InterBase / Firebird: el uso de eventos

Este artículo pretende dar una breve introducción acerca del uso de los eventos de Firebird e InterBase y como usarlos desde Delphi.
Los eventos permiten a las aplicaciones responder a acciones y cambios en las Bases de Datos hechos por algún usuario, cuando más de una aplicación se está ejecutando al mismo tiempo, evitando, de esta manera, la necesidad de que las aplicaciones se comuniquen directamente unas con otras y sin incurrir en la pérdida de tiempo que requiere un polling periódico para determinar un cambio en la Base de Datos.

En Firebird/InterBase, un evento es un mensaje pasado por un trigger o un procedimiento almacenado al manejador de eventos de Firebird/InterBase para anunciar la ocurrencia de una condición o acción específica, usualmente un cambio en la Base de Datos como por ejemplo, Insert, Update o Delete. Hay que tener en cuenta que los eventos son registrados solamente después de que la transacción bajo la cual este evento ocurre es llevada a cabo (Commit). Cada vez que se produce un evento, el servidor de Firebird/InterBase notifica a las aplicaciones interesadas la ocurrencia del mismo.

La declaración que se utiliza en Firebird/InterBase para eventos es POST_EVENT. Se debe tener en cuenta que la declaración POST_EVENT solamente puede ser utilizada en triggers y procedimientos almacenados.

Un ejemplo.

Supongamos que estamos realizando una aplicación que debe trabajar en red, y que nos interesa que cuando se está ejecutando más de una aplicación al mismo tiempo, los cambios que se producen en una aplicación se reflejen en las aplicaciones restantes. Podemos crear un trigger que produzca un evento cada vez que cualquier aplicación inserte datos en una tabla. El código que deberíamos utilizar sería como el que sigue.

```
CREATE TRIGGER TR_POST_NEW_REG FOR TABLA AFTER INSERT
AS
BEGIN
  POST_EVENT 'new_reg';
END
```

Donde TR_POST_NEW_REG es el nombre que le damos al trigger que estamos creando, TABLA corresponde al nombre de la tabla para la que estamos creando el trigger y new_reg es el nombre que le damos a nuestro evento. El nombre de los eventos está restringido a un máximo de 15 caracteres.

Hasta aquí creamos un trigger que se va a activar cada vez que se inserte un nuevo registro en nuestra base de datos. Ahora vamos a ver como manejamos esto desde Delphi.

Delphi 5 incorpora una nueva paleta de componentes, conocidos como componentes IBX. Dentro de esta paleta contamos con el componente IBEvents que será quien nos permitirá manejar los eventos que hayamos realizado con Firebird/InterBase.

Usando IBEvents.

Una vez que hayamos incorporado a nuestra aplicación un componente IBEvents debemos poner en su propiedad IBDataBase el nombre del componente IBDataBase que está relacionado con la Base de Datos que generará el evento en el que estamos interesados. El componente IBEvents posee un único Evento, como podremos comprobar en el Object Inspector, el evento OnEventAlert, aquí deberemos escribir el código que nos interese que realice nuestra aplicación cuando se produce el evento de interés.

Otro punto muy importante a tener en cuenta, es que nuestra aplicación debe registrar interés en la ocurrencia de eventos. Esto podemos hacerlo mediante el siguiente código:

```
IBEvents.Events.Add('nombre_del_evento');
IBEvents.RegisterEvents;
```

Donde nombre_del_evento, es el mismo nombre que le dimos cuando creamos el trigger o un procedimiento almacenado, siguiendo con nuestro ejemplo anterior en nombre_del_evento, deberíamos poner new_reg.

Si en algún momento deseamos que nuestra aplicación no reciba notificación acerca de la ocurrencia de un evento, simplemente deberemos usar IBEvents.UnRegesterEvents.

Con esta pequeña introducción, ya estamos en condiciones de comenzar a utilizar los eventos de Firebird/InterBase, pero debido a que siempre todo queda más claro mediante un ejemplo, vamos a desarrollar ahora un pequeño ejemplo acerca del uso de eventos.

