Corrigé du TD/TP BDA (Séance n° 1)
Vues et Privilèges d'accès avec Rappels SQL et PLSQL

Première étape : Consultation des tables systèmes

Q1 Quel est le nom de tous les attributs de la relation PROF ?

```
SELECT COLUMN_NAME
FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'PROF' ;
```

Q2 Donnez la liste des fonctions et des procédures stockées de la base IUT ?

```
SELECT OBJECT_NAME
FROM USER_OBJETS
WHERE OBJECT_TYPE IN ('TABLE', 'FUNCTION', 'PROCEDURE');
```

Q3 Quelles sont toutes les contraintes d'intégrité définies sur les relations de données ?

```
SELECT CONSTRAINT_NAME
FROM USER_CONSTRAINTS ;
```

Q4 Retrouvez le nom des contraintes définies sur toutes les relations de la base IUT ayant un attribut de type NUMBER.

```
SELECT CONSTRAINT_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN (
       SELECT TABLE_NAME
       FROM USER_TAB_COLUMNS
       WHERE DATA_TYPE = 'NUMBER') ;
```

Q5 Retrouvez le nom des contraintes et le nom de l'attribut de type NUMBER sur lequel elles portent pour toutes les relations de la base IUT.

```
SELECT DISTINCT CC. CONSTRAINT_NAME, CC. COLUMN_NAME, DATA_TYPE
FROM USER_CONS_COLUMNS CC, USER_TAB_COLUMNS ATT
WHERE CC. COLUMN_NAME = ATT.COLUMN_NAME AND
       CC.TABLE_NAME = ATT.TABLE_NAME AND DATA_TYPE = 'NUMBER' ;
```

Q6 Retrouvez le nom des contraintes, le nom de l'attribut sur lequel elles portent et le type de cet attribut, pour toutes les relations de la base IUT ayant un attribut de type NUMBER.

```
SELECT CI. CONSTRAINT_NAME, CI. COLUMN_NAME, DATA_TYPE
FROM USER_CONS_COLUMNS CI, USER_TAB_COLUMNS ATT
```

```
WHERE CI. COLUMN_NAME = ATT.COLUMN_NAME
AND CI. TABLE_NAME = ATT.TABLE_NAME AND CI. TABLE_NAME IN (
            SELECT TABLE_NAME
            FROM USER_TAB_COLUMNS
            WHERE DATA_TYPE = 'NUMBER') ;
```

Q7 Trouvez le nom et la spécification des contraintes de domaine dans la base IUT.

```
SELECT CONSTRAINT_NAME, SEARCH_CONDITION
FROM USER_CONSTRAINTS
WHERE CONSTRAINT_TYPE = 'C' ;
```

Q8 Quels sont les attributs portant le même nom dans des relations différentes de la base ? Affichez nom attribut 1, nom relation 1, nom attribut 2, nom relation 2.

```
SELECT AT1.COLUMN_NAME, AT1.TABLE_NAME, AT2.COLUMN_NAME, AT2.TABLE_NAME
FROM USER_TAB_COLUMNS AT1, USER_TAB_COLUMNS AT2
WHERE AT1.COLUMN_NAME = AT2.COLUMN_NAME AND
       AT1.TABLE_NAME <> AT2.TABLE_NAME ;
```

Q9 Comment formuler la requête précédente en évitant que les couples soient affichés deux fois ((A1, A2) et (A2, A1)) ?

```
SELECT AT1.COLUMN_NAME, AT1.TABLE_NAME, AT2.COLUMN_NAME, AT2.TABLE_NAME
FROM USER_TAB_COLUMNS AT1, USER_TAB_COLUMNS AT2
WHERE AT1.COLUMN_NAME = AT2.COLUMN_NAME AND
       AT1.TABLE_NAME < AT2.TABLE_NAME ;
```

Deuxième étape : Création et manipulation de vues

Q10 Créez une vue appelée PROF_INFO2 donnant le numéro, le nom et le prénom des professeurs du département Informatique enseignant en deuxième année.
Puis consultez la vue par une requête d'interrogation.

Création de la vue :

```
CREATE VIEW PROF_INFO2 (NUM, NOM, PRENOM) AS
SELECT PROF.NUM_PROF, NOM_PROF, PRENOM_PROF
FROM PROF, ENSEIGNT, ETUDIANT
WHERE ANNEE = 2 AND
            PROF.NUM_PROF = ENSEIGNT.NUM_PROF AND
            ENSEIGNT.NUM_ET = ETUDIANT.NUM_ET ;
```

Consultation de la vue :

```
SELECT * FROM PROF_INFO2 ;
```

Q11 Il s'agit de prendre en compte l'intégrité de domaine de l'attribut DISCIPLINE dans la relation MODULE. En effet, ces valeurs admissibles ne peuvent appartenir qu'à l'ensemble suivant : {Informatique, Gestion, Maths}. Créez une vue DIS, permettant d'assurer que toute insertion à travers cette vue vérifie la contrainte de domaine énoncée. Vérifiez l'opération réalisée en tentant une insertion invalide.

