# RT 1A - Bases de données - TP3

Samir Loudni

IUT RT Caen

## 1 Objectifs

Les objectifs de ce TP sont :
- la création, l'interrogation et la manipulation de vues mono-table et multi-tables ;
- la gestion des droits d'accès ;

## 2 Généralités sur les vues

Une vue est une table virtuelle sans existence propre. Elle est constituée à partir du résultat d'une clause `SELECT` sur une table réelle. Les vues sont utilisées pour :
- modifier la présentation de la base de données à certains utilisateurs. On parlera alors de schéma externe (en opposition au schéma physique correspondant aux tables) ;
- simplifier la construction de certaines requétes difficiles ;
- assurer la confidentialité de la base de données en construisant des sous-schémas adaptés à chaque utilisateur.

### 2.1 Syntaxe de `CREATE VIEW`

Il existe deux types de vues :
- **les vues mono-table sans contrainte** :
  `CREATE[OR REPLACE] VIEW` nom_de_la_vue(colonne_1, ..., colonne_n)
  `AS SELECT` $a_1$ ,..., $a_n$
  `FROM` nom_de_table_source `WHERE` ...
  `[ WITH CHECK OPTION]` ;
- **les vues multi-table sans contrainte** :
  `CREATE[OR REPLACE] VIEW` nom_de_la_vue(colonne_1, ..., colonne_n)
  `AS SELECT` $t_1.a_1$ ,..., $t_n.a_n$
  `FROM` table_source_1 $t_1$, table_source_2 $t_2$, ... `WHERE` ...
  `[WITH CHECK OPTION]` ;

**Remarques** :
- Si la vue doit avoir la même structure que la table source, vous pouvez vous servir de * dans la requête `SELECT` et ne pas donner de noms aux attributs de la vue (ils porteront les mêmes noms que les attributs de la table). Dans le cas contraire les colonnes du `SELECT` donnera la structure de la vue à créer (les noms des colonnes de la vue et de la table source ne sont pas forcément les mêmes) ;
- Penser à mettre l'ordre `DROP VIEW` avant l'ordre `CREATE VIEW` afin de pouvoir recréer la vue à la demande ;
- L'option `WITH CHECK OPTION` de la commande `CREATE VIEW` permet de définir des contraintes spécifiques pour chaque vue (le prédicat servant alors de contrainte). Ainsi, lors de la manipulation d'une vue mono-table par `INSERT`, `UPDATE` ou `DELETE`, ces ordres sont exécutés dans la table sur laquelle la vue porte si les enregistrements respectent la contrainte posée sur la vue.

### 2.2 Vues mono-table sans contrainte

1. Créer la vue `LOGICIELS_UNIX` à partir de la table `LOGICIEL` ne contenant que les logiciels de type `UNIX`.

   Toutes les colonnes sont conservées et portent le même nom que les colonnes de la table. Interroger la vue avec (`DESC` et `SELECT`).

2. Créer la vue `POSTE_0` à partir de la table `POSTE` ne contenant que les postes du rez-de-chaussée (restriction avec `ETAGE = 0` de la table `SEGMENT`). Faire une jointure de la forme procédurale (en utilisant l'opérateur `IN`) sinon la vue sera considérée comme une vue multi-table. Toutes les colonnes ne sont pas conservées et leurs ordres et noms changent :

   $$\texttt{POSTE\_0}(\texttt{NPOS\_0}, \texttt{NSAL\_0}, \texttt{TYP\_0}, \texttt{NSEG\_0}, \texttt{A\_0})$$

   Vérifier la structure et contenu de la vue avec `DESC` et `SELECT`.

3. Insérer avec `INSERT` deux nouveaux postes dans la vue (`p15` et `p16` par exemple, tels que `p15` appartienne au segment de l'étage 0 et `p16` à un segment d'un autre étage).
   - Vérifier le contenu de la vue et celui de la table. Conclusion ?
   - Supprimer ensuite ces deux enregistrements de la table `POSTE`.
   - Vérifier à nouveau le contenu de la vue. Conclusion ?

4. Créer la vue `SALLE_PRIX` à partir de la table `SALLE`. Cette vue doit contenir toutes les salles avec le prix de location pour une journée (en fonction du nombre de postes, le montant de la location d'une salle à la journée est calculé sur la base de 100 Euros par poste présent). Servez-vous de `100*NB_POSTE` dans la requête `SELECT`.

   $$\texttt{SALLE\_PRIX}(\texttt{N\_SALLE}, \texttt{NOM\_S}, \texttt{NB\_POSTE}, \texttt{PRIX\_LOCATION})$$

   - Vérifier le contenu de la vue,
   - Afficher les salles dont le prix de location dépasse un montant donné.