Ejemplo de uso de eventos de Firebird/InterBase.

Vamos a crear una base de datos firebird/InterBase, y una tabla a la que llamaremos Clientes. Supongamos que la información que nos interesa de los clientes es la siguiente: Nombre, Teléfono, Número de documento y Dirección. Entonces, para crear nuestra tabla hacemos:

```
CREATE TABLE CLIENTES (
  CLAVECLIENTE INTEGER NOT NULL,
  NOMBRE VARCHAR(70),
  TELEFONO VARCHAR(15),
  DOCUMENTO VARCHAR(20),
  DIRECCION VARCHAR(100)
)
```

Una vez creada la tabla vamos a crear tres eventos, manejados por triggers, para avisar a nuestra aplicación cuando se inserte, modifique y borre un registro.

1. Trigger para la inserción de registros:

```
CREATE TRIGGER NUEVO_CLIENTE FOR CLIENTES AFTER INSERT
AS
BEGIN
  POST_EVENT 'NEW_CLIENTE';
END
```

2. Trigger para la modificación de registros:

```
CREATE TRIGGER CAMBIOS_EN_CLIENTE FOR CLIENTES AFTER UPDATE
AS
BEGIN
  POST_EVENT 'CAMBIO_CLIENTE';
END
```

3. Trigger para la eliminación de registros:

```
CREATE TRIGGER BORRO_CLIENTE FOR CLIENTES AFTER DELETE
AS
BEGIN
  POST_EVENT 'DEL_CLIENTE';
END
```

Finalmente vamos a hacer un generador y un trigger que nos incremente automáticamente la clave (llave primaria) de nuestra base de datos.

Primero hacemos el generador:

```
CREATE GENERATOR G_CLAVECLIENTE
```

Y ahora el trigger:

```
CREATE TRIGGER CLIENTE_BI FOR CLIENTES BEFORE INSERT
AS
BEGIN
  NEW.CLAVECLIENTE=GEN_ID(G_CLAVECLIENTE,1);
END
```

Ahora vamos a ver como usar estos eventos desde Delphi. Abrimos una nueva aplicación desde Delphi y ponemos, por ahora, los siguientes componentes: un IBDataBase y un IBTransaction.

Conectamos el componente IBDataBase a nuestra base de datos, ponemos su propiedad Connected a true, y lo asociamos con el componente IBTransaction. También asociamos el componente IBTransaction con el componente IBDataBase.

Ahora colocamos un componente IBDataSet y lo asociamos al IBDataBase, y escribimos lo siguiente:

* Propiedad SelectSQL

```
SELECT * FROM CLIENTES
```

* Propiedad InsertSQL

```
INSERT INTO CLIENTES(CLAVECLIENTE, NOMBRE,DOCUMENTO,
TELEFONO, DIRECCION) VALUES(:CLAVECLIENTE, :NOMBRE,
:DOCUMENTO,:TELEFONO, :DIRECCION)
```

* Propiedad ModifySQL

```
UPDATE CLIENTES
SET NOMBRE=:NOMBRE,
DOCUMENTO=:DOCUMENTO,
TELEFONO=:TELEFONO,
DIRECCION=:DIRECCION
WHERE CLAVECLIENTE=:OLD_CLAVECLIENTE
```

* Propiedad DeleteSQL

```
DELETE FROM CLIENTES
WHERE CLAVECLIENTE=:OLD_CLAVECLIENTE
```

Poner la propiedad Active del IBDataSet a true.

