Notas sobre creación de usuarios "clientes" de Trew@

JUNTA DE ANDALUCIA

CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

5 de abril de 2006

GuadalTEL
INGENIERÍA DE SISTEMAS

ÍNDICE

Introducción

Propósito

El presente documento tiene como objetivo definir brevemente las necesidades y características que, en general, deben tenerse en cuenta para la creación de usuarios "consumidores" de la base de datos que da soporte al motor Trew@.

Para completar estas notas, se remite al lector a la documentación técnica detallada que acompaña a cada componente.

Alcance

Este documento va dirigido a:

- La dirección del proyecto w@ndA.
- El colectivo de usuarios-desarrolladores Trew@
- El colectivo de usuarios administradores de sistemas y base de datos de aplicaciones "clientes" de Trew@

General

Como es de esperar, para trabajar con el motor Trew@, ya sea como usuario administrador de entidades de un sistema desde las pantallas de administración (Forms de Oracle o JSP de administración en J2EE) o ya sea con las APIs que se ofrecen (por ejemplo, las JTrApis), es necesario hacer una conexión a la base de datos de Trew@.

Lo que se define a continuación, es lo que en condiciones normales se recomienda para la creación y uso de usuarios de base de datos "consumidores" de una conexión a Trew@, concretamente los permisos, sinónimos, etc. que estos usuarios "reales" de base de datos necesitarían tener. Todo esto está también orientado a evitar que estas conexiones no se hagan con el usuario propietario de los objetos de Trew@, práctica que se viene siguiendo muchas veces y poco recomendable por razones de seguridad sobre todo.

Estas notas no pretenden ser de obligado cumplimiento pero sí servir de guía y ayudar a la comprensión de la gestión de usuarios "típica" que puede seguirse cuando se utiliza el motor de tramitación Trew@.

Usuarios para pantallas de administración

Si hay que crear un usuario para usarlo como conexión con las pantallas de administración de Trew@ (Forms de Oracle o JSP de administración en J2EE), lo primero es crear el usuario de base de datos como tal. Al menos este usuario debería tener permisos para conectarse y crear sinónimos.

Como debe ser usuario "*administrador*", es lógico entender que este usuario debe tener acceso a los objetos de Trew@, por tanto deberán crearse sinónimos y darse permisos a dichos objetos. No obstante, en este punto puede haber variaciones dependiendo de la política de cada Consejería y de las necesidades concretas.

Por ejemplo:

- Si está claro que el usuario puede administrar "*todo*" pues se pueden dar los sinónimos y todos permisos necesarios sobre todos los objetos de Trew@.
- En cambio, si lo que se quiere es crear un usuario de consulta y que vea todos los datos, pues sólo habría que dar los permisos correspondientes sobre los objetos de Trew@ (normalmente sólo "*select*").
- En otros casos, si además se quiere que cuando alguien utilice este usuario como conexión desde otro tipo de aplicaciones, por ejemplo un SQL*Plus o TOAD, sólo vea parte de los datos, pues es necesario saber qué es lo que el usuario en cuestión debería "ver" y lo más evidente sería crear sinónimos a vistas filtradas que hagan lo que se necesita, e igualmente darle los permisos correspondientes que se estimen oportunos sobre estas vistas. A partir de la versión v1.1.0 de Trew@ se suministran los scripts necesarios para crear estas vistas, crear los sinónimos para usuarios y dar los permisos sobre las mismas

Esto, como puede verse, no es más que una administración de usuarios de base de datos, según las necesidades y sin tener nada que ver en principio con que estemos hablando de un sistema Trew@, lo único a tener en cuenta sería que para que las pantallas de administración no ofrezcan un mal funcionamiento, el usuario creado debería tener sinónimos para cada uno de los objetos de Trew@, y al menos permisos de "select" sobre ellos.

Por el hecho de ser un usuario Trew@, además, también debe existir como usuario en la tabla GN_USUARIOS de Trew@ ya que esto se comprueba al entrar en las pantallas de administración. Esto puede hacerse con el propietario de los objetos (si no tenemos otro usuario con el que entrar), desde un TOAD o similar o bien desde las propias pantallas de administración.

