

TP : Création des tables, instructions LMD et LDD

Énoncé :

1- Soit la base de données « **personnel** » comportant les trois tables suivantes : **Table employe**: contient

- nemp : entier de trois chiffres ; //numéro de l'employé
- nom : chaîne de longueur max 15 caractères, doit être renseigné ; // son nom
- prenom : chaîne de longueur max 15 caractères, doit être renseigné ; // son prénom
- username : chaîne de longueur max 10 caractères;
- salaire : réel de 7 chiffres au max dont 2 après la virgule ; //le salaire de l'employé
- prime : réel de 7 chiffres au max dont 2 après la virgule ; // la prime de l'employé
- daterecrut : une date dont sa valeur par défaut est la date du jour ; // date de recrutement de l'employé

Table departement :

- codedep : entier de 2 chiffres ; // code du département
- nomdep : chaîne de 15 caractères ; // nom du département
- lieu : chaîne de 15 caractères // le lieu où est situé le département
- Table grade** : contient
- grade : entier de 2 chiffres ; // le grade d'un employé
- salmin : entier; // salaire minimal d'un grade
- salmax : entier; // le salaire max d'un grade

2- la population de la table employe est

Nemp	Nom	prenom	username	salaire	prime	daterecrut
001	Laribi	Mohammed		2500.50		12/10/1992
002	Bossiaux	Mathias		3560.30		06/09/1997
003	KHUDOYAN	Kylian		4180.69		03/12/1999
004	KWIZERA	Soumaya		2920.98		18/09/2000
005	DUSSY	Lila		5300.55		11/11/2002

3- La population de la table departement est :

Codedep	Nomdep	lieu
01	Administration	Paris
02	Recherche	Amiens
03	Comptabilité	Lille
04	Vente	Paris
05	Achat	Abbeville

4- La population de la table grade est :

Grade	Salmin	Salmax
01	700.00	1200.00
02	1201.00	1400.00
03	1401.00	2000.00
04	2001.00	3000.00
05	3001.00	9000.00

5- on a les règles de gestion :

- Chaque employé est associé à un seul département
- Chaque département peut avoir plusieurs employés

// Création des tables//

Q1) Créer ces trois tables.

```
CREATE TABLE Employe (
Nemp INTEGER PRIMARY KEY,
Nom VARCHAR(50) NOT NULL,
Prenom VARCHAR(100) NOT NULL,
Username VARCHAR(50),
Salaire REAL NOT NULL,
Prime REAL,
Daterecrut VARCHAR(20)
);
CREATE TABLE Departement (
```

```
Codedep INTEGER PRIMARY KEY,  
Nomdep VARCHAR(50) NOT NULL,  
Lieu VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE Grade (  
Grade INTEGER PRIMARY KEY,  
Salmin REAL NOT NULL,  
Salmax REAL NOT NULL  
);
```

Q2) Vérifier la création de chaque table (utiliser l'instruction `desc nom_table`).

```
PRAGMA table_info(Employe);  
PRAGMA table_info(Departement);  
PRAGMA table_info(Grade);
```

// Modification du schéma et manipulation des données//

Q3) Ajouter la colonne fono : une chaîne de 10 caractères //fonction d'un employé

```
ALTER TABLE Employe  
ADD Fono VARCHAR(10) NOT NULL;
```

Q4) Insérer les populations des tables employe, departement et grade.

Table Employe :

```
INSERT INTO Employe ("Nemp","Nom","Prenom","Salaire","Daterecrut") VALUES  
(001,"Laribi","Mohammed",2500.50,"12/10/1992");
```

```
INSERT INTO Employe ("Nemp","Nom","Prenom","Salaire","Daterecrut") VALUES  
(002,"Bossiaux","Mathias",3560.30,"06/09/1997");
```

```
INSERT INTO Employe ("Nemp","Nom","Prenom","Salaire","Daterecrut") VALUES  
(003,"KHUDOYAN","Kylian",4180.69,"03/12/1999");
```

```
INSERT INTO Employe ("Nemp","Nom","Prenom","Salaire","Daterecrut") VALUES  
(004,"KWIZERA","Soumaya",2920.98,"18/09/2000");
```

```
INSERT INTO Employe ("Nemp","Nom","Prenom","Salaire","Daterecrut") VALUES  
(005,"DUSSY","Lila",5300.55,"11/11/2002");
```

