Gestion des privilèges

1. La notion d'utilisateur

La notion d'utilisateur possède une lacune importante dans SQL car elle fait l'impasse sur la façon dont on créé un utilisateur...

NOTA : SQL 2 a prévu des règles d'accès spécifiques pour les différentes familles d'ordre SQL. Ainsi pour la création d'un schéma (une base de donnée en fait) ou d'un catalogue (un ensemble de bases de données), il laisse la règle à l'appréciation de l'éditeur du SGBDR. En revanche pour les ordres CREATE, ALTER et DROP, l'utilisateur courant doit être le même que l'utilisateur propriétaire (auteur) du schéma modifié.

Rapellons que la création d'un utilisateur se fait au moment de la création du schéma. Par défaut ce dernier est le créateur des objets.

Souvent dans les implémentation commerciales, on trouve un pseudo ordre SQL du genre *CREATE USER nomUtilisateur*, ou encore une procédure stockée permettant de définir un nouvel utilisateur.

Ainsi, pour MS SQL Server :

sp_adduser 'nomConnexion', 'nomNouvelUtilisateur'

Permet d'ajouter un nouvel utilisateur affecté à la connexion donnée.

La norme SQL2 propose trois fonctions pour connaître l'utilisateur (c'est à dire celui qui se connecte au serveur) et l'auteur, c'est à dire le créateur des objets :

SYSTEM_USER	nom de connexion
SESSION_USER	nom du créateur
CURRENT_USER	utilisateur courant

Ainsi, par défaut, SQL Server utilise les noms suivants :

SELECT SYSTEM_USER, SESSION_USER, CURRENT_USER	SYSTEM_USER SESSION_USER CURRENT_USER ---------- ---------- ---------- sa dbo dbo

Notons en marge le super utilisateur SQL PUBLIC qui concerne tous les utilisateurs passés et à venir.

2. Octroyer des privilèges

Les privilèges sont, pour un ou plusieurs utilisateurs la possibilité d'utiliser certains objets et parmi ces objets, certains ordres SQL.

2.1. Les différents privilèges

Voici une liste des différents privilèges que SQL permet et les objets sur lesquels ces privilèges portent :

USAGE	domaine, jeu de caractères, collation, translation
SELECT, INSERT, UPDATE, REFERENCES	nom du créateur
CURRENT_USER	table, vue, colonne(s)

Notons qu'il n'est pas possible de définir des droits sur la création, la modification ou la suppression des éléments de schema (base de données), ceci étant définit lors de la création du schema.

2.2. Attribution de privilèges

C'est l'ordre SQL GRANT qui permet d'attribuer un privilège à différents utilisateurs sur différents objets.

Voici la syntaxe de l'ordre SQL GRANT :

```
GRANT <privileges>
  TO <gratifié> [ { , <gratifié> }... ]
    [ WITH GRANT OPTION ]
```

```
  <privileges> ::=
    <privilège> [ { , <privilège> }... ]
```

```

```

```
   <privilege> ::=
      ALL PRIVILEGES ON <objet table>
    | <privilège d'action>
    | <privilege d'usage>
    | <privilege de référence>
```

```
    <privilège d'action> ::=
      <action> [ { , <action> }... ] ON <objet table>
```

```

```

```
     <privilege d'usage> ::=
      USAGE ON <objet d'usage>
```

```
    <privilege de référence> ::=
      REFERENCES [ ( <liste de colonne> ) ] <nom de table>
```

```
       <action> ::=
           SELECT
         | DELETE
         | INSERT [ ( <liste de colonne> ) ]
         | UPDATE [ ( <liste de colonne> ) ]
```

```
       <objet table> ::=
         [ TABLE ] <nom de table ou de vue>
```

```
       <objet d'usage> ::=
         DOMAIN <nom de domaine>
         | COLLATION <nom de collation>
         | CHARACTER SET <nom de jeu de caractères>
         | TRANSLATION <nom de translation>
```

```
<gratifié> ::=
   PUBLIC
  | <utilisateur>
```

La clause WITH GRANT OPTION, est utilisée pour autoriser la transmission des droits. La clause ALL PRIVILEGES n'a d'intérêt que dans le cadre de la transmission des droits.

