ventas (este es, pues, el nombre), después selecciona un controlador para Microsoft Access e indica el archivo de base de datos está en "c:\LaEmpresa\Administración\Negocio.mdb". Eso es básicamente de lo que se trata.

ODBC asegura una conexión continua desde un cliente, servidor o aplicaciones Web.

ODBC proveé una solución completa e independiente para el acceso a datos, porque define estándares para el proceso y acceso físico a las bases de datos. ODBC permite a las aplicaciones cliente desarrollar en una única y común API.

Casi todas las DB actuales tienen un ODBC. Debido a que este elemento impone ciertas limitaciones, ya que no todo lo que la DB sabe hacer es compatible con la aplicación, como velocidad de proceso, tiempos de espera, máxima longitud de registro, número máximo de registros, versión de SQL, etc., está cayendo en desuso a cambio de otras técnicas de programación, pero aún le quedan muchos años de buen servicio.

Todo lo referido aquí funciona con Windows NT Server 4.0 con el Service Pack 4 o superior instalado (actualmente existe el 6). El Option Pack 4 para actualizar el IIS y las extensiones ASP. SQL Server 6.5 y Access 97. Esas otras técnicas de programación antes mencionadas, se utilizan ya en el nuevo Windows 2000, Office 2000 y SQL Server 7.0, que además de ODBC pueden utilizar.... pero esa es otra historia.

Esta es la idea: por un lado el ODBC provee de unas caracteríisticas siempre homogéneas, y por el otro permite distintos controladores que aseguran la conectividad de la aplicación con diferentes bases de datos.

ODBC proveé una solucióm completa e independiente para el acceso a datos, porque define estándares para el proceso y acceso físico a las bases de datos. ODBC permite a las aplicaciones cliente desarrollar en una única y común API.

La tecnología ODBC es utilizada en múltiples plataformas, incluyendo Windows 3.1, Windows NT, OS/2, Macintosh y UNIX, y DBMS´s como DB2, Oracle, SYBASE, INFORMIX, Microsoft SQL Server, DEC, Apple DAL, dBase, Excel, etc.

3.2.1Características

- ◊ La conectividad de datos para aplicaciones Web y aplicaciones server críticas más escalable y confiable. −Connect ODBC 3.5 soporta Microsoft Transaction Server para Oracle 8, SQL Server 7, y los sistemas de bases de datos Sybase Adaptive Server 12.
- ◊ Es el primer y único ODBC driver para conectividad directa desde aplicaciones UNIX hacia Microsoft SQL Server − Este nuevo soporte permite a las organizaciones desarrollar o desplegar sus datos SQL Server para cliente/servidor − y aplicaciones basadas en el Web residiendo sobre UNIX.
- ◊ Cursor scrollable es soportado para todas las bases de datos. Del mismo modo muchas bases de datos y ODBC drivers no soportan cursores de éste tipo. Con Connect ODBC 3.5 usted obtiene soporte para todas sus aplicaciones de ODBC.
- ◊ DataDirect Connect ODBC libera el ODBC driver para aplicaciones en plataforma Irix y bases de datos como Informix y Sybase.

3.2.2 Conexión a un recurso ODBC

A partir de este momento, trabajaremos con nuevos objetos, los cuales nos permiten establecer la comunicación entre el cliente y servidor de bases de datos.

El primero es el Objeto Connection**,** el cual nos permitirá conectarnos a la base de datos, este objeto tiene una serie de colecciones, propiedad y métodos propios que nos permiten manipular las operaciones con la bases de datos.

Luego tenemos el Objeto Recorset (Conjunto de registros), mediante el cual realizaremos las operaciones sobre las tablas de la base de datos.

Ademas se encuentra el Objeto Error, con el cual podremos controlar los eventos no deseados que ocurran sobre un recordset o una conexión.

3.2.3 Campo de aplicación y alcance

La tecnología ODBC proporciona una interfaz común para acceder a bases de datos SQL heterogéneas(18). ODBC se basa en SQL como estándar de acceso a los datos. Esta interfaz proporciona una interesante interoperabilidad, ya que gracias a ella una misma aplicación puede acceder a diferentes SGBDs SQL (Sistema de Gestión de Base de Datos) a través de un conjunto común de instrucciones. Esto permite a los desarrolladores construir y distribuir aplicaciones

cliente/servidor no ligadas específicamete a una base de datos determinada. Cada usuario añade luego el 'driver' correspondiente, el cual se encarga de enlazar la aplicación con el SGBD de su elección. Los drivers aíslan a la aplicación de llamadas específicas para una base de datos determinada, de la misma manera que los drivers de impresora evitan que los procesadores de texto tengan que incorporar comandos específicos de impresión para cada tipo de máquina.

Así pues, ODBC es la especificación de una API para bases de datos, independiente del SGBD y del sistema operativo y, aunque Microsoft presenta la especificación en C, también del lenguaje de programación. ODBC se basa en las especificaciones CLI (Call Level Interface) de ISO y X/Open. ODBC 3.0 implementa completamente estas dos especificaciones públicas (las versiones anteriores de ODBC estaban basadas en versiones preliminares de estas especificaciones pero no las implementaban completamente) y añade algunas prestaciones que normalmente necesitan los desarrolladores como cursores de 'scroll' en pantalla.

Las funciones en la API ODBC son implementadas por los desarrolladores de drivers específicos para un SGBD. Las aplicaciones llaman a las funciones contenidas en estos drivers para acceder a la información de una manera no dependiente de la naturaleza específica de la base de datos.

3.3Conectividad de Bases de Datos de Java (JDBC)

Se considera el primer producto estándar de Java con DBMS, creado y ofrecido por primera vez en marzo de 1996.

