

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

TD1

Exercice 1 :

1. Ecrire un script SQL pour la création de tables suivantes :

Produit (Numprod, Desprod, Poids, Qte_stk, #CodMag)

Magasin (NumMag, AdresseM, Surface)

Client (NumClt, NomC, AgeClt, AdresseC)

Commande (NumClt, Numprod, DateC, QteC, Prix_vente)

Avec :

- Les attributs soulignés sont des clés primaires
- Les attributs précédés par le caractère « # » sont des clés étrangères
- Numprod, NumMag et NumClt: sont de type numérique de taille 6
- Desprod, NomC, AdresseC et AdresseM: sont de type chaîne de caractère de longueur maximale 30,
- L'âge de client est de type numérique de taille 2
- Poids, Qte_stk, Surface, QteC, Prix_vente : sont des réels de taille 8 chiffres au total dont trois chiffres après la virgule
- NumClt et NumProd : Clé primaire multiple
- DateC est de type date

On suppose que :

- deux produits différents ne peuvent pas avoir la même désignation
- le poids d'un produit et le Prix_vente doivent être positif
- Le domaine de valeurs de la colonne AgeClt de la table Client s'étend de 20 à 50.
- Le champ AdresseC n'accepte pas de valeurs nulles
- La valeur par défaut de l'adresse du magasin est « Tunis »

2. Ecrire un script SQL pour supprimer le champ AdresseM de la table Magasin

3. Ecrire un script SQL pour ajouter un champ Couleur de type caractère à la table Produit

4. Ecrire un script SQL pour modifier la taille de champ Nomclt de 30 à 20

5. Ecrire un script SQL pour renommer le champ Qte_stk to QteS de la table Produit

6. Ecrire un script SQL pour ajouter à la table Magasin la contrainte suivante : la surface doit être comprise entre 10 et 100 m².

7. Ecrire un script SQL pour afficher la table Produit

Exercice 2 :

Une société souhaite utiliser une base de données pour gérer son équipement informatique. Le bâtiment de son siège est composé de trois étages. Chaque étage possède son propre réseau Ethernet (ou un segment de réseau distinct). A ces réseaux sont connectées des salles équipées de postes de travail. Un poste de travail est une machine sur laquelle sont installés certains logiciels. La base de données doit également décrire l'installation du logiciel.

Les noms et types des colonnes sont comme suit:

Table Segment		
Colonne	Commentaires	Types
<u>indIP</u>	trois premiers bloc IP (exemple : 130.120.80)	VARCHAR2(11)
nomSegment	nom du segment	VARCHAR2(20)
etage	étage du segment	NUMBER(2)

Table Salle		
Colonne	Commentaires	Types
<u>nSalle</u>	numéro de la salle	VARCHAR2(7)
nomSalle	nom de la salle	VARCHAR2(20)
nbPoste	nombre de postes de travail dans la salle	NUMBER(2)
#indIP	trois premiers bloc IP (exemple : 130.120.80)	VARCHAR2(11)

Table Poste		
Colonne	Commentaires	Types
<u>nPoste</u>	code du poste de travail	VARCHAR2(7)
nomPoste	nom du poste de travail	VARCHAR2(20)
#indIP	trois premiers bloc IP (exemple : 130.120.80)	VARCHAR2(11)
ad	dernier bloc de chiffres IP (exemple : 11)	VARCHAR2(3)
typePoste	type du poste (Unix, TX, PCWS, PCNT)	VARCHAR2(9)
#nSalle	numéro de la salle	VARCHAR2(7)

Table Logiciel		
Colonne	Commentaires	Types
<u>nLog</u>	code du logiciel	VARCHAR2(5)
nomLog	nom du logiciel	VARCHAR2(20)
dateAch	date d'achat du logiciel	DATE
version	version du logiciel	VARCHAR2(7)
typeLog	type du logiciel (Unix, TX, PCWS, PCNT)	VARCHAR2(9)
prix	prix du logiciel	NUMBER(6,2)

Table Installer		
Colonne	Commentaires	Types
<u>nPoste</u>	code du poste de travail	VARCHAR2(7)
<u>nLog</u>	code du logiciel	VARCHAR2(5)
numIns	numéro séquentiel des installations	NUMBER(5)

dateIns	date d'installation du logiciel	DATE
delai	intervalle entre achat et installation	INTERVAL DAY(5) TO SECOND(2),

Table Type		
Colonne	Commentaires	Types
typeLP	types des logiciels et des postes	VARCHAR2(9)
nomType	noms des types (Terminaux X, PC Windows...)	VARCHAR2(20)

1. Ecrire un script SQL pour la création des tables avec leurs clés primaires et les contraintes suivantes :

- Les noms des segments, des salles et des postes n'acceptent pas de valeurs nulles.
- Le domaine de valeurs de la colonne ad de la table Poste s'étend de 0 à 255.
- La colonne prix au niveau de la table Logiciel doit être supérieure ou égale à 0.
- La colonne dateIns de la table Installer sera égale à la date du jour par défaut.

2. Ecrire un script SQL qui affiche la description de toutes les tables.

3. Enrichir le script créé dans la question n°1 par des requêtes permettant l'alimentation de la base de données par les lignes suivantes :

Table Salle			
nSalle	nomSalle	nbPoste	indIP
S01	Salle 1	3	130.120.80
S02	Salle 2	2	130.120.80
S03		2	130.120.80
S11	Salle 11	2	130.120.81
S12	Salle 12	1	130.120.81
S21	Salle 21	2	130.120.82

Table Poste					
nPoste	nomPoste	indIP	Ad	TypePoste	nSalle
P1	Poste 1	130.120.80	01	TX	S01
P2	Poste 2	130.120.80	02	UNIX	S01
P3	Poste 3	130.120.80	03	TX	S01
P4	Poste 4	130.120.80	04	PCWS	S02
P5	Poste 5	130.120.80	05	PCWS	S02
P6	Poste 6	130.120.80	06	UNIX	S03
P7	Poste 7	130.120.80	07	TX	S03
P8	Poste 8	130.120.81	01	UNIX	S11

Table Logiciel					
nLog	nomLog	dateAch	version	typeLog	prix
Log1	Oracle 10	12/02/2018	10i	UNIX	3000

Log2	Oracle 11	06/07/2017	11i	UNIX	4500
Log3	SQL SERVER	11/12/2017	8	PCWS	2300
Log4	Front Page	05/02/2011	6	PCWS	900
Log5	Windev	08/12/2014	7	BeOS	1200
Log6	SQL*NET	19/12/2014	2.0	UNIX	1400
Log7	IIS	15/07/2014	7	PCWS	400
Log8	DreamWeaver	08/09/2013	2.0	BeOS	200

Table Segment		
indIP	nomSegment	etage
130.120.80	Brin RDC	
130.120.81	Brin 1er étage	
130.120.82	Brin 2ème étage	

Table Type	
typeLP	nomType
TX	Terminal X Windows
UNIX	Système Unix
PCWS	PC Windows
BeOS	OS Haiku

4. Dans ce même script, créez la séquence sequenceIns commençant à la valeur 1, d'incrément 1. Utilisez cette séquence pour estimer la colonne numIns de la table Installer. Insérez les enregistrements suivants :

Table Installer				
nPoste	nLog	numIns	dateIns	delai
P2	Log1	1	14/02/2018	
P2	Log2	2	17/07/2017	
P4	Log5	3	17/07/2017	
P6	Log6	4	20/05/2015	
P6	Log1	5	13/02/2018	
P8	Log2	6	19/07/2017	
P7	Log6	7	20/05/2015	
P3	Log7	11	01/04/2014	

5. Modifier le contenu des tables déjà créées comme suit :

- Donner le nom «Salle 3» à la salle numéro « S03».
- Augmenter le prix d'achat de logiciel numéro « Log8 » de 200.