Voici maintenant une batterie d'exemples afin de mieux comprendre comment utiliser cet ordre... Pour nos exemples, nous avons considéré que les utilisateurs DUPONT, DURAND, DUBOIS, DUVAL, DULAC et DUFOUR était créé dans la base de données. Celui qui lance les ordres (sauf indication contraire) est l'utilisateur DUHAMEL.

```
GRANT SELECT
    ON T_CHAMBRE
    TO DUBOIS ;
```

Autorise DUBOIS à lancer des ordres SQL SELECT sur la table T_CHAMBRE. Notez l'absence du mot TABLE.

```
GRANT INSERT, UPDATE, DELETE
    ON TABLE T_CHAMBRE
    TO DUVAL, DUBOIS ;
```

Autorise DUVAL et DUBOIS à modifier les données par tous les ordres SQL de mise à jour (INSERT, UPDATE, DELETE) mais pas à les lire !

```
GRANT SELECT
    ON TABLE T_CMAMBRE
    TO DUFOUR WITH GRANT OPTION ;
```

Autorise DUFOUR à lancer des ordres SQL SELECT sur la table T_CHAMBRE mais aussi à transmettre à tout autre utilisateur les droits qu'il a acquis dans cet ordre.

```
GRANT SELECT, INSERT, DELETE
    ON TABLE T_CHAMBRE
    TO DURAND WITH GRANT OPTION ;
```

Autorise DURAND à lancer des ordres SQL SELECT, INSERT, DELETE sur la table T_CHAMBRE.

```
GRANT SELECT, UPDATE
    ON TABLE T_CHAMBRE
    TO PUBLIC ;
```

Autorise tous les utilisateurs présent et à venir à lancer des ordres SQL SELECT et UPDATE sur la table T_CHAMBRE.

Exemple : DURAND lance l'ordre suivant :

```
GRANT ALL PRIVILEGES
    ON TABLE T_CHAMBRE
    TO DUBOIS
```

Ce qui autorise DUBOIS à lancer sur la table T_CHAMBRE, les mêmes ordres SQL, que ceux autorisé à DURAND (SELECT, INSERT, DELETE).

On parle alors d'**héritage de droits** c'est à dire que l'utilisateur dotés de ces droits peut à nouveau les céder à un ou plusieurs autres utilisateurs.

Exemple : DURAND lance l'ordre suivant :

```
GRANT UPDATE
    ON TABLE T_CHAMBRE
    TO DUBOIS
```

Cet ordre va provoqué **une erreur**, car DURAND n'est pas autorisé à lancer des ordres UPDATE sur la table T_CHAMBRE et ne peut donc transmettre un droit qu'il n'a pas !

2.3. Gestion fine des privilèges

Est-il possible de gérer des privilèges plus fins que sur l'intégralité de la table ou de vue ? En particulier, peut-on gérer des privilèges au niveau de certaines colonnes d'une table ?
La réponse est OUI, mais il faut utiliser un peu d'astuce...

2.3.1. Privilèges INSERT et UPDATE sur colonne

On peut employer l'ordre GRANT pour ce faire :

```
GRANT UPDATE (CHB_POSTE_TEL, CHB_COUCHAGE)
    ON TABLE T_CHAMBRE
    TO DULAC ;
```

Cet ordre permet à DULAC de modifier uniquement les colonnes "poste téléphonique" et "nombre de place de couchage" de la table T_CHAMBRE.

Plus curieux, on peut définir les colonnes utilisables pour un ordre d'insertion pour un utilisateur :

```
GRANT INSERT (CHB_NUMERO, CHB_ETAGE, CHB_BAIN, CHB_DOUCHE, CHB_WC)
    ON TABLE T_CHAMBRE
    TO DULAC
```

Cet ordre permet à DULAC d'insérer une nouvelle ligne dans la table, uniquement en spécifiant les colonnes listées. Le problème est que dans cette liste ne figure pas la colonne clef... Autrement dit, DULAC ne pourra jamais rien insérer du tout, sauf si la clef est calculée par un déclencheur avant insertion.