Crea una interfaz con un nivel de programación que le permite comunicarse con las bases de datos mediante un concepto similar al de componentes ODBC, el cual se ha convertido en el estándar que se utiliza en computadoras personales o en redes locales.

El estándar de JDBC está basado en un nivel de interfaz con instrucciones SQL X/Open, que es básicamente lo mismo que en ODBC.

Las clases de objetos para iniciar la transacción con la base de datos, están escritas completamente en Java, lo cual permite mantener la seguridad, robustez y portabilidad de este ambiente.

El puente JDBC−ODBC manipula la traducción de llamadas JDBC a aquellas que puedan ser entendidas por el cliente ODBC a un nivel de lenguaje C.

3.4JDBC vs. ODBC

ODBC(Open DataBase Connectivity:Conectividad abierta de BD) es la interface para conectarse con BD relacionales mas usada por los programadores de aplicaciones(19).Nos preguntaremos que significado tiene entonces JDBC si ya existe una interficie popular que supuestamente hace lo mismo.La respuesta es que usaremos JDBC por diferentes razones:

- ◊ ODBC usa una interface escrita con el lenguaje de programación C.Por lo tanto como que C no es un lenguaje portable las aplicaciones JAVA perderian tambien automaticamente su portabilidad.
- ◊ ODBC se ha de instalar manualmente en cada maquina, en cambio los drivers de JDBC como estan escritos en JAVA son automaticamente instalables, portables y seguros.

Hay que decir tambien, que existen drivers puente entre JDBC−ODBC.Estos drivers traducen las llamadas de JDBC a ODBC permitiendo comunicarse con BD propietarias que no tienen ni idea de que existe JAVA.De esta manera por ejemplo podemos trabajar con una BD Access de Microsoft que usa ODBC, con el lenguaje JAVA.

3.5 PUENTE JDBC/OBDC

JDBC es la API estándar de acceso a Bases de Datos con Java, y se incluye con el Kit de Desarrollo de Java (JDK) a partir de la versión 1.1. Sun optó por crear una nueva API, en lugar de utilizar APIs ya existentes, como ODBC, con la intención de obviar los problemas que presenta el uso desde Java de estas APIs, que suelen ser de muy bajo nivel y utilizar características no soportadas directamente por Java, como punteros, etc. Aunque el nivel de abstracción al que trabaja JDBC es alto en comparación, por ejemplo, con ODBC, la intención de Sun es que sea la base de partida para crear librerías de más alto nivel, en las que incluso el hecho de que se use SQL para acceder a la información sea invisible.

Para trabajar con JDBC es necesario tener controladores (drivers) que permitan acceder a las distintas Bases de Datos: cada vez hay más controladores nativos JDBC. Sin embargo, ODBC es hoy en día la API más popular para acceso a Bases de Datos: Sun admite este hecho, por lo que, en colaboración con Intersolv (uno de principales proveedores de drivers ODBC) ha diseñado un puente que permite utilizar la API de JDBC en combinación con controladores ODBC. Un último detalle: algunos fabricantes, como Microsoft, ofrecen sus propias APIs, en lugar de JDBC, como RDO, etc. Aunque estas APIs pueden ser muy eficientes, y perfectamente utilizables con Java, su uso requiere tener muy claras las consecuencias, sobre todo la pérdida de portabilidad.

JDBC−ODBC bridge plus ODBC driver (tipo 1): este driver fue desarrollado entre Sun e Intersolv y permite al programador acceder a fuentes de datos ODBC existentes mediante JDBC. El JDBC−ODBC Bridge (puente JDBC−ODBC) implementa operaciones JDBC traduciéndolas a operaciones ODBC, se encuentra dentro del paquete sun.jdbc.odbc (que se encuentra incluido dentro del JDK a partir de la versión 1.1) y contiene librerías nativas para acceder a ODBC. Se debe señalar que en cada máquina cliente que utilice el driver es necesaria una configuración previa, es decir, deberemos definir la fuente de datos utilizando para ello el gestor de drivers ODBC que los podemos encontrar dentro del Panel de Control de Windows. Debido a esta configuración en las máquinas clientes, este tipo de driver no es adecuado para utilizarlo dentro de applets, su utilización está más encaminada a aplicaciones

Java dentro de pequeñas intranets en las que la instalaciones en los clientes no sean un gran problema o para aplicaciones Java que pueden actuar de capa intermedia dentro de un modelo de tres capas, como se veía anteriormente. Además debido a que el puente JDBC−ODBC contiene parte de código nativo, no es posibible utilizarlos dentro de applets, debido a las restricciones de seguridad de éstos. Desde Javasoft (una división de Sun) nos sugieren que el uso del puente JDBC−ODBC se limite al tiempo que tarden en aparecer drivers JDBC de otro tipo para la base de datos a la que queremos acceder, esta advertencia es debida a que muchos drivers ODBC no están programados teniendo en cuanta la programación multiproceso (multithreading), por lo que se pueden producir problemas a la hora de utilizar el puente JDBC−ODBC.

3.6 Preparación del entorno de desarrollo

A continuación se va abordar la creación de una fuente de datos ODBC, a través de la cuál accederemos a una Base de Datos Interbase. Hemos escogido este modo de acceso en lugar de utilizar un driver JDBC nativo para Interbase porque ello nos permitirá mostrar el uso de drivers ODBC: de este modo, aquél que no disponga de Interbase, podrá crear una Base de Datos Access, BTrieve o de cualquier otro tipo para el que sí tenga un driver ODBC instalado en su sistema, y utilizar el código fuente incluido con el diskette que acompaña a la revista.

