1 Création et manipulation d'une base de données

Dans ce TP, nous allons créer la base de données de l'exercice 3 du TD 4 d'algèbre relationnelle. Nous rappelons que cette base est constituée des relations suivantes :

Tables
$Produits(Constructeur : caractere, \underline{Modele} : entier, Type : chaine)$
$PC(\underline{Modele} : entier, Vitesse : entier, RAM : entier, HD : reel, CD : chaine, Prix : reel)$
$Portables(\underline{Modele} : entier, Vitesse : entier, RAM : entier, HD : reel, Ecran : reel, Prix : reel)$
$Imprimantes(\underline{Modele} : entier, Couleur : booleen, Type : chaine, Prix : reel)$

Contraintes
$Les\ clefs\ donnees\ ci-dessus$
$Pas\ de\ valeur\ nulle$
$PC[Modele] \subseteq Produits[modele]$
$Portables[Modele] \subseteq Produits[modele]$
$Imprimantes[Modele] \subseteq Produits[modele]$
$dom(Produits.Type) = \{laptop, pc, printer\}$
$dom(PC.CD) = \{6x, 8x\}$
$dom(Imprimantes.Type) = \{dry, ink-jet, laser\}$

1.1 Connexion à votre base de données `ORACLE`

Pour manipuler votre base de données, vous devez utiliser l'interpréteur de commande d'`ORACLE` : *sqlplus*. Pour lancer sqlplus, utilisez la commande suivante : *sqlplus* (votre login est aussi votre mot de passe). Vous êtes dans l'interpréteur `SQL`, et maintenant, les seules commandes acceptées sont des commandes `SQL`. Vous pouvez obtenir la liste des commandes SQL en tapant *help index* et l'aide pour une commande particulière en tapant *help nom_commande*.

1.2 Manipulation des tables via le SGBD

Nous allons voir comment créer une table, la modifier et la supprimer.

1.2.1 Création d'une table

La création d'une table s'effectue grâce à la commande *create table*. Celle-ci s'utilise de la façon suivante :

CREATE TABLE nom_table ($attribut_1$ $type_1, \ldots, attribut_n$ $type_n$);

Vous utiliserez les types de données *int*, *float* et $varchar2(longueur\ max)$.

Créez à l'aide de cette commande les tables de la base (sans clef ni contrainte). Tapez $select * from\ tab$; pour obtenir un descriptif du contenu courant de la base. Vous pouvez taper $describe\ nom\_table$; pour obtenir le descritif d'une table.

Insérer les trois premiers tuples (voir énoncé du TD) de la table *produits* à l'aide de la commande :

INSERT INTO nom_table VALUES ($val\_attribut_1, \ldots, val\_attribut_n$);

Nous rappelons que pour voir le contenu d'une table vous pouvez utiliser la requête : SELECT * FROM nom_table;.

Changer le nom du constructeur du premier tuple à l'aide de la commande :

UPDATE *nom_table* SET *attribut* = *nouvelle_valeur* WHERE *condition*;

La commande UPDATE instancie la valeur de *attribut* à *nouvelle_valeur* pour tous les tuples vérifiant la condition.

1.2.2 Modification d'une table – Ajout de contraintes

À l'aide de la commande ALTER TABLE, modifiez les tables que vous venez de créer afin d'y indiquer les clefs primaires et étrangères, les droits à la valeur nulle, ainsi que les contraintes de domaine d'attribut.

Notez qu'une clef ne doit pas pouvoir prendre la valeur null. Si vous n'avez pas indiqué, lors de la création d'une table, que la valeur null est interdite pour les attributs utilisez la commande suivante :

ALTER TABLE *nom_table* MODIFY *attribut* NOT NULL;

La syntaxe à utiliser pour les clefs primaires est :

ALTER TABLE *nom_table* ADD PRIMARY KEY (*attribut*);

La syntaxe à utiliser pour les clefs étrangères est :

ALTER TABLE nom_table_1 ADD FOREIGN KEY ($attibut_1$) REFERENCES nom_table_2 ($attribut_2$);

1.2.3 Suppression d'une table

La suppression d'une table s'effectue grâce à la commande DROP TABLE *nom_table*.
Supprimez toutes vos tables en commençant par produit. Que se passe-t-il ? Pourquoi ?

