Sistemas y Aplicaciones Informáticas

Tema 39. Desarrollo de Aplicaciones mediante Bases de Datos Relacionales.

1. Ámbito de docencia.

- Sistemas informáticos monousuario y multiusuario (ASI 1).
- Sistemas informáticos multiusuario y en red (DAI 1).
- Instalación y mantenimiento de equipos y sistemas informáticos (ESI 1).

2. Desarrollo de aplicaciones con lenguajes de alto nivel.

2.1. Lenguajes de alto nivel y bases de datos. Tipos de comunicación.

- Una aplicación es un conjunto de programas informáticos que realizan una tarea. Una aplicación de base de datos es una aplicación que utiliza un sistema gestor de base de datos (SGBD) para consultar, introducir o modificar datos de una base de datos. La mayoría de los lenguajes de alto nivel permiten incluir instrucciones y objetos para procesar los datos de la base de datos.
- Hay varios tipos de comunicación entre un lenguaje de alto nivel y una base de datos:
  - *Acceso directo*. El lenguaje y el entorno de programación reside en el propio SGBD. Es el caso del lenguaje Pro*C incluido con Oracle, desde el cual se accede directamente a la base de datos. El código sólo es ejecutable por el SGBD para el que se realiza la aplicación.
  - *Acceso mediante un controlador intermedio*. Existe un software (controlador) que se comunica con el SGBD utilizando su lenguaje particular. La comunicación entre el lenguaje de programación y el controlador es la misma para todas las bases de datos. Habrá un controlador para cada SGBD que se utilice, pero la aplicación será la misma para todos.
  - *Acceso mediante máquina virtual*. Existe un software (máquina virtual) que emula un SGBD con un acceso estándar para el cual se requiere un controlador de acceso común. Por su parte, la máquina virtual se encarga de la comunicación con los respectivos controladores de acceso a cada SGBD. Es el caso del acceso a bases de datos ODBC con un driver JDBC.

*2.2. SQL embebido en C (Pro*C).*

- Pro*C es una versión de C que permite incluir sentencias SQL de acceso a bases de datos precediéndolas por 'EXEC SQL'. Un programa Pro*C se compila en dos pasos:
  - El precompilador traduce las sentencias SQL por llamadas a librerías específicas.
  - La salida es un programa en C que se compila y enlaza, obteniendo un programa ejecutable.
- El intercambio de información con el SGBD se realiza a través de variables del lenguaje, denominadas variables anfitrionas, que dentro de las sentencias SQL van precedidas de ':'. Las variables anfitrionas se declaran en una sección especial:

```
EXEC SQL BEGIN DECLARE SECTION;
..declaraciones de variables anfitrionas
EXEC SQL END DECLARE SECTION;
```

  Para las variables se pueden utilizar los tipos de datos de C típicos y el pseudotipo VARCHAR para manipular cadenas. VARCHAR cadena[20] es equivalente a

```
struct{
unsigned short len;
char arr[20];
} cadena;
```

- Aparte de las sentencias SQL básicas, es posible utilizar otros tipos de sentencias:

- *Sentencia de conexión a la BD*. `CONNECT :usuario IDENTIFIED BY :clave;`
- *Cursores*. Cuando una consulta puede devolver más de una fila es necesario declarar un cursor (`DECLARE ncursor CURSOR FOR SELECT …`). Después debe abrirse (`OPEN ncursor`) y leer cada una de las filas recuperadas una por una (`FETCH ncursor INTO :variable1, :variable2, …`), hasta llegar al fin del cursor. Por último es necesario cerrar el cursor (`CLOSE ncursor`) para liberar los recursos utilizados.

– Para el control de los errores producidos durante la ejecución de un programa se utiliza la estructura sqlca (área de comunicación SQL), que se actualiza después de la ejecución de cada sentencia SQL, mediante `#include <sqlca.h>`. Existen dos alternativas.

- *Comprobación explícita de errores*. Implica revisar el valor del campo sqlcode de la estructura sqlca cada vez que se ejecute una sentencia SQL. Los posibles valores de sqlca.sqlcode son cero (sin error), valores positivos (avisos) y negativos (errores).
- *Comprobación implícita de errores*. Mediante `WHENEVER <CONDICION> <ACCION>`, que debe estar antes de las sentencias SQL para las que queremos controlar el error:
  - Las condiciones posibles son `SQLWARNING` (aviso), `SQLERROR` (error) y `NOT FOUND` (filas no encontradas).
  - Las acciones posibles son `CONTINUE`, `DO sentencia`, `DO BREAK`, `DO CONTINUE`, `GOTO etiqueta` y `STOP`.