También para hacer uso de las pantallas de administración, el usuario creado debe además tener el role TR_R_ADMINISTRADOR de Trew@ (aunque este role no tenga ningún permiso, más adelante se habla sobre esto). Este permiso puede darse al crear el usuario de base de datos, o bien se puede hacer también desde las propias pantallas de administración cuando se da de alta como usuario Trew@ (señalando en el apartado "*Permiso*" la opción de "*Administrador*"). Además se debe tener permisos de "*select*" sobre "*DBA_ROLE_PRIVS*".

Como alternativa al apartado anterior a partir de la versión v.1.1.0 de Trew@ se permite para los usuarios de base de datos "reales" un tratamiento similar al de los usuarios que no existen realmente en base de datos (sólo en la tabla correspondiente), permitiendo la comprobación de permisos no a través de los roles de base de datos sino a través de los perfiles de usuario "*TR_R_ADMINISTRADOR*" y "*TR_R_USUARIO*" asociados al sistema por defecto "*TREW@*". Este tratamiento de permisos es el que se sigue para trabajar además con la base de datos PostgreSQL. En caso de usar esta "funcionalidad" téngase en cuenta que no sería necesario dar roles de base de datos ni permisos sobre DBA_ROLE_PRIVS.

Hasta aquí lo que se necesita para un usuario que haga uso de las pantallas de administración. Si se dispone de las versiones actualizadas de Trew@ v0.0.1 o v1.0.1 y posteriores, se puede hacer además que este usuario sólo vea determinados sistemas. Hablamos del caso en que se entre con las pantallas de administración, si se entra desde otra herramienta (TOAD, SQL*Plus, etc.) y no hay vistas que lo impidan, evidentemente el usuario tendrá acceso a todo aquello sobre lo que se tenga permiso, incluso a datos de otros sistemas. Para aprovechar esta funcionalidad de las pantallas de administración, basta con asociar al usuario (en la pantalla de mantenimiento de usuarios) algún perfil del sistema o sistemas en los que se quiere que el usuario sea administrador (Perfil de usuario). Esto hará que al entrar con las pantallas sólo se permita administrar los sistemas en los que se tiene permiso.

Además, las pantallas de administración de las versiones de Trew@ a partir de la v0.0.1, v1.0.1 y posteriores hacen un encriptado de claves, de forma que la clave a utilizar desde las pantallas sea distinta a la clave "real" de base de datos, para evitar conexiones que no sean desde las propias pantallas de administración.

Como esta separación de sistemas sólo se hace en el ámbito de pantallas y no de base de datos, también podría darse el caso de necesitar además que el usuario sólo tenga acceso a determinados datos, lo que nos lleva de nuevo a crear sinónimos a vistas filtradas.

Por ejemplo, si imaginamos que queremos hacer lo mismo en base de datos que lo que se hace desde las pantallas de administración (ver sólo los datos de determinados sistemas), pues habría que crear vistas filtradas para casi todas las tablas de Trew@ que incluyeran una cláusula parecida a:

```
SELECT ....
FROM ....
WHERE STMA_X_STMA IN
(SELECT DISTINCT P.STMA_X_STMA FROM
TR_USUARIOS_X_PERFILES_USUARIO UPU,
TR_PERFILES_USUARIOS P
WHERE UPU.PEUS_X_PEUS = P.X_PEUS
AND UPU.USUA_C_USUARIO = USER)
```

y crear para el usuario sinónimos a estas vistas. En cualquier caso, todo esto dependería de qué necesidades haya y qué tipo de usuarios "administradores" haya que crear para las pantallas de administración Trew@.

Por defecto los scripts que se facilitan para creación de sinónimos y permisos, están orientados a un usuario administrador que puede acceder a todos los datos existentes en Trew@, pero a partir de la versión v1.1.0 se incluyen scripts para vistas como se ha comentado anteriormente.

Conexiones con JTrApis

Para hacer estas conexiones está claro que hay que disponer de un usuario de base de datos, que va a representar la conexión que utilizará la API para hacer su trabajo.

NOTA: no confundir con la conexión que hace el sistema "cliente" de Trew@ a su propio modelo de datos, ya que la gestión de esta conexión es competencia de dicha aplicación cliente.