1.3 Remplissage d'une base de données

Récupérez et lisez le fichier `creation.sql` sur la page web de votre chargé de TP. Ce fichier contient les commandes de création des tables de la base de données. Chargez ce fichier depuis *sqlplus* grâce à la commande *start creation.sql*. Affichez le contenu de la base.

Insérez le premier n-uplet de chacune des tables *produits*, *pc*, *portables* et *imprimantes*, *dans cet ordre*. Que constatez-vous ? Pourquoi ?

Insérer les trois premiers n-ulplets de la table *imprimantes* en insérant d'abord les tuples appropriés dans la table *produits*.

Supprimer les n-uplets du constructeur *D* de la table *produits* en utilisant la commande :

DELETE FROM *nom_table* WHERE *condition*

Que constatez-vous ? Pourquoi ?

Supprimez tous les tuples que vous avez insérés dans les tables (sans supprimer les tables).

Récupérez le fichier `remplissage.sql` sur la page web de votre chargé de TP. Ce fichier contient les commandes de remplissage des tables de la base de données. Lisez et chargez ce fichier.

2 Interrogation d'une base de données

Sur la base de données précédemment créée, nous allons poser en SQL les requêtes suivantes.

1. Quels sont les portables (laptop en anglais) référencés dans la relation Produits ?
   La réponse doit être :

```
 modele
--------
   2001   2008
```

```
   2002   2005
   2003   2006
   2004   2007
(8 lignes)
```

2. Quelle est la liste des constructeurs ?
   La réponse doit être :

```
 constructeur
--------------
 A
 B
 C
 D
 E
 F
 G
 H
 I
(9 lignes)
```

3. Quels sont les modèles des ordinateurs référencés ?
   La réponse doit être :

```
 modele
--------
   1001   1010
   1002   2001
   1003   2002
   1004   2003
   1006   2004
   1005   2008
   1007   2005
   1008   2006
   1009   2007
(18 lignes)
```

4. Quels sont les PC avec 256 Mo de RAM ?
   La réponse doit être :

```
 MODELE
----------
      1001
      1002
      1005
      1010
(4 lignes)
```

5. Quels sont les PC à exactement moins de 1100 Euros ?
   La réponse doit être :

```
    MODELE
----------
      1001
      1002
      1003
      1004
      1005
      1010
(6 lignes)
```

6. Quels sont les portables avec un disque d'au moins (ou égal à) 50 Go et à moins de (ou égal à) 1400 Euros ?
   La réponse doit être :

```
 modele
--------
   2001
   2004
   2005
   2008
(4 lignes)
```

7. Quelles sont les imprimantes laser couleur ?
   La réponse doit être :

```
 modele
--------
   3004
(1 ligne)
```

8. Quels sont les portables du constructeur D ayant au moins 256 Mo de RAM ?
   La réponse doit être :

```
modele
--------
   2001
   2003
(2 lignes)
```

9. Quels sont les ordinateurs du constructeur D ayant au moins 256 Mo de RAM ?
   La réponse doit être :

```
 modele
--------
   1008
   1009
   1010
   2001
   2003
(5 lignes)
```

10. Quels sont les ordinateurs coûtant exactement moins de 1200 Euros ?
    La réponse doit être :

```
 MODELE
----------
      1001
      1002
      1003
      1004
      1005
      1006
      1010
      2001
      2006
      2008
(10 lignes)
```

11. Quels sont les constructeurs fournissant des imprimantes et des pc ?
    La réponse doit être :

```
 constructeur
--------------
 B
 D
(2 lignes)
```

12. Quels sont les constructeurs couvrant tous les types de produits ?
    La réponse doit être :

```
 constructeur
--------------
 D
(1 ligne)
```

