IMPLEMENTACIÓN DE MECANISMOS DE SEGURIDAD E INTEGRIDAD EN UN SISTEMA DE BASES DE DATOS (Aplicación de conceptos)

ALONSO TAMAYO ALZATE*, NÉSTOR DARÍO DUQUE MÉNDEZ *

PC: Seguridad en bases de datos, trigger, auditoría, reglas de integridad, autocontrol.

RESUMEN

Los sistemas de bases de datos proveen a usuarios, administradores y diseñadores, elementos que apoyan la integridad y seguridad de la información. Este articulo busca presentar, usando el motor InterBase de la casa Inprise, la facilidad de implementar estos mecanismos en los diseños de los sistemas de información basados en bases de datos.

La mayoría de los Sistemas de Gestión de Bases de Datos SGBD importantes permiten en una forma muy parecida implantar estos controles. Llaves primarias, integridad referencial, dominios, triggers, procedimientos almacenados y auditoría entre otras características son plasmadas en una construcción concreta que puede servir de guía y motivación para usuarios y administradores en la búsqueda de sistemas autocontrolados.

ABSTRACT

Database Systems provide users, managers, and designers with elements that support integrity and data security. The current paper seeks to show, by using an InterBase engine of the Inprise Corporation, the facility of implementing these mechanisms in the systems information design based on database system.

The majority of important Database Manager System DBMS allow for the implementation of these controls in a very similar manner. Primary keys, referential

* Universidad Nacional de Colombia. Sede Manizales. Departamento de Informática y Computación

integrity, domains, triggers, stored procedure, and audit among other characteristics are materialized within a concrete construction, which can serve as a guide and motivation for users and administrators in the search for self-controled systems.

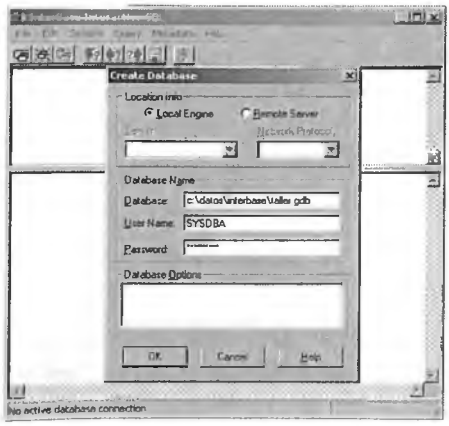
Introducción

En un artículo anterior se hizo referencia a los elementos que poseen los Sistemas de Gestión de Bases de Datos para apoyar la seguridad y/o integridad de la información, pero no basta con saber que existen, es importante ver que su implementación es fácil y requiere de pocos esfuerzos. Esto nos coloca en la necesidad de mostrar en una construcción real como pueden ser incluidos. Para ello trabajaremos con el DBMS InterBase 5.5, potente motor de base de datos de la empresa Inprise. Es seguro que el uso de otros sistemas no generará grandes diferencias en su montaje.

Se mostrará paso a paso la materialización de los conceptos en estudio, a través del desarrollo de un sencillo ejercicio, una pequeña aplicación de venta de productos, donde se incluyen estos elementos de control. Además se muestra que es posible crear nuestra propia bitácora de auditoría de acuerdo a necesidades concretas y conocimiento del sistema. Se tomó como base el paradigma cliente/servidor, a partir del cual se construyó inicialmente la estructura de la parte servidora en el motor de base de datos, luego se desarrolla la aplicación cliente que solicita sus servicios.

En el lado del Servidor

Para iniciar se debe crear la base de datos, donde se almacenarán los datos e información administrativa del propio sistema.



Crear la Base de Datos Taller, en directorio Local C:\Datos\interbase. Este directorio debe ser creado previamente.

Para tener acceso a la ventana para introducción de comandos SQL, Inicio/ Programas/InterBase/InterBase Windows ISQL:

El usuario definido por defecto en InterBase es SYSDBA y su clave de acceso

es masterkey, correspondiente al administrador del sistema de base de datos. Es importante tener presente que cualquier persona que haya usado InterBase conoce este nombre de usuario y la clave, por lo tanto puede ser un hueco de seguridad el no cambiar éstas predefiniciones.

Al crear una nueva Base de Datos, se realiza la conexión a esta. En otro momento, antes de poder realizar alguna acción sobre la misma se debe realizar la conexión: File/Connect to DataBase. Aparecerá una ventana similar a la anterior, donde deben digitarse los datos para acceso a la misma. Si la conexión es correcta en la parte inferior de la ventana de ISQL, aparece el nombre de la Base de Datos activa.

Creación de Tablas con SQL:

Escribir en la ventana superior las siguientes instrucciones SQL:

```
CREATE TABLE CLIENTE(
NumeroCliente  int     NOT NULL,
Nombre         char(30),
Apellido       char(30),
Direccion      char(30),
Ciudad         char(20))
```

Luego se presiona el botón de Execute Query  . Si existe error se informa, en caso contrario la acción ha sido ejecutada.

