Université Paris 13 L2
Institut Galilée Année 2014-2015
Licence 2 - 2eme semestre

Traitement de données appliqué à la finance
TP5 - SQL avec OOBase
Insertion de données avec SQL

Dans ce TP, vous utiliserez Open Office Base afin d'insérer des données dans une base à partir d'un fichier texte, et de répartir ces données dans les différentes tables de votre modèle. Tous les fichiers sont situés ici : `http://goo.gl/Uig9PB`

1 Mini base : étudiants

Pour commencer de façon simple, nous allons étudier le fichier *mini_ etudiant.csv*. Ouvrez le fichier et étudiez le dans un éditeur de texte (ou dans un tableur). Le but sera de ranger les données de ce fichier dans différentes tables.

1.1 Construction de tables

Cette base met en relation des Etudiants et des UE : chaque étudiant, repéré par son numéro et son nom, est inscrit à différentes UE, identifiées par leur nom et leur code. Commencez par créer une nouvelle base de données et y construire trois tables :

```
Etudiant(numero_etud, nom_etud)
UE(code_ue, matiere_ue)
Inscription(numero_etud, code_ue)
```

1.2 Importation du fichier csv

Créez maintenant une nouvelle base de données qui ne servira qu'à importer le fichier CSV précédemment téléchargé. Pour cela, commencez par **placer le fichier tout seul dans un nouveau dossier** (à l'aide de l'explorateur de fichier). Ensuite, allez dans le menu *Fichier > Nouveau > Base de données*, et sélectionnez la troisième option *Se connecter à une base de données existante*. Sélectionnez dans la liste *Texte*, puis cliquez sur *Suivant*.

Dans l'écran suivant, sélectionnez bien l'option *Fichiers *.csv*, et sélectionnez la point virgule **comme séparateur de champs**. Cliquez ensuite sur *Parcourir*, et sélectionnez le dossier contenant le fichier précédemment téléchargé. Ensuite, cliquez sur *Terminer*, enregistrez la base de données dans un nouveau fichier (temporaire) et validez.

Une nouvelle base de données s'ouvre, avec un table s'appelant *mini_ etudiants*. Faîtes glisser cette table vers votre base de début (celle contenant les trois tables) : une nouvelle fenêtre s'affiche, cliquez alors sur le bouton *Creer*. Si le programme vous demande de créer une clef primaire, répondez par la négative. Votre fichier CSV, sous forme de table de données, est maintenant importé dans votre table.

1.3 Transfert des données vers *Etudiants* et *UE*

Nous allons commencer par remplir la table *Etudiants* avec les données issues de *mini_ etudiants*. Pour ce faire, commencez par construire une requête permettant de sélectionner tous les étudiants présents dans le base *mini_ etudiants*. Créer une nouvelle requête en mode SQL, et écrivez

```
SELECT "Nom Etudiant", "Num Etudiant" FROM "mini_etudiants"
```

Le résultat de la requête devrait produire 17 résultats, alors que nous n'avons que 6 étudiants ! Comme vous pouvez le voir, nous avons de nombreux doublons... Pour les éviter, nous allons demande à n'afficher qu'une seule fois les résultats dupliqués :

```
SELECT DISTINCT "Nom Etudiant", "Num Etudiant" FROM "mini_etudiants"
```

La requête devrait cette fois-ci contenir 6 résultats, qui sont les étudiants qui nous intéressent. Nous allons utiliser le résultat de cette requête pour insérer des données dans la table *Etudiant*. Pour ce faire, nous utiliserons une requête INSERT à laquelle nous combinerons le SELECT précédent. Comme il s'agit d'un INSERT, nous ne pouvons pas l'exécuter en faisant une nouvelle requête, il faut passer par le menu *Outils* > *SQL* :

```
INSERT INTO "Etudiant" ("numero_etud", "nom_etud")
SELECT DISTINCT "Nom Etudiant", "Num Etudiant" FROM "mini_etudiants"
```

Flûte, ça ne fonctionne pas ! En effet, le SELECT choisit d'abord la colonne de nom puis de numéro de l'étudiant, tandis que le INSERT insère d'abord les données dans la colonne de numéro puis de nom de l'étudiant. Modifiez la requête précédente afin que l'INSERT fonctionne et vérifiez que les 6 étudiants ont bien été insérés dans la table *Etudiant*. Sauvegardez votre requête dans une nouvelle requête (afin de la conserver quelque part).