13. Quels sont les constructeurs ne fournissant que des PC ?
La réponse doit être :

```
 constructeur
--------------
 A
 C
(2 lignes)
```

14. Quels sont les constructeurs ne fournissant qu'un type de materiel ?
La réponse doit être :

```
 constructeur
--------------
 A
 C
 E
 F
 G
 H
 I
(7 lignes)
```

15. Quels sont les constructeurs disposant de pc avec au moins 1024 Mo de RAM, un cd 56x et d'un portable avec un écran supérieur à 15 pouces et pouvant fournir une imprimante à moins de 300 Euros ?
La réponse doit être :

```
 constructeur
--------------
D
(1 ligne)
```

16. Sachant que pour faire de la PAO il faut au moins 1024 Mo de RAM, une vitesse de 3 Ghz ou 3.5 , 160 Go minimum de hd ainsi qu'une imprimante couleur, quel est le constructeur fournissant cette configuration et à quel prix ?
La réponse doit être :

```
 constructeur | prix
--------------+------
 B            | 1168
 B            | 1490
(2 lignes)
```

17. Quelles sont les tailles de disques durs qui sont utilisées par au moins 2 PC ?
La réponse doit être :

```
 hd
-----
 120
 160
 180
(3 lignes)
```

18. Quels sont les constructeurs qui vendent au moins deux ordinateurs differents avec une fréquence d'au moins 3 Ghz ?
La réponse doit être :

```
 constructeur
--------------
 A
 B
 C
 D
(4 lignes)
```

19. Quels sont les constructeurs vendant l'ordinateur le plus rapide ?
    La réponse doit être :

```
 constructeur
--------------
 B
 C
 D
(3 lignes)
```

20. Quels sont les constructeurs vendant exactement trois types de PC ?
    La réponse doit être :

```
 constructeur
--------------
 A
 D
(2 lignes)
```

3 Réorganisation, schémas et tables

3.1 Réorganisation

Réorganiser le travail à faire selon les fichiers suivants :
- `creation.sql` contenant la définition du schéma de la base de données relationnelle en prenant en compte ses contraintes. Afin de pouvoir travailler en groupe lors de ce TP et échanger les données avec les membres de votre groupe, respecter les noms des relations et attributs de l'énoncé,
- `insertion.sql` contenant les n-uplets,
- `affichage.sql` permettant d'afficher le contenu des tables créées,
- `requete.sql` pour les requêtes qui auront dû être testées une à une,
- `nettoyage.sql` dont l'exécution permet de détruire toutes les tables que vous avez créés.

3.2 Connection

Connectez vous à Oracle depuis deux fenêtres de terminal distinctes (on les appelera `F1` et `F2`).

1) Exécuter dans `F1` et dans `F2`, votre fichier `nettoyage.sql` pour détruire toutes vos tables, et vérifier sur `F1` et sur `F2` que vos tables sont bien détruites.
2) Exécuter seulement dans `F1`, vos fichiers `creation.sql` et `insertion.sql` pour recréer le schéma et l'instance de la base du TP précédent. Vérifier que le schéma et l'instance créés correspondent aux schéma et instance souhaités. Vérifier que la base est aussi lisible à partir de la fenêtre `F2`. Qu'en déduisez-vous ?
3) Dans la fenêtre `F1` insérer en ligne de commande dans une de vos tables un nouveau n-uplet t sans faire `commit`, et regarder si ce n-uplet se trouve bien dans la table.
   Vérifier dans la fenêtre `F2` si le n-uplet t se trouve dans la table. Qu'en déduisez-vous ?
   Faut-il faire `commit` dans la fenêtre `F1` ?

3.3 Schéma

Un schéma `SQL` est un ensemble de tables. Le nom du schéma est le nom de login du créateur de la table. Le créateur a tous les privilèges sur ses tables, et il peut céder un ou plusieurs de ces privilèges à un autre utilisateur. La personne qui a un privilège sur une de vos tables doit

préfixer votre table par votre nom lors de son utilisation, ou bien créer un synonyme de votre table auparavant (voir `ANNEXE`).