Se hace lo propio con las demás tablas:

```
CREATE TABLE VENTAS(
NVenta         int     NOT NULL,
Fecha          date,
NCliente       int     NOT NULL,
NItem          int     NOT NULL,
Total          float)
```

Antes de crear la tabla Productos se pueden definir dominios con algunas restricciones, esto es crear un tipo de dato que puede ser utilizado posteriormente como cualquiera de los predefinidos:

```
CREATE DOMAIN PrecioPr
    AS float
    CHECK (VALUE > 1000)
```

```
CREATE TABLE PRODUCTO(
Codigo          int     NOT NULL,
descripción     char(30),
precio          PrecioPr)
```

Las dos anteriores instrucciones determinan que todo producto debe tener un precio superior a 1000.

```
CREATE TABLE ITEM(
NVenta          int NOT NULL,
CodigoP         int NOT NULL,
Cantidad        float)
```

Se debe hacer un cambio en la Tabla Ventas, que lleve a una definición como:

```
TABLE VENTAS (
    nventa      integer NOT NULL,
    fecha       date,
    ncliente    integer NOT NULL,
    total       float);
```

Entonces:

```
Alter table  VENTAS drop Nitem;
Alter table  VENTAS drop precio;
Alter table  VENTAS add Total Float;
```

Es posible definir las **llaves primarias** de la siguiente forma:

```
alter table cliente add PRIMARY KEY (NumeroCliente);
alter table producto add primary key (Codigo);
alter table ventas add primary key (NVenta);
```

Otra forma consiste en borrar inicialmente la tabla y luego crearla como se desee:

```
drop table item;

CREATE TABLE ITEM
(NVenta  int  NOT NULL,
CodigoP  int  NOT NULL,
Cantidad float,
Unitario  float,
PRIMARY KEY (Nventa, CodigoP),
FOREIGN KEY (NVenta)
```

```
REFERENCES ventas(NVenta),
FOREIGN KEY (CodigoP)
REFERENCES Producto(Codigo));

alter table item add subtotal float
```

Se puede observar que se está usando una llave primaria compuesta y se está definiendo la **integridad referencial**. Ahora, si en las tablas referenciadas los atributos de la llave primaria compuesta, no están definidos como llave primaria, no se podrá tener esta relación.

Para que se permita que un atributo se convierta en llave primaria, se debe condicionar que este no sea nulo (Regla de la Entidad).

```
CREATE TABLE historia
(Productoint,
fecha          date,
usuario   char (10),
viejo          float,
porcentaje     float);
```

Esta tabla será usada para almacenar la información al momento que se presente un cambio en el precio de un producto. El mecanismo para adicionar registros es un trigger asociado a la tabla producto, el cual puede ser creado usando un guión SQL, que es un archivo ASCII con una serie de instrucciones, así:

```
CONNECT taller.gdb USER SYSDBA PASSWORD masterkey;
SET TERM !!;

CREATE TRIGGER historia FOR producto
AFTER UPDATE AS
BEGIN
IF (old.precio <> new.precio) THEN
INSERT INTO historia
(producto, fecha,usuario,viejo,porcentaje)
VALUES (old.codigo, "now", USER, old.precio,(new.precio - old.precio) * 100 / old.precio);
END !!

SET TERM ; !!
```

La especificación del trigger permite apreciar lo siguiente:

Actúa para la tabla producto.

Se dispara después de una actualización (after update).

Verifica que la actualización sea un cambio de precio.

Si lo anterior ha sucedido, inserta un registro en la tabla historia, almacenando el código del producto, la fecha en que se realizó la modificación, el usuario que lo hizo, el precio anterior y el porcentaje de cambio). No se debe olvidar que este disparador actuará independientemente, solo basta que se den las condiciones que se han especificado.

Realmente se ha creado, en forma sencilla, una tabla como log de auditoría a la medida, que permite seguir el rastro a las acciones deseadas.

Para facilitar la tarea del auditor en cuanto a recuperación de la información en la tabla historia es posible construir un procedimiento almacenado. El usuario que crea el procedimiento es el propietario del mismo y puede conceder derechos sobre éste a los demás usuarios.

Para una aplicación concreta este será el procedimiento de auditoría, audit1:

```
CREATE PROCEDURE audit1
AS
 BEGIN
     DELETE from salida;

     INSERT INTO salida
     SELECT * FROM HISTORIA
     WHERE PORCENTAJE >= 0.2;
end;
```

Para el caso solo serán reportados los productos cuyo cambio de precio haya superado el 20% del valor anterior.