6. Ecrire des requêtes permettant de formuler les besoins en informations ci-dessous à l'aide d'instructions SELECT:

- Numéros des logiciels installés sur le poste 'p6'.
- Nom, adresse IP, numéro de salle des postes de type 'Unix' ou 'PCWS'.
- Même requête pour les postes du segment '130.120.80' triés par numéros de salles décroissants.
- Pour chaque poste, le nombre de logiciels installés (en utilisant la table Installer).

- Moyenne des prix des logiciels 'Unix'.
- Plus récente date d'achat d'un logiciel.

7. Ecrire un script SQL pour la destruction des tables.

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

Correction TD1

Exercice 1 :

1. Ecrire un script SQL pour la création de tables suivantes

```
CREATE TABLE Produit
(Numprod number(6), /* Inline Numprod number(6) primary key, */
Desprod varchar(30) unique, /* Inline*/
Poids number(8,3),
Qte_stk number(8,3),
/* Out line Constraint stNumprod check(Numprod is not null), */
/* Out line Constraint un_Desprod unique(Desprod), */
Constraint Ck1_Produit CHECK (Poids >=0),
Constraint PK_Produit primary key (NumProd),
Constraint FK_Produit Foreign Key (CodMag) references Magasin(NumMag));
*****

Create Table Magasin
(NumMag number(6),
AdresseM varchar(30) default 'Tunis',

Surface number(8,3),
constraint PK_ NumMag primary key (NumMag) );
*****

create table Client
(NumClt number(6),
NomC VARCHAR2(30),
AgeClt number(2) check(AgeClt between 20 and 50), /* Inline*/
AdresseC VARCHAR2(30) not null, /* Inline*/
constraint PK_CodC primary key (NumClt)
);
/*out Line
Constraint st_sex_ch check(st_sex in('m','f')),
Constraint st_age_ch check(st_age between 20 and 30)
Constraint st_ AdresseC check(AdresseC is not null)*/
****
```

```

CREATE TABLE Commande
(
NumClt number(6),
NumProd number(6),
QteC number(8,3),
Prix_vente number(8,3),
DateC DATE,
/* in line NumProd number(6)references Produit (NumProd)*/
)
Constraint Ck1_ Prix_vente CHECK (Prix_vente >=0));
constraint Pk_LigneC primary key (NumClt, NumProd)) ,
/*out line*/
constraint fk_IN1 foreign key (NumProd e) references Produit(NumProd) ,
constraint fk_In2 foreign key (NumClt) references Client(Numclt) );

```

2. Ecrire un script SQL pour supprimer le champ AdresseM de la table Magasin

```

ALTER TABLE Magasin
DROP AdresseM;

```

3. Ecrire un script SQL pour ajouter un champ Couleur de type caractère à la table Produit

```

ALTER TABLE Produit
add(Couleur char(1));

```

4. Ecrire un script SQL pour modifier la taille de champ Nomclt de 30 à 20

```

ALTER TABLE Client
Modify NomClt number(20);

```

5. Ecrire un script SQL pour renommer le champ Qte_stk to QteS de la table Produit

```

Alter table Produit Rename column Qte_stk To QteS;

```

6. Ecrire un script SQL pour ajouter à la table Magasin la contrainte suivante : la surface doit être comprise entre 10 et 100 m².

```

ALTER TABLE Magasin
ADD Constraint ck1_magasin check(surface between 10 and 100) ;

```

7. Ecrire un script SQL pour afficher la table Produit

```

Desc Produit;

```

Exercice 2 :

1. Ecrire un script SQL pour la création des tables :

```

CREATE TABLE Segment(
indIP VARCHAR2(11),
nomSegment VARCHAR2(20)not null ,
Etag NUMBER(2),
constraint pk_segment primary key (indIP));

```

```

CREATE TABLE Salle(
nSalle VARCHAR2(7),
nomSalle VARCHAR2(20),
nbPost NUMBER(2),
indIP VARCHAR2(11),
constraint pk_salle primary key(nSalle),
constraint fk_salle foreign key (indIP) references Segment(indIP));

```

```

CREATE TABLE Poste(
nPoste VARCHAR2(7),
nomPoste VARCHAR2(20),
indIP VARCHAR2(11),
Ad VARCHAR2(3),
typePoste VARCHAR2(9),
nSalle VARCHAR2(7),
constraint pk_poste primary key(nPoste),
CONSTRAINT ck_ad CHECK (ad BETWEEN '000' AND '255'),
CONSTRAINT fk_indIP FOREIGN KEY (indIP) REFERENCES segment (indIP),
constraint fk_poste foreign key (nSalle) references Salle(nSalle));

```

```

CREATE TABLE Logiciel(
nLog VARCHAR2(5),
nomLog VARCHAR2(20),
dateAch DATE,
version VARCHAR2(7),
typeLog VARCHAR2(9),
Prix NUMBER(6,2),
constraint pk_Logiciel primary key (nLog),
constraint ck_Prix CHECK (Prix>=0));

```

```

CREATE TABLE Installer(
nPoste VARCHAR2(7),
nLog VARCHAR2(5),
numIns NUMBER(5),
dateIns DATE default SYSDATE,
Delai INTERVAL DAY(5) TO SECOND (2),
constraint pk_Installer primary key (nPoste,nLog),
constraint fk_IN1 foreign key (nPoste) references Poste(nPoste) ,
constraint fk_In2 foreign key (nLog) references Logiciel(nLog) );

```



```
CREATE TABLE Types(
typeLP VARCHAR2(9),
nomType VARCHAR2(20),
constraint pk_Types primary key(typeLP));
```

2. Ecrire un script SQL qui affiche la description de toutes les tables.

```
DESC Segment;
DESC Salle;
DESC Poste;
DESC Logiciel;
DESC Installer;
DESC Types;
```

3. Enrichir le script crée dans la question n°1

```
INSERT INTO Segment VALUES ('130.120.80','Brin RDC',NULL);
INSERT INTO Segment VALUES ('130.120.81','Brin 1er étage',NULL);
INSERT INTO Segment VALUES ('130.120.82','Brin 2ème étage',NULL);

INSERT INTO Salle VALUES ('s01','Salle 1',3,'130.120.80');
INSERT INTO Salle VALUES ('s02','Salle 2',2,'130.120.80');
INSERT INTO Salle VALUES ('s03','Salle 3',2,'130.120.80');
INSERT INTO Salle VALUES ('s11','Salle 11',2,'130.120.81');
INSERT INTO Salle VALUES ('s12','Salle 12',1,'130.120.81');
INSERT INTO Salle VALUES ('s21','Salle 21',2,'130.120.82');
INSERT INTO poste VALUES ('p1','Poste 1','130.120.80','01','SOL','s01');
INSERT INTO poste VALUES ('p2','Poste 2','130.120.80','02','UNIX','s01');
INSERT INTO poste VALUES ('p3','Poste 3','130.120.80','03','IBM','s01');
INSERT INTO poste VALUES ('p4','Poste 4','130.120.80','04','SOL','s02');
INSERT INTO poste VALUES ('p5','Poste 5','130.120.80','05','PCWS','s02');
INSERT INTO poste VALUES ('p6','Poste 6','130.120.80','06','UNIX','s03');
INSERT INTO poste VALUES ('p7','Poste 7','130.120.80','07','IBM','s03');
INSERT INTO poste VALUES ('p8','Poste 8','130.120.81','01','UNIX','s11');
INSERT INTO poste VALUES ('p9','Poste 9','130.120.81','02','IBM','s11');
INSERT INTO poste VALUES ('p10','Poste 10','130.120.81','03','UNIX','s12');
INSERT INTO poste VALUES ('p11','Poste 11','130.120.82','01','IBM','s21');
INSERT INTO poste VALUES ('p12','Poste 12','130.120.82','02','PCWS','s21');

INSERT INTO logiciel VALUES
('log1','Oracle OLAP', TO_DATE('12-02-2018','DD-MM-
YYYY'),'12c','UNIX',3000);
INSERT INTO logiciel VALUES
```

```

('log2','Oracle Ent.', TO_DATE('06-07-2017','DD-MM-
YYYY'),'12c','UNIX',4500);
INSERT INTO logiciel VALUES
('log3','SQL Server', TO_DATE('11-12-2017','DD-MM-YYYY'),'2017','IBM',8300);
INSERT INTO logiciel VALUES
('log4','Front Page', TO_DATE('05-02-2011','DD-MM-YYYY'),'6','PCWS',900);
INSERT INTO logiciel VALUES
('log5','WinDev', TO_DATE('08-12-2014','DD-MM-YYYY'),'7','SOL',1200);
INSERT INTO logiciel VALUES
('log6','SQL*Net', NULL, '2.0','UNIX',1400);
INSERT INTO logiciel VALUES
('log7','I. I. S.', TO_DATE('15-07-2014','DD-MM-YYYY'),'7','IBM',400);
INSERT INTO logiciel VALUES
('log8','DreamWeaver',TO_DATE('08-09-2013','DD-MM-YYYY'),'2.0','PCWS',200);

INSERT INTO Types VALUES ('IBM', 'Terminal IBM');
INSERT INTO Types VALUES ('UNIX','Système Unix');
INSERT INTO Types VALUES ('PCWS','PC Windows');
INSERT INTO Types VALUES ('SOL', 'SOLARIS');