El **Listado A** muestra el SQL utilizado para crear nuestra pequeña Base de Datos de ejemplo, incluida en el diskette. Nótese que el campo ID de la tabla CONTACTOS es del tipo contador o autoincremento, y requiere de la definición de un trigger y un contador en Interbase: otros sistemas de Base de Datos pueden tener soporte directo para este tipo de campo, lo que hará innecesario parte del código SQL del **Listado A**.

```
/* No se hace commit automatico de sentencias de definicion de datos, DDL */

SET AUTODDL OFF;

/********** Creacion de la base de datos **********/

/* Necesario USER y PASSWORD si no se ha hecho login previo */

CONNECT "h:\work\artículos\jdbc\1\fuente\agenda.gdb"

USER "SYSDBA" PASSWORD "masterkey";

/* Eliminación de la Base de datos */

DROP DATABASE;

CREATE DATABASE

"h:\work\artículos\jdbc\1\fuente\agenda.gdb"

USER "SYSDBA" PASSWORD "masterkey";

CONNECT "h:\work\artículos\jdbc\1\fuente\agenda.gdb"

USER "SYSDBA" PASSWORD "masterkey";
```

```
/***** Creación de la tabla de Contactos *****/
CREATE TABLE CONTACTOS (
ID INTEGER NOT NULL,
NOMBRE CHAR(30) NOT NULL,
EMPRESA CHAR(30),
CARGO CHAR(20),
DIRECCION CHAR(40),
NOTAS VARCHAR(150),
PRIMARY KEY( ID )
);
CREATE INDEX I_NOMBRE ON CONTACTOS( NOMBRE );
/* Creamos un contador para CONTACTOS.ID */
CREATE GENERATOR CONTACTOS_ID_Gen;
SET GENERATOR CONTACTOS_ID_Gen TO 0;
/* Creamos trigger para insertar código del contador en CONTACTOS */
SET TERM !! ;
CREATE TRIGGER Set_CONTACTOS_ID FOR CONTACTOS
BEFORE INSERT AS
BEGIN
New.ID = GEN_ID( CONTACTOS_ID_Gen, 1);
END !!
SET TERM ; !!
/**** Creación de la tabla de teléfonos de contactos ****/
CREATE TABLE TELEFONOS (
CONTACTO INTEGER NOT NULL,
TELEFONO CHAR(14) NOT NULL,
```

```
NOTAS VARCHAR(50),

PRIMARY KEY( CONTACTO, TELEFONO )

);

/* Se hace COMMIT */

EXIT;
```

Listado A: SQL utilizado para crear la Base de Datos de ejemplo, AGENDA.GDB, con Interbase.

El primer paso, evidentemente, será crear la Base de Datos, que en nuestro caso se llama AGENDA.GDB. Una vez hecho esto, debemos crear una fuente de datos ODBC: para ello, se debe escoger en el Panel de Control el icono "32bit ODBC", correspondiente al gestor de fuentes de datos ODBC, y hacer click sobre el botón "Add..." de la página "User DSN", con lo que aparecerá la ventana de la **Figura A**.

Figura A: Ventana del gestor de fuentes de datos ODBC utilizada para añadir nuevas fuentes de datos.

Hecho esto, queda configurar la fuente de datos, indicando el nombre que le vamos a dar (que utilizaremos a la hora de conectarnos a la Base de Datos), así como otros datos. La **Figura B** muestra la ventana de configuración de una fuente de datos Interbase, utilizando un driver de Visigenic.

Data Source Name es el nombre que utilizaremos para identificar la fuente de datos, y en nuestro caso será *JDBC_AGENDA*. *Description* es un texto explicativo optativo, y *Database* especifica cómo se llama y dónde se encuentra la Base de Datos que vamos a acceder a través de esta fuente de datos. Para poder utilizar los programas de ejemplo, deberemos introducir aquí el directorio donde copiemos los ejemplos, seguido del nombre de la Base de Datos, AGENDA.GDB. En mi caso, *Database* es *h:\work\artículos\jdbc\1\fuente\AGENDA.GDB*.

El hecho de que para acceder una Base de Datos mediante ODBC se utilice el nombre de la fuente de datos, en lugar del nombre de la Base de Datos, nos permite cambiar la ubicación e incluso el nombre de la misma sin tener que modificar el código fuente de nuestros programas, ya que estos utilizarán el nombre de la fuente de datos para identificarla.

En cuanto a los demás parámetros de la fuente de datos, se debe escoger como *Network Protocol* la opción *<local>*. El nombre de usuario (*User name*) y la contraseña (*Password*), se indicarán manualmente desde dentro de nuestro programa: no suele ser aconsejable especificarlo en la misma fuente de datos, dado que esto daría acceso a la Base de Datos a cualquier programa que acceda a la misma a través de la misma.

Figura B: Ventana de configuración de una fuente de datos ODBC.

Como último paso, no debemos olvidar arrancar el servidor de Base de Datos, si utilizamos un gestor que incluye uno, como sucede con Interbase.

3.6.1 El primer programa con JDBC

Como es lógico, la API JDBC incluye varias clases que se deben utilizar para conseguir acceso a una Base de Datos. La **Tabla A** muestra la lista de clases e interfaces más importantes que JDBC ofrece, junto con una breve descripción. Estas clases se encuentran en el paquete *java.sql*.

Clase/Interface	**Descripción**
Driver	Permite conectarse a una Base de Datos: cada gestor de Base de Datos requiere un Driver distinto.
DriverManager	Permite gestionar todos los Drivers instalados en el sistema.
DriverPropertyInfo	Proporciona diversa información acerca de un Driver.
Connection	Representa una conexión con una Base de Datos. Una aplicación puede tener más de una conexión a más de una Base de Datos.
DatabaseMetadata	Proporciona información acerca de una Base de Datos, como las tablas que contiene, etc.
Statement	Permite ejecutar sentencias SQL sin parámetros.
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada.
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada y salida, típicamente procedimientos almacenados.
ResultSet	Contiene las filas o registros obtenidos al ejecutar un SELECT.
ResultSetMetadata	Permite obtener información sobre un ResultSet, como el número de columnas, sus nombres, etc.

