Rendu TP Base de données répartie ORACLE, Thomas LACAZE, Jean-Michel REMEUR

TP2

I - Description et création de la base données :

1) Ajouter les contraintes de clés primaires et de clés étrangères.

Clefs primaires :

Méthode 1, utilisation de `ALTER` pour modifier la structure de la table :

```
ALTER TABLE Etudiant ADD PRIMARY KEY (NumEtu);
ALTER TABLE Matiere ADD PRIMARY KEY (Matiere);
ALTER TABLE Epreuve ADD PRIMARY KEY (NumEpreuve);
ALTER TABLE Epreuve ADD PRIMARY KEY (NumEtu, NumEpreuve);
```

Méthode 2, création des clés primaires lors de la création :

```
CREATE TABLE Etudiant (
    NumEtu NUMBER(8) PRIMARY KEY,
    Nom VARCHAR2(20),
    Prenom VARCHAR2(20),
    DateNais DATE,
    Rue VARCHAR2(50),
    CP CHAR(5),
    Ville VARCHAR2(25)
);


CREATE TABLE Matiere (
    CodeMat CHAR(3) PRIMARY KEY,
    Libelle VARCHAR2(20),
    Coef NUMBER(3,2)
);

CREATE TABLE Epreuve (
    NumEpreuve NUMBER(10) PRIMARY KEY,
    DateEpreuve DATE,
    Lieu VARCHAR2(20),
    CodeMat CHAR(3)
);

CREATE TABLE Notation (
    NumEtu NUMBER(8),
```

```
    NumEpreuve NUMBER(10),
    Note NUMBER(4,2),
    PRIMARY KEY (NumEtu,NumEpreuve)
);
```

Clefs étrangères :

```
ALTER TABLE Epreuve ADD CONSTRAINT FkEpreuveMatiere FOREIGN KEY(CodeMat)
REFERENCES Matiere(CodeMat);
ALTER TABLE Notation ADD CONSTRAINT FkNotationEtudiant FOREIGN KEY(NumEtu)
REFERENCES Etudiant(NumEtu);
ALTER TABLE Notation ADD CONSTRAINT FkNotationEpreuve FOREIGN KEY(NumEpreuve)
REFERENCES Epreuve(NumEpreuve);
```

2) Ajouter les contraintes d'intégrité suivantes dans le script SQL.

```
ALTER TABLE Notation ADD CONSTRAINT ChkNote check (Note >= 0 and Note <= 20);
ALTER TABLE Etudiant MODIFY Nom NOT NULL;
ALTER TABLE Matiere ADD CONSTRAINT LibelleUnique UNIQUE (LIBELLE);
ALTER TABLE Epreuve MODIFY DateEpreuve DEFAULT SYSDATE NOT NULL;
```

II - Instantiation de la base de données

1) Insérer les données correspondantes dans différentes tables et valider votre insertion par COMMIT.

Nous avons bien exécuté le script et finalisé l'insertion avec un `COMMIT;`

2) Insérer l'étudiant : `(1100 | TOTO | Albert | 01-07-1992| Rue de Crimée | 75019|Paris)`.

```
INSERT INTO ETUDIANT VALUES ('1100', 'TOTO', 'Albert', '01-07-1992', 'Rue de
Crimée', '75019', 'Paris');
```

Cet étudiant a obtenu la note 13 à l'épreuve de numéro « 31031 », mettre à jour cette information dans la base de données.

```
INSERT INTO NOTATION VALUES ('1100', '31031', '13');
```

Que constatez-vous ? Justifier.

On constate qu'il y a une violation de clef étrangère. En effet, l'épreuve avec le numéro "31031" n'existe pas dans la base de données. On ne peut donc pas insérer une notation sans que l'épreuve ne soit créée au préalable.