```

4. Dans ce même script, créez la séquence sequenceIns commençant à la valeur 1, d'incrément 1

```

CREATE SEQUENCE sequenceIns INCREMENT BY 1 START WITH 1 MAXVALUE 10000
NOCYCLE;
INSERT INTO installer VALUES ('p2', 'log1', sequenceIns.NEXTVAL,
TO_DATE('14-02-2018','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p2', 'log2', sequenceIns.NEXTVAL,
TO_DATE('17-07-2017','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p4', 'log5', sequenceIns.NEXTVAL, NULL,NULL);
INSERT INTO installer VALUES ('p6', 'log6', sequenceIns.NEXTVAL,
TO_DATE('20-05-2015','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p6', 'log1', sequenceIns.NEXTVAL,
TO_DATE('13-02-2018','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p8', 'log2', sequenceIns.NEXTVAL,
TO_DATE('19-07-2017','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p8', 'log6', sequenceIns.NEXTVAL,
TO_DATE('20-05-2015','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p11','log3', sequenceIns.NEXTVAL,
TO_DATE('20-04-2017','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p12','log4', sequenceIns.NEXTVAL,
TO_DATE('20-04-2013','DD-MM-YYYY'),NULL);

```

```

INSERT INTO installer VALUES ('p11','log7', sequenceIns.NEXTVAL,
TO_DATE('20-04-2014','DD-MM-YYYY'),NULL);
INSERT INTO installer VALUES ('p7', 'log7', sequenceIns.NEXTVAL,
TO_DATE('01-04-2014','DD-MM-YYYY'),NULL);
COMMIT;

```

5. Modifier le contenu des tables déjà créés comme suit :

- Donner le nom «Salle 3» à la salle numéro « S03». `UPDATE Salle SET nomSalle= 'Salle 3' WHERE nSalle='S03' ;`
- Augmenter le prix d'achat de logiciel numéro « Log8 » de 200. `UPDATE Logiciel SET prix =prix + 20. WHERE nlog='log8' ;` //si tout les logiciels : `UPDATE Logiciel SET prix = prix + 200 ;`

6. Ecrire des requêtes à l'aide d'instructions SELECT:

- Numéros des logiciels installés sur le poste 'p6'. `SELECT nLog FROM Installer WHERE nPoste = 'p6';`
- Nom, adresse IP, numéro de salle des postes de type 'Unix' ou 'PCWS'.
`SELECT nomPoste, indIP, ad, nSalle FROM poste WHERE typePoste = 'UNIX' OR typePoste = 'PCWS';`
- Même requête pour les postes du segment '130.120.80' triés par numéros de salles décroissants.
`SELECT nomPoste, indIP, ad, nSalle FROM poste WHERE (typePoste = 'UNIX' OR typePoste = 'PCWS') AND indIP = '130.120.80' ORDER BY nSalle DESC;`
- Pour chaque poste, le nombre de logiciels installés (en utilisant la table Installer).
`SELECT nPoste, COUNT(nLog) FROM installer GROUP BY (nPoste);`
- Moyenne des prix des logiciels 'Unix'.
`SELECT AVG(prix) FROM Logiciel WHERE typeLog = 'UNIX';`
- Plus récente date d'achat d'un logiciel. `SELECT MAX(dateAch) FROM Logiciel;`

7. Ecrire un script SQL pour la destruction des tables.

```

DROP TABLE Installer;
DROP TABLE Types;
DROP TABLE Poste;
DROP TABLE Salle;
DROP TABLE Logiciel;
DROP TABLE Segment;

```

Matière : Base de données Avancée	
Chargée de cours : Farah Barika Ktata Chargée de TD : Emna Ammar Elhadjamor	TD2

Exercice 1 :

1. Parmi les déclarations de variables suivantes, déterminer celles qui sont incorrectes :

- `v_id NUMBER(4);`
- `v_x, v_y, v_z VARCHAR2(10);`
- `name VARCHAR2(30);`
- `Pi CONSTANT NUMBER := 3.14159;`
- `v_en_stock BOOLEAN := 1;`

2. Soit les déclarations de variables suivantes :

`DECLARE`

`nom VARCHAR2(6);`

`nom2 VARCHAR2(6);`

`prime NUMBER(5,2);`

- Affecter la chaîne de caractères 'Ahmed' à `nom`
- Affecter la variable `nom` à `nom2`
- Affecter le nombre 500.50 à `prime`

3. A l'aide d'une instruction `FOR`, écrivez un bloc anonyme pour afficher les entiers de l'intervalle 10..50.

4. A l'aide d'une instruction `LOOP`, écrivez un bloc PL/SQL permettant d'insérer les chiffres de 1 à 100 dans la table `Compte(NO)`.

5. A l'aide d'une instruction `For..LOOP`, écrivez un bloc PL/SQL permettant d'afficher la somme des nombres entre 10 et 20.

6. Ecrivez des fonctions permettant de convertir une date selon un format bien déterminé :

- `FUNCTION Lib_Jour:` retourne le libellé du jour de la date système
- `FUNCTION Lib_Month:` retourne le libellé du mois de la date système

7. Réécrivez des fonctions précédentes en des procédures paramétrées.

Exercice 2 :

Une société souhaite utiliser une base de données pour gérer son parc de véhicules. Soit la table suivante:

Table TRAJET		
Colonne	Commentaires	Types
NOTRAJ	numéro du trajet	INTEGER
VILLEDEP	ville de départ	CHAR(20)
VILLEARR	ville d'arrivée	CHAR(20)
DATETRAJET	date du trajet	DATE
NBKM	Nombre de kilomètres	INTEGER

1. Ecrivez un bloc PL/SQL pour :
 - Entrez un nom d'une ville,
 - Affichez le nombre moyen, le nombre minimum, le nombre maximum de kilomètres des trajets qui sont partis de cette ville.
2. Ecrivez un bloc PL/SQL pour :
 - Entrer un numéro de trajet ;
 - Afficher la date, la ville de départ et la ville d'arrivée de ce trajet. En utilisant l'exception NO_DATA_FOUND, afficher un message d'erreur si le trajet n'existe pas.

