FIREBIRD: ADMINISTRACIÓN Y SEGURIDAD

FIREBIRD: ADMINISTRACIÓN Y SEGURIDAD

1.- SEGURIDAD 1
2.- TABLAS Y VISTAS DE SISTEMA 5
3.- MONITORIZACIÓN 6
4.- CONFIGURACIÓN 8
5.- HERRAMIENTAS 9
5.1.- ISQL 9
5.2.- GBAK y 10
5.3.- GFIX 10
5.4.- GSTAT 10
5.5.- GSEC 11

1.- SEGURIDAD

Firebird no viene con ninguna utilidad para encriptar y desencriptar datos (salvo para las contraseñas de usuario) por lo que esta operación la tiene que hacer el cliente. Así no se tienen mecanismos de seguridad sobre los ficheros salvo los establecidos por el propio sistema operativo en el que se encuentra instalado el servidor.

Es por ello, que se aconseja instalar el servidor en máquinas dedicadas en las que se pueda establecer un buen nivel de seguridad: sistema operativo seguro (mejor linux que windows), sistema de ficheros apropiado (ext o ntfs), establecer limitaciones de acceso por parte de usuarios a los ficheros (definición de permisos de acceso a las carpetas y/o ficheros, no compartir las carpetas en red), configuración de un cortafuegos (habilitar sólo lo mas imprescindible como el acceso al servidor a través del puerto 3050), trabajo sobre líneas seguras y aplicar configuraciones de seguridad a Firebird (directorios en los que pueden estar los ficheros de bases de datos, etc).

La instalación de servidor de Firebird incluye una base de datos de autentificación de usuario (fichero **security.fdb**) para almacenar las definiciones de todos los usuarios que tienen acceso al servidor. Para cada usuario se establece un nombre y una contraseña sensible al contexto.

Esta seguridad se aplica cuando se conecta un usuario al servidor mediante conexión TCP/IP o local, siendo necesario identificarse con un usuario y contraseña. Este mecanismo adolece de un problema básico: si copian un fichero de base de datos y lo llevan a otro sistema, pueden acceder al mismo con los datos de seguridad de éste último (usuario y contraseñas), por lo que es fundamental hacer segura la máquina servidor.

Por defecto, Firebird viene con un usuario administrador SYSDBA con contraseña MASTERKEY, aconsejándose como primera operación cambiar la contraseña. Este usuario tiene permiso para hacer cualquier operación sobre cualquier objeto de la base de datos. Por ello, siempre se debe definir otro usuario con el que trabajar cuando, por ejemplo, accedemos desde un cliente. Si capturan el nombre de usuario y/o contraseña, podrán hacer menos cosas sobre la base de datos.

Un usuario se crea mediante la sentencia CREATE USER, se modifica mediante ALTER USER y se borra mediante DROP USER

CREATE USER <nombre usuario> {PASSWORD 'contraseña'}
[FIRSTNAME 'nombre']
[MIDDLENAME 'primer apellido']
[LASTNAME 'segundo apellido'];

CREATE USER <nombre usuario>
[PASSWORD 'contraseña']
[FIRSTNAME 'nombre']
[MIDDLENAME 'primer apellido']
[LASTNAME 'segundo apellido'];

DROP USET <nombre usuario>

Un usuario con permisos SYSDBA puede crear, modificar y borrar usuarios. Un usuario puede modificar su contraseña.

En Firebird, se trabaja con el concepto de role. Un role es creado en una base de datos estanado sólo disponible en la misma. No es más que un contenedor de privilegios asignados. Una vez que se rellena algún privilegio, estos pasarán a estar disponibles para algunos usuarios.

Un role se usa para controlar permisos asignados a conjuntos de usuarios, por ejemplo, se consideran usuarios de perfil administrativos, otros cajeros, etc. Así se asignan los permisos a role y se incluyen los usuarios en el role. Cualquier cambio se realiza sobre el role y no sobre los usuarios individuales.

