TP N°02 (semaine 39-40 – 4h) :
Création d'une Bases de Données Oracle

Exercice 1 : Création de la base HOTEL – Modification de script

1) Il s'agit dans un premier temps de supprimer toutes les tables de **votre schéma** (c'est-à-dire de votre compte oracle sur BD19**)** éventuellement créées en S2.

   a) Lister le nom des tables (`table_name`) de votre schéma en utilisant la table `USER_TABLES` :

   ```
   SELECT table_name FROM USER_TABLES ;
   ```

   b) Pour supprimer les tables de votre schéma, compléter et utiliser le bloc PLSQL suivant :

   ```
   BEGIN
       FOR tab IN (SELECT … FROM …) LOOP
         EXECUTE IMMEDIATE 'DROP TABLE '|| tab.… ||' CASCADE CONSTRAINTS';
         DBMS_OUTPUT.PUT_LINE('Table '||...||' supprimée');
       END LOOP;
   END;
   /
   ```

2) Récupérer le script de création **`hotel.sql`** fourni (généré à partir du MCD grâce à POWER AMC).
   Remarque : Ne pas ajouter les tables manquantes pour l'instant !

   Tenter d'exécuter le script complet : Plusieurs erreurs se produisent lors de la **création** des tables. Corriger ces erreurs pour pouvoir exécuter le script correctement et créer les tables.

   Ajouter, au début du script, les ordres de **suppressions** (`DROP`) de chacune des tables (afin de pouvoir relancer plusieurs fois ce script de création des tables) ou utiliser le script PL/SQL précédent.

Tant que les tables n'auront pas été créées une première fois, **les ordres de suppression (`DROP`) généreront des erreurs**. Il ne faut pas en tenir compte tant que toutes les tables n'ont pas été créées sans erreur.

Clé primaire : Ensemble minimal de colonnes qui permet d'identifier de manière unique chaque enregistrement d'une table.

Clé étrangère : Colonne(s) d'une table qui référence la clé primaire d'une autre table.

IMPORTANT : Une clé étrangère doit faire référence à une clé primaire définie au préalable

3) Afin de tester les différentes syntaxes possibles, modifier les contraintes ***de table*** en contrainte ***de colonne*** chaque fois que cela est possible dans les tables **TELEPHONE**, **ADRESSE** et **FACTURE**. Relancer le script.

Contraintes de colonne :

```
CREATE TABLE maTable (
      monChp1  typemonChp1
            CONSTRAINT maContrainteMonChp1  définition de la contrainte,
      monChp2 typemonChp2
            CONSTRAINT maContrainteMonChp2 définition de la contrainte,
      …) ;
```

Contraintes de table :

```
CREATE TABLE maTable (
      monChp1  typemonChp1,
      monChp2 typemonChp1,
      …,
      CONSTRAINT maContrainteMonChp1 définition de la contrainte,
      CONSTRAINT maContrainteMonChp2 définition de la contrainte) ;
```

Attention : **pour les contraintes de PK et de FK la syntaxe diffère si l'on est en contrainte de table ou en contrainte de colonne. Elle est identique pour les contraintes CHECK.**

4) Ajouter les requêtes de création et de suppression de la table **EMAIL** au bon endroit dans le script.

Champ	Type	Nullable	Lg	Contrainte
EML_ID	ENTIER	NON		Clé Primaire
CLI_ID	ENTIER	NON		Clé étrangère
EML_ADRESSE	Chaîne de caractère	NON	64	Contient forcement @ puis .
EML_LOCALISATION	Chaîne de caractère	OUI	20	« domicile » ou « bureau »

Il est recommandé de nommer les contraintes lors de leur création (bien que ce soit facultatif) car il sera difficile de faire évoluer les contraintes déclarées (désactivation, réactivation, suppression) et la lisibilité des programmes en sera affectée.

5) Modifier le script en ajoutant les contraintes de validation (**CHECK**) dans les tables concernées :
   - Ajouter sur la table **CHAMBRE** les contraintes suivantes :
     - Le champ **CHB_ETAGE** ne peut prendre que les valeurs : « RDC », « 1er » ou « 2e»
     - Le champ **CHB_NUMERO** ne peut pas prendre la valeur 13.
     - Les champs **CHB_BAIN** et **CHB_WC** ne peuvent prendre que les valeurs 0 ou 1.
   - Ajouter sur la table **FACTURE** la contrainte suivante : La date de paiement (**FAC_DAT_PMT**) est toujours supérieure ou égale à la date de facture (**FAC_DATE**).