Exercice 3 :

1. Proposez un script PL/SQL qui affiche, à l'aide du paquetage DBMS_OUTPUT, les détails de la dernière installation sous la forme suivante :

```
Dernière installation en salle : numérodSalle
```

```
-----
```

```
Poste : numérodPoste Logiciel : nomduLogiciel en date du dateInstallation
```

Vous utiliserez les directives %TYPE pour extraire directement les types des colonnes et pour améliorer ainsi la maintenance du bloc.

Exercice 4 :

Soit la table Enseignant (Id_enseignant, nom, grade, age, nombre_modules, # Id_Departement), et la table Departements (Id_Departement, nom_Departement)

1. Ecrivez un bloc PL/SQL permettant de :
 - Trouver le nom de l'enseignant et le nombre des modules du Id_enseignant =100.
 - Utilisez IF SQL%Found pour mettre à jour le nombre des modules (5) du l'enseignant numéro 200.
 - Trouver les enseignants dont leurs noms « Mohamed », écrire une exception en cas ou l'instruction Select retourne plusieurs valeurs.
 - Compter le nombre total de n-uplets dans la table Enseignant et stocker le résultat dans une variable
 - Compter le nombre d'enseignants dont le grade (grade) est PROFESSEUR dans la table Enseignant et stocker le résultat dans une deuxième variable.
 - Calculer la proportion (en pourcentage), stocker le résultat dans une troisième variable et afficher le résultat à l'écran.
2. Écrivez une procédure pour ajouter un nouveau département.

3. Supprimez la procédure de la question précédente.
4. Écrivez une fonction pour compter le nombre d'enseignants pour un département donné.
5. En utilisant un Curseur, on veut extraire les numéros et les noms des enseignants du département
numéro 5
6. Écrivez une procédure PLSQL qui prends en paramètre un NUMBER (age limite) et qui affiche pour
chaque département le nombre des enseignants qui dépassent l'âge limite. Utilisez un curseur avec
paramètre l'âge limite.

Exercice 5 :

Soient les tables suivantes représentant un sous-ensemble des données utilisées par une application de gestion commerciale.

Article (codearticle, designation, prixunitaire, qtestock)

Commande(numcommande, datecommande, idclient)

Lignecommande(numcommande, numligne, codearticle, qtecommandee)

On souhaite écrire une unité de traitement PL/SQL permettant d'afficher la liste des commandes triées par client et par date de commande.

1. Écrivez une procédure permettant d'afficher les informations relatives à une commande dont on donne le numéro.
2. Écrivez une procédure permettant l'affichage de toutes les commandes d'un client dont on donne le numéro. Les commandes doivent être triées par date.
3. Écrivez un bloc PL/SQL permettant l'affichage de toutes les commandes de tous les clients. Cette liste doit être triées par numéro du client.
4. Écrivez un trigger permettant de tenir à jour la quantité en stock automatiquement après les opérations de mise à jour de la table (insertion, modification ou suppression).

Exercice 6 :

On considère une entreprise d'exploitation de films cinématographique qui souhaite développer une application informatique pour la planification de la projection des films dans les salles. On considère les tables suivantes :

Film (codefilm, titre, datesortie, typefilm)

Salle(codesalle, capacite, adresse)

Affectation (codefilm, codesalle, datedebut, datefin)

1. Écrivez un bloc PL/SQL permettant d'afficher le titre et la date de sortie des films de type 'action' dont la date de sortie est comprise entre le 01/10/2004 et le 31/12/2004. L'affichage doit être fait par ordre chronologique.

2. Écrivez une fonction permettant de retourner le nombre de films ayant été affectés à une salle donnée pendant une période donnée.
3. Écrivez un bloc PL/SQL permettant l’affichage en synthèse de l’activité de toutes les salles pendant l’année 2004. Pour chaque salle, ce bloc doit afficher le nombre de films ayant été affectés pendant l’année 2004.
4. On souhaite garder un historique de toutes les mises à jour effectuées dans les trois tables Film, Salle et Affectation. Pour chaque opération, on veut mémoriser la table, le type de l’opération, le nom de l’utilisateur et la date de l’opération. Donner la structure de la table qui va stocker cet historique ainsi que le code du trigger correspondant.

Exercice 7 :

Soient les tables suivantes :

VOL (Numvol, Heure_départ, Heure_arrivée, Ville_départ, Ville_arrivée)

PILOTE (Matricule, Nom, Age, Salaire)

1. Écrivez un déclencheur qui vérifie que le NumVol commence par les lettres 'AF' avant son l'insertion dans la table VOL.
2. Écrivez un programme PL/SQL qui insère le vol AF110 partant de Paris à 21h40 et arrivant à Tunis à 23h10 (hypothèse : le vol n’est pas déjà présent dans la table).
3. Écrivez un programme PL/SQL qui calcule la moyenne des salaires des pilotes dont l’âge est entre 30 et 40 ans.

Exercice 8 :

On désire connaître, pour chaque logiciel installé, le temps (nombre de jours) passé entre l’achat et l’installation. Ce calcul devra renseigner la colonne delai de la table Installer pour l’instant nulle.

Les résultats devront être affichés (par DBMS_OUTPUT.PUT_LINE) ainsi que les incohérences (date d’installation antérieure à la date d’achat, date d’installation ou date d’achat inconnue).

1. Écrivez la procédure calculTemps pour programmer ce processus.

Un exemple d’état de

sortie est présenté ci-après :

Logiciel Oracle 6 sur Poste 2, attente 2924 jour(s).

Logiciel Oracle 8 sur Poste 2, attente 1463 jour(s).

*Date d'achat inconnue pour le logiciel SQL*Net sur Poste 2*

Pas de date d'installation pour le logiciel WinDev sur Poste 4

...

Logiciel I. I. S. installé sur Poste 7 11 jour(s) avant d'être acheté

...

2. Écrivez le déclencheur Trig_Après_DI_Installer sur la table Installer permettant de faire la mise à jour automatique des colonnes nbLog de la table Poste, et nbInstall de la table Logiciel. Prévoir les cas de désinstallation d'un logiciel sur un poste, et d'installation d'un logiciel sur un autre.
3. Écrivez le déclencheur Trig_Après_DI_Poste sur la table Poste permettant de mettre à jour la colonne nbPoste de la table Salle à chaque ajout ou suppression d'un nouveau poste.
4. Écrivez le déclencheur Trig_Après_U_Salle sur la table Salle qui met à jour automatiquement la colonne nbPoste de la table Segment après la modification de la colonne nbPoste.
Ces deux derniers déclencheurs vont s'enchaîner : l'ajout ou la suppression d'un poste entraînera l'actualisation de la colonne nbPoste de la table Salle qui conduira à la mise à jour de la colonne nbPoste de la table Segment.
5. Ajouter un poste pour vérifier le rafraîchissement des deux tables (Salle et Segment).
6. Supprimer ce poste puis vérifier à nouveau la cohérence des deux tables.
7. Écrivez le déclencheur Trig_Avant_UI_Installer sur la table Installer permettant de contrôler, à chaque installation d'un logiciel sur un poste, que le type du logiciel correspond au type du poste, et que la date d'installation est soit nulle soit postérieure à la date d'achat.