– Existen tres métodos para definir sentencias SQL dinámicas:

- No consultas no, no variables anfitrionas.

```
Cadena = “insert into mitabla values(‘test’)”;
EXEC SQL EXECUTE IMMEDIATE :cadena;
```

- No consultas, número conocido de variables anfitrionas.

```
Cadena = “delete from emp where empno = :n”
EXEC SQL PREPARE sent_sql FROM :cadena;
Emp_no =3;
EXEC SQL EXECUTE sent_sql USING :emp_no;
```

- Consultas, número conocido de elementos a seleccionar y de variables anfitrionas.

```
Consulta =“select idemp from emp where sal < :salario”;
EXEC SQL PREPARE sent_sql FROM :consulta;
EXEC SQL DECLARE mi_cursor FOR sent_sql;
EXEC SQL OPEN mi_cursor USING :mi_salario;
EXEC SQL FETCH mi_cursor INTO :no_emp;
EXEC SQL CLOSE mi_cursor;
```

2.3. Controladores de acceso a bases de datos.

– **ODBC (Open Database Connectivity)**. Se trata de uno de los interfaces más populares. Ideado por Microsoft en 1989 para permitir el acceso estándar a bases de datos en diferentes SGBD simultáneamente. Cada SGBD necesita su controlador específico para trasladar las llamadas del ODBC en llamadas específicas del sistema gestor de bases de datos.

- **JDBC (Java Database Connectivity)**. Es la respuesta del lenguaje Java a la tecnología ODBC. Es una biblioteca de clases que permite la conexión mediante Java con bases de datos que soporten SQL. El controlador correspondiente es el encargado de interactuar con el SGBD.
- **DAO (Data Access Objects)**. Permite el acceso a las bases de datos utilizando objetos. En realidad es una interfaz que se coloca entre ODBC y el código del programa para permitir que el programa utilice objetos de acceso a la base de datos, lo que permite acceder a la base de datos utilizando programación orientada a objetos. Es una interfaz obsoleta.
- **OLE DB (Object Link Embedding for Databases)**. Se trata de un tipo de controlador creado en 1996 que permite al acceso a bases de datos usando un modelo de acceso por componentes. Admite el acceso a bases de datos de todo tipo y está creado en C++ (utiliza punteros), por lo que no es apto para ser utilizado por lenguajes como Visual Basic.
- **ADO (ActiveX Data Objects)**. Es una interfaz que se sitúa entre el controlador OLE DB (también admite ODBC) y el programa para permitir que la comunicación con éste último sea más sencilla. Los programas Visual Basic anteriores a la plataforma .NET utilizan esta interfaz para acceder a bases de datos.
- **ADO.NET**. Mejora de ADO escrita en XML y pensada para soportar la creación de servicios Web bajo la plataforma .NET de Microsoft.

3. Desarrollo de aplicaciones Web.

3.1. Modelo de tres capas. Esquema de comunicación. HTTP y HTML.

- En la World Wide Web las aplicaciones intercambian la información basándose en un modelo cliente/servidor de tres capas, integrando bases de datos y otras aplicaciones externas:
  - *Capa de presentación (cliente)*. Formada por los navegadores web. Efectúan las peticiones al servidor web, recibiendo e interpretando las respuestas de éste.
  - *Capa de proceso (servidor web)*. Son los que realizan las operaciones de la aplicación web, e interactúan con los servidores de datos.
  - *Capa de datos (servidores de datos)*. Puede estar formada por servidores de base de datos, servidores de correo electrónico, servidores de ficheros, etc.
- El esquema de comunicación está formado por la parte cliente (navegador), la parte servidora (servidor Web) y el canal de comunicación (red de área local o amplia):
  - El navegador realiza una petición al servidor Web (GET para solicitar al servidor que envíe un recurso web al cliente, o POST para enviar al servidor datos recogidos por el cliente mediante un formulario) utilizando el protocolo HTTP (HyperText Transfer Protocol).
  - El servidor Web responde enviando un código de respuesta, una cabecera MIME (código utilizado para que el cliente sepa cómo tratar los datos enviados) y un documento en forma de página codificada en el lenguaje estándar HTML (HyperText Markup Language) junto con los datos solicitados, que es interpretada y mostrada en pantalla por el navegador.

3.2. Páginas dinámicas mediante Common Gateway Interface (CGI).

- Se trata de un estándar que define cómo interaccionan los servidores Web con programas especiales que son ejecutados por el sistema operativo del servidor Web, destinados a procesar la

información que envían los clientes Web y generar documentos de forma dinámica. A estos programas especiales se les llama aplicaciones o módulos CGI, o simplemente CGI.