Table Departement :

```
INSERT INTO Departement ("Codedep","Nomdep","Lieu") VALUES (01,"Administration","Paris");
```

```
INSERT INTO Departement ("Codedep","Nomdep","Lieu") VALUES (02,"Recherche","Amiens");
```

```
INSERT INTO Departement ("Codedep","Nomdep","Lieu") VALUES (03,"Comptabilité","Lille");
```

```
INSERT INTO Departement ("Codedep","Nomdep","Lieu") VALUES (04,"Vente","Paris");
```

```
INSERT INTO Departement ("Codedep","Nomdep","Lieu") VALUES (05,"Achat","Abbeville");
```

Table Grade :

```
INSERT INTO Grade ("Grade","Salmin","Salmax") VALUES (01,"700.00","1200.00");
INSERT INTO Grade ("Grade","Salmin","Salmax") VALUES (02,"1201.00","1400.00");
INSERT INTO Grade ("Grade","Salmin","Salmax") VALUES (03,"1401.00","2000.00");
INSERT INTO Grade ("Grade","Salmin","Salmax") VALUES (04,"2001.00","3000.00");
INSERT INTO Grade ("Grade","Salmin","Salmax") VALUES (05,"3001.00","9000.00");
```

Q5) Confirmer l'ajout des lignes. (utiliser l'instruction select * from nom_table;)

```
SELECT *
FROM Employe
```

```
SELECT *
FROM Departement
```

```
SELECT *
FROM Grade
```

Q6) Concaténer la première lettre de la colonne nom avec les 7 premières lettres de la colonne prenom de la table employe pour produire le username (utiliser l'instruction select* from nom_table ; avant et après la génération du username)

```
UPDATE Employe
SET username =
SUBSTR(Nom, 1, 1) || SUBSTR(Prenom, 1, 7);
```

Q7) Changer les salaires de tous les employés de salaires inférieurs à 3000 pour le porter à 3200.45

```
UPDATE Employe
SET Salaire = 3200.45
WHERE Salaire < 3000 ;
```

Q8) Pour tous les employés, calculer la prime par la formule suivante : pour les employés dont le salaire dépasse 4000,00 la prime est de 4% du salaire ; pour les autres, la prime est de 5 % du salaire.

```
UPDATE Employe
SET Prime = Salaire * 0.04
WHERE Salaire > 4000 ;
```

```
UPDATE Employe
SET Prime = Salaire * 0.05
WHERE Salaire <= 4000 ;
```

Q9) créer une nouvelle table qu'on appellera **emp** à partir de la table **employe**, qui contient les colonnes numemp, nom, salaire, daterecrut.

```
CREATE TABLE emp AS
SELECT nemp AS numemp, nom, salaire, daterecrut
FROM employe;
```

Q10) Supprimer la colonne daterecrut de la table emp. Vérifier que la suppression a été effectuée.

```
ALTER TABLE emp DROP COLUMN daterecrut;
```

DESC emp; -- Verification

Q11) Supprimer la dernière ligne de la table emp. Vérifier votre suppression.

DELETE FROM emp WHERE numemp = (SELECT MAX(numemp) FROM emp);

Q12) vider toute la table emp. Confirmer que la table est vidée.

TRUNCATE TABLE emp;
-- OR: **DELETE FROM emp;**

Q13) Modifier la table employé en ajoutant une colonne « codedep » pour indiquer le département dans lequel est affecté l'employé

ALTER TABLE employe ADD codedep INT;

Q14) Insérer dans la table employé la ligne (6,'houem', 'hasna', 'h.houem',3000.00, 20). Que constatez vous ?

**INSERT INTO employe (nemp, nom, prenom, username, salaire, codedep)
VALUES (6, 'houem', 'hasna', 'h.houem', 3000.00, 20);**

Q15) Ajouter la contrainte d'intégrité référentielle à la table employé (codedep est clé étrangère référençant la colonne codedep dans la table département

-- First, we must remove the invalid row from Q14 or the constraint will fail to add
DELETE FROM employe WHERE nemp = 6;

**ALTER TABLE employe
ADD CONSTRAINT fk_dept
FOREIGN KEY (codedep) REFERENCES departement(codedep);**

Q16) Insérer dans la table employé la ligne (7,'bouchemal', 'Kylian ','f.bouchemal', 20). Que constatez vous ?

**INSERT INTO employe (nemp, nom, prenom, username, codedep)
VALUES (7, 'bouchemal', 'Kylian', 'f.bouchemal', 20);**

Q17) Ré essayer l'insertion dans la table employé de la ligne (7,'bouchemal', 'Kylian ','f.bouchem', 2). Que constatez vous ?

**INSERT INTO employe (nemp, nom, prenom, username, codedep)
VALUES (7, 'bouchemal', 'Kylian', 'f.bouchem', 2);**

Q18) Ajouter la contrainte not null à la colonne username. Que constatez vous ?