Hasta ahora viene siendo una práctica común hacer dicha conexión con el usuario propietario de los objetos de Trew@, pero se recomienda que dicha conexión sea distinta al propietario Trew@ por razones evidentes, sobre todo por seguridad.

¿Qué se necesitaría para un usuario de este tipo?, pues como va a ser la conexión para las APIs deben ver todos los objetos de Trew@ y deberían tener todos los permisos sobre los mismos, ya que es el uso que se haga de la API quien gobernará qué se hace y qué no. Evidentemente también se podrían hacer vistas para no permitir ver datos de otros sistemas pero habría que afinar muy bien.

Estos usuarios pueden crearse de la misma forma que los usuarios ya comentados para las pantallas de administración, la única diferencia es que si las aplicaciones que trabajan con Trew@, vía API, **siempre** establecen un usuario distinto al de la conexión (ver guía de referencia de las APIs para más detalles), ni siquiera necesitan de ningún role de Trew@, ni tampoco estar dados de alta en la tabla de usuarios, ya que sólo se van a utilizar para que haya una conexión que tenga acceso a los objetos de Trew@.

Apis PL/SQL (PQ-TrApis en Oracle)

Estos usuarios sólo necesitan tener permisos de "execute" sobre los paquetes PL/SQL que representan las Apis en Trew@. Si además se hace **siempre** un "*establecerUSuario*" de un usuario *no-Oracle*, ni siquiera necesitan ningún role de Trew@. Sólo necesitarían un role de Trew@ en el caso exclusivo de que la llamada a la API sea con el propio "user" de base de datos y no se haya hecho este "*establecerUsuario*". Aún así tampoco necesitarían de roles de Trew@ si se han dado los permisos a través de los perfiles de usuario del sistema por defecto "TREW@".

Si es necesario, habrá que crear los sinónimos a este usuario a los paquetes PL/SQL a los que vaya a llamar.

Además si estos usuarios van a crear paquetes, funciones o procedimientos almacenados en su esquema, y van a hacer uso de las APIs PL/SQL, deberán tener permisos directos sobre los PL de Trew@, ya que Oracle no deja crear estos paquetes si no se tienen permisos directos sobre los objetos a los que se hace referencia.

Roles de Trew@

Los roles de Trew@ *TR_R_USUARIO* y *TR_R_ADMINISTRADOR*, tienen "herencias" de versiones anteriores, es decir, antes sólo existía una API PL/SQL y los usuarios eran siempre usuarios de base de datos, entonces por compatibilidad con estas versiones anteriores, se ha mantenido agrupados ciertos permisos sobre los roles de Trew@ (básicamente "execute" sobre los paquetes PL), de forma que dado el role dado los permisos sobre las APIs PL/SQL.

Esta agrupación de permisos pierde un poco el sentido en aplicaciones J2EE que sólo necesitan de un usuario "real" de base de datos para hacer la conexión, máxime cuando se instalan 2 o más sistemas Trew@ en la misma instancia de base de datos, ya que los roles llevarían permisos sobre objetos de distintos sistemas Trew@. En este caso podrían quitarse los permisos a los roles, teniendo en cuenta que cualquier permiso que haya que dar a los usuarios sobre objetos de Trew@ debería darse uno a uno.

¿Qué role de Trew@ dar en caso que sea necesario?, pues la única diferencia es que el *TR_R_ADMINISTRADOR* tiene permisos para ciertas Apis que son de administración y un usuario "normal" no tiene permisos para ejecutar. Además en otras APIS, con un role *TR_R_ADMINISTRADOR* se permite evitar ciertos controles.

Como alternativa a estos roles de base de datos ya se ha descrito anteriormente que a partir de la versión v1.1.0 se hace posible la comprobación de permisos a través de los perfiles de usuarios del modelo de datos Trew@, concretamente asociando al usuario uno de los perfiles asociados al sistema por defecto "TREW@".