Un usuario de Firebird se puede asignar a mas de un role, aunque sólo se le aplicará los permisos para el role con el que se conectó.

Además de los roles Firebird soporta los grupos UNIX y recientemente los WINDOWS.

Firebird trae por defecto un role para administración RDB$ADMIN. Este role se utiliza para dar permisos de SYSDBA a un usuario. Para ello será necesario asignarlo previamente al usuario.

A nivel de un objeto de base de datos se trabaja con los privilegios. Un privilegio representa un permiso para realizar una operación de DML. Se tienen los siguientes privilegios:

- ***SELECT***: Leer datos.
- ***INSERT***: Crear nuevas filas.
- ***UPDATE:*** Modificar datos existentes.
- ***DELETE:*** Borrar filas.
- ***REFERENCES:*** Referirse a una clave primaria desde una clave foránea. Siempre se tiene que acompañar de los privilegios sobre las tablas que contienen la tabla foránea.
- ***ALL:*** Todos los privilegios anteriores.
- ***EXECUTE:*** Ejecución un procedimiento almacenado o llamarlo desde una sentencia SELECT. Nunca se asigna cuando se indica ALL.
- ***ROLE:*** Se adquieren todos los privilegios asignados al role. Nunca se asigna cuando se indica ALL.

Cada objeto defininido en la base de datos (tabla, vista, procedimiento almacenado y role) tiene definido una serie de privilegios que se pueden asignar sobre él. Así un UPDATE no es aplicable a un procedimiento almacenado mientras que un EXECUTE no es aplicable a una tabla o vista. De esta forma tenemos que:

- SELECT, INSERT, UPDATE y DELETE sólo se pueden aplicar a tablas y vistas.
- REFERENCES solo se aplica a tablas, en principio solo a aquellas que son referenciadas por claves foráneas.
- En las vistas, el usario además de tener permiso sobre la misma debe tener permisos sobre las tablas usadas para generar la misma. Estos permisos puede tenerlos el propietario de la vista, la vista en sí misma o el usuario de la vista.
- La vistas actualizables tienen que tener los permisos SELECT, INSERT, UPDATE y DELETE sobre las tabla base.
- EXECUTE solo se puede puede aplicar a procedimientos almacenados.,

Los privilegios se asignan a los usuarios. Un usuario será un usuario definido en la base de datos de seguridad, una cuenta o un grupo UNIX/WINDOWS, un usuario especial o un objeto de base de datos.

La asignación de los permisos a un objeto sólo la puede realizar el usuario SYSDBA o el propietario del mismo. Si alguno de ellos quiere permitir que otro usuario distinto pueda asignar permisos se utiliza la cláusula WITH GRANT OPTION. A partir de ese momento el usuario puede asignar los permisos que le han dado a otro usuario. Esto mismo se puede aplicar para administrar el objeto con la cláusula WITH ADMIN OPTION.

Existe un usuario especial PUBLIC que representa cualquier usuario en la base de datos y que se puede utilizar para asignar permisos a todos en un solo paso.

Los privilegios se pueden asignar tanto a objetos enteros como a columnas dentro de una tabla o vista. Para asignarlos se tiene la siguiente sintaxis:

GRANT <privilegios>
{ON {{[TABLE] <tabla>}
| <vista>
| {PROCEDURE procedimiento}
}
}| <role con privilegios>
TO <usuario generico>
[{WITH GRANT OPTION} | {WITH ADMIN OPTION}]

<privilegios> = <privilegio> [, <privilegio> ...] | <role> | ALL
<privilegio> = INSERT | DELETE | UPDATE [(columna [, columna ....])]
| REFERENCES [(columna [, columna ....])]
| EXECUTE

<usuario generico> = <usuario>[, usuario …] | PUBLIC
| <usuario LINUX> | <usuario WINDOWS>
| GROUP <grupo LINUX o WINDOWS>
| <role>
| {TRIGGER triger}
| {PROCEDURE procedimiento}
| {VIEW vista}

Por ejemplo podríamos tener

GRANT SELECT, UPDATE, INSERT, DELETE ON USUARIOS TO profesor; -- da los privilegios indicados a la tabla usuarios para el usuario profesor.
GRANT UPDATE (CODIGO) ON USUARIOS TO PUBLIC -- da permiso para actualizar la columna codigo de la tabla usuarios a PUBLIC

GRANT REFERENCES (codigo) ON USUARIOS TO oficina; -- da el permiso para referenciar la columna codigo de la tabla usuarios al usuario oficina.