1) Former un groupe de 2 à 4 utilisateurs (avec vos voisins) et donner différentes sortes de privilèges (1 par 1) sur vos tables à un autre membre de votre groupe. Tester ensemble des commandes `SQL` sur les tables qui ne sont pas les votres.
2) Annuler tous les privilèges que vous avez cédés aux membres de votre groupe. Céder à un de ces membres seulement le privilège `insert` pour une table. Refaire l'exercice 1, en supposant que maintenant vous êtes `F1` et l'autre membre est `F2`. Qu'en déduisez-vous ?
   Remarque : Cet exercice n'a de sens que si vous avez bien respecté les exigeances du TP précédent et que les schémas de vos tables sont tout à fait identiques.
3) Annuler tous les droits que vous avez cédés aux membres de votre groupe. Céder à tous les membres de votre groupe seulement le privilège `select` pour toutes les tables de votre base.
   Maintenant vous avez une base de données commune dont les données sont réparties sur différents sites, où un membre est vu comme un site.
   Reformuler les requêtes 1, 2 et 3 du TP précédent sur cette "base répartie".
   Remarque : Cet exercice n'a de sens que si vous avez bien respecté les exigeances du TP précédent et que les schémas de vos bases sont tout à fait identiques.

3.4 Tables/Vues systèmes

`Oracle` gère les informations concernant ses différents utilisateurs, ainsi que ses propres informations, sous forme d'une base de données. Les caractéristiques de certaines de ces tables sont données en Annexe.

- Regarder la descrition de ces tables et consulter leurs contenus.
- Répondez aux questions suivantes en utilisant ces tables.
  1. Donnez le nom des tables que vous avez créées.
  2. Donnez le nom et le type des colonnes que vous avez créées ainsi que le nom de la table auxquelles elles appartiennent.
  3. Donnez le nom des tables auxquelles vous avez accès et leur propriétaire.
  4. Donnez le nom et le propriétaire de toutes les tables auxquelles vous avez accès mais que vous n'avez pas créées.
  5. En dehors des tables systèmes, quelles sont les tables sur lesquelles vous avez des privilèges.
  6. Parcourez la table DICTIONARY (qui a pour synonyme DICT), et pour les tables dont vous ne voyez pas l'utilité, lire le commentaire associé avec help (si cette commande est accessible).

Annexe : `DATE` Le type de données `DATE` correspond aux informations calendaires et horaires.

Bien que ces dernières puissent être représentées par une chaîne de caractères ou par des entiers, le type de données `DATE` a des propriétés particulières associées. Pour chaque valeur de ce type, Oracle enregistre les champs suivants : siècle, année, jour, heure, minute et seconde.

Une valeur peut-être donnée comme un littéral (au format 'YYYY-MM-DD') ou convertie d'une chaîne de caractères ou d'un entier avec la fonction `TO_DATE` :

```
DATE '2005-03-22'
TO_DATE('05-MAR-22:14:30','YY-MON-DD:HH24:MI')
```

Le format par défaut des littéraux sous Oracle est défini par le paramètre `NLS_DATE_FORMAT`. Dans l'exemple ci-dessus, le format indique les deux derniers chiffres de l'année, une abréviation pour le mois, le jour du mois sur deux chiffres, l'heure sur 24 heures et les minutes.

1) Oracle convertit les valeurs des chaînes de caractères au format par défaut en valeur de type `DATE` quand elles sont utilisées dans les expressions de type `DATE`.

   Lorsque l'on spécifie une date sans composante horaire la valeur par défaut est minuit. Lorsque l'on spécifie une date sans composante de jour, la valeur par défaut est le premier jour du mois en cours.