A esta altura la parte de la lógica de la aplicación está completa. La información de control será insertada en los momentos determinados y los procedimientos están

listos para actuar. Cualquier usuario que interaccione con la base de datos está sujeto a lo anteriormente señalado.

Creación de la Aplicación Cliente

La segunda fase del proceso de creación del programa cliente/servidor consiste en estructurar adecuadamente un proyecto basado en Delphi 5.0, con el fin de realizar la lógica de la presentación que permita operar y visualizar en forma gráfica, las tareas encomendadas al proyecto.

Se usa el Módulo de Datos para programar la lógica de la aplicación (Definición de tablas, acciones a ejecutar e imágenes a utilizar)

1. Ingresar a Delphi 5.0
   File | New Application
   Creándose un nuevo proyecto.

2. Cambiar las propiedades Nombre (Formulario) y caption (FrmVentas)

3. Grabar: File/Save All
   Elegir para el formulario el nombre UFmPpal, y para el proyecto el nombre Ventas.

4. Agregar un DataMódulo al proyecto
   File | New

En la carpeta New seleccionar Data Module y pulsar el botón OK.
Name: MdlVentas y salvar nuevamente, con nombre Mventas.

Se coloca un objeto Database DbVentas 

El Objeto TDatabase sirve para centralizar el control de la Base de Datos. Provee un acceso permanente a la Base de Datos; al estar asociado a un Alias, los cambios

que se puedan requerir no afectarán las aplicaciones, además permite el manejo de transacciones.

Todos los demás componentes que requieran conexión a la Base de Datos, lo realizarán por intermedio del TDatabase.

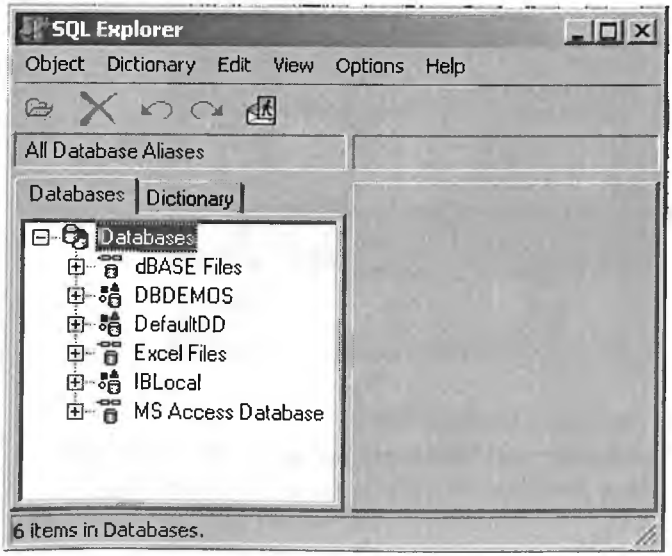
Las propiedades de importancia para la actual tarea son:

AliasName: Ventas
DataBaseName: VentasDB
LoginPrompt: Permite definir si cada que se haga la conexión con la Base de Datos la aplicación solicita Login y Password (propiedad en True). Si está en False, la conexión toma la información de la propiedad Params, empleando el evento OnLogin del TdataBase.

Para el caso se incluirá algo como lo siguiente:
USER NAME=SYSDBA
PASSWORD=masterkey

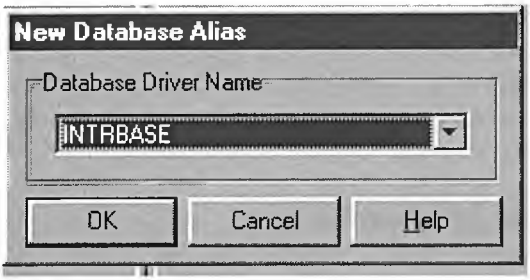
Se adiciona al DataModulo un objeto tipo StoredProc, 

para ejecutar el procedimiento almacenado audit1, creado anteriormente. Name: SPHist



Se debe recordar que la importancia del Procedimiento almacenado radica en que es parte de la base de datos y por lo tanto sujeto a toda la seguridad de un servidor.

5. Crear un alias para la Base de Datos. En barra de menús:
Database/ Explore
En el icono Database.

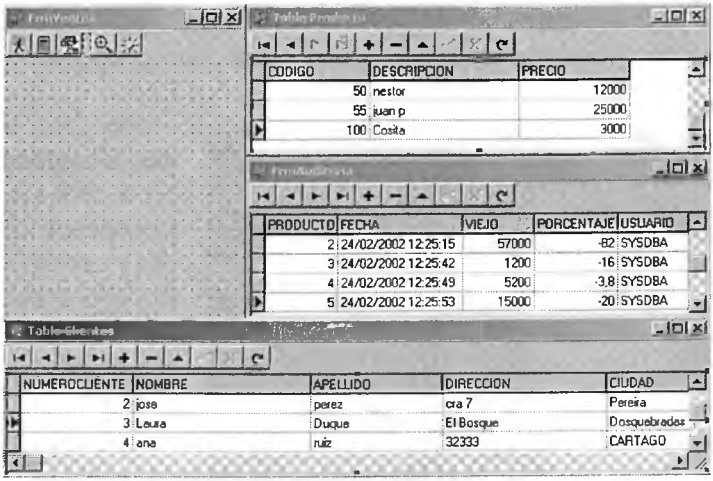


Con el botón derecho de mouse y seleccionando New (para nuevo alias) se obtiene la ventana que permite escoger el Driver o controlador para la Base de Datos