En cuanto al role *GN_R_GESTION*, por defecto, agrupa todos los permisos sobre las tablas generales (GN_*), pero no quiere decir que obligatoriamente deba existir este role ni tener los permisos que tiene, puede variarse los permisos, o el nombre del role, etc.., o puede incluso establecerse otra política para dar estos permisos a aquellos usuarios que necesiten "ver" estas tablas generales. En este aspecto cabe destacar para que se tenga en cuenta, que con las Apis a partir de Trew@ v1.0.0 se puede hacer consultas a estas tablas vía API, que si se utilizan no se necesitaría ni siquiera dar estos permisos. Otra asunto sería que el usuario en cuestión necesite hacer otras operaciones con las tablas, pero en cualquier caso, la distribución de permisos sobre las tablas GN se deja a elección de cada caso.

Establecer un usuario con las Apis

Esta API de "establecerUsuario" que está disponible tanto en las PQ-TrApis (Apis PL/SQL) y en las JTrApis (APIs java) de Trew@, permite establecer para las Apis un usuario de "trabajo" con las mismas. Por defecto, ya sea en java o en pl/sql, se hace un "establecerUsuario" interno del "*USER*" de la conexión. Esta API se utiliza, por ejemplo, para establecer un usuario que no es usuario de base de datos "real", lo que hace que el mantenimiento de usuarios se centre en datos de tablas de base de datos más que en un mantenimiento de usuarios de base de datos, aligerando así la labor de los administradores de base de datos. En este caso de usuarios no "reales" de base de datos, tanto las APIs java como las pl/sql, ya no comprueban los roles de Trew@ sino que lo que se comprueba es si el usuario tiene un "*Perfil de usuario*" asociado en Trew@ (es por ello que en la tabla *TR_PERFILES_USUARIOS* existen 2 perfiles creados por defecto asociados al sistema Trew@ que se llaman igual que los roles de base de datos). Esta comprobación se hace de esta forma incluso para los usuarios de base de datos "reales" que tengan asociados alguno de estos perfiles primando esta comprobación de permisos sobre la de roles de base de datos.

En resumen...(recomendaciones para creación de usuarios de base de datos)

Resumiendo un poco todo lo comentado anteriormente, para crear un usuario de base de datos y darle los permisos y sinónimos correctos sobre objetos de Trew@, podrían seguirse los siguientes pasos, teniendo en cuenta que la política existente en cada Consejería puede hacer que varíen en algún aspecto:

1. Conocer qué es lo que se va a hacer con el usuario que se pretende crear y cómo va a acceder este usuario a Trew@ (¿pantallas de administración o APIs?). Esto es **muy importante** saberlo y definirlo antes de crear dicho usuario. Como nota importante, cabe destacar que cualquier otro acceso que no sea vía API o vía pantallas de administración, no garantiza compatibilidad con versiones posteriores de Trew@, ya que la filosofía de este sistema es la de "caja negra" accesible sólo vía API.

2. Si el usuario tiene que llamar a las Apis PL/SQL (sólo en Base de datos Oracle), debe tener permisos de "*execute*" y normalmente tener sinónimos, a los paquetes que representan dichas Apis. Si además estas llamadas las va a hacer desde un procedimiento almacenado, el permiso de "*execute*" sobre los PL no se permite que se haya dado a través de un role, tiene que ser directo sobre el objeto (así lo exige una base de datos Oracle).

   Si no se hace un "establecer usuario" (se tomará por defecto el *USER* de la conexión), pues necesitará además bien el role de base de datos *TR_R_USUARIO*, o bien el role *TR_R_ADMINISTRADOR* (o en su defecto alguno de los perfiles de usuario del sistema por defecto). Si en cambio sí se hace un "establecer usuario" y el usuario establecido no es un usuario real de base de datos, no se necesita ninguno de los roles ya que Trew@ en este caso mirará los perfiles de usuario en vez de los roles de base de datos. Si el usuario establecido sí que es usuario de base de datos, pero distinto al de la conexión, el usuario establecido sí que necesitará tener uno de los roles de Trew@. Es decir, siempre que se haga llamada a la API PL/SQL y el usuario que haga la llamada sea usuario de base de datos "real", necesitará uno de los 2 roles (dependiendo ya de qué competencias tenga).