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

Correction TD2

Exercice 1 :

1. Parmi les déclarations de variables suivantes, déterminer celles qui sont incorrectes :

- `v_id NUMBER(4);` Correcte
- `v_x,v_y,v_z VARCHAR2(10);` Incorrecte : un seul identifiant par ligne
- `name VARCHAR2(30);` correcte
- `Pi CONSTANT NUMBER := 3.14159;` correcte
- `v_en_stock BOOLEAN := 1;` Incorrecte : 1 n'est pas une valeur booléenne

2. Soit les déclarations de variables suivantes :

- Affecter la chaîne de caractères 'Ahmed' à nom

```
BEGIN
```

```
nom := 'Ahmed';
```

- Affecter la variable nom à nom2

```
nom2 := nom; Affectation d'une variable.
```

- Affecter le nombre 500.50 à prime

```
prime := 500.50;
```

3. A l'aide d'une instruction FOR, écrivez un bloc anonyme pour afficher les entiers de l'intervalle

10..50

```
BEGIN
```

```
FOR I IN 10..50 LOOP
```

```
DBMS_OUTPUT.PUT_LINE (I);
```

```
END LOOP;
```

```
END
```

4. A l'aide d'une instruction LOOP, écrivez un bloc PL/SQL permettant d'insérer les chiffres de 1 à 100 dans la table Compte(NO).

```
DECLARE
```

```
nb NUMBER := 1 ;
```

```
BEGIN
```

```

LOOP
INSERT INTO Compte
VALUES (nb) ;
nb = nb + 1 ;
EXIT WHEN nb > 100 ;
END LOOP
END

```

5. A l'aide d'une instruction For..LOOP, écrivez un bloc PL/SQL permettant d'afficher la somme des nombres entre 10 et 20.

```

DECLARE
somme NUMBER := 0 ;
BEGIN
FOR i IN 10..20 LOOP
somme = somme + i ;
END LOOP
DBMS_OUTPUT.PUT_LINE('Somme = ' || somme) ;
END

```

6. Ecrivez des fonctions permettant de convertir une date selon un format bien déterminé :

- FUNCTION Lib_Jour: retourne le libellé du jour de la date système

```

FUNCTION Lib_Jour (P_date DATE DEFAULT SYSDATE)
RETURN VARCHAR2 IS
BEGIN
RETURN (TO_CHAR(P_DATE, 'Day')) ;
END Lib_Jour ;
//test de la fonction DBMS_OUTPUT.PUT_LINE (Lib_Jour (SYSDATE));

```

- FUNCTION Lib_Month: retourne le libellé du mois de la date système

```

FUNCTION Lib_Month (P_date DATE DEFAULT SYSDATE)
RETURN VARCHAR2 IS
BEGIN
RETURN (TO_CHAR(P_DATE, 'Month')) ;
END Lib_Month;

```

7. Réécrivez des fonctions précédentes en des procédures paramétrées :

```

PROCEDURE Lib_Jour (P_Libelle OUT VARCHAR2, P_Date date DEFAULT
SYSDATE) IS
BEGIN P_Libelle:= TO_CHAR (P_DATE, 'Day')) ;
END Lib_Jour ;

```

```

PROCEDURE Lib_Month (P_Libelle OUT VARCHAR2, P_Date date DEFAULT
SYSDATE) IS
BEGIN P_Libelle:= TO_CHAR (P_DATE, 'Month')) ;
END Lib_Month;

```

Exercice 2 :

1. Ecrivez un bloc PL/SQL pour :

- Entrez un nom d'une ville,
- Affichez le nombre moyen, le nombre minimum, le nombre maximum de kilomètres des trajets qui sont partis de cette ville.

```

DECLARE
nom_ville CHAR(20) ;
nombre_min INTEGER ;
nombre_max INTEGER ;
nombre_moyen INTEGER ;
BEGIN
-- Entrez un nom d'une ville
nom_ville := '&Entrez_un_nom_ville';
-- Selectionnez
SELECT AVG(nbkm), MIN(nbkm), MAX(nbkm)
INTO nombre_moyen, nombre_min, nombre_max
FROM trajet
WHERE villedep = nom_ville ;
-- Affichez
DBMS_OUTPUT.PUT_LINE('Ville départ : '|| nom_ville) ;
DBMS_OUTPUT.PUT_LINE(' - Nombre moyen : '||
TO_CHAR(nombre_moyen)) ;
DBMS_OUTPUT.PUT_LINE(' - Nombre minimum : '||
TO_CHAR(nombre_min)) ;
DBMS_OUTPUT.PUT_LINE(' - Nombre maximum : '||
TO_CHAR(nombre_max)) ;
END ;

```

2. Ecrivez un bloc PL/SQL pour :

- Entrer un numéro de trajet ;
- Afficher la date, la ville de départ et la ville d'arrivée de ce trajet. En utilisant l'exception NO_DATA_FOUND, afficher un message d'erreur si le trajet n'existe pas.

```

DECLARE
m_notraj trajet.notraj%TYPE := &Entrez_numero_trajet;
m_datetrajet trajet.datetrajet%TYPE;
m_villedep trajet.villedep%TYPE;
m_villearr trajet.villearr%TYPE;
BEGIN
-- Sélectionnez les informations de ce trajet:
SELECT datetrajet, villedep, villearr
INTO m_datetrajet, m_villedep, m_villearr
FROM trajet WHERE m_notraj = notraj ;

```

```

IF SQL%FOUND THEN
-- Affichez les informations
DBMS_OUTPUT.PUT_LINE ('No trajet: '||m_notraj);
DBMS_OUTPUT.PUT_LINE ('Date : '||m_datetrajet);
DBMS_OUTPUT.PUT_LINE ('Départ : '||m_villedep);
DBMS_OUTPUT.PUT_LINE ('Arrivée : '||m_villearr);
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Ce trajet n'existe pas ');
END ;

```

Exercice 3 :

1. Proposez un script PL/SQL qui affiche, à l'aide du paquetage DBMS_OUTPUT, les détails de la dernière installation

```

SET SERVEROUTPUT ON

DECLARE

v_sequenceInsMax Installer.numIns%TYPE;
v_nPoste          Installer.nPoste%TYPE;
v_nLog            Installer.nLog%TYPE;
v_dateIns         Installer.dateIns%TYPE;
v_nSalle          Poste.nSalle%TYPE;
v_nomLog          Logiciel.nomLog%TYPE;

BEGIN

SELECT  numIns, nPoste, nLog, dateIns INTO v_sequenceInsMax,
v_nPoste, v_nLog, v_dateIns
FROM    Installer WHERE v_dateIns = (SELECT MAX(dateIns) FROM
Installer);

--

SELECT  nSalle INTO v_nSalle
FROM    Poste  WHERE nPoste = v_nPoste;

--

SELECT  nomLog INTO v_nomLog
FROM    Logiciel WHERE nLog = v_nLog;

--

DBMS_OUTPUT.PUT_LINE('Dernière installation en salle : '||
v_nSalle);

DBMS_OUTPUT.PUT_LINE('-----

');

```

```

DBMS_OUTPUT.PUT_LINE('Poste : ' || v_nPoste || ' Logiciel : ' ||
v_nomLog || ' en date du ' || v_dateIns);
END;
/

```

Exercice 4 :

1. Ecrivez un bloc PL/SQL permettant de :

- Trouver le nom de l'enseignant et le nombre des modules du Id_enseignant =100

```

DECLARE
V_Name Enseignant.nom %Type ;
V_NBM Number;
V_Id Enseignant. Id_enseignant %Type := 100 ;
BEGIN
Select nom, nombre_modules into V_Name , V_NBM
From Enseignant
Where Id_enseignant = V_Id ;
DBMS_OUTPUT.PUT_LINE( V_Name || ' and His number of mod is : ' ||
NBM) ;
END ;

```

- Utilisez IF SQL%Found pour mettre à jour le nombre des modules (5) du l'enseignant numéro 200

```

DECLARE
V_Name Varchar2 (20);
BEGIN
Select nom into V_Name
From Enseignant
Where Id_enseignant = 200;
IF SQL%Found Then
Update Enseignant
Set nombre_modules = 5
Where Id_enseignant = 200;
End IF;
END;
*****