Tabla A: Clases e interfaces definidos por JDBC.

El mejor modo de comenzar a estudiar la API de JDBC es empezar con un pequeño programa. El **Listado B** corresponde a un programa que carga el driver utilizado para conectarnos a bases de datos utilizando controladores ODBC.

```
// Importamos el paquete que da soporte a JDBC, java.sql

import java.sql.*;

import java.io.*;

class jdbc1 {

public static void main( String[] args ) {

try {

// Accederemos a la fuente de datos

// ODBC llamada JDBC_AGENDA

String urlBD = "jdbc:odbc:JDBC_AGENDA";
```

```
String usuarioBD = "SYSDBA";

String passwordBD = "masterkey";

// Cargamos la clase que implementa

// el puente JDBC=>ODBC

Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );

// Establecemos una conexion con la Base de Datos

System.out.println( "Estableciendo conexión con " +

urlBD + "..." );

Connection conexion = DriverManager.getConnection(

urlBD, usuarioBD, passwordBD );

System.out.println( "Conexión establecida." );

conexion.close();

System.out.println("Conexión a " + urlBD " cerrada.");

}

catch( Exception ex ) {

System.out.println( "Se produjo un error." );

}

}

}
```

Listado B: Código básico para llevar a cabo la conexión con una Base de Datos a través de ODBC.

Establecer una conexión con una Base de Datos mediante JDBC es sencillo: en primer lugar, registramos el *Driver* a utilizar (que en nuestro caso es el puente JDBC/ODBC), mediante el código

```
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
```

Si no se registra este driver, se producirá un error intentar la conexión. A continuación, se lleva a cabo la conexión a la Base de Datos mediante el código

```
Connection conexion = DriverManager.getConnection( urlBD, usuarioBD, passwordBD );
```

La clase *DriverManager* gestiona los *Drivers* registrados en el sistema: al llamar a *getConnection*, recorre la lista de *Drivers* cargados hasta encontrar uno que sea capaz de gestionar la petición especificada por urlBD, que en nuestro ejemplo es "jdbc:odbc:JDBC_AGENDA". Los parámetros usuarioBD y passwordBD corresponden al nombre del usuario y su contraseña, necesarios la mayor parte de las veces para acceder a cualquier Base de Datos.

Cómo se especifica la Base de Datos a utilizar depende del *Driver* utilizado: en nuestro caso, en que utilizamos el puente ODBC/JDBC, todas las peticiones serán de la forma "jdbc:odbc:NOMBRE_FUENTE_DATOS".

La cadena utilizada para indicar la Base de Datos siempre tiene tres partes, separadas por el carácter ":". La primera parte es siempre "jdbc". La segunda parte indica el *subprotocolo*, y depende del Sistema de Gestión de Base de Datos utilizado: en nuestro caso, es "odbc", pero para *SQLAnyware*, por ejemplo, es "dbaw". La tercera parte identifica la Base de Datos concreta a la que nos deseamos conectar, en nuestro caso la especificada por la fuente de datos ODBC llamada "JDBC_AGENDA". Aquí también se pueden incluir diversos parámetros necesarios para establecer la conexión, o cualquier otra información que el fabricante del SGBD indique.

Siguiendo con el **Listado B**, a continuación se establece una conexión con la Base de Datos, mediante el código

```
Connection conexion = controlador.connect( urlBD, usuarioBD, passwordBD );
```

Acto seguido cerramos la conexión, con

```
conexión.close();
```

Es conveniente cerrar las conexiones a Bases de Datos tan pronto como dejen de utilizarse, para liberar recursos rápidamente. Sin embargo, ha de tenerse en cuenta que establecer una conexión es una operación lenta, por lo que tampoco se debe estar abriendo y cerrando conexiones con frecuencia.

3.6.2 Consultas en JDBC

Un programa que realice una consulta y quiera mostrar el resultado de la misma requerirá del uso de varias clases: la priemra es *DriverManager*, que permitirá llevar a cabo una conexión con una Base de Datos, conexión que se representa mediante un objeto que soporta el interface *Connection*. También será necesario además ejecutar una sentencia SELECT para llevar a cabo la consulta, que se representará por un objeto que soporte el interface *Statement* (o *PreparedStatement*, o *CallableStatement*, que estudiaremos más adelante). Una sentencia SELECT puede devolver diversos registros o filas: esta información es accesible mediante un objeto que soporte el interface *ResultSet*.