Répétez (et adaptez) les opérations précédentes afin d'importer, dans la table UE, les différentes UE de notre jeu de données : procédez de la même façon, c'est à dire commencez par construire la requête servant à sélectionner les données que vous souhaitez, puis insérez-les. Vérifiez votre insertion en examinant la table *UE*.

Sauvegardez votre requête d'insertion à la suite de la précédente requête, afin de les conserver dans un même fichier.

1.4 Construction de l'association

Pour importer les données vers l'association, il suffit de sélectionner, dans la table *mini_ etudiants*, les colonnes de numéro de l'étudiant et de nom de l'UE, et de les insérer dans la relation *Inscription* :

```
INSERT INTO "Inscription" ("code_ue", "numero_etud")
SELECT DISTINCT "Code UE", "Num Etudiant" FROM "mini_etudiants"
```

Vérifiez votre insertion en examinant la table *Inscription*. Une fois l'insertion réalisée avec succès, vous pouvez supprimer la table *mini_ etudiants* de votre base.

1.5 De nouvelles données

Récupérez les données du fichier*mini_ etudiants2.csv*.

En répétant les mêmes commandes que précédemment, importez le fichier comme une table dans votre base de données, puis utilisez la commande INSERT élaborée précédemment afin d'insérer les étudiants dans la table *Etudiant* (pensez à modifier le nom *mini_ etudiants* en *mini_ etudiants2*) : est-ce que cela a fonctionné ?

En effet, des étudiants déjà présents dans la base *Etudiant* sont présents dans le fichier *mini_ etudiants2*, et votre commande tente de les insérer de nouveau, ce qui est interdit. Il va d'abord falloir modifier la commande SELECT afin de ne sélectionner, dans *mini_ etudiants2*, que les étudiants qui ne sont pas déjà présents dans *Etudiant* : ceci se fait avec la commande EXCEPT.

```
SELECT DISTINCT "Nom Etudiant", "Num Etudiant" FROM "mini_etudiants2"
EXCEPT
SELECT DISTINCT "nom_etud", "numero_etud" FROM "Etudiant"
```

Il suffit d'incorporer cette commande SELECT à une commande INSERT afin d'insérer uniquement les nouveaux étudiants dans la table *Etudiant* :

```
INSERT INTO "Etudiant" ("nom_etud", "numero_etud")
SELECT DISTINCT "Nom Etudiant", "Num Etudiant" FROM "mini_etudiants2"
EXCEPT
SELECT DISTINCT "nom_etud", "numero_etud" FROM "Etudiant"
```

Pour incorporer les nouvelles unités à la table UE, nous pourrions faire exactement la même chose. Néanmoins, nous allons procéder avec une autre commande produisant les mêmes résultats que le EXCLUDE (simplement pour essayer... en pratique, choisissez la méthode que vous préférez) :

```
SELECT DISTINCT "Code UE", "Nom UE" FROM "mini_etudiants2"
WHERE "Code UE" NOT IN (
        SELECT "code_ue" FROM "UE")
```

Cette commande devrait produire une seule nouvelle unité. Enveloppez-la dans un INSERT afin d'ajouter la nouvelle unité à la base des unités (et vérifiez ensuite que la commande a fonctionné).

Faîtes de même pour ajouter les nouvelles associations à la table *Inscription* : vous devriez obtenir en tout 22 enregistrements dans la base *Inscription*.