Création de la vue :

```
CREATE VIEW DIS AS
SELECT * FROM MODULE
WHERE DISCIPLINE IN ('INFORMATIQUE', 'GESTION', 'MATHS')
WITH CHECK OPTION ;
```

Tentative d'insertion dans la vue d'un tuple erroné :

```
INSERT INTO DIS(CODE, DISCIPLINE) VALUES (99, 'SPORT') ;
```

Q12 Dans la base actuellement définie, la contrainte suivante n'est pas vérifiée : "Le responsable d'une matière doit forcément enseigner cette matière". Comment prendre en compte cette contrainte d'intégrité dynamique par une vue ?

Création de la vue :

```
CREATE VIEW MAT AS
SELECT * FROM MODULE M
WHERE RESP IN (
            SELECT NUM_PROF
            FROM ENSEIGNT
            WHERE M. CODE = ENSEIGNT.CODE)
 WITH CHECK OPTION ;
```

Vérification par insertion d'un tuple erroné :

```
INSERT INTO MAT (CODE, RESP) VALUES ('XXX', 99) ;
```

Remarque : l'obligation dans une vue d'exprimer les jointures sous forme imbriquée impose l'utilisation d'un alias pour MODULE dans la définition de la vue MAT : le responsable de la matière traitée au niveau du premier bloc doit enseigner cette matière là.

Autre possibilité pour la création de la vue :

```
CREATE VIEW MAT AS
SELECT * FROM MODULE M
WHERE (CODE, RESP) IN (
            SELECT CODE, NUM_PROF
            FROM ENSEIGNT)
 WITH CHECK OPTION ;
```

Q13 Définissez la vue nécessaire pour prendre en compte toutes les contraintes d'intégrité de référence mises en jeu lors d'une insertion dans la relation ENSEIGNT ?

Création de la vue :

```
CREATE VIEW ENS AS
SELECT *
FROM ENSEIGNT
WHERE CODE IN (SELECT CODE FROM MODULE) AND
              NUM_ET IN (SELECT NUM_ET FROM ETUDIANT) AND
              NUM_PROF IN (SELECT NUM_PROF FROM PROF)
 WITH CHECK OPTION ;
```

Vérification par insertion d'un tuple erroné :

```
INSERT INTO ENS (CODE, NUM_ET, NUM_PROF) VALUES ('SPO', 2112, 12) ;
```

Q14 Définissez la vue nécessaire pour prendre en compte toutes les contraintes d'intégrité de référence mises en jeu lors d'une suppression de matière dans la relation MODULE ?

La relation MODULE est, via sa clef primaire CODE, associée aux relations :
- PROF par la clef étrangère MAT_SPEC ; - ENSEIGNT par la clef étrangère CODE ;

- NOTATION par la clef étrangère CODE ; - MODULE par la clef étrangère CODEPERE

La prise en compte de l'intégrité de référence en suppression signifie qu'une matière ne peut être supprimée de la relation que s'il n'existe :
- aucun professeur dont cette matière est la spécialité ;
- aucun enseignement réalisé dans cette matière ;
- aucune note attribuée dans cette matière ;
- aucune matière dépendant de cette matière.

Création de la vue :

```
CREATE VIEW MAT AS
SELECT * FROM MODULE
WHERE CODE NOT IN
              (SELECT DISTINCT CODE FROM ENSEIGNT)
      AND CODE NOT IN
              (SELECT DISTINCT MAT_SPEC FROM PROF)
      AND CODE NOT IN
              (SELECT DISTINCT CODE FROM NOTATION)
      AND CODE NOT IN
              (SELECT DISTINCT CODEPERE FROM MODULE)
WITH CHECK OPTION ;
```

Vérification par tentative de suppression d'une matière enseignée :

```
DELETE FROM MAT WHERE CODE = 'ACSI' ;
```

Troisième étape : Gestion des privilèges d'accès aux données

Q15 Donnez l'autorisation à tous de consulter la relation ETUDIANT.

```
GRANT SELECT ON ETUDIANT TO PUBLIC ;
```

Q16 Effectuez la consultation de tous les étudiants résidant à Marseille, dans l'espace de travail d'un autre utilisateur de login GALAXE. Vérifiez que vous n'avez que le droit de consultation, en tentant une opération de mise à jour sur cette relation.

```
SELECT NOM_ET, PRENOM_ET FROM OPS$GALAXE.ETUDIANT
WHERE VILLE_ET = 'MARSEILLE' ;
```

Vérification de l'accord des privilèges :

```
 SELECT * FROM GALAXE.ETUDIANT
/
UPDATE GALAXE.ETUDIANT SET VILLE_ET = 'PARIS' WHERE NUM_ET = 2401
/
```

L'ordre de modification est rejeté, faute d'avoir le droit nécessaire.