```

- Trouver les enseignants dont leurs noms « Mohamed », écrire une exception en cas ou l'instruction Select retourne plusieurs valeurs.

```

DECLARE
V_Name Varchar2(20);
BEGIN
Select nom into V_Name
From Enseignant
Where nom Like 'Mohamed' ;
DBMS_OUTPUT.PUT_LINE ('V_Name');
EXCEPTION
When Too_Many_Rows Then
DBMS_OUTPUT.PUT_LINE ('Query Retrieved Multiple Rows');
END;

```

- Compter le nombre total de n-uplets dans la table Enseignant et stocker le résultat dans une variable
- Compter le nombre d'enseignants dont le grade (grade) est PROFESSEUR dans la table Enseignant et stocker le résultat dans une deuxième variable;
- Calculer la proportion (en pourcentage), stocker le résultat dans une troisième variable et afficher le résultat à l'écran.

```

DECLARE
ntot INTEGER; -- Nombre total d'enseignants
nprf INTEGER; -- Nombre de Profs
pprf REAL;    -- Proportion de Profs
personne EXCEPTION; -- Exception : pas d'Enseignants
BEGIN
SELECT COUNT(*) INTO ntot FROM Enseignant;
IF (ntot = 0) THEN
RAISE personne;
END IF;
SELECT COUNT(*) INTO nprf FROM Enseignant WHERE grade = 'Professeur';
pprf := 100 * nprf / ntot;
DBMS_OUTPUT.PUT_LINE('Proportion de Profs = ' || pprf || ' %');
EXCEPTION
WHEN personne THEN
RAISE_APPLICATION_ERROR(-20500, 'La table Enseignant est vide !');
END;

```

2. Ecrire une procédure pour ajouter un nouveau département

```

CREATE PROCEDURE Add_Dept(V_Dept_Id in Number, V_Dept_Name
Varchar2)
IS
BEGIN
Insert into Departments ( Department_Id,Department_Name)
Values (V_Dept_Id , V_Dept_Name ); DBMS_OUTPUT.PUT_LINE ('Inserted
'||
SQL%Rowcount ||' row ');
END;

/* Pour l'exécution écrire soit - Execute Add_Dept ; ou bien
- BEGIN Add_Dept ; END; */

```

3. Supprimer la procédure de la question précédente.

```

DROP Procedure Add_Dept;

```

4. Ecrire une fonction pour compter le nombre d'enseignants pour un département donné.

```

CREATE OR REPLACE FUNCTION proc_dept (p_no IN
Department.Department_Id %TYPE)
RETURN NUMBER AS v_no NUMBER;
BEGIN
SELECT COUNT(Id_enseignant)
INTO v_no
FROM Enseignant
WHERE Department_Id =p_no;
RETRUN (v_no);
END;

```

5. En utilisant un Curseur, extraire les numéros et les noms des enseignants du département numéro 5

```

DECLARE
Cursor Ens_Cursor is Select Id_enseignant, nom
From Enseignant where Department_Id = 5 ;
V_Id Enseignant. Id_enseignant %Type ;
V_Name Enseignant.nom%Type ;
BEGIN
Open Ens_Cursor ;
LOOP
Fetch Ens_Cursor into V_Id , V_Name ;
Exit When Ens_Cursor %Notfound ;
DBMS_OUTPUT.PUT_LINE ( V_Id || ' and his name is :' || V_Name );

```

```

End Loop ;
Close Ens_Cursor ;
END;
*****

DECLARE
Cursor Emp_Cursor is Select Employee_Id , Last_Name
From Employees
Where Department_Id = 30 ;
V_Rec Emp_Cursor %Rowtype ;
BEGIN
Open Emp_Cursor ;
LOOP
Fetch Emp_Cursor into V_Rec;
Exit When Emp_Cursor %Notfound ;
DBMS_OUTPUT.PUT_LINE ( V_Rec.Employee_Id || " " || V_Rec.Last_Name
);
End Loop ;
Close Emp_Cursor ;
END;

```

6. Écrivez une procédure PLSQL qui prends en paramètre un NUMBER (age limite) et qui affiche pour chaque département le nombre des enseignants qui dépassent l'age limite. Utilisez un curseur avec paramètre l'age limite.

```

CREATE OR REPLACE PROCEDURE moyenneAge(AgeLim IN NUMBER)
IS
CURSOR CS(Age_Limite NUMBER) IS
SELECT DEPARTEMENT AS DNOM, COUNT(*) AS NB
FROM Enseignant
WHERE AGE > Age_Limite
GROUP BY DEPARTEMENT ;
BEGIN
FOR DEPT IN CS(AgeLim) LOOP
DBMS__OUTPUT.PUT_LINE(DEPT.DNOM || ' ' || DEPT.NB)
END FOR
END

```

Exercice 5 :

Soient les tables suivantes représentant un sous-ensemble des données utilisées par une

application de gestion commerciale.

Article (codearticle, designation, prixunitaire, qtestock)

Commande(numcommande, datecommande, idclient)

Lignecommande(numcommande, numligne, codearticle, qtecommandee)

On souhaite écrire une unité de traitement PL/SQL permettant d'afficher la liste des commandes triées par client et par date de commande.

1. Écrivez une procédure permettant d'afficher les informations relatives à une commande dont on donne le numéro.

```
Create or replace procedure AfCmd(N IN number)
as
C Commande%rowtype; --enregistrement: structure de la table commande
Begin
select * INTO C
from Commande
where numcommande=N;
dbms_output.put_line('le numcommande est : '||C.numcommande);
dbms_output.put_line('la datecommande est : '||C.datecommande);
dbms_output.put_line('le idclient est : '||C.idclient);
exception
when NO_DATA_FOUND then dbms_output.put_line('numcommande
introuvable');
end;
```

2. Écrivez une procédure permettant l'affichage de toutes les commandes d'un client dont on donne le numéro. Les commandes doivent être triées par date.

```
create or replace procedure commcl (nc in number)
is
ncomm commande.numcommande%type;
dtecomm commande.datecommande%type;
cursor c1 is select numcommande,datecommande into ncomm,dtecomm
from commande
where idclient=nc
order by datecommande;
null exception;
begin
open c1;
```

```

loop
fetch c1 into ncomm,dtecomm;
exit when c1%notfound;
dbms_output.put_line(ncomm||' | '||dtecomm );
dbms_output.put_line('-----');
end loop;
if(c1%rowcount=0) then RAISE null;
end if;
close c1;
exception
when null then dbms_output.put_line('idclient introuvable');
end;;

```

3. Écrivez un bloc PL/SQL permettant l’affichage de toutes les commandes de tous les clients. Cette liste doit être triées par numéro du client.

```

declare
cursor c2 is select * from Commande order by idclient ASC;
C Commande%rowtype;
begin
dbms_output.put_line(' IdClient '||' numcommande '||'
datecommande ');
open c2;
loop
fetch c2 INTO C;
exit when c2%notfound;
dbms_output.put_line(C.idclient||' '||C.numcommande||'
'||C.datecommande);
end loop;
dbms_output.put_line('Les lignes traitées = '||c2%rowcount);
close c2;
end;
/

```

4. Écrivez un trigger permettant de tenir à jour la quantité en stock automatiquement après les opérations de mise à jour de la table **Lignecommande** (insertion, modification ou suppression).

```

create or replace trigger qtstock
after insert or update or delete on Lignecommande
for each row