ALTER TABLE employe MODIFY username VARCHAR(10) NOT NULL;

Q19) Insérer dans la table la ligne (13,' KHUDOYAN', 'mostefa'). Que constatez vous ?

INSERT INTO employe (nemp, nom, prenom) VALUES (13, 'KHUDOYAN', 'mostefa');

Q20) Ajouter une contrainte pour vérifier que le salaire est positif

ALTER TABLE employe ADD CONSTRAINT chk_salaire CHECK (salaire > 0);

Q21) Insérer dans la table employé la ligne (14, 'DANTREVAUX', Théo, 't.dantrevaux', -50). Que constatez vous?

```
INSERT INTO employe (nemp, nom, prenom, username, salaire)
VALUES (14, 'DANTREVAUX', 'Théo', 't.dantrevaux', -50);
```

Q22) Ajouter une contrainte dans la table grade pour vérifier toujours que le salaire minimum d'un grade doit toujours être < au salaire maximum

```
ALTER TABLE grade ADD CONSTRAINT chk_grade_logic CHECK (salmin < salmax);
```

Q23) Ajouter à la table grade la ligne (6, 4500.34, 3600.50) ; que constatez vous ?

```
INSERT INTO grade VALUES (6, 4500.34, 3600.50);
```

Q24) Ajouter la colonne responsable de la table département avec l'intégrité référentielle vers la tab le employé.

```
ALTER TABLE departement ADD responsable INT;
ALTER TABLE departement
ADD CONSTRAINT fk_responsable
FOREIGN KEY (responsable) REFERENCES employe(nemp);
```

// Modification du schéma et manipulation des données//

Q25) Créer une vue Vue_Salaire_Sup qui affiche :

- le numéro de l'employé
- le nom
- le prénom
- le salaire

Pour tous les employés ayant un salaire supérieur ou égal à 4000.

Résultat attendu : une vue filtrée uniquement sur les hauts salaires.

```
CREATE VIEW Vue_Salaire_Sup AS
SELECT nemp, nom, prenom, salaire
FROM employe
WHERE salaire >= 4000;
```

Q26) Afficher le contenu de la vue Vue_Salaire_Sup.

```
SELECT * FROM Vue_Salaire_Sup;
```

Q27) Créer une vue Vue_Employe_Departement affichant :

- le nom de l'employé
- le prénom
- le nom du département
- le lieu du département

```

CREATE VIEW Vue_Employe_Departement AS
SELECT e.nom, e.prenom, d.nomdep, d.lieu
FROM employe e
JOIN departement d ON e.codedep = d.codedep;

```

Q28) Afficher uniquement les employés travaillant à Paris à l'aide de la vue créée précédemment.

```

SELECT * FROM Nom_De_Ta_Vue
WHERE ville = 'Paris';

```

Q29) Créer une vue Vue_Employes_Bien_Payes contenant uniquement :

- nemp
- nom
- salaire

Pour les employés dont le salaire est strictement supérieur à **3500**, avec l'option **WITH CHECK OPTION**.

```

CREATE VIEW Vue_Employes_Bien_Payes AS
SELECT nemp, nom, salaire
FROM Employes
WHERE salaire > 3500
WITH CHECK OPTION;

```

Q30) À partir de la vue Vue_Employes_Bien_Payes, tenter d'insérer :

- un employé avec un salaire de **3000**
- puis un employé avec un salaire de **4200**

Que constatez-vous dans chaque cas ?

Expliquez le rôle de **WITH CHECK OPTION**.

Cas	Sal aire	Résultat	Pourquoi ?
Emplo yé 1	300 0	Échec (Erreur)	Le salaire est inférieur à 3500. Le WITH CHECK OPTION bloque l'insertion.
Emplo yé 2	420 0	Succès	Le salaire est supérieur à 3500, ce qui respecte la condition de la vue.

Le rôle du WITH CHECK OPTION :

C'est un garde-fou. Il garantit que toute modification (insertion ou mise à jour) effectuée à travers la vue respecte la condition définie dans le WHERE de cette vue.

Sans cette option, tu pourrais insérer un employé avec un salaire de 3000 via la vue ; il irait dans la table de base, mais il "disparaîtrait" immédiatement de la vue car il ne remplit pas la condition \$salaire > 3500\$. Le WITH CHECK OPTION empêche cette incohérence.

Q31) Supprimer la vue Vue_Employes_Bien_Payes.

```

DROP VIEW Vue_Employes_Bien_Payes;

```