GRANT EXECUTE ON PROCEDURE procedimiento
TO profesor,
PROCEDURE otro_proc; -- da el privilegio de ejecución sobre procedimiento al usuario profesor y al procedimiento otro_proc;

GRANT ALL ON USUARIOS TO profesor; -- da todos los privilegios sobre la tabla usuarios al usuario profesor.

Para trabajar con los roles se tiene una serie de sentencias. Por ejemplo podríamos tener:

CREATE ROLE papel; -- crea el role papel
I
GRANT ALL ON USUARIOS TO papel; --asigna todos los privilegios al role papel;

GRANT papel TO profesor; -- asigna al usuario profesor el role papel.

DROP ROLE papel; -- borra el role

Igual que se pueden dar permisos con GRANT se pueden quitar los permisos dados mediante la sentencia REVOKE. Solamente el usuario SYSDBA o el usuario que da el privilegio puede revocarlos. Los permisos que se indicaron por bloque (con ALL , para un grupo, un role o a PUBLIC), no pueden revocarse de forma individual.

La sentencia REVOKE tiene una sintaxis similar a la de GRANT en la que se sustituye la cláusula TO por FROM. Por ejemplo, la sintaxis básica sería:

REVOKE <privilegios>
ON <objeto>
FROM <usuario generico>

```
REVOKE SELECT ON USUARIOS TO profesor;

REVOKE EXECUTE ON PROCEDURE procedimiento
    FROM profesor,
        PROCEDURE otro_proc;
```

2.- TABLAS Y VISTAS DE SISTEMA

Como ya se ha dicho en temas anteriores toda la información relativa a la base de datos, los metadatos, es almacenada mediante unas tablas y vistas especiales llamadas de sistema. Todas ellas empiezan por RDB$.

Algunas de las tablas y vistas de sistema:

- **RDB$CHARACTER_SET**: Conjuntos de caracteres disponibles para la base de datos.
- **RDB$COLLATIONS**: Secuencias de ordenación definidas.
- **RDB$DATABASE**: Tabla que contiene una sola fila con información de la base de datos.
- **RDB$FILES**: Detalle de los ficheros secundarios y ficheros sombra de la base de datos.
- **RDB$FORMAT**: Mantiene una cuenta del numero de cambios en los metadatos de las tablas. Cada vez que una tabla o vista se cambia se incrementa el número. Este número permite a las aplicaciones acceder a una tabla cambiada sin necesidad de recompilar. El límite de este número es 255. Si se alcanza este valor se convierte en inaccesible. Este valor sólo se resetea cuando se hace un restore de la base de datos.
- **RDB$GENERATORS**: Nombres e identificadores de los generadores.
- **RDB$INDICES**: Definiciones de todos los índices.
- **RDB$PAGES**: Información sobre las páginas de la base de datos.
- **RDB$PROCEDURES**: Definiciones de los procedimientos almacenados.
- **RDB$PROCEDURE_PARAMETERS**: Parámetros de los procedimientos almacenados.
- **RDB$REF_CONSTRAINTS**: Contiene las accones para la restricciones referenciales (claves foráneas).
- **RDB$RELATION_CONSTRAINTS**: Restricciones de intregridad a nivel de tabla.
- **RDB$RELATION_FIELDS**: Definiciones de las columnas.
- **RDB$RELATIONS**: Definiciones de cabecera de las tablas y vistas.
- **RDB$SECURITY_CLASSES**: Listas de control de acceso.
- **RDB$TRANSACTIONS**: Transacciones en base de datos
- **RDB$TRIGGERS**: Definiciones de los triggers.
- **RDB$TYPES**: Definición de los tipo enumerados usados en Firebird.
- **RDB$USER_PRIVILEGES**: Permisos SQL.
- **CHECK_CONSTRAINTS**: Vista con todas las restricciones CHECK definidas.
- **CONSTRAINT_COLUMN_USAGE**: Vista con las listas de columnas usadas por claves primarias, unicas y foráneas.
- **REFERENTIAL_CONSTRAINTS**: Vista con todas las restricciones referenciales definidas.
- **TABLE_CONSTRAINTS**: Vista con todas las restricciones a nivel de tabla.