1.6 Quelques commandes SQL

Afin de vérifier que vos données sont complètes, construisez les requêtes suivantes :

1. La liste des noms de matières choisies par Lucas. *Vous devriez obtenir deux réponses.*
2. La liste des noms d'élèves ayant choisi les Maths. *Vous devriez obtenir quatre réponses.*
3. La liste des noms d'élèves n'ayant pas choisi l'Anglais. *Vous devriez obtenir quatre réponses.*
4. La liste des noms d'élèves ayant choisi les Maths mais pas l'Anglais. *Vous devriez obtenir trois réponses.*
5. Je cherche à obtenir la liste des étudiants ayant choisi Coreen ou Espagnol... Que pensez-vous de cette commande ? Pourquoi ne fonctionne-t-elle pas ? Corrigez-la. *Vous devriez obtenir cinq réponses.*

```
SELECT "Etudiant"."nom_etud" FROM "Etudiant"
WHERE "UE"."code_ue" = "Inscription"."code_ue"
AND "Inscription"."numero_etud" = "Etudiant"."num_etud"
AND ( "UE"."matiere_ue" = 'Coreen' OR
"UE"."matiere_ue" = 'Espagnol' )
```

2 Base de données des vols d'avions

Nous allons tenter de faire de nouveau la même chose : importer un fichier de données sous format CSV, et répartir ses informations dans différentes tables.

2.1 Analyser les données et construire un modèle Entité-Association, puis un schéma relationnel

Analysez le fichier de données *avion_data.csv* afin de déduire un modèle E/A puis un schéma relationnel. Voici certains éléments à prendre en compte dans ce travail :

- On ne sera pas totalement en 1FN dans ce travail (ce ne sera pas l'objectif) : on enregistrera par exemple directement nom et prénom des pilotes. Comme on ne divisera pas les colonnes du fichier original en sous-colonnes, pouvez-vous indiquer quelles données vous empêcheront d'être en 1FN (nous verrons plus tard comment résoudre ce problème) ?

- Il sera nécessaire d'avoir une table regroupant le nom des villes.
- Les avions appartiennent à une seule compagnie aérienne. Pour les pilotes, ce n'est pas forcément le cas.

Une fois votre modèle E/A et votre schéma relationnel prêt, construisez vos tables dans OOBase.

2.2 Importer le fichier dans le programme

Importez les données du fichier textitavion_data.csv, et répartissez les informations dans vos tables. Gardez en tête que vous serez amené à insérer des données supplémentaires avec les mêmes commandes.

Attention Quand vous importez un fichier CSV avec Base en utilisant la méthode décrite en début de ce TP, il ne vous est pas possible de préciser l'encodage du fichier, ce qui va poser problème dans cet exercice. Pour gérer ce problème, il faut tout d'abord ouvrir le fichier CSV dans Calc, et choisir le bon encodage. Ensuite, enregistrez le fichier en tant que feuille de calcul (.ods).

Suivez ensuite toutes les étapes d'importation du fichier comme décrit dans la première partie, mis à part qu'il faudra choisir, dans la première liste en dessous de *Se connecter à une base de données existante*, l'option *Classeur* et non pas l'option *Texte*.

2.3 Commandes SQL

Afin de s'assurer que l'import s'est bien passé, voici quelques commandes SQL à tester :

1. Affichez les informations des vols (Ville de départ, Ville d'arrivée, Compagnie) effectués par *Inès Lacroix. Vous devriez obtenir trois résultats.*
2. Comptez le nombre de vols assurés par *Korean Air. Vous devriez en obtenir 14. Attention, on souhaite avoir directement le nombre de vols comme réponse de la requête : la commande SELECT COUNT devrait vous aider.*
3. Comptez le nombre de vols assurés par une compagnie américaine. *Vous devriez en obtenir 171.*
4. Affichez les numéros de vols où l'avion atterrit dans sa ville de localisation. *Vous devriez en obtenir 4.*
5. Affichez les noms des pilotes qui ont déjà piloté deux avions (ou plus) de deux compagnies aériennes différentes. *Vous devriez en obtenir 2.*

2.4 Fichier et commandes SQL supplémentaires

Importez les données du fichier textitavion_data2.csv dans votre base :

Afin de s'assurer que l'import s'est bien passé, testez les commandes SQL précédemment écrites. *Vous devriez obtenir respectivement 6 vols, 25 vols, 231 vols, 6 vols et 3 pilotes.*