NOTA : dans le cas de l'attribution de privilèges d'insertion sur colonne, il est indispensable de faire figurer toutes les colonnes NOT NULL n'ayant ni clause de valeur par défaut, ni remplissage par un trigger avant insertion. Sans cela cette autorisation est illusoire !

2.3.2. Privilèges SELECT sur colonne

Ce type de privilège n'est pas géré directement par un ordre SQL.
En effet, il n'est pas possible d'écrire :

```
GRANT SELECT (CHB_NUMERO, CHB_ETAGE, CHB_BAIN, CHB_DOUCHE, CHB_WC)
    ON TABLE T_CHAMBRE
    TO DULAC
```

Cet ordre n'est pas légal au niveau de SQL.

Mais un ordre GRANT peut porter sur une vue !
Nous voila donc sauvé : créer une vue pour gérer les privilèges de sélection de l'utilisateur DULAC..

```
CREATE VIEW V_CHAMBRE
AS
  SELECT CHB_NUMERO, CHB_ETAGE, CHB_BAIN, CHB_DOUCHE, CHB_WC
  FROM   T_CHAMBRE
```

```
GRANT SELECT
    ON V_CHAMBRE
    TO DULAC
```

2.5. Privilèges de référence

Lorsque l'on crée des tables en liaisons les unes aux autres, on utilise très souvent le mécanisme d'intégrité référentiel afin de gérer les clefs étrangères.

Voyons ce qui se passe si, dans notre base exemple, nous attribuons les droits ainsi :

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON TABLE T_CHAMBRE
    TO DUMONT
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON TABLE T_PLANNING
    TO DUMONT
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON TABLE TJ_CHB_PLN_CLI
    TO DUMONT
```

Dumont pourra sélectionner et supprimer sans problèmes dans toutes les tables. Il pourra mettre à jour les données et insérer sans aucun problème dans les les tables T_CHAMBRE et T_PLANNING.
En revanche il se heurtera parfois à un refus de la base de données pour la mise à jour de la table de jointure TJ_CHB_PLN_CLI. Pire , il lui sera impossible d'insérer des données dans cette dernière table...
Quel en est la raison ?

Regardons comment à été créée cette table de jointure :

```
 CREATE TABLE TJ_CHB_PLN_CLI
(   CHB_ID              INTEGER             NOT NULL,
    PLN_JOUR            DATE                not null,
    CLI_ID              INTEGER             not null,
    CHB_PLN_CLI_NB_PERS  SMALLINT             not null,
    CHB_PLN_CLI_RESERVE  NUMERIC(1)           not null       default 0,
    CHB_PLN_CLI_OCCUPE   NUMERIC(1)           not null       default 1,
    CONSTRAINT PK_TJ_CHB_PLN_CLI    PRIMARY KEY  (CHB_ID, PLN_JOUR) ,
    CONSTRAINT FK_CHB_ID            REFERENCES   T_CHAMBRE (CHB_ID) ,
    CONSTRAINT FK_PLN_JOUR          REFERENCES   T_PLANNING (PLN_JOUR) ,
    CONSTRAINT FK_CLI_ID            REFERENCES   T_CLIENT CLI_ID)
) ;
```

Elle utilise 3 tables en référence : T_CHAMBRE, T_PLANNING, T_CLIENT. Or notre utilisateur DUMONT n'a aucun privilège sur la table T_CLIENT. Il lui sera donc impossible lors de l'insertion, comme lors de la mise à jour de préciser une valeur pour cette colonne sans qu'il se voit automatiquement infligé un refus du serveur.

Or donc, pour pouvoir définir une valeur pour la colonne CLI_ID lors de l'exécution des ordres UPDATE et INSERT, notre utilisateur DUMONT, doit avoir un privilège supplémentaire définit comme suit :

```
 GRANT REFERENCES (CLI_ID)
     ON TABLE T_CLIENT
     TO DUMONT
```

NOTA : le privilège de référence ne porte pas exclusivement sur les contraintes d'intégrité mais sur toute contrainte faisant référence à une colonne d'une table externe.

3. Révocation des privilèges

L'ordre SQL REVOKE permet de révoquer, c'est à dire "retirer" un privilège.
Sa syntaxe est la suivante :