3.- MONITORIZACIÓN

A partir de la introducción de las tablas temporales globales en la versión 2.1, se incorporó a Firebird la posibilidad de monitorizar y controlar diferente información relativa a la base de datos y a las consultas que se lanzan durante el trabajo normal en una sesión por parte del cliente. De esta forma es posible tener información tal como base de datos conectada, sentencias preparadas, trnsacciones, etc.

Esta información es accesible completamente por el usuario SYSDBA y el propietario de la base de datos mientras que cualquier otro usuario sólo tiene acceso a la información relativa a su conexión.

Todo esta gestión es posible gracias a las tablas de monitorización MON$:

- **MON$DATABASE:** base de datos conectada. Con información como ruta de la base de datos, versión de ODS, identificadores de proceso, etc.
- **MON$ATTACHMENTS**: Conexiones realizadas. Con información de estado, usuario que hace la conexión, protocolo de conexión, etc.
- **MON$STATEMENTS:** Sentencias preparadas para ejecutar. Con información como identificadores, texto, estado, etc.
- **MON$CALL_TACK:** Pila de llamadas de las sentencias PSQL activas.
- **MON$IO_STATS:** Estadísticas de I/O.
- **MON$RECORD_STATS:** Estadísticas a nivel de fila.
- **MON$MEMORY_USAGE:** Uso de memoria actual.
- **MON$CONTEXT_VARIABLES:** Variables de contexto conocidas

A estas tablas se puede acceder como a cualquier otra tabla que podamos encontrar en nuestra base de datos. Por ejemplo:

Para saber todos los identificadores de procesos con carga de CPU actuales:

```
SELECT MON$SERVER_PID
    FROM MON$ATTACHMENTS
    WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
        AND MON$STATE = 1
```

Para obtener información sobre las aplicaciones clientes

```
SELECT MON$USER, MON$REMOTE_ADDRESS,
        MON$REMOTE_PID,
        MON$TIMESTAMP
    FROM MON$ATTACHMENTS
    WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

Para obtener el nivelde aislamiento de la transacción actual

```
SELECT MON$ISOLATION_MODE
    FROM MON$TRANSACTIONS
    WHERE MON$TRANSACTION_ID = CURRENT_TRANSACTION
```

Para obtener las sentencias actualmente activas

```
SELECT ATT.MON$USER,
        ATT.MON$REMOTE_ADDRESS,
        STMT.MON$SQL_TEXT,
        STMT.MON$TIMESTAMP
    FROM MON$ATTACHMENTS ATT
        JOIN MON$STATEMENTS STMT
            ON ATT.MON$ATTACHMENT_ID = STMT.MON$ATTACHMENT_ID
    WHERE ATT.MON$ATTACHMENT_ID <> CURRENT_CONNECTION
        AND STMT.MON$STATE = 1
```

Para borrar todas las sentencias actualmente en ejecución

```
DELETE FROM MON$STATEMENTS
    WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

4.- CONFIGURACIÓN

Firebird puede ser instalado en dos versiones de arquitectura: Classic Server y Superserver.