3) Mettre à jour le coefficient de Statistique à 0,3 et Informatique à 0,5 sachant qu'on doit toujours garder la somme des coefficients de toutes les matières doit rester à 1

```
UPDATE MATIERE set COEF = 0.30 WHERE LIBELLE = 'Statistique';
UPDATE MATIERE set COEF = 0.50 WHERE LIBELLE = 'Informatique';
```

4) Mettre à jour toutes les notes de l'épreuve 11031 à 14

```
UPDATE NOTATION set NOTE = 14 WHERE NUMEPREUVE = 11031;
```

5) Est-il possible de rajouter une épreuve avec la participation de tous les étudiants (Certification Oracle de la matière Informatique avec une note de 10 + 2% du numéro d'étudiant s'il est inferieur 300 et de 10 + 1% du numéro d'étudiant s'il est supérieur ou égal à 300 en considérant que 18 sera la note maximale). Comment ?

```
DECLARE
    v_codeMat MATIERE.CodeMat%type;
    v_numEpreuve EPREUVE.NUMEPREUVE%type := 666;
    v_note NOTATION.NOTE%type;
    CURSOR c_student IS SELECT * FROM ETUDIANT;
BEGIN
    SELECT CODEMAT INTO v_codeMat FROM MATIERE WHERE LIBELLE = 'Informatique';
    INSERT INTO EPREUVE(NUMEPREUVE, LIEU, CODEMAT) VALUES (v_numEpreuve, 'EFREI',
v_codeMat);
    FOR student in c_student
    loop
        IF student.NUMETU < 300 then
            v_note := 10 + student.NUMETU * 0.02;
        end if;
        IF student.NUMETU > 300 then
            v_note := 10 + student.NUMETU * 0.01;
        end if;
        IF v_note > 18 then
            v_note := 18;
        end if;
        INSERT INTO NOTATION VALUES (student.NUMETU, v_numEpreuve, v_note);
    end loop;
    commit;
END;
```

III - Interrogation

1) Vérifier la création de vos 4 tables et leur instanciation (Nombres de lignes par table par exemple).

```
SELECT COUNT(*) FROM ETUDIANT;
SELECT COUNT(*) FROM MATIERE;
SELECT COUNT(*) FROM EPREUVE;
SELECT COUNT(*) FROM NOTATION;
```

2) Liste des notes supérieures ou égales à 10.

```
SELECT * FROM NOTATION WHERE NOTE >= 10;
```

3) Liste des épreuves dont la date se situe entre le 1er janvier et le 30 juin 2019.

```
SELECT * FROM EPREUVE where DATEEPREUVE between '01-01-2019' and '30-06-2019';
```

Ou

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY';
SELECT * FROM EPREUVE WHERE DATEEPREUVE BETWEEN TO_DATE('01-01-2019', 'DD-MM-
YYYY') AND TO_DATE('30-06-2019', 'DD-MM-YYYY');
```

4) Nombre total d'épreuves.

```
SELECT COUNT(*) FROM EPREUVE;
```

5) Nombre de notes indéterminées (NULL).

```
SELECT COUNT(*) FROM NOTATION where NOTE IS NULL;
```

6) Liste des notes en précisant pour chacune le nom et le prénom de l'étudiant qui l'a obtenue.

```
SELECT NOTE, NOM, PRENOM FROM NOTATION
INNER JOIN ETUDIANT E on E.NUMETU = NOTATION.NUMETU;
```

Ou

```
SELECT NOM, PRENOM, NOTE FROM ETUDIANT
INNER JOIN NOTATION ON ETUDIANT.NUMETU = NOTATION.NUMETU;
```

7) Moyennes des notes de chaque étudiant (indiquer le nom et le prénom), classées de la meilleure à la moins bonne (les notes à NULL sont à considérer comme 0).

```
SELECT AVG(NVL(NOTE, 0)) MEAN, NOM, PRENOM FROM NOTATION
INNER JOIN ETUDIANT E on E.NUMETU = NOTATION.NUMETU
GROUP BY E.NUMETU, NOM, PRENOM
ORDER BY MEAN desc;
```

8) Liste des étudiants qui ont obtenu la plus petite moyenne en Informatique

```
SELECT ETUDIANT.NUMETU,ETUDIANT.NOM,ETUDIANT.PRENOM, CS_AVERAGE.AVERAGE
FROM ETUDIANT
INNER JOIN (
    SELECT NUMETU, AVG(NOTE) AS AVERAGE
    FROM (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
    INNER JOIN MATIERE M ON M.CODEMAT = E.CODEMAT
    WHERE M.LIBELLE = 'Informatique'
    GROUP BY NUMETU
) CS_AVERAGE ON ETUDIANT.NUMETU = CS_AVERAGE.NUMETU
WHERE CS_AVERAGE.AVERAGE = (
    SELECT MIN(AVERAGE)
    FROM (
        SELECT NUMETU, AVG(NOTE) AS AVERAGE
        FROM (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
        INNER JOIN MATIERE M ON M.CODEMAT = E.CODEMAT
        WHERE M.LIBELLE = 'Informatique'
        GROUP BY NUMETU
    )
);
```

9) Liste des étudiants qui ont obtenu la meilleure moyenne