- El funcionamiento básico es el siguiente:
  - El usuario rellena un formulario HTML y envía los datos. La acción va asociada a la Uniform Resource Locator (URL) de una aplicación CGI, que tiene el siguiente formato:

    ```
    protocolo://[usuario:password@]<maquina>[:puerto]/<ruta>/recurso[?parámetro1=valor1&parámetro1=valor2][#sección]
    ```

  - El navegador codifica los datos (URLEncoded) antes de enviarlos:
    - Los caracteres de letras y números de la tabla ASCII estándar se dejan intactos.
    - Los espacios en blanco son sustituidos por +.
    - Los caracteres especiales y aquellos con un significado especial ('+', '&', '=', ';', '/', '?', '#', etc.) son sustituidos por su valor hexadecimal con el prefijo '%'.
  - El servidor Web lanza la aplicación CGI como un proceso independiente, pasando los datos del formulario a través de variables de entorno, línea de comandos y entrada estándar.
  - La aplicación CGI recoge los datos, accede a la base de datos y devuelve al servidor su resultado a través de la salida estándar. El resultado consta normalmente de una cabecera HTTP simple (normalmente `Content-Type: text/html`) seguida de una línea en blanco (muy importante) y un documento completo del tipo indicado.
  - El servidor completa la información de la cabecera, y el resultado final es enviado al cliente.
- Los programas de la aplicación CGI pueden ser:
  - *Compilados*. Utilizando C, C++, Pascal, etc. Son programas más eficientes, directamente ejecutables, que suelen encontrarse en un directorio /cgi-bin.
  - *Interpretados.* Utilizando Perl, bash, etc. La ejecución de la aplicación la realiza el intérprete de órdenes, que se pone previamente en funcionamiento. Cuando acaba la ejecución del CGI, también acaba la del intérprete. El script suele estar también en /cgi-bin.
- El servidor Web distingue un módulo CGI de cualquier otro fichero convencional:
  - *Por su ubicación (Windows/Linux)*. Residen en directorios virtuales especiales del sistema. El nombre más habitual es /cgi-bin.
  - *Por su extensión (Windows)*. El nombre del fichero suele tener extensión .cgi.
  - *Por sus permisos (Linux)*. El módulo CGI debe tener permisos de lectura y ejecución para todo el mundo. Esto implica un cierto riesgo para la seguridad del sistema.
- La manera más habitual de invocar un módulo CGI es como acción asociada a un formulario HTML. El método de invocación establece la manera en que el CGI recibirá los parámetros:

  ```
  <form action="/cgi-bin/test.cgi" method="POST">
  ```

  - *Método GET*. La información se incluye codificada en la propia URL (a continuación de '?') dentro de la cabecera HTTP. Es el método utilizado por defecto, suele estar limitada a 200 caracteres como máximo. En este caso los datos codificados son accesibles por el CGI a través de la variable de entorno QUERY_STRING (texto de la URL desde la primera '?' hasta el final). Si no se ha utilizado '=', el servidor también hace llegar los datos decodificados a la aplicación CGI como argumentos (en C argc y argv).

- *Método POST*. La información se incluye en el cuerpo del mensaje HTTP. La cantidad de información enviada puede ser mucho mayor. Los datos llegan codificados por la entrada estándar. La variable CONTENT_LENGTH indica el número de bytes de la información.

3.3. Páginas dinámicas mediante Java Servlets.

- Son la alternativa Java a los CGIs. Son aplicaciones Java especiales, que extienden la funcionalidad del servidor Web, dedicadas a:
  - Leer los datos enviados por el cliente.
  - Extraer cualquier información útil incluida en la cabecera HTTP o en el cuerpo del mensaje de petición enviado por el cliente.
  - Generar dinámicamente resultados.
  - Formatear los resultados en un documento HTML.
  - Establecer los parámetros HTTP adecuados incluidos en la cabecera de la respuesta (por ejemplo: el tipo de documento, cookies, etc.).
  - Enviar el documento final al cliente.
- Crean contenido HTML con sentencias 'out.print', y se ejecutan como módulos software en el mismo proceso. La conexión a base de datos se puede realizar a través de JDBC.
- Las ventajas respecto a los CGIs son:
  - *Eficiencia*. Cada petición por parte de un cliente crea un hilo, no un nuevo proceso como ocurría con los CGIs tradicionales.
  - *Potencia*. Son programados en Java, por lo que se puede emplear todas las clases y herramientas disponibles para esta plataforma.
  - *Portabilidad*. Pueden ser utilizados sobre cualquier SO y en la mayoría de servidores Web.
  - *Seguridad*. Controlada por la máquina virtual de Java, la mayoría de problemas de seguridad encontrados en los CGIs no aparecen en los Servlets.
  - *Precio*. Normalmente todo el software necesario es gratis.