   Une colonne de type `DATE` contient toujours les composantes calendaires et horaires. Si une requête utilise une date sans sa composante horaire alors cette composante est traitée comme ayant la valeur par défaut. Il faut toujours s'en préoccuper afin que les requêtes exprimées sur des dates renvoient bien ce qui est demandé.

Annexe : `ALTER TABLE`

```
ALTER TABLE nom_de_table une_alteration ;
une_alteration ::=
    ADD nom_de_colonne domaine [DEFAULT expression] [ contrainte_de_colonne ] |
    ADD contrainte_de_table |
    DROP COLUMN nom_de_colonne [ RESTRICT | CASCADE ] |
    MODIFY nom_de_colonne [domaine] [DEFAULT expression] [contrainte_de_colonne] |
    RENAME COLUMN nom_de_colonne TO nouvelle_colonne |
    RENAME TO nouveau_nom |
    DROP CONSTRAINT nom_de_contrainte [ RESTRICT | CASCADE ] |
```

`ALTER TABLE` change la définition d'une table existante. Il y a plusieurs variantes :

- `ADD nom_de_colonne domaine` et `ADD (liste_de_colonnes)` ajoutent une nouvelle colonne à la table en utilisant la même syntaxe que `CREATE TABLE`. Une nouvelle colonne ne peut avoir la contrainte `NOT NULL` que si la table est vide.
- `ADD contrainte_de_table` ajoute une nouvelle contrainte à une table en utilisant la même syntaxe que `CREATE TABLE`.
- `DROP COLUMN nom_de_colonne` supprime une colonne d'une table. Les indexes et les contraintes de table référençant cette colonne sont automatiquement supprimés. Il faut utiliser l'option `CASCADE` si certains objets hors de la table dépendent de cette colonne, comme par exemple des références de clés étrangères ou des vues.
- `DEFAULT expression` ne modifient pas les lignes déjà présentes dans la table. Les valeurs par défaut ne s'appliquent qu'aux prochaines commandes `INSERT`. Des valeurs par défaut peuvent aussi être créées pour les vues.
- `RENAME TO` change le nom d'une table (ou d'un index, d'une séquence, d'une vue) ou le nom d'une colonne de la table. Elle est sans effet sur les données stockées.
- `DROP CONSTRAINT` supprime des contraintes d'une table. Actuellement, les contraintes n'ont pas nécessairement un nom unique, si bien qu'il peut y avoir plusieurs contraintes qui ont le nom donné en paramètre. Toutes ces contraintes sont supprimées.

`ALTER TABLE` effectue une copie temporaire de la table originale. Les modifications sont faites sur cette copie, puis l'originale est effacée, et enfin la copie est renommée pour remplacer l'originale. Cette méthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes. Durant l'exécution de `ALTER TABLE`, la table originale est lisible par d'autres clients. Les modifications et insertions sont reportées jusqu'à ce que la nouvelle table soit prête.

Annexe : SELECT

- `SELECT [ALL|DISTINCT] colonne | (fct(colonne) [AS nom]) [, liste_col_ou_fct(col)]`
  Liste de colonnes ou fonctions sur une colonne (`AVG|MAX|MIN|SUM [ALL|DISTINCT]`) ou sur le nombre de lignes d'une table (`COUNT [ALL|DISTINCT]`). S'il y a des regroupements de colonnes `GROUP BY` alors les fonctions se font sur chaque regroupement (`AVG, MAX, MIN`) ou sur le nombre de lignes de chaque regroupement (`COUNT`).

- `FROM nom_de_table [alias] [,liste_de_nom_de_table]`
  Indique les tables à partir desquelles récupérer les données. Un alias permet un renommage local. On utilise l'alias à la place du nom de la table afin d'enlever des ambiguités, de simplifier l'écriture ou d'être plus explicite.