Q17 Accordez le droit de consultation et de mise à jour des données de la relation PROF à un autre utilisateur, qui vous accordera les mêmes privilèges.
Consultez puis effectuez une modification sur la relation PROF de l'autre utilisateur.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON PROF TO GALAXE
WITH GRANT OPTION ;
```

Vérification de l'accord des privilèges :

```
SELECT * FROM GALAXE.PROF
/
INSERT INTO GALAXE.PROF(NUM_PROF, NOM_PROF, PRENOM_PROF)
VALUES (21,'EINSTEIN', 'ALBERT')
/
```

Remarque : la visualisation du tuple inséré par l'utilisateur `OPS$GALAXE` n'est possible qu'après Validation de votre transaction par **`COMMIT`**.

Quatrième étape : Développement en PL/SQL

Q18 Ecrire un bloc anonyme permettant d'afficher les codes et libellé des matières en gérant l'exception. Ecrire une version avec un curseur déclaré et une version avec un curseur non déclaré.

Version avec un curseur déclaré (pour tester l'exception enlever **`WHERE`** `CODEPERE` **`IS NOT NULL`** **de la requête)**:

```
DECLARE

CURSOR Mat IS SELECT CODE, LIBELLE
              FROM MODULE
              WHERE CODE NOT IN (SELECT CODEPERE
                                 FROM MODULE
                                 WHERE CODEPERE IS NOT NULL) ;
pas_de_mat EXCEPTION;
Trouver BOOLEAN : = False ;
BEGIN
     FOR M IN Mat LOOP
       Trouver : = true ;
       DBMS_OUTPUT.PUT_LINE (M. CODE ||' '|| M. LIBELLE) ;
     END LOOP ;
     IF NOT Trouver THEN RAISE pas_de_mat ; END IF ;
EXCEPTION
     WHEN pas_de_mat THEN
     DBMS_OUTPUT.PUT_LINE ('Aucune matière') ;
END;
```

Version avec un curseur non déclaré (**curseur dynamique**) :

```
DECLARE
pas_de_mat EXCEPTION;
Trouver BOOLEAN : = False ;
BEGIN
     FOR M IN (SELECT CODE, LIBELLE
               FROM MODULE
               WHERE CODE NOT IN (SELECT CODEPERE
                                  FROM MODULE
                                  WHERE CODEPERE IS NOT NULL)) ;
 LOOP
     Trouver : = true ;
     DBMS_OUTPUT.PUT_LINE (M. CODE ||' '|| M. LIBELLE) ;
 END LOOP ;
     IF NOT Trouver THEN RAISE pas_de_mat ; END IF ;
EXCEPTION
     WHEN pas_de_mat THEN
     DBMS_OUTPUT.PUT_LINE ('Aucune matière') ;
END;
```

Q19 Programme d'insertion automatique de tuples dans GROUPE

Création de la nouvelle relation :

```
CREATE TABLE GROUPE (
NUMERO NUMBER (1,0),
EFFECTIF NUMBER (3,0),
CONSTRAINT CP_GROUPE PRIMARY KEY(NUMERO));

DECLARE
      eff_calc    GROUPE.EFFECTIF%TYPE ; -- Nombre d'etudiants par groupe
      nb_groupe   NUMBER(1,0) ; -- Nombre de groupes de deuxieme annee
      aucun_etudiant    EXCEPTION ;

BEGIN
      COMMIT ;
      SELECT COUNT (DISTINCT GROUPE) INTO nb_groupe
            FROM ETUDIANT WHERE ANNEE = 2 ;
      IF nb_groupe = 0 THEN RAISE aucun_etudiant ; END IF ;
      FOR i IN 1..nb_groupe LOOP
                  /* Recherche de l'effectif du ieme groupe */
SELECT COUNT (*) INTO eff_calc FROM ETUDIANT
                  WHERE GROUPE = i AND ANNEE = 2 ;
/* Insertion dans la relation GROUPE du tuple décrivant le ième groupe */
INSERT INTO GROUPE VALUES (i, eff_calc) ;
      END LOOP ;
      COMMIT ;
EXCEPTION
      WHEN aucun_etudiant
      THEN DBMS_OUTPUT.PUT_LINE ('Il n''y a pas d''étudiant') ;
END ;
/
```