6) Ajouter les contraintes de clés primaires (PK) et étrangères (FK) manquantes aux tables **CHAMBRE** et **PLANNING** (**pour cela consulter le MPD**).

Exercice 2 : Insertions de données dans la base HOTEL

1) **Dans un fichier nommé *insertion.sql*,** écrire les requêtes d'insertions permettant de charger les données dans vos tables à partir des tables du schéma HOTEL. (*conseil : les exécuter une par une au fur et à mesure*)

   ATTENTION : Pour les tables TRF_CHB, FACTURE ainsi que PLANNING, **ne récupérer que les informations de l'année 2007.**

   Attention, vous devrez bien évidement, respecter les contraintes d'intégrité référentielle et résoudre les éventuels problèmes rencontrés (certaines tables ne comportent pas exactement les mêmes champs, vous devrez vérifier et ne sélectionner parfois que certains champs et dans le bon ordre)

Pour insérer des données dans une table qui proviennent d'une autre table, on utilise INSERT INTO … SELECT … ;

Par exemple : `INSERT INTO TITRE SELECT * FROM hotel.titre;`

Remarque : **SELECT *** ne fonctionne que si les tables ont exactement les mêmes champs et dans le même ordre.

2) Ajouter les requêtes de création et de suppression de la table **TARIF** à partir de la table TARIF du schéma HOTEL en ne récupérant, là encore, que les tarifs de 2007.

Pour créer une table contenant les données d'une autre table on utilise :

```
CREATE TABLE matable AS
      (SELECT .. FROM … );
```

Consulter les contraintes de votre table **TARIF** : ajouter la contrainte manquante à cette table.
Ajouter également les contraintes nécessaires et manquantes à la table **TRF_CHB**.

a)

Pour ajouter une contrainte à une table existante on utilise :

```
ALTER TABLE matable ADD
      CONSTRAINT …;
```

Exercice 3 : Modification de la base HOTEL – Contraintes (suite et fin)

a) Ecrire la requête permettant d'ajouter la contrainte suivante sur la table **TELEPHONE** (utiliser la commande **ALTER TABLE**) : l'attribut **TEL_LOCALISATION** ne peut prendre comme valeur que : « domicile », « portable » ou « bureau » et la valeur par défaut est « domicile ».

Pour ajouter une valeur par défaut à un champ :

```
ALTER TABLE matable
MODIFY macolonne DEFAULT valeurParDéfaut;
```

b) Tester en ajoutant la ligne suivante pour provoquer une erreur :

TEL_ID	CLI_ID	TYP_CODE	TEL_NUMERO	TEL_LOCALISATION
300	1	GSM	07-12-58-96-00	« INCONNU »

Tester en ajoutant la ligne suivante puis vérifier le contenu de la table :

TEL_ID	CLI_ID	TYP_CODE	TEL_NUMERO	TEL_LOCALISATION
300	1	GSM	07-12-58-96-00	NULL

Ajouter à nouveau **en omettant** cette fois la localisation. Vérifier.

Lorsqu'on insère la valeur NULL, l'éventuelle valeur par défaut associée au champ n'est pas utilisée.

c) Tenter de supprimer le client nommé **Alain DUPONT**. **Pourquoi une erreur se déclenche-t-elle ?**

d) Faire les modifications nécessaires pour que lorsqu'un client est supprimé, ses téléphones soient automatiquement supprimés de la table TELEPHONE : supprimer la contrainte de clé étrangère et la recréer avec la directive utile dans ce cas.

Il n'est pas possible de modifier une contrainte, il faudra supprimer la contraintes FK concernée puis la recréer avec la directive ON DELETE CASCADE

Pour ajouter une contrainte à une table existante on utilise :

```
ALTER TABLE matable DROP CONSTRAINT maContrainte;
```

Pour quelle(s) autre(s) table(s), la modification est-elle également nécessaire ?
(Remarque : on ne fera les modifications demandées que pour les 4 tables directement impliquées pour ne pas supprimer par erreur un client qui aurait des factures)

e) Tenter de supprimer à nouveau le client nommé Alain DUPONT. Tenter de supprimer un client ayant une facture. Annuler les suppressions (ROLLBACK)

Exercice 4 : Ajouts d'Index

1) **Index existants** : Contrôler les index générés par ORACLE lors de la création des tables.

Les index sur les clés primaires sont générés automatiquement.
Il est préférable de créer des index sur les clés étrangères, souvent utilisées dans les jointures

2) Ajout d'index
   - Aux index déjà prévus pour les clés étrangères en fin de script, ajouter les index manquants sur les clés étrangères de la table **EMAIL**, **TRF_CHB** et **CHAMBRE**.