El **Listado C** muestra el código fuente correspondiente a un pequeño programa que muestra los nombres de todos los contactos que hemos almacenado en la Base de Datos AGENDA.GDB, y que utiliza las clases anteriormente mencionadas.

```
import java.sql.*;

import java.io.*;
```

```
class jdbc2 {

public static void main( String[] args ) {

try {

// Accederemos al alias ODBC llamado JDBC_DEMO

String urlBD = "jdbc:odbc:JDBC_AGENDA";

String usuarioBD = "SYSDBA";

String passwordBD = "masterkey";

// Cargamos el puente JDBC=>ODBC

Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");

// Intentamos conectarnos a la base de datos JDBC_DEMO

Connection conexion = DriverManager.getConnection (

urlBD, usuarioBD, passwordBD );

// Creamos una sentencia SQL

Statement select = conexion.createStatement();

// Ejecutamos una sentencia SELECT

ResultSet resultadoSelect = select.executeQuery(

"SELECT * FROM CONTACTOS ORDER BY NOMBRE" );

// Imprimimos el nombre de cada contacto encontrado

// por el SELECT

System.out.println( "NOMBRE" );

System.out.println( "------" );

int col = resultadoSelect.findColumn( "NOMBRE" );

boolean seguir = resultadoSelect.next();

// Mientras queden registros...

while( seguir ) {

System.out.println(
```

```
resultadoSelect.getString( col ) );
seguir = resultadoSelect.next();
};
// Liberamos recursos rápidamente
resultadoSelect.close();
select.close();
conexion.close();
}
catch( SQLException ex ) {
// Mostramos toda la informaci¢n sobre
// el error disponible
System.out.println( "Error: SQLException" );
while (ex != null) {
System.out.println ("SQLState: " +
ex.getSQLState ());
System.out.println ("Mensaje: " +
ex.getMessage ());
System.out.println ("Vendedor: " +
ex.getErrorCode ());
ex = ex.getNextException();
System.out.println ("");
}
}
catch( Exception ex ) {
System.out.println( "Se produjo un error inesperado" );
}
```

```
}

}
```

Listado C: Ejecución de una sentencia SELECT e impresión de los resultados con java.

Echemos un vistazo al código en el **Listado C**. El código para registrar el controlador (*Driver*) a utilizar es el mismo del **Listado B**. A continuación se crea una sentencia SQL (*Statement*), y se ejecuta, obteniendo seguidamente el resultado de la misma (*ResultSet*), para lo que se utiliza el siguiente código:

```
// Creamos una sentencia SQL

Statement select = conexion.createStatement();

// Ejecutamos una sentencia SELECT

ResultSet resultadoSelect = select.executeQuery(

"SELECT * FROM CONTACTOS ORDER BY NOMBRE" );
```

Además del método *executeQuery*, utilizado para ejecutar sentencias SELECT, *Statement* también proporciona otros métodos: *executeUpdate* ejecuta una sentencia UPDATE, DELETE, INSERT o cualquier otra sentencia SQL que no devuelva un conjunto de registros, y retorna el número de registros afectados por la sentencia (o −1 si no los hubo). El método *getResultSet* devuelve el *ResultSet* de la sentencia, si lo tiene, mientras que *getUpdateCount* devuelve el mismo valor que *executeUpdate*.

Es posible limitar el número máximo de registros devuelto al hacer un *executeQuery* mediante *setMaxRows*, y averiguar dicho número mediante *getMaxRows*. Es posible también obtener el último aviso generado al ejecutar una sentencia, mediante *getWarning*, así como limitar el tiempo en segundos que el controlador esperará hasta que el SGBD devuelva un resultado, mediante *setQueryTimeout*. Por último, el método *close* libera los recursos asociados a la sentencia.

El interface *ResultSet* es el que encapsula el resultado de una sentencia SELECT. Para recuperar la información es necesario acceder a las distintas columnas (o campos), y recuperarla mediante una serie de métodos *getString*, *getFloat*, *getInt*, etc. Al utilizar estos métodos debe indicarse el número correspondiente a la columna que estamos accediendo: si lo desconocemos, podemos averiguar el número correspondiente a una columna, dado el nombre, mediante *findColumn*. Por último, dado que un *ResultSet* puede contener más de un registro, para ir avanzando por la lista de registros que contiene deberemos utilizar el método *next*, que devuelve un valor booleano indicando si existe otro registro delante del actual. El uso de los métodos anteriores se ilustra en el **Listado C**, que recorre la lista de contactos en la Base de Datos, obteniendo el valor del campo (o columna) "NOMBRE", e imprimiéndolo, mediante un bucle en el que se llama repetidamente al método *next* y a *getString*.

Además de estos métodos, *ResultSet* también cuenta con *getWarnings*, que devuelve el primer aviso obtenido al manipular los registros, así como *wasNull*, que indica si el contenido de la última columna accedida es un NULL SQL. Por último, el método *close* libera los recursos asociados al *ResultSet*.

Excepción	Descripción
SQLException	Error SQL.
SQLWarning	Advertencia SQL.
DataTruncation	Producida cuando se truncan datos inesperadamente, por ejemplo al intentar almacenar un texto demasiado largo en un campo.

Tabla B: Excepciones que puede generar la API de JDBC

Al utilizar la API JDBC es posible obtener diversos errores debido a que se ha escrito incorrectamente una sentencia SQL, a que no se puede establecer una conexión con la Base de Datos por cualquier problema, etc.

El paquete *java.sql* proporciona tres nuevas excepciones, listadas en la **Tabla A**. En el **Listado C** se muestra un ejemplo de uso de las excepciones del tipo *SQLException*: es posible que se encadenen varias excepciones de este tipo, motivo por el que esta clase proporciona el método *getNextException*.

La excepción *SQLWarning* es silenciosa, no se suele elevar: para averiguar si la Base de Datos emitió un aviso se debe utilizar el método *getWarnings* de *Statement* y otras clases, que devuelve excepciones de esta clase. Dado que es posible que se obtengan varios avisos encadenados, esta clase proporciona el método *getNextWarning*, que devuelve el siguiente aviso, si lo hay.

Por último, las excepciones del tipo *DataTruncation* se producen cuando se trunca información inesperadamente, ya sea al leerla o al escribirla.

CAPITULO IV MANEJADORES DE BASE DE DATOS

4.1 ORACLE

El lider del mercado de bases de datos, Oracle, de Oracle Corporation, es la base de datos relacional de uso más extendido. Esta base de datos ofrece varias características y proporciona a los usuarios muchas sutilezas(20).