```
REVOKE [ GRANT OPTION FOR ] <privileges>
  FROM <gratifié> [ { , <gratifié> }... ]
  [ RESTRICT
    | CASCADE ]
```

```
  <privileges> ::=
    <privilège> [ { , <privilège> }... ]
```

```
   <privilege> ::=
      ALL PRIVILEGES ON <objet table>
    | <privilège d'action>
    | <privilege d'usage>
    | <privilege de référence>
```

```
    <privilège d'action> ::=
      <action> [ { , <action> }... ] ON <objet table>
```

```
     <privilege d'usage> ::=
       USAGE ON <objet d'usage>
```

```
    <privilege de référence> ::=
      REFERENCES [ ( <liste de colonne> ) ] <nom de table>
```

```
       <action> ::=
            SELECT
          | DELETE
          | INSERT [ ( <liste de colonne> ) ]
          | UPDATE [ ( <liste de colonne> ) ]
```

```
       <objet table> ::=
         [ TABLE ] <nom de table ou de vue>
```

```
       <objet d'usage> ::=
          DOMAIN <nom de domaine>
          | COLLATION <nom de collation>
          | CHARACTER SET <nom de jeu de caractères>
          | TRANSLATION <nom de translation>
```

```
  <gratifié> ::=
     PUBLIC
   | <utilisateur>
```

La grande différence réside en fait dans l'usage des mots clefs RESTRICT et CASCADE. En cas de RESTRICT, si le ou les utilisateurs visés ont cédés leurs droits à d'autres, un message d'erreur apparaîtra et le SGBDR refusera de révoquer le ou les droits. En revanche l'usage du mot clef CASCADE entrainera la révocation des droits cédés à la manière d'un chateau de carte.

A noter que l'utilisation de l'expression GRANT OPTION FOR ne révoke pas les droits mais supprime la possibilité de cession des droits lorsque ces droits ont été définis en utilisant la clause WITH GRANT OPTION.

3.1. Quelques exemples simples

```
REVOKE SELECT
    ON T_CHAMBRE
    FROM DUBOIS ;
```
Supprime le privilège de selection de la table T_CHAMBRE attribué à DUBOIS dans l'exemple 1.

```
REVOKE INSERT, DELETE
    ON TABLE T_CHAMBRE
    FROM DUVAL, DUBOIS ;
```
Supprime les privilèges d'insertion et de suppression de la table T_CHAMBRE attribué à DUVAL et DUBOIS dans l'exemple 2, mais pas celui de mise à jour (UPDATE).

```
REVOKE GRANT OPTION FOR SELECT
    ON TABLE T_CMAMBRE
    FROM DUFOUR ;
```
Supprime la possibilité pour DUFOUR de transmettre le privilège de sélection sur la table T_CHAMBRE.

3.2. Problématique de révocation

Il existe cependant quelques pièges dans l'utilisation du mécanisme de révocation. Nous allons en montrer quelques uns à l'aide de différents exemples. Rapellons simplement que celui qui lance les ordres (sauf indication contraire) est l'utilisateur DUHAMEL.

Contrairement aux droits "systèmes" les privilèges sont cumulatifs. On peut ainsi obtenir plusieurs fois le même privilège sur le même objet en provenance de différents utilisateurs. Le privilège sera totalement retiré lorsque tous les utilisateurs ayant donné ce privilége l'auront retiré.

```
GRANT SELECT
    ON T_CLIENT
    TO DUCROS WITH GRANT OPTION
```

```
GRANT SELECT
    ON T_CLIENT
    TO DUGLAND
```

C'est maintenant DUCROS qui est l'utilisateur qui va lancer l'ordre suivant :

```
GRANT SELECT
    ON T_CLIENT
    TO DUGLAND ;
```
Enfin, DUHAMEL reprend la main pour révoquer ainsi :