3.4. Páginas dinámicas mediante lenguajes de script.

- **Scripts de cliente**. Son pequeños programas que se incrustan en la página HTML con la que el servidor web responde a una petición de un cliente, y que son interpretados por el navegador. Ejemplos de lenguajes de este tipo son JavaScript y VBScript.
- **Scripts de servidor**. Los ficheros HTML almacenados en el servidor Web contienen un código que es interpretado por éste para insertar la información precisa. El navegador nunca ve el contenido de los scripts del lado del servidor, sólo la salida que éstos producen. Cuando un cliente (navegador) solicita al servidor Web una URL que no es una página estática sino un script (reconocido por su extensión especial), el servidor Web utiliza el intérprete de scripts configurado para ejecutar ese código. El resultado es una página Web HTML que el servidor devuelve como respuesta. La ventaja sobre los CGIs es que al estar integrado el módulo intérprete en el servidor, no se ejecuta una copia del programa por cada ejecución:
  - *ASP (Active Server Pages)*. Es el lenguaje de scripts de servidor creado por Microsoft. Separa el lenguaje de script del modelo de objetos. El acceso a base de datos (Oracle, SQL

Server, Access, etc.) es a través de ODBC u OLE DB. Sólo soportado por el servidor Web Microsoft Internet Information Services (IIS).

- *PHP (Hypertext Preprocessor)*. Es un lenguaje interpretado multiplataforma orientado a objetos, que se encuentra embebido en páginas HTML. El módulo PHP del servidor analiza cada instrucción dentro de la página solicitada, y devuelve sólo código HTML. Permite el acceso a las principales bases de datos a través de funciones específicas de PHP escritas en C, y también mediante ODBC. Es un código fuente abierto de libre distribución.
- *JSP (Java Server Pages)*. Permite mezclar HTML con código Java de forma similar a PHP o ASP. Las páginas JSP se traducen en aplicaciones que ejecuta el servidor en cada petición. Puede utilizar todas las funcionalidades que ofrece Java para acceder a bases de datos.

4. Herramientas específicas de los sistemas gestores de bases de datos.

4.1. Lenguajes de cuarta generación (4GL).

- Son lenguajes en los que el programador no especifica el procedimiento a seguir, puesto que el propio lenguaje es capaz de indicar al ordenador cómo debe ejecutar el programa. Entre este tipo de lenguajes se pueden destacar actualmente SQL, Visual Basic y Forms.
- Se caracterizan por lo siguiente:
  - El programador debe definir una serie de parámetros que, mediante la utilización de un conjunto de herramientas del lenguaje, permite generar un programa de aplicación.
  - Son más fáciles de utilizar que los de tercera generación, ya que suelen incluir interfaces gráficos y capacidades de gestión avanzadas, pero consumen muchos más recursos.
  - No requieren una capacitación especial por parte del programador, mejorando su productividad, y están diseñados para resolver problemas específicos.
- En los lenguajes de cuarta generación están incluidos:
  - Lenguajes de presentación, como lenguajes de consultas y generadores de informes.
  - Lenguajes especializados, como hojas de cálculo y lenguajes de bases de datos.
  - Generadores de aplicaciones que definen, actualizan y obtienen datos de una base de datos.
  - Lenguajes utilizados para generar el código de una aplicación.

4.2. Tipos de herramientas de desarrollo de aplicaciones.

4.2.1. Generadores de aplicaciones.

- Un generador de aplicaciones es una herramienta para crear programas que hagan de interfaz entre el usuario y la base de datos. El usuario especifica qué debe hacer el programa y el generador de aplicaciones es quien determina cómo realizar la tarea.
- Constan de procedimientos que realizan las funciones fundamentales que se utilizan en la mayoría de los programas. Estos procedimientos están escritos en un lenguaje de programación de alto nivel y forman una librería de funciones entre las que escoger.

4.2.2. Generadores de formularios.

- Un generador de formularios es una herramienta interactiva que permite crear rápidamente formularios de pantalla para consultar, introducir y modificar la información que contiene una

base de datos. Los generadores de formularios permiten que el usuario defina el aspecto de la pantalla, qué información se debe visualizar y en qué lugar de la pantalla debe visualizarse.