Presionar Ok. Nombre para el alias: Ventas. Asociarlo con el archivo de la Base de Datos, expandiendo la pequeña caja de puntos asociado con SERVER NAME. En este caso \datos\interbase\taller.gdb.

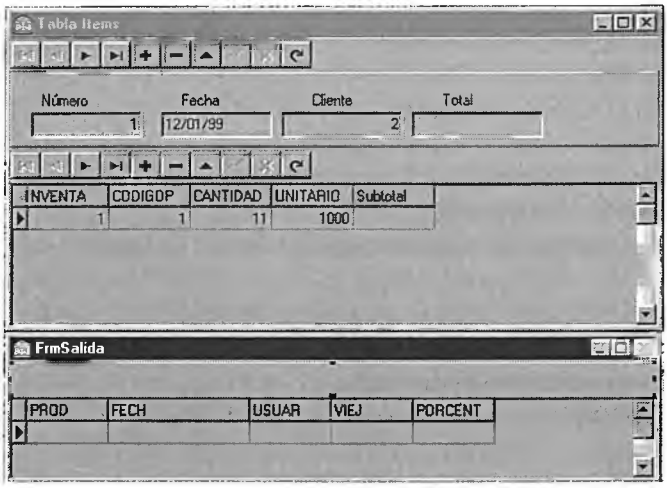
USER NAME SYSDBA
Grabar esta información: Object/|Save as
Aceptar el nombre VENTAS y pulsar el botón OK.
Cerrar el programa Explore.

Para la presentación se deben crear estos formularios.



Los objetos utilizados son Navigator, DBGrid, Panel.
Las acciones asociados a los botones del formulario principal son, en su orden: Formulario Clientes, formulario Producto, formulario Items, formulario Auditoría, Formulario Listado.

Estos son sus contenidos.



La programación que se muestra a continuación permite el uso de las características implementadas en el Servidor de Bases de Datos (parte inicial).

```
procedure TMdlVentas.ValidarPositivo(Sender: TField);
begin
//como sender no es flotante, lo convertimos.
  if Sender.AsFloat <=0 then
  raise Exception.Create('El valor unitario debe ser mayor que cero');
end;
procedure TMdlVentas.TotalizarFactura(DataSet: TDataSet);
 (*Controlar que se ubique nuevamente donde estaba situado. Utilizar un bookmark.
Evitar el efecto de scroll de muchos registros con la orden DisableControls *)
  var
  Total1:Currency;
  BM:TBookMark;
begin
  total1:=0;
  BM:=TblItems.GetBookmark;
  TblItems.DisableControls;
  TblItems.First;
  while not TblItems.Eof do
  begin
    Total1:=Total1+TblItemsSubtotal.Value;
```

```
    TblItems.Next;
  end;
  If TblVentas.State=DsBrowse then
    TblVentas.Edit;
  TblVentasTOTAL.Value:=Total1;
  TblVentas.Post;
  TblItems.GotoBookMark(BM);
  TblItems.FreeBookMark(BM);
  TblItems.EnableControls; // habilitar controles de la pantalla
end;
```

Para usar el procedimiento almacenado.

```
procedure TMdlVentas.ListadoExecute(Sender: TObject);
begin
   SPHist.Prepare;
   SPHist.ExecProc;
   FrmSalida.ShowModal;
end;
```

Nota:

Para programar la vista de la Historia de cambios se debe mantener cerrada la tabla, en otro caso los cambios no se ven reflejados hasta que se actualice.

BIBLIOGRAFÍA

Codd, E. F. A Relational Model for Large Shared Data Banks. ACM. 1970.

Ozsu, Valduriez. Principles of Distributed Database System. Prentice Hall. 1991.

Orfali, Harkey, Edwards. Cliente/Servidor. Guía de Supervivencia. McGrawHill. 1997.

Marcon Gomes Vaz María Salete, Valter Schastai, Duque Méndez Néstor Darío. Ponencia Día Internacional de Seguridad en Cómputo. Pereira. 1999.

Inprise. Manual Interbase Server.

Korth. Fundamentos de Bases de Datos. McGraw-Hill. 1998.

Groff, Weinberg. Guía LAN Times de SQL. McGraw-Hill. 1996.