```
REVOKE SELECT
    ON T_CLIENT
    FROM DUGLAND;
```
DUGLAND peut-il sélectionner des lignes de la table T_CLIENT ? La réponse est OUI, par ce qu'il possède encore un droit de sélection venant de DUCROS !

4. Retrouver les privilèges

Les vues d'information de schema permettent de retrouver les privilèges, les objets et les utilisateur visés (originaires et destinataires). Pour cela la norme SQL 2 à prévue 3 vues spécifiques :

INFORMATION_SCHEMA	Colonnes
TABLE_PRIVILEGES	GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
COLUMN_PRIVILEGES	GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
USAGE_PRIVILEGES	GRANTOR, GRANTEE, OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,OBJECT_TYPE, PRIVILEGE_TYPE, IS_GRANTABLE

```
SELECT GRANTOR, GRANTEE, TABLE_NAME, ' <TABLE>' AS COLUMN_NAME, PRIVILEGE_TYPE,
IS_GRANTABLE
FROM   INFORMATION_SCHEMA.TABLE_PRIVILEGES
WHERE  GRANTEE IN ('DUBOIS', 'DUVAL', 'DULAC')
UNION
SELECT GRANTOR, GRANTEE, TABLE_NAME, COLUMN_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
FROM   INFORMATION_SCHEMA.COLUMN_PRIVILEGES
WHERE  GRANTEE IN ('DUBOIS', 'DULAC', 'DUVAL)
ORDER  BY GRANTEE, TABLE_NAME, PRIVILEGE_TYPE, COLUMN_NAME;
```

```
GRANTOR     GRANTEE   TABLE_NAME     COLUMN_NAME     PRIVILEGE_TYPE IS_GRANTABLE
----------- --------- -------------- --------------- -------------- ------------
DUHAMEL     DUBOIS    T_CHAMBRE      <TABLE>         SELECT         False
DUHAMEL     DUBOIS    T_CHAMBRE      <TABLE>         INSERT         False
DUHAMEL     DUBOIS    T_CHAMBRE      <TABLE>         UPDATE         False
DUHAMEL     DUBOIS    T_CHAMBRE      <TABLE>         DELETE         False
DUHAMEL     DUVAL     T_CHAMBRE      <TABLE>         INSERT         False
DUHAMEL     DUVAL     T_CHAMBRE      <TABLE>         UPDATE         False
DUHAMEL     DUVAL     T_CHAMBRE      <TABLE>         DELETE         False
DURAND      DUBOIS    T_CHAMBRE      <TABLE>         SELECT         False
DURAND      DUBOIS    T_CHAMBRE      <TABLE>         INSERT         False
DURAND      DUBOIS    T_CHAMBRE      <TABLE>         UPDATE         False
DURAND      DUBOIS    T_CHAMBRE      <TABLE>         DELETE         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_POSTE_TEL   UPDATE         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_COUCHAGE    UPDATE         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_NUMERO      INSERT         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_ETAGE       INSERT         Fals
DUHAMEL     DULAC     T_CHAMBRE      CHB_BAIN        INSERT         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_DOUCHE      INSERT         False
DUHAMEL     DULAC     T_CHAMBRE      CHB_WC          INSERT         False
DUHAMEL     DULAC     V_CHAMBRE      <TABLE>         SELECT         False
```

Si vous avez suivi à la lettre tous les exemples d'octroi de privilèges jusqu'à l'exemple 13, alors la requête ci avant donnera le résultat ci dessus...

5. Critiques diverses

Pour aussi simple qu'il soit, ce système qui revèle quelques pièges est assez complet mais insatisfaisant.
Voyons quels en sont les limites...

5.1. Pas de privilèges simultanés sur plusieurs objets

Ainsi il n'est pas possible de d'octroyer des privilèges à plusieurs objet simultanément :

```
GRANT INSERT, DELETE
    ON TABLE T_CHAMBRE, T_CLIENT, T_PLANNING
    FOR DUVAL, DUBOIS ;
```

Un tel ordre est syntaxiquement incorrect du fait de la présence de plusieurs tables...
La conséquence est que la gestion fine des droits de 5 utilisateurs sur une base de données comportant une centaine de table se traduit en au moins une centaines d'ordres à passer...

5.2. Pas de privilèges "négatif"

Il n'est pas possible d'appliquer un privilège de déniement à un objet. Cela n'est pas définit par la norme SQL.

```
DENY DELETE
   TO DUVAL, DUBOIS
```

N'existe pas en SQL. Ce serait pourtant bien pratique de passer par un tel mécanisme...