5. Ajouter la colonne `TARIF` de type INT(3) à la table `TYPE`. Mettre à jour cette table avec l'ordre `UPDATE` de manière à modifier les valeurs suivantes :

   | TYP_LP | TARIF |
   |---|---|
   | TX | 50 |
   | PCWS | 100 |
   | PCNT | 120 |
   | UNIX | 200 |
   | NC | 80 |
   | BeOS | 400 |

   - Créer la vue `SALLE_1(SALLE, TYPEP, NOMBRE, TARIF)` telle que :

   - Le contenu de la vue doit être le suivant :

```
mysql> SELECT * FROM salle_1;

N_SALLE   TYPEP   NOMBRE     TARIF
-------   ------  ---------  -----
s01       TX      2          50
s01       UNIX    1          200
s02       PCWS    2          100
s03       TX      1          50
...
```

- A partir de cette vue créer la vue

SALLE_PRIX_TOTAL(N_SALLE, PRIX_LOCATION)

- Cette vue devra tenir compte des types de postes de travail présents dans les salles (par exemple la salle 1 sera facturé $2 * 50 + 1 * 200 = 300$Euros). Vérifier le contenu de cette vue.

### 2.3 Vues multi-tables sans contrainte

1. Créer la vue SALLE_POSTE permettant d'avoir la liste des installations des postes dans toutes les salles sous la forme suivante :

```
mysql> SELECT * FROM salle_poste;

NOM_S        NOM_POSTE         ADRESSE_IP           NOM_TYPE_POSTE
-------------------------------------------------------------------
Salle 1      Poste 1           130.120.80.01        Terminal X-Window
Salle 1      Poste 2           130.120.80.02        Système Unix
Salle 1      Poste 3           130.120.80.03        Terminal X-Window
```

### 2.4 Vues mono-table avec contrainte

1. Reprendre la vue POSTE_0 de la deuxième question en rajoutant l'option de contrôle. Tenter d'insérer un poste appartenant à un étage différent de 0.

2. Créer la vue INSTALLER_0(N_POSTE, N_LOG, DATE_INS) ne permettant de travailler qu'avec les postes du rez-de-chaussée. Interdire aussi l'installation d'un logiciel de type PCNT.

3. Tenter d'insérer deux postes dans cette vue ne correspondant pas à ces deux contraintes (pour DATE_INS, utilisez current-date qui renvoi la date système) :
   – un poste d'un étage,
   – puis un logiciel de type PCNT.

## 3 Droits d'accès (GRANT et REVOKE)

### 3.1 Comment fonctionne le système de droits

Le contrôle d'accès de MySQL se fait en deux étapes :
- Étape 1 : Le serveur vérifie que vous êtes autorisé à vous connecter.
- Étape 2 : Une fois la connexion établie, le serveur vérifie chaque requête que vous soumettez, pour vérifier si vous avez les droits suffisants pour l'exécuter.

Les droits des utilisateurs sont stockés dans les tables user, db, host, tables_priv et columns_priv de la base mysql.

### 3.2 Vérification de la requête et contrôle d'accès

Pour chaque requête, le serveur vérifie si vous avez les droits suffisants pour exécuter une commande, en fonction du type de commande. Ces droits peuvent provenir des tables indiquées ci-dessus. Ces tables sont manipulées avec les commandes GRANT et REVOKE :

- La table user donne les droits aux utilisateurs au niveau global, c'est à dire que ces droits s'appliquent quelle que soit la base de données courante.
- La table db donnent des droits au niveau des bases. Les champs de la table déterminent quels utilisateurs peuvent accéder à quelles bases, depuis quel hôte. Les champs de droits indiquent alors les opérations permises.

- La table `host` est utilisée comme extension de la table `db` lorsque vous voulez qu'une ligne de la table `db` s'applique à plusieurs hôtes. Par exemple, si vous voulez qu'un utilisateur soit capable d'utiliser une base depuis plusieurs hôtes dans votre réseau, il suffit de laisser la colonne Host vide dans la table `db`.
- Les tables `tables_priv` et `columns_priv` spécifient les droits au niveau des tables et des colonnes, plutôt qu'au niveau des bases.

### 3.3 La commande `GRANT`

Considérons un utilisateur userRT1 qui veut donner un privilège (GRANT) à userRT2 sur sa base T2. Le privilège peut être (SELECT, UPDATE, INSERT, DELETE...) :

`GRANT` privilège `ON` T2.* `TO` 'userRT2' ;

1. Travailler par binôme ,donner à votre binôme (userRT_i) le droit de travailler sur votre table `TYPE` en lecture (SELECT).
   - userRT_i affiche la table de userRT_j
   - userRT_i essaye de modifier la table de userRT_j en ajoutant par exemple un nouveau type (faire préfixer le nom de la table par son propriétaire dans la clause INSERT).
   - Quel est le message d'erreur ?
2. Donner à votre binôme (userRT_i) le droit de travailler sur votre table TYPE en modification (UPDATE).
   - userRT_i modifie la table de userRT_j en changeant par exemple un nom de type
   - userRT_i réaffiche la table de userRT_j
   - userRT_j affiche sa table, que se passe t'il et pourquoi ?
   - userRT_j exécute la même modification que userRT_i a fait
   - userRT_j réaffiche la table, que se passe t'il et pourquoi ?