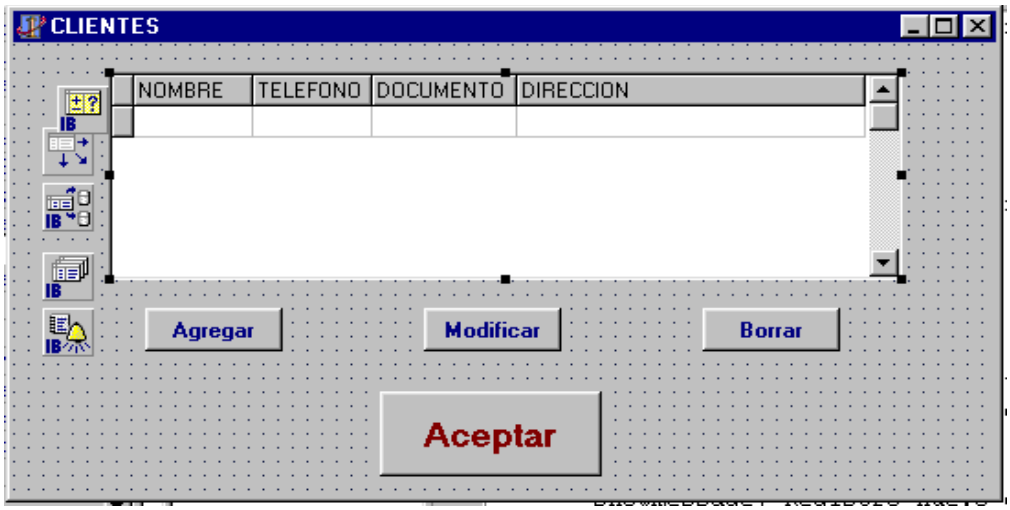
Colocamos un componente DataSource de la paleta Data Access y lo asociamos al componente IBDataBase. Agregamos un DBGrid, y los asociamos con el DataSource.

Ahora agregamos al Form un componente IBEvents y lo asociamos con el DataBase mediante

su propiedad DataBase.

Vamos a poner cuatro botones, uno para agregar un nuevo cliente, otro para borrar un cliente, otro para modificar los datos de un cliente y finalmente un botón para llevar a cabo la transacción.

El Form nos tiene que quedar como muestra la siguiente figura.



Finalmente, vamos a ver el código que utilizamos desde Delphi:

Primero, registramos los eventos en los que nuestra aplicación tiene interés. Esto lo hacemos en el evento FormShow de nuestro Form mediante el siguiente código:

```
IBEvents1.Events.Add('NEW_CLIENTE');
IBEvents1.Events.Add('DEL_CLIENTE');
IBEvents1.Events.Add('CAMBIO_CLIENTE');
IBEvents1.RegisterEvents;
```

Ahora, vamos a escribir el código necesario para insertar, modificar y borrar registros de nuestra base de datos. Para esto vamos a usar el evento Onclick de los botones que pusimos en el Form.

1. Botón Agregar:

```
IBDataSet1.Insert;
IBDataSet1.FieldByName('CLAVECLIENTE').asinteger:=1;
```

2. Botón Borrar:

```
IBDataset1.Delete;
IBTransaction1.CommitRetaining;
```

3. Botón Modificar:

```
IBDataSet1.Edit;
```

El botón Aceptar lo vamos a usar para llevar a cabo las transacciones cuando insertamos o modificamos un registro, entonces en su evento Onclick escribimos el siguiente código:

```
IBDataset1.Post;
IBTransaction1.CommitRetaining;
```

Finalmente, vamos a hacer que nuestra aplicación lleve a cabo alguna acción cuando se registra un evento. Para esto en el evento OnEventAlert del IBEvent, que es el único evento con el que cuenta este componente, escribimos el siguiente código:

```
if EventName='NEW_CLIENTE' then
  showmessage('Registro nuevo en la base de datos')
else if EventName='DEL_CLIENTE' then
  showmessage('Se borró un registro de la tabla')
else if EventName='CAMBIO_CLIENTE' then
  showmessage('Se modificaron los datos del cliente');
```

Y a probar.....

Con esto finalizó esta pequeña introducción al uso de Eventos de Firebird e InterBase. Espero realmente que les sea de utilidad.

(por Erika Martínez)
Buenos Aires, Argentina