Una base de datos oracle es un conjunto de datos. Oracle proporciona la capacidad de almacenar y acceser a estos datos de forma consecuente con el modelo definido conocido como relacionarl(relational model). Por ello, oracle se conoce como un gestor de base de datos relacionales(RDBMS: Relational Database Management System). La mayoria de las referencias en una base de datos no ser refiere nadamas a los datos fisicos sino tambien a la combinacion de los objetos fisicos, de memoria y de proceso.

Los datos de una base de datos se almacenan en tablas. Las tablas relacionales se definen mediante sus columnas, y tienen un nombre. Los datos se almacenan como filas de la tabla. Las tablas pueden estar relacionadas entre ellas y pueden utilizarse la base de datos para imponer estas relaciones.

Una base de datos oracle almacena sus datos en archivos. Internamente, existen estructuras de la base de datos que proporcionan una asignación lógica de los datos con los archivos, lo que permite almacenar de forma separada diferente tipos de de datos. Estas divisiones lógicas se llaman espacio de tablas y archivos.

Para acceder a los datos de la Base de Datos, Oracle utiliza un conjunto de procesos de fondo compartido por todos los usuarios, además existen ciertas estructuras de memoria que se utilizan para almacenar los datos de la base de datos mas consultadas recientemente. Estas áreas de memoria ayudan a mejorar el rendimiento de la base de datos al disminuir la cantidad de operaciones de e/s de acceso a los archivos.

Oracle*8*i, la base de datos para Internet, cambia la forma en la que se accede y gestiona la información para satisfacer las demandas de. Con soluciones desarrolladas y desplegadas con Oracle*8*i, cualquier organización puede explotar las oportunidades ilimitadas que ofrece Internet. Oracle*8i* no sólo presenta

las herramientas necesarias para gestionar los tipos de datos que se encuentran en las direcciones web más utilizadas en la actualidad, sino que también ofrece el rendimiento y la escalabilidad que necesarias para soportar estas ubicaciones de gran tamaño y otras aplicaciones críticas. Los desarrolladores se han visto obligados a producir aplicaciones avanzadas, no sólo de forma rápida, sino también con flexibilidad para satisfacer las necesidades de las empresas en constante cambio.

Oracle8i presenta soporte adicional de Java! −el lenguaje de programación de mayor uso en la actualidad− al incluir en el servidor una máquina virtual Java sólida, integrada y escalable. Esto amplía el soporte de Oracle para Java en todos los niveles de aplicaciones, permitiendo desplegar programas Java donde se ejecuten mejor, en el nivel del cliente, en el servidor de aplicaciones o en el servidor de b base de datos, sin tener que recompilar ni modificar el código Java.

4.2 MICROSOFT SQL SERVER

4.2.1 Características

La versión de SQL Server de Microsoft se esta convirtiendo en un serio competidor de las aplicaciones reales de bases de datos . Esta base de datos es en cierta forma menos costosa que el resto de los productos de base de datos y se usa más para el desarrollo de aplicaciones más pequeñas.

La explosiva popularidad de SQL es una de las tendencias más importantes de industria de la informática actual. En los últimos años, SQL se ha convertido en el lenguaje estándar de bases de datos. Alrededor de 100 productos de base de datos ofrecen SQL, ejecutándose en un rango de sistemas informáticos que van desde computadoras personales a los sistemas basados en una computadora central. SQL juega un papel principal en la arquitectura de base de datos de los principales proveedores de computadoras, y esta en el núcleo de la estrategia de base de datos de Microsoft.

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. El nombre SQL es una abreviación de Structured Query Languaje(Lenguaje Estructurado de Consultas). Por razones históricas, SQL se pronuncia generalmente sequel, pero también se utiliza para interactuar con una base de datos. De hecho SQL funciona con un tipo especifico de base de datos, llamado base de datos relacional.

4.2.2Funciones

El nombre lenguaje estructurado de consultas es realmente un termino algo equivocado. En primer lugar, SQL es mucho mas que una herramienta de consulta, aunque ese fue

originalmente su propósito y la recuperación de datos es todavía una de sus funciones mas importantes. SQL se utiliza para controlar todas las funciones que suministra un DBMS a sus usuarios incluyendo:

- ◊ Definición de datos. SQL permite que un usuario defina la estructura u la organización de los datos almacenados, así como las relaciones existentes entre ellos.
- ◊ Recuperación de datos. SQL permite a un su usuario o a un programa recuperar y utilizar los datos almacenados en una base de datos.
- ◊ Manipulación de datos. SQL permite a un usuario o a un programa actualizar la base de datos añadiendo datos nuevos, borrando los viejos y modificando los almacenados previamente.
- ◊ Control de acceso. SQL puede ser utilizado para restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo los datos almacenados contra accesos no autorizados.
- ◊ Compartición de información. SQL es utilizado para coordinar la compartición de datos entre usuarios concurrentes, asegurando que no hay interferencias entre ellos.
- ◊ Integridad de datos. SQL define restricciones de integridad en la base de datos de, protegiéndolas de alteraciones debidas a actualizaciones inconsistentes o fallos del sistema.

♦ **SYBASE SQL**

Sybase SQL Server es uno de los manejadores relacionales más conocido y poderoso del mercado actual. Su capacidad de almacenamiento, ambiente multiusuario y excelente desempeño lo han convertido en uno de los DBMS más exitosos de la historia de la computación, ganándose una enorme reputación en menos de 10 años, gracias a sus innovaciones, confiabilidad y facilidad de uso.

4.3.1 Soluciones IPS (Integrated Product Solutions) de Sybase para la Integración de Datos de la Empresa

Las empresas necesitan obtener ventajas competitivas en el mercado altamente cambiante. Uno de los principales problemas al que se enfrentan los tomadores de decisiones es cómo obtener esta ventaja sin necesidad de invertir una gran cantidad de tiempo y trabajo en procesos costosos para lograr integrar el ambiente tecnológico ya instalado.