```
SELECT ETUDIANT.NUMETU, ETUDIANT.NOM, ETUDIANT.PRENOM, MAIN_AVERAGE.AVERAGE
FROM ETUDIANT
INNER JOIN (
    SELECT NUMETU, AVG(NOTE) AS AVERAGE
    FROM (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
    GROUP BY NUMETU
) MAIN_AVERAGE ON ETUDIANT.NUMETU = MAIN_AVERAGE.NUMETU
WHERE MAIN_AVERAGE.AVERAGE = (
    SELECT MAX(AVERAGE)
    FROM (
```

```
        SELECT NUMETU, AVG(NOTE) AS AVERAGE
        FROM (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
        GROUP BY NUMETU
    )
);
```

10) Liste des étudiants qui ont participé à toutes les épreuves

```
SELECT ETUDIANT.NOM, ETUDIANT.PRENOM
FROM ETUDIANT,(
    SELECT * FROM (
        SELECT COUNT(*) NB FROM EPREUVE) EP, (
            SELECT COUNT(NUMETU) NB, NUMETU FROM NOTATION GROUP BY NUMETU
        ) ETU
        WHERE EP.NB = ETU.NB
    ) RES
WHERE RES.NUMETU = ETUDIANT.NUMETU;
```

Ou plus propremment :

```
SELECT ETUDIANT.NUMETU, ETUDIANT.NOM, ETUDIANT.PRENOM
FROM ETUDIANT INNER JOIN NOTATION N ON ETUDIANT.NUMETU = N.NUMETU
GROUP BY ETUDIANT.NUMETU,ETUDIANT.NOM, ETUDIANT.PRENOM
HAVING COUNT(*) = (SELECT COUNT(DISTINCT NUMEPREUVE) FROM EPREUVE);
```

III - Gestion des vues

1) Créer la vue renfermant tous les étudiants ayant eu des épreuves en Informatique ainsi que les notes obtenues.

```
CREATE VIEW ETU_INFO AS (
    SELECT ETU.NUMETU AS NUMERO,ETU.PRENOM AS PRENOM ,ETU.NOM AS NOM,NVL(N.NOTE,
0) AS NOTE
    FROM (
        (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
        INNER JOIN ETUDIANT ETU ON ETU.NUMETU = N.NUMETU
    )
    INNER JOIN MATIERE M ON E.CODEMAT = M.CODEMAT
    WHERE M.LIBELLE = 'Informatique'
    AND E.NUMEPREUVE IS NOT NULL
);
```

2) Donner la moyenne et le nombre d'épreuves en informatique de chaque étudiant ayant passé au moins une épreuve dans cette matière.

```
SELECT NUMERO,NOM,PRENOM,COUNT(*) AS "NOMBRE D'EXAMENS",AVG(NOTE) AS "MOYENNE EN
INFORMATIQUE"
FROM ETU_INFO
GROUP BY NUMERO,NOM,PRENOM
```

4) Donner les noms d'étudiants ainsi que leur moyenne ayant eu une moyenne supérieure ou égale à 10 en Informatique et classés par ordre de mérite.

```
SELECT NUMERO, PRENOM, NOM, AVG(NOTE) AS "MOYENNE EN INFORMATIQUE"
FROM ETU_INFO GROUP BY NUMERO,PRENOM, NOM HAVING AVG(NOTE)>=10 ORDER BY AVG(NOTE)
DESC;
```

5) Donner les noms d'étudiants qui n'ont passé aucune épreuve en Informatique.

```
SELECT * FROM ETUDIANT WHERE NUMETU NOT IN (SELECT NUMERO FROM ETU_INFO GROUP BY
NUMERO)
```

IV – Gestion de la confidentialité

1) Créer la vue EtudLyonnais renfermant tous les étudiants lyonnais (La vue obtenue contient tous les champs de la table Etudiant).