```

```

begin
if inserting then
update article
set qtestock=qtestock-:new.qtecommandee
where codearticle=:new.codearticle;
elsif updating then
update article
set qtestock=qtestock+:old.qtecommandee-:new.qtecommandee
where codearticle=:new.codearticle;
elsif deleting then
update article
set qtestock=qtestock+:old.qtecommandee
where codearticle=:new.codearticle;
end if;
end;

```

Exercice 6

On considère une entreprise d'exploitation de films cinématographique qui souhaite développer une application informatique pour la planification de la projection des films dans les salles. On considère les tables suivantes :

Film (codefilm, titre, datesortie, typefilm)

Salle(codesalle, capacite, adresse)

Affectation (codefilm, codesalle, datedebut, datefin)

1. Écrivez un bloc PL/SQL permettant d'afficher le titre et la date de sortie des films de type 'action' dont la date de sortie est comprise entre le 01/10/2004 et le 31/12/2004. L'affichage doit être fait par ordre chronologique.

```

declare
cursor c3 is select * from film where typefilm='action' and
datesortie between
'01/12/2004' and '31/12/2004' order by titre;
C film%rowtype;
begin
dbms_output.put_line(' titre '||' datesortie ');
open c3;
loop
fetch c3 INTO C;

```

```

exit when c3%notfound;
dbms_output.put_line(C.titre||' '||C.datesortie);
end loop;
close c3;
end;
/

```

2. Écrivez une fonction permettant de retourner le nombre de films ayant été affectés à une salle donnée pendant une période donnée.

```

create or replace function nbrefilm(d1 IN date,d2 IN date,salle IN
number)
return number
is
nb number;
Begin
select count(codofilm) INTO nb from Affectation where (d1>= datedebut
and
d2<=datefin) and codesalle=salle;
return nb;
end;
/
declare
dt1 date;
dt2 date;
s Affectation.codesalle%type;
begin
dbms_output.put_line('le nbre de film dans cette periode est
'||nbrefilm(dt1,dt2,s));
end;
/

```

3. Écrivez un bloc PL/SQL permettant l’affichage en synthèse de l’activité de toutes les salles pendant l’année 2004. Pour chaque salle, ce bloc doit afficher le nombre de films ayant été affectés pendant l’année 2004.

```

declare
cursor c3 is select distinct Codesalle from Affectation;
s c3%rowtype;

```

```

C number;
begin
dbms_output.put_line('Codesalle'||' '||'Nombre_film');
open c3;
loop
fetch c3 INTO s;
exit when c3%notfound;
select count(idefilm) INTO C
from Affectation
where To_char(datedebut,'fmYYYY')=2004 and Codesalle=s.codesalle;
dbms_output.put_line(s.codesalle||' '||C);
end loop;
close c3;
end;
/

```

4. On souhaite garder un historique de toutes les mises à jour effectuées dans les trois tables **Film**, **Salle** et **Affectation**. Pour chaque opération, on veut mémoriser la table, le type de l'opération, le nom de l'utilisateur et la date de l'opération. Donner la structure de la table qui va stocker cet historique ainsi que le code du trigger correspondant.

a. Structure de la table de stockage :

```

create table t1
(tab varchar2(15),
typ varchar2(3),
user varchar2(15),
dateMaj date);

```

b. Déclencheur pour stocker l'historique des opérations.

```

create or replace trigger histo
after insert or update or delete on Affectation
for each row
begin
if inserting then
INSERT INTO t1
VALUES('Affectation','I',USER,sysdate);
elsif updating then
INSERT INTO t1

```

```
VALUES('Affectation','U',USER,sysdate);
elsif deleting then
INSERT INTO t1
VALUES('Affectation','D',USER,sysdate);
end if;
end;
/
--De même pour les autres tables
```

Exercice 7 :

- Écrivez un déclencheur qui vérifie que le NumVol commence par les lettres 'AF' avant son l'insertion dans la table VOL.

```
CREATE OR REPLACE TRIGGER VERIFIE_CODE_VOL
BEFORE INSERT OR UPDATE ON VOL
FOR EACH ROW
WHEN (:NEW. Numvol NOT LIKE 'AF%')
BEGIN
RAISE_APPLICATION_ERROR(-20001,'VOL_CODE doit commencer par AF');
END;
```
- Écrivez un programme PL/SQL qui insère le vol AF110 partant de Paris à 21h40 et arrivant à Tunis à 23h10 (hypothèse : le vol n'est pas déjà présent dans la table).

```
DECLARE
v vol%ROWTYPE;
BEGIN
v.numvol := 'AF110';
v.heure_départ := to_date('21/11/2013 21:40', 'DD/MM/YYYY
hh24:mi');
v.heure_arrivée := to_date('21/11/2013 23:10', 'DD/MM/YYYY
hh24:mi');
v.ville_départ := 'Paris';
v.ville_arrivée := 'Tunis';
INSERT INTO vol VALUES v;
END;
```
- Écrivez un programme PL/SQL qui calcule la moyenne des salaires des pilotes dont l'âge est entre 30 et 40 ans.

```

DECLARE

CURSOR curseur1 IS SELECT salaire FROM pilote
WHERE (Age >= 30 AND Age <=40);
salairePilote Pilote.Salaire%TYPE;
sommeSalaires NUMBER(11,2) := 0;
moyenneSalaires NUMBER(11,2);
BEGIN
OPEN curseur1;
LOOP
FETCH curseur1 INTO salairePilote;
EXIT WHEN (curseur1%NOTFOUND OR curseur1%NOTFOUND IS NULL);
sommeSalaires := sommeSalaires + salairePilote;
END LOOP;
moyenneSalaires := sommeSalaires / curseur1%ROWCOUNT;
CLOSE curseur1;
DBMS_OUTPUT.PUT_LINE('Moyenne salaires (pilotes de 30 <E0> 40 ans)
: ' ||
moyenneSalaires);
END;

```

Exercice 8 :

1. Écrivez la procédure calculTemps pour programmer ce processus.

Un exemple d'état de

sortie est présenté ci-après :

Logiciel Oracle 6 sur Poste 2, attente 2924 jour(s).

Logiciel Oracle 8 sur Poste 2, attente 1463 jour(s).

Date d'achat inconnue pour le logiciel SQL*Net sur Poste 2

Pas de date d'installation pour le logiciel WinDev sur Poste 4

...

```

CREATE OR REPLACE PROCEDURE calculTemps IS
    CURSOR curseur IS SELECT
        l.nomLog, p.nomPoste, l.dateAch, i.dateIns, i.nLog, i.nPoste
        FROM Installer i, Logiciel l, Poste p
        WHERE i.nPoste = p.nPoste AND i.nLog = l.nLog;
    atte NUMBER(4);
BEGIN

```

```

    FOR enreg IN curseur LOOP
    IF enreg.dateIns IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Pas de date d''installation pour le
logiciel '
        || enreg.nomLog || ' sur ' || enreg.nomPoste);
    ELSE
        IF enreg.dateAch IS NULL THEN
DBMS_OUTPUT.PUT_LINE('Date d''achat inconnue pour le logiciel ' ||
enreg.nomLog || ' sur ' || enreg.nomPoste);
        ELSE
            atte := enreg.dateIns - enreg.dateAch;
            IF atte < 0 THEN
                DBMS_OUTPUT.PUT_LINE('Logiciel ' || enreg.nomLog || '
installé sur ' || enreg.nomPoste || ' ' || -atte || ' jour(s) avant
d''être acheté!');
            ELSE
                IF atte = 0 THEN
                    DBMS_OUTPUT.PUT_LINE('Logiciel ' || enreg.nomLog || ' sur '
|| enreg.nomPoste || ' acheté et installé le même jour!');
                ELSE
                    DBMS_OUTPUT.PUT_LINE('Logiciel ' || enreg.nomLog || ' sur '
|| enreg.nomPoste || ', attente ' || atte || ' jour(s).');
                    UPDATE Installer SET delai = NUMTODSINTERVAL(enreg.dateIns -
enreg.dateAch, 'DAY')
                        WHERE nPoste = enreg.nPoste AND nLog = enreg.nLog;
                END IF;
            END IF;
        END IF;
    END IF;
    END LOOP;
    COMMIT;
END calculTemps;
////////////////////
SET SERVEROUT ON
EXECUTE calculTemps;
SELECT * FROM Installer;