3. Si el usuario va a ser utilizado para las pantallas de administración de Trew@ (Forms de Oracle o jsp de admisnitración), necesitará "ver" todos los objetos de Trew@, esto es, dependiendo de si se quiere más o menos seguridad podrían crearse sinónimos a vistas filtradas de los objetos de Trew@ (típicamente filtrando por el sistema).

En cuanto a los permisos (ya sea sobre vistas o sobre las tablas) dependerán de qué se necesita para ese usuario, porque si sólo se necesita consulta se puede dar sólo este permiso. En este caso, aunque las pantallas no están preparadas para detectar qué permiso tiene el usuario, la base de datos elevará el mensaje correspondiente de lo que no se permite hacer. Por ejemplo, si se intenta insertar desde las pantallas, pues se elevará el error correspondiente de que no se tienen los permisos de inserción, pero en cualquier caso no significa esto que las pantallas fallen.

Sí que se necesita permisos de "*select*" en *SYS.DBA_ROLE_PRIVS* (o en una vista de la misma) y role *TR_R_ADMINISTRADOR* de base de datos. Aunque como se comenta anteriormente, como alternativa se pueden dar alguno de los perfiles de usario pertenecientes al sistema por defecto "TREW@".

4. Si el usuario se va a crear para hacer una conexión a Trew@ vía API de java (JTrApis), éste será parecido al comentado para las pantallas de administración, es decir, se pueden utilizar para ellos los mismos scripts para sinónimos y permisos. Este usuario sólo necesitará uno de los roles en el caso de no hacer un establecimiento de un usuario distinto al de la conexión (vía API desde la aplicación cliente). Sí que necesita permiso de "*select*" en *SYS.DBA_ROLE_PRIVS* por si acaso alguna vez tiene que comprobar los roles de un usuario establecido que sea usuario de base de datos "real" o en su defecto al menos unos de los perfiles de usuario comentados anteriormente. En cuanto a qué sinónimos y permisos se pueden igualmente hacer vistas filtradas por sistemas, pero con los permisos habría que afinar mucho, por lo que se recomienda que tenga todos y que sea la aplicación cliente la que controle qué está haciendo vía API.

5. En cuanto a las tablas generales (GN_*), se podrían dar los permisos y sinónimos a "gusto del consumidor", son los únicos objetos del esquema de Trew@ a los que se permiten accesos directos. Aunque se recomienda que se den siempre que no haya otra posibilidad. Por ejemplo, a partir de Trew@ v1.0.0, ya se ofrecen Apis que permiten obtener datos de estas tablas, y desde las pantallas de administración se pueden también administrar estos datos, por lo que si no se necesita, casi mejor no dar estos permisos.

Otras recomendaciones

1. Por cada sistema "cliente" que vaya a hacer uso de Trew@, crear un usuario administrador para las pantallas de administración.

2. También por cada sistema "cliente" crear otro usuario para la conexión java (JTrApis).

NOTA: Se puede utilizar el mismo, pero hay que tener en cuenta que si no se ha filtrado con vistas se puede tener acceso a todos los datos (incluso de otros sistemas) desde otras herramientas que no sean las pantallas de administración de Trew@ (Forms de Oracle o jsp de administración) tales como SQL*Plus, TOAD, etc.. Esto puede evitarse en las pantallas de administración de las versiones v0.0.1, v1.0.1 y posteriores, debido a que estas versiones hacen un encriptado de claves. También puede evitarse para las JTrApis utilizando "DataSorce" para las conexiones, de esta forma aunque se utilice el mismo usuario, alguien que no sea administrador de sistemas podría no conocer la clave real del usuario de base de datos, y podría trabajar con las pantallas y las APIs sin necesidad de conocer esta clave.

3. Utilizar vistas filtradas siempre que la seguridad y la visibilidad del usuario sea un factor muy importante.

4. Utilizar "DataSource" y pool de conexiones en el servidor de aplicaciones para las JTrApis en vez de ficheros ".properties", siempre que así se decida como mejor alternativa.

Historia de versiones

VERSION	FECHA	AUTOR	DESCRIPCIÓN
1.0.0	17/02/2006	GuadalTEL	Primera versión
1.1.0	05/04/2006	GuadalTEL	Ajustes para la versión v1.1.0 de Trew@