- La mayoría de los generadores de formularios constan de los siguientes módulos:
  - *Navegador de objetos*. Se usa fundamentalmente para desplazarse rápidamente entre las diferentes utilidades que se utilizan para la creación de la aplicación. Permite acceder a los objetos y bibliotecas que se encuentran en el disco y en la base de datos.
  - *Visor de pantallas*. Es el lugar donde se diseña el aspecto y el estilo de cada una de las pantallas de la aplicación, permitiendo modificar el color, el tamaño y tipo de fuente, acceso a los datos y el estilo de la aplicación.
  - *Hoja de propiedades*. Permite establecer los atributos de cada objeto del formulario como el título, el formato de los botones y las acciones a realizar después de un clic.
  - *Editor de código fuente*. Permite introducir código a la aplicación para controlar o completar la funcionalidad de un formulario o de un control.
- Los generadores de formularios son los componentes principales de los generadores de aplicaciones, y contienen muchas clases distintas de elementos:
  - *Disparadores*. Son bloques de código en lenguaje de programación nativo de base de datos que se asocia a otro elemento: un formulario, un bloque de datos o un elemento de un bloque de elementos. El disparador se ejecuta cuando se produce un cierto evento en la base de datos. Los disparadores de una aplicación de formularios contiene código que se añade al código residente en las unidades de programa independientes y en las bibliotecas. Deben definirse las reglas de comportamiento del disparador según la secuencia evento-condición-acción. En SQL un disparador se define de la siguiente manera:

    ```
    CREATE [OR REPLACE] TRIGGER nombre_regla
    {BEFORE|AFTER |INSTEAD OF}
    {evento_ddl | evento_dml | evento_db}
    [[REFERENCING OLD AS nuevonombre [NEW AS otronombre]]
    [FOR EACH ROW [WHEN (condicion]]
    {bloque PL/SQL| CALL nombreprocedimiento}
    ```

    siendo

    ```
    evento_ddl:={CREATE|ALTER|DROP|GRANT|REVOKE}
    evento_dml:={INSERT|DELETE|UPDATE [OF lista de atributos]}
    evento_db:={SERVERERROR|LOGON|LOGOFF|STARTUP|SHUDOWN}
    ```

  - *Bloques de datos*. Es una unidad de construcción intermedia de los formularios. Un bloque de datos puede ser una colección de elementos o una colección de registros, cada uno de los cuales tiene la misma estructura.
  - *Canvas y ventanas*. Un canvas es la base sobre la que se sitúa el texto y los elementos del formulario. Cada elemento hace referencia a un único canvas en su hoja de propiedades. Los elementos de un bloque de datos se pueden dividir entre diferentes canvas.
  - *Avisos*. Son cuadros de diálogo especiales que muestran un icono y varios botones de acción, con el fin de informar de un evento, un error o una confirmación de una acción.

- *Menús emergentes*. Son menús flotantes que emergen cuando se pulsa el botón derecho del ratón en un canvas o en un elemento. Permiten que el usuario ejecute de forma rápida alguna acción relevante de la aplicación relacionada con la posición de ratón.
- *Editores*. Son cuadros de diálogo con un editor de texto sencillo que permite la introducción de líneas de texto en un elemento de texto, con la posibilidad de especificar sus propiedades.

4.2.3. Generadores de informes.

- Un informe es una presentación de datos orientada a páginas. Mientras que un formulario proporciona una herramienta interactiva de manejo de datos, el propósito de un informe es dar un formato legible a los datos sin permitir su manejo.
- Un generador de informes es una herramienta para crear informes a partir de los datos almacenados en la base de datos. Se parece a un lenguaje de consultas en que se permite al usuario hacer preguntas sobre la base de datos y obtener información de ella.
- Sin embargo, en el generador de informes se tiene un mayor control sobre el aspecto de la salida. Se puede dejar que el generador determine automáticamente el aspecto de la salida o se puede diseñar ésta para que tenga el aspecto que desee el usuario final.

4.2.4. Generadores de gráficos.

- Un generador de gráficos es una herramienta para obtener datos de la base de datos y visualizarlos en un gráfico mostrando tendencias y relaciones entre datos. Normalmente se pueden diseñar distintos tipos de gráficos: barras, líneas, etc.
- Un módulo pantalla de una aplicación puede ser uno o más gráficos que se derivan de datos que forman parte de una base de datos, o puede contener cualquier combinación de elementos gráficos con o sin referencia a la base de datos.