```
CREATE VIEW ETU_LYONNAIS AS (
    SELECT * FROM ETUDIANT WHERE VILLE = 'Lyon'
);
```

2) Donner la moyenne en informatique des étudiants lyonnais

```
SELECT E.NUMETU, AVG(N.NOTE) FROM ETU_LYONNAIS E, (
        SELECT NUMETU, NVL(NOTE, 0) AS NOTE FROM NOTATION
        INNER JOIN (EPREUVE E2 INNER JOIN MATIERE M ON E2.CODEMAT = M.CODEMAT AND
M.LIBELLE = 'INFORMATIQUE'
    )
    ON NOTATION.NUMEPREUVE = E2.NUMEPREUVE
) N WHERE N.NUMETU = E.NUMETU
GROUP BY E.NUMETU
```

3) L'étudiant Dupond a quitté l'établissement. Est-il possible de le supprimer depuis la vue EtudLyonnais ? Vérifier ?

Non, il y a une violation de contrainte d'intégrité - enregistrement fils existant, car l'utilisateur ne fait pas parti de la view

```
DELETE FROM ETU_LYONNAIS WHERE NOM = 'Dupond';
```

4) L'étudiant Durand déménage lors de son stage à 1, Rue de Lyon Paris 75002. Est-il possible de mettre à jour ses informations depuis la vue EtudLyonnais ? Vérifier ?

Oui cela est possible.

```
UPDATE ETU_LYONNAIS SET VILLE = 'Paris' where nom = 'Durand';
```

5) Est-il possible d'ajouter l'étudiant `(700, Gates, Bill, 1980-09-1, Rue de Paris, 69005, Lyon)` depuis la vue EtudLyonnais ? Vérifier ?

Oui cela est possible.

```
INSERT INTO ETU_LYONNAIS VALUES  (700, 'Gates', 'Bill', '09-01-1980', 'Rue de
Paris', 69005, 'Lyon');
```

Vérification :

```
SELECT * FROM ETU_LYONNAIS;
```

6) Donner la vue MoyLyonnais contenant la moyenne par matière des étudiants lyonnais.

```
CREATE VIEW MOYLYONNAIS AS (
    SELECT ETU.NUMETU,ETU.PRENOM,ETU.NOM,ETU.RUE, AVG(N.NOTE) AS AVERAGE
    FROM (NOTATION N INNER JOIN EPREUVE E ON N.NUMEPREUVE = E.NUMEPREUVE)
    INNER JOIN ETUDIANT ETU ON N.NUMETU = ETU.NUMETU
    WHERE ETU.NUMETU IN (SELECT NUMETU FROM ETU_LYONNAIS)
    GROUP BY ETU.NUMETU,ETU.PRENOM,ETU.NOM,ETU.RUE
)
```

7) Est-il possible de mettre à jour la moyenne en informatique de l'étudiant Dupont depuis la vue MoyLyonnais ?

Oui c'est possible car il est dans la view Moy_Lyonnais

8) Donnez un droit d'accès en consultation à la vue EtudLyonnais à votre binôme (Un second compte que vous allez créer si nécessaire sous Oracle). Vérifier le contenu de la vue depuis les

2 comptes.

On donne le privilege d'update à jm: GRANT UPDATE ON MoyLyonnais to jm; On donne le privilege de select à jm: GRANT SELECT ON MoyLyonnais to jm;

9) Mettez à jour le nom de rue de Dupont qui passe à Rue de la Meuse, 69008, Lyon. Vérifier sur les 2 comptes si la mise à jour est faite. Le second compte essaye de remettre Dupont à son ancienne adresse. Que se passe –t-il ? Quoi faire ?

Chaque user voit uniquement ses changements, pour pouvoir voir les changements des autres users il faut que l'autre user commit après modification; Si user1 fait une modification il faut qu'il commit pour que user2 puisse voir les modifications. C'est un peu comme git. L'user1 quand il insert/delete/update/... il commit en 'local' et pour pousser les informations sur origin il doit push (commit pour sql).

10) Votre binôme insère un nouvel étudiant lyonnais. Que doit-il faire pour que vous puissiez consulter ce nouvel étudiant ?

Il faut utiliser l'instruction COMMIT; pour mettre fin à la transaction et l'enregistrer dans la base.

11) Une erreur de saisie entre 2 notes de 2 étudiants sur une même épreuve doit être corrigée. Comment procéder ?

On peut utiliser le mécanisme de rollback qui permet d'annuler les instructions de la transaction en cas d'erreur.

12) Que doit-on faire pour éviter des mises à jour erronées des notes des étudiants par vous ou votre binôme (conflit de mise à jour) ?

Il faut mettre en place des transactions en désactivant l'autocommit sur la session. Le moteur transactionnel du SGBD permet d'assurer la cohérence de la base entre l'exécution de plusieurs instructions.