Las empresas necesitan reducir los costos operacionales aprovechando los sistemas operacionales existentes − que generalmente incluyen las aplicaciones y los sistemas de bases de datos relacionales que éstas utilizan. Independientemente del motor de base de datos que conduce los sistemas operacionales, la necesidad del negocio de integrar y replicar datos es la misma. Sybase tiene dos soluciones basadas en los tipos de sistemas operacionales en uso. Para clientes con sistemas operacionales que tienen bases de datos Sybase, las soluciones de productos son:

- ◊ Sybase Adaptive Server Enterprise − Una solución de administración de datos escalable y de alta performance.
- ◊ Sybase Adaptive Server Enterprise Replication − Una solución integrada para administración y replicación de datos para ambientes de bases de datos Sybase.
- ◊ Sybase Enterprise Studio − Una solución completa para administrar, replicar e integrar datos entre distintas bases de datos de sistemas operacionales.
- ◊ Sybase Mainframe Connect − Una solución optimizada para integrar los datos de mainframe a la empresa.

♦ **INFORMIX**

La base de datos relacional Informix Software, Inc. Esta disponible tanto para las plataformas UNIX como para Windows NT. Esta base de datos se usa más para aplicaciones en un rango que va desde pequeñas a medianas, pero se puede emplear para proyectos de desarrollo mayores(21).

Informix proporciona un conjunto completo de herramientas de desarrollo de aplicaciones que están totalmente integradas con Informix Dynamic Server, permitiendo la creación de poderosas aplicaciones comerciales para el entorno de bases de datos corporativas. Al usar las herramientas de desarrollo Informix, puede crear rápidamente una amplia gama de aplicaciones, incluyendo la administración dinámica de contenidos preparados para Web y sistemas basados en Java.

- **DB2**

DB2 es el sistema realcional de IBM y es una de las bases de datos realcionales más antifuas en el mercado. Se usa principalmente en sistemas de computadoras mainframe como AS/400 y RS/6000. Esta base de datos proporciona características avanzadas y se usa principalmente para soluciones de base de datos a gran escala(22).

Se dice también ser la base de datos más utilizada en el mundo. Más que Oracle. Más que Microsoft SQL.¿Por qué? Porque es la que mejor responde a las exigencias del e−business de hoy. Detrás del e−business está siempre una base de datos. Con la versión 7.1 de DB2 Universal Database, se reduce a la mitad el tiempo de implantación de las soluciones y la velocidad de búsqueda es ahora 10 veces superior que la de sus competidores DB2 cubre todas las áreas imaginables y todos los productos necesarios para responder a las necesidades de un e−business del futuro. Este paquete contiene todas las herramientas que usted necesita para construir una aplicación de datos para la Web , incluyendo DB2 Extenders, DB2 Connect, Net.Data, Visual Age para Java y Websphere.

4.5.1 Escalabilidad

DB2 Universal Database de IBM es el primer y el único servidor de bases de datos del mundo cuya escalabilidad va desde un computador de bolsillo a una laptop, a un servidor de rango mediano, a clusters de servidores para servidores empresariales masivamente paralelos.... a través de 23 plataformas en 14 lenguajes con una sólida confiabilidad. Plataformas que soportan Windows NT en español, OS/2, y "sabores" populares de UNIX incluyendo Linux, AS/400 y OS/390.

4.6 DBASE

DBASE es una base de datos que a diferencia de las bases de datos de datos SQL, las bases de datos dBase no pueden cambiar su definición. Una vez creado el fichero, la definición de la base de datos es fija. No hay indices que aceleren la busqueda u organicen los datos de distinto modo.

4.7ACCESS

Microsoft Access es un potente sistema de administración de Bases de Datos relacionales. Las Bases de Datos de Access son documentos combinados donde se divide la información por parcelas de objetos especializados.

4.7.1Características

Como elemento primario de información se encuentra las tablas. Normalmente, se crea una tabla para cada tipo de datos de los que componen la Base de Datos. Aunque las tablas se crean como elementos independientes, pueden crearse relaciones entre distintas tablas para recuperar datos de todas ellas mediante una consulta, formulario o informe.

Las consultas se utilizan para localizar y recuperar los datos específicos que cumple unas determinadas condiciones especificadas por el usuario. Las consultas permiten, además, actualizar varios registros al mismo tiempo, así como realizar operaciones de muy diversas índole con los datos almacenados en las tablas.

Por otro lado, los formularios permiten visualizar, introducir y modificar los datos de las tablas de una forma muy sencilla y amena. Al abrir un formulario, Access recupera en él los datos de una o varias tablas y les muestra en un diseño de ficha creado, bien de forma automática por el Asistente para formularios, o manualmente desde el principio por el propio usuario.

Los informes se utilizan primordialmente para presentar, resumir e imprimir los datos en la forma que resulte más apropiada para cada proyecto. Se pueden crear informes que incorporen cálculos basados en los datos de las tablas para mostrar resultados totales o promedios, o bien para generar e imprimir catálogos, listas de nombres y direcciones o etiquetas postales.

4.7.2 Crear una Base de Datos

Access cuenta con dos métodos básicos para crear una Base de Datos.

- ◊ Un método sería crear una Base de Datos en blanco y agregarle más tarde las tablas, formularios, informes y otros objetos. Este es el método más flexible, sin duda, pero requiere que cada elemento de la Base de Datos sea definido por separado.
- ◊ El segundo método consiste en usar un Asistente que crea en una sola operación las tablas, formularios e informes necesarios para el tipo de Base de Datos seleccionado entre una amplia variedad de soluciones típicas. Esta es la forma más sencilla de empezar para crear una Base de Datos.