En **Classic Server** se ejecuta un proceso de servidor por cada conexión. Cuando un cliente intenta conectarse a una base de datos, se inicia una instancia del ejecutable fb_inet_server que permanece en ejecución mientras la conexión no se finalice. Esto hace que se tenga una cache local de páginas por cada proceso y que se tengan que habilitar mecanismos de sincronización usando las características del sistema operativo en el que se instala, como por ejemplo señales en LINUX. Esta arquitectura es muy eficiente en caso de tener un número bajo de conexiones.

En **Superserver**, hay una instancia del servidor fbserver en ejecución. Cuando se establece una conexión por parte de un cliente se inicia un flujo para su control. Se tiene una sola cache compartida y se utiliza un flujo adicional para la sincronización. Es la indicada cuando el número de conexiones es grande.

Una vez seleccionada la arquitectura apropiada e instalado en el sistema, se pueden configurar diferentes parámetros a través del **fichero de configuración** "firebird.conf". Normalmente una vez realizada una configuración inicial no es necesario posteriores cambios.

Cuando se inicia el servidor se lee el fichero de configuración y se actualizan aquellos flacs para los que se haya indicado un valor al leido (se sobrescribe el valor por defecto).

El fichero se utiliza para rellenar los parámetros en el formato "parámetro = valor", donde valor será booleano (1=trae, 0=false) o una cadena que especifica el valor. Las lineas que no se quieren aplicar vendrán precedidas de un carácter de comentario (#).

Algunos de los parámetros mas importantes son:

- ***RootDirectory:*** Cadena que indica la ruta absoluta del directorio raiz en el sistema de ficheros. Normalmente no se indica.
- ***DatabaseAccess:*** Restringe las ubicaciones en la que se podrá colocar los archivos de bases de datos en el sistema de ficheros local para procesarlos el servidor. Puede valer: **Full** (por defecto) en cualquier posición, **None** solo se pemiten las bases de datos registradas en el fichero Aliases.conf, **Restrict** se restringe a las rutas que indiquen.
- ***TempDirectories:*** Lista de uno o varios directorios separados por ; con argumento opcional el número de bytes máximo a utilizar. P.e. c:\tmp 100000000; c:\windows\temp
- ***CpuAffinityMask:*** Procesador con el que trabajará el servidor. Valor 1 (CPU 0), 2 (CPU 1), 3 (CPU 0 y 1), etc…
- ***RemoteServiceName:*** Nombre del servicio del servidor por el que se escucha las peticiones. Por defecto gds_db.
- ***RemoteServicePort:*** Puerto usado en el servidor por el que se escucha las peticiones. Por defecto 3050.
- ***RemoteBindAddress:*** Dirección ip en la que el servidor estará a la escucha. Por defecto todas.

5.- HERRAMIENTAS

Firebird viene con una serie de herramientas en línea de comandos útiles para realizar operaciones varias sobre las bases de datos, como son copias de seguridad, estadísticas, consultas, reparaciones, etc. Estas herramientas se pueden encontrar en el directorio bin de la instalación.

5.1.- ISQL

Esta herramienta proporciona una interfaz interactiva no gráfica para la ejecución de sentencias DDL y DML, así como para unas sentencias de administración no disponibles en DSQL.

Se inicia mediante el comando:
Isql [base de datos] [-u[ser] usuario –pas[sword] contraseña

Una vez abierto el programa nos aparecerá un prompt (SQL>) a la espera de que le indiquemos una operación a realizar. Las sentencias que introduzcamos deberán finalizarse con ;.