- `[WHERE condition]`

  Exemples d'opérateurs utilisables dans une condition :
  - opérateurs logiques (`AND, OR, NOT, IS NULL, IS NOT NULL`)
  - opérateurs de chaînes (`[NOT] IN|BETWEEN|LIKE`)
  - opérateurs arithmétiques (`+, -, *, /, mod(,)`)
  - comparateurs arithmétiques (`<, >, =, <>, <=, >=, [NOT] IN`)

  Conditions avec sous-requêtes :
  ```
  WHERE [NOT] EXISTS|IN (clause_select)
  WHERE colonne | (fct(colonne) [AS nom]) op [ALL|ANY] (clause_select)
  ```
  avec
  `ALL` : la condition est vraie seulement si elle est vraie pour toutes les valeurs de la clause select. Donc la condition est vraie si la sous-requête est vide. Donc : `<ALL` = moins que le minimum, `>ALL` = plus que le maximum, `<>ALL` équivalent à `NOT IN`.
  `ANY` : la condition est vraie seulement si elle est vraie pour au moins l'une les valeurs de la clause select. Donc : `<ANY` = moins que le maximum, `>ANY` = plus que le minimum, `=ANY` équivalent à `IN`.

- `GROUP BY colonne [,list_col]`
  Regroupe des résultats selon une(des) colonne(s).

- `HAVING condition`
  est à `GROUP BY` ce que `WHERE` est à `FROM`.

- `[ORDER BY colonne [ASC|DESC] [,list_col]]`
  Tri des résultats selon une (des) colonne(s). Toute colonne du select qui n'est pas sous une fonction doit être spécifiée dans la liste des colonnes de `ORDER BY` si on souhaite effectuer un tri.

Clauses obtenues par opération emsembliste :
```
(clause_select) UNION [ALL] (clause_select)
(clause_select) INTERSECT [ALL] (clause_select)
(clause_select) MINUS [ALL] (clause_select)
```
Sans l'option `ALL` seuls les n-uplets distincts sont conservés (à tester : support de `ALL` sous Oracle 10i pour les deux dernières opérations). Rappel : il faut respecter les types (i.e. les domaines en algèbre relationnelle) colonne à colonne.

Annexe : GRANT

`SQL> GRANT <privilège> ON <objet>` `TO <utilisateur_x>`	donne un privilège à un utilisateur_x sur l'objet mentionné parmi les privilèges : `insert, select, update,delete, alter`
`SQL> GRANT ALL ON <obj> TO <x>`	donne tous les privilèges à x
`SQL> GRANT <priv> ON <obj> TO PUBLIC`	donne droit à tout le monde
`SQL> CREATE SYNONYM <tab1> FOR <u>.<tab>`	crée un synonyme de nom `tab1` pour la table `tab` créée par l'utilisateur `u`.
`SQL> REVOKE <privilège> ON <objet>` `FROM <utilisateur_x>`	révocation des droits de manière identique au `GRANT`

Quelques tables de système :

- Table contenant la description des tables (colonnes) auxquelles vous avez accès :
  `ALL_TABLES (owner, table_name,...)`
  `ALL_TAB_COLUMNS (owner, table_name, column_name, data_type, nullable,...)`
- Table contenant la description des tables, des vues, des contraintes et des colonnes que vous avez créées :
  `USER_TABLES (table_name,...)`
  `USER_TAB_COLUMNS (table_name, column_name, data_type, nullable,...)`
  `USER_CONSTRAINTS(....)`
  `USER_CONSTRAINTS_COLUMNS(....)` `USER_VIEWS_COLUMNS(....)`
- Table contenant tous les utilisateurs de la base de données :
  `ALL_USERS (user_name, user_id, created)`
- Table contenant toutes les tables (colonnes) sur lesquelles vous avez des privilèges :
  `TABLE_PRIVILEGES (grantee, owner, table_name, grantor, select_priv, insert_priv, delete_priv, update_priv, alter_priv, created,...)`
  `COLUMN_PRIVILEGES (grantee, owner, table_name, column_name, grantor, update_priv, created,...)`
- Table contenant toutes les tables (colonnes) systèmes :
  `DICTIONARY (table_name, comments)`
  `DICT_COLUMNS (table_name, column_name, comments)`