En ambos casos existe la posibilidad de ampliar y modificar la estructura de la Base de Datos creada en un principio.

CONCLUSION

Hoy en día se puede observar la cantidad de usuarios en el world Wide Web, porque dentro de este ambiente se puede encontrar gran cantidad de información. Es por eso que muchas compañías están usando bases de datos relacionales para manejar la información en sus sitios web.

JDBC tiene la habilidad de Java para integrarse con DBMS comerciales y su naturaleza orientada al manejo de red, es posible crear un ambiente ideal tipo Cliente−Servidor. Al combinar Java con JDBC, se presentan nuevas expectativas para comunicarse con las bases de datos. Se puede entonces tener acceso de una maquina Cliente con tecnología Sun Microsystem a un servidor con tecnología Microsoft en donde se estará almacenada la base de datos a la cual el cliente desea accesar, por medio de Servlets. Para lograr esto Microsoft creó OBDC que es una conectividad abierta de Base de Datos siendo un intermediario entre base de datos y aplicaciones cuya tarea es sostener una conversación de preguntas y respuestas entre dos sujetos que no hablan el mismo idioma y que gestionan sus recursos de forma

diferente. OBDC es entonces como una caja que alberga controladores y los controladores son lo que saben hablar con las bases de datos. Por otro lado Sun Microsystem crea una nueva API ya antes mencionada JDBC para acceso a base de datos. De esta manera podemos acceder a una bases de datos entre dos maquinas con diferentes tecnologías, creándose de esta manera el puente JDBC/OBDC, es decir que desde una maquina cliente que tiene tecnología Microsoft y es la que desea acceder a una base de datos va a poder acceder a datos de un servidor con tecnología Sun Microsystem, permitiendose así la comunicación entre diferentes tecnologías.

BIBLIOGRAFIA

MICHAEL Afergan.

Java Soluciones Instantaneas

Editorial Prentice Hall.

México 1997, pag. 7

ASHTON Hobbs.

Programación para Bases de Datos con JDBC

Editorial Prentice Hall

México 1998, pag. 57 −59

http://docencia.dgsca.unam.mx/cursos/cursos/temarios/bases/sybase.html

http://www.monografias.com/trabajos/protocolotcpip/protocolotcpip.shtml

http://www.javahispano.com/tutoriales/servlets.html

http://www.tyssa.com.ar/soluciones/ebusiness/sun.htm

http://www.ok.cl/cgi/chap0/

http://www.eitig.com/lomasweb/cursillo/ASP.htm

http://members.nbci.com/_XMCM/analista/publica/doc5.htm#1

http://members.nbci.com/_XMCM/analista/publica/doc5.htm#0

http://www.fie.us.es/~mrueda/articulos/seminario−2.html#4

http://www.informix.com/international/caribbean/es/products/appdev.htm

http://www.sybase.es/

http://www.sybase.com.ar/sybasecomar/sybase.com.ar/tqry_contenido_novedad.html?nota=51

http://docencia.dgsca.unam.mx/cursos/cursos/temarios/bases/sybase.html

http://www.br.ibm.com/e−business/espanol/infrastructure/db2/

http://www.map.es/csi/caibi/ibst/estandar/3/ibodbc.htm

http://168.243.1.4/investigacion/bdweb/tecnolog.html

http://www.abits.com.mx/Fabs/Merant/DATADIRECT%20ODBC.htm?tm

http://intranet.adm.ula.ve/cursoasp/sesion5.htm

http://www.audisoft.com/soporte/tips/tipjava01.htm

http://sestud.uv.es/manual.esp/gestion3.htm

(1) http://www.osmosislatina.com/aplicaciones/servidor_web.htm

(2)

(3) http://www.ok.cl/cgi/chapo

(4) http://www.javahispano.com/tutoriales/servlets.html

(5) Michel Afergan Java Soluciones Instantáneas 2ª edición Edit. Prentice Hall México 1997 Pag. 8.

(6) http://www.eitig.com/lomasweb/cursillo/ASP.htm

(7) **http://members.nbci.com/_XMCM/analista/publica/doc5.htm#1**

(8)http://members.nbci.com/_XMCM/analista/publica/doc5.htm#2

(10) Obr. Cit. Pag 19 Capitulo II

(11) http://www.fie.us.es/~mrueda/articulos/seminario−2.html#4

(12) http://monografias.com/trabajos/protocolotcpip/protocolotcpip.shtml

(13) http://java.programación .net/jdbc1.html

(14) Obr.Cit. Pag. 34 Capitulo III

(15) http://industry.java.sun.com/products/jdbc/drivers

(16)Obr. Cit. Pag 34 Capitulo III

(17) **http://skyscraper.fortunecity.com/rofl/434/articulos/odbc/odbc.htm**

(18) http://www.map.es/csi/caibi/ibst/estandar/3/ibodbc.htm

(19) Obr. Cit. Pagina 34 Capitulo III

(20) ASHTON Hobbs Programación para bases de datos con JDBC 1ª edicion Editoria

Prentice Hall México 1997 Pag. 58

(21) Obr. Cit. Pag 67 Capitulo III

(22) Obr. Cit. Pag 67 Capitulo III

3

(1)http://www.osmosislatina.com/lenguajes/perl/basico.htm

(9)Obr. Cit. Pag. 18 Capítulo II

(5) http://skyscraper.fortunecity.com/rofl/434/articulos/odbc/odbc.htm

Obr. Cit. Pag 16 Capitulo I

(11) http://www.fie.us.es/~mrueda/articulos/seminario−2.html#4

(13) http://java.programacion.net/jdbc1.html

(22) Obr. Cit. Pag 67 Capitulo III