   - On considère la requête suivante :

```
SELECT pln_jour, chb_id
FROM client c JOIN planning p ON (c.cli_id = p.cli_id)
WHERE UPPER(cli_nom)    = 'DUPONT'
AND   UPPER(cli_prenom)= 'ALAIN';
```

Vérifier le plan d'exécution de la requête avec la commande EXPLAIN.

Pour utiliser la commande EXPLAIN :

```
EXPLAIN plan SET statement_id ='planReq' FOR
      SELECT … ;
```

Pour afficher le plan d'exécution :

```
SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY);
```

Créer un index **i_pln_client** qui permette d'accélérer l'exécution de cette requête. Vérifier en affichant le plan d'exécution.

   - Même chose pour la requête suivante : Vérifier le plan d'exécution et créer un index qui permette de l'accélérer.

```
SELECT fac_date, cli_id
FROM facture
ORDER BY fac_date DESC, cli_id;
```

Exercice 5 : Rôles et Privilèges

1) Donner les ordres SQL assignant le privilège de lecture (SELECT) des tables **CLIENT** et **ADRESSE** à l'ensemble des utilisateurs (**PUBLIC**).

   Vérifier en faisant tester par son voisin, l'accès à vos tables **CLIENT et ADRESSE** en utilisant la requête suivante (si votre login est **dieu0012** par exemple) :

```
SELECT cli_nom, adr_ville
FROM dieu0012.client c
    JOIN dieu0012.adresse a ON a.cli_id = c.cli_id
ORDER BY cli_nom;
```

2) Assigner ensuite les privilèges supplémentaires suivants à votre voisin :

- Privilège d'insertion, de suppression et de modification du **nb_pers** pour la table **PLANNING** ,

Attention : Pour pouvoir mettre à jour ou supprimer des lignes d'une table, les privilèges **UPDATE** ET **DELETE** ne suffisent pas.

Le privilège **SELECT** est nécessaire.

- Privilège d'insertion dans la table **CLIENT**,
- Privilège de modification de la table **CLIENT** (sauf colonne *cli_id*),
- Privilège de lecture de la table **CLIENT** en cascade (en autorisant la redistribution du privilège).

Faire ensuite vérifier par votre voisin grâce aux requêtes suivantes (où **dieu0012** est votre login):

```
INSERT INTO dieu0012.planning (CHB_ID, PLN_JOUR, CLI_ID, NB_PERS)
VALUES (1,SYSDATE,1,2);
```

```
UPDATE dieu0012.planning
SET nb_pers = 10
WHERE trunc(pln_jour) = trunc (SYSDATE);
```

```
DELETE
FROM dieu0012.planning
WHERE trunc(pln_jour) = trunc (SYSDATE);
```

```
INSERT INTO dieu0012.client (CLI_id, TIT_CODE, CLI_nom)
VALUES (105, 'M.', 'HAZARD');
```

```
UPDATE dieu0012.client
SET cli_prenom = 'Tristan'
WHERE cli_id = 105;
```

3) Reprendre le privilège de lecture sur la table **CLIENT** à votre voisin.
Pourquoi peut-il toujours accéder en lecture à votre table **CLIENT** en lecture ? (vérifier).
Reprendre tous les privilèges sur les tables **CLIENT** et **PLANNING**. (vérifier)

Attention : on ne reprendre que les privilèges qui ont été donnés

Par exemple, si on donne un privilège à PUBLIC on ne peut reprendre le privilège qu'à PUBLIC et pas à un utilisateur en particulier.

4) Donner un privilège à votre voisin qui lui autorise la lecture de votre table **AGENT_ENTRETIEN** (sauf colonne **AGT_SALAIRE**), et uniquement pour les agents **encore en poste**. (Vérifier)

5) Créer un rôle appelé **RO_HOTEL_*login*** (où *login* est votre nom d'utilisateur, par ex **RO_HOTEL_DIEU0012**) et alimenter ce rôle avec les privilèges suivants :
   - Privilège de modification sur **CHAMBRE** (uniquement la colonne *agt_id*),
   - Privilège de lecture, insertion, modification et suppression sur **PLANNING** et **TARIF**.

6) Assigner le rôle **RO_HOTEL_*login*** à une personne de votre groupe. Vérifier puis faire tester.

7) Supprimer le rôle **RO_HOTEL_*login***.