Junto a las sentencias normales de DDL y DML podemos utilizar, entre otras:

- **CONNECT**: Para conectar a una base de datos. La sintaxis sería ***CONNECT base_datos USER usuario PASSWORD contraseña***.
- **HELP:** Para mostrar la información de los comandos disponibles.
- **QUIT:** Para salir del programa.
- **INPUT fichero:** Lee y ejecuta un script.
- **OUTPUT [fichero]:** Redirecciona la salida a un fichero o al dispositivo de salida estándar.
- **SHOW:** Muestra información sobre un objeto de la base de datos. Se tienen los siguientes valores:
    - ***SHOW DATABASE***
    - ***SHOW {DOMAINS | DOMAIN nombre}***
    - ***SHOW {EXCEPTIONS | EXCEPTION nombre}***
    - ***SHOW {FUNCTIONS | FUNCTION nombre}***
    - ***SHOW {GENERATORS | GENERATOR nombre}***
    - ***SHOW GRANT {objeto | role}***
    - ***SHOW {INDICES | INDEX {tabla | vista}}***
    - ***SHOW {PROCEDURES | PROCEDURE nombre}***
    - ***SHOW ROLES***
    - ***SHOW SYS***
    - ***SHOW {TABLES | TABLE nombre}***
    - ***SHOW {TRIGGERS | TRIGGER nombre}***
    - ***SHOW VERSION***
    - ***SHOW {VIEWS | VIEW nombre}***
- **SET**: Para establecer como se realizan diversas operaciones en la herramienta.
    - ***SET AUTODDL [ON | OFF]***: Por defecto ON. Si las operaciones de DDL se confirman de forma automática.
    - ***SET BLOBDISPLAY [n | ALL | OFF]***: Subtipo blob que se mostrará. Por defecto valor 1 (texto).
    - ***SET COUNT [ON | OFF]***: Muestra el número de filas devuelta por la consulta. Por defecto OFF.
    - ***SET ECHO [ON | OFF]***: Se mostrará o no la sentencia que se ejecuta. Se usa cuando se cargan ficheros por lotes. Valor por defecto ON.
    - ***SET NAMES conjunto_caracteres***: Fija el conjunto de caracteres con el que trabajar.

- ***SET PLAN [ON | OFF]***: Permite mostrar o no el plan de ejecución para una consulta. Valor por defecto ON.
- ***SET PLANONLY [ON | OFF]***: Indica si se muestra únicamente el plan de ejecución y no se ejecuta la sentencia. Por defecto OFF.
- ***SET SQL DIALECT N***: Establece el dialecto de trabajo al indicado. Por defecto 3.
- ***SET STATS [ON | OFF]***: Muestra o no las estadísticas de eficiencia. Por defecto OFF.

5.2.- GBAK y

Son herramientas que se utilizan para realizar copias de seguridad y para restaurar las mismas.
Se pueden realizar las copias de seguridad completas (gbak) o incrementales ().

Gbak se puede utilizar para realizar operaciones de limpieza como eliminar transacciones zombie, procesos de sweep o para realizar actualizaciones de la base de datos a formato mas nuevos (cambio en el ODS).

Estos comandos sólo los puede ejecutar el usuario SYSDBA o el propietario de la base de datos.

5.3.- GFIX.

Esta herramienta es usada por el usuario SYSDBA o el propietario de una base de datos para realizar las siguientes operaciones:

- Realizar un sweep, es decir, limpiar las version antiguas de los registros que no ha eliminado el recolector de basura.
- Cambiar el intervalo de sweep.
- Iniciar un shutdown de la base de datos para acceso en exclusividad y ponerla de nuevo en línea.
- Cambiar entre escrituras sincronías (forzadas) o asíncronas.
- Cambiar la base de datos de lectura-escritura a solo lectura y viceversa.
- Cambiar el dialecto.
- Cambiar el tamaño de cache de la base de datos.
- Encontrar y confirmar o recuperar transacciones en limbo.
- Recuperar bases de datos y sus datos corruptos bajo ciertas condiciones.
- Activar o eliminar un shadow de una base de datos.

5.4.- GSTAT

Es una herramienta por la que se puede administrar a los usuarios existentes en el sistema. Mediante ella podremos dar de alta nuevos usuarios, borrar existentes, cambiar la contraseña, etc

5.5.- GSEC

Permite obtener diversas estadísticas sobre una base de datos y los objetos de ella. Se puede usar para comprobar los accesios a la misma, los identificadores de transacción, el uso de los indices, etc