```


2. Écrivez le déclencheur Trig_Après_DI_Installer sur la table Installer permettant de faire la mise à jour automatique des colonnes nbLog de la table Poste, et nbInstall de la table Logiciel. Prévoir les cas de désinstallation d'un logiciel sur un poste, et d'installation d'un logiciel sur un autre.

```
CREATE OR REPLACE TRIGGER Trig_Après_DI_Installer
AFTER INSERT OR DELETE ON Installer
FOR EACH ROW
BEGIN
    IF DELETING THEN
        UPDATE Poste SET nbLog=nbLog - 1
            WHERE nPoste = :OLD.nPoste;
        UPDATE Logiciel SET nbInstall = nbInstall - 1
            WHERE nLog = :OLD.nLog;
    ELSE
        IF INSERTING THEN
            UPDATE Poste SET nbLog = nbLog + 1
                WHERE nPoste = :NEW.nPoste;
            UPDATE Logiciel SET nbInstall = nbInstall + 1
                WHERE nLog = :NEW.nLog;
        END IF;
    END IF;
END;
```

3. Écrivez le déclencheur Trig_Après_DI_Poste sur la table Poste permettant de mettre à jour la colonne nbPoste de la table Salle à chaque ajout ou suppression d'un nouveau poste.

```
CREATE OR REPLACE TRIGGER Trig_Après_DI_Poste
AFTER INSERT OR DELETE ON Poste
FOR EACH ROW
BEGIN
    IF DELETING THEN
        UPDATE Salle SET nbPost = nbPost - 1
            WHERE nSalle = :OLD.nSalle;
    ELSE
        UPDATE Salle SET nbPost = nbPost + 1
            WHERE nSalle = :NEW.nSalle;
    END IF;
END;
```

4. Écrivez le déclencheur Trig_Après_U_Salle sur la table Salle qui met à jour automatiquement la colonne nbPoste de la table Segment après la modification de la colonne nbPoste.

```
CREATE OR REPLACE TRIGGER Trig_Après_U_Salle
AFTER UPDATE OF nbPoste ON Salle
FOR EACH ROW
DECLARE
    differ NUMBER;
BEGIN
    differ := :NEW.nbPoste - :OLD.nbPoste;
    UPDATE Segment SET nbPoste = nbPoste + differ
        WHERE indIP = :NEW.indIP;
END;
```

5. Ajouter un poste pour vérifier le rafraîchissement des deux tables (Salle et Segment).

```
INSERT INTO installer VALUES ('p8', 'log7',
sequenceIns.NEXTVAL, SYSDATE, NULL);
```

6. Supprimer ce poste puis vérifier à nouveau la cohérence des deux tables.

```
DELETE FROM installer WHERE nPoste='p8' AND nLog='log7';
```

7. Écrivez le déclencheur Trig_Avant_UI_Installer sur la table Installer permettant de contrôler, à chaque installation d'un logiciel sur un poste, que le type du logiciel correspond au type du poste, et que la date d'installation est soit nulle soit postérieure à la date d'achat.

```
CREATE OR REPLACE TRIGGER Trig_Avant_UI_Installer
BEFORE INSERT OR UPDATE OF nPoste, nLog ON installer
FOR EACH ROW
DECLARE
    type_log    Types.typeLP%TYPE;
    type_pos    Types.typeLP%TYPE;
    date_achat  DATE;
BEGIN
    SELECT typeLog, dateAch INTO type_log, date_achat
        FROM logiciel WHERE :NEW.nLog = nLog;
    SELECT typePoste INTO type_pos
        FROM Poste WHERE :NEW.nPoste = nPoste;
```

```
IF type_log != type_pos THEN      RAISE_APPLICATION_ERROR(-
20111,'Poste : '||type_pos||' différent du logiciel :
'||type_log);
END IF;
IF :NEW.dateIns IS NOT NULL THEN
    IF :NEW.dateIns < date_achat THEN
        RAISE_APPLICATION_ERROR(-20112,'Installation ('|| :NEW.dateIns
||') inférieure a la date d'achat ('||date_achat||')');
    END IF;
END IF;
END;/
```

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

TD3

Exercice 1 :

1. Ecrire un package qui contient 2 fonctions:

- pour calculer l'aire d'un carré,
- pour calculer l'aire d'un rectangle

2. En se basant sur la table « student » et la séquence « student_req » qui permet l'insertion automatique des valeurs de la colonne « student_id » (Create sequence student_req), Ecrire un package (spécification et body) nommé General_student qui contient :

- une procédure « insert_student » pour l'insertion d'un nouveau étudiant
- une procédure « delete_student » pour la suppression d'un étudiant
- une fonction « get_name » pour retourner le nom de l'étudiant à partir de son numéro

Table student	
Student_id	Number, primary key
First_name	Varchar2(100)
Birthday	Date

3. Ecrire le code correspondant pour exécuter les procédures de ce package

Exercice 2:

1. Donner toutes les raisons pour lesquelles le code suivant est incorrecte, corriger le !

```
Create or replace package proc_rules_calling
Is
Procedure print_emp_details (p_emp_id number)
End
Create or replace package body proc_rules_calling
Is
function get_no_work_days (p_emp_id number)
Return varchar2
Is
V_hiredate date;
Begin
```

```

Select hire_date into v_hiredate
From employees
Where employee_id=id;
Return round (sysdate-v_hiredate);
End;

Procedure print_emp_details (p_emp_id number)
Is
V_details employees%rowtype;
Begin
Select * into v_details from employees
Where employee_id=p_emp_id;
Dbms_output.put_line('id:' || v_details.employee_id);
Dbms_output.put_line('fname:' || v_details.first_name);
Dbms_output.put_line('salary:' || v_details.salary);
Dbms_output.put_line('hire date:' || v_details.Hire_date);
Dbms_output.put_line('nbr of days work:' || get_no_work_days(p_emp_id));
End;

Execute package.print_em_details(101);

```

Exercice 3 :

1. En utilisant SQL dynamique :

- Écrire une procédure « delete_any_table » qui prend en paramètre le nom du tableau à supprimer et qui affiche combien de ligne ont été supprimés dans cette table.
- Écrire une procédure « create_any_table » qui prend en paramètre le nom du tableau à créer et les détails (les colonnes avec leurs types).
- Ecrire le code correspondant pour exécuter ces procédures

2. Expliquer le rôle de ref cursor et commenter le code PLSQL suivant:

```

Declare
Type c_emp_dept is ref cursor ;
D_cursor c_emp_dept;
v_empno employees.employee_id%type;
v_first_name employees.first_name%type;
begin
open D_cursor for select employee_id, first_name
from employees where department_id=10;
loop

```

```

fetch D_cursor into v_empno, v_first_name;
exit when D_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;
close d_cursor;11

```

```

open D_cursor for select employee_id, first_name
from employees where department_id=30;
loop
fetch D_cursor into v_empno, v_first_name;
exit when D_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;
close d_cursor;

```

```

create or replace procedure emp_list( p_dept_id number default
null)
is
type c_emp_dept is ref cursor;
d_cursor c_emp_dept;
v_empno employees.employee_id%type;
v_first_name employees.first_name%type;
v_sql varchar2(1000):='select employee_id, first_name from
employees';
begin
if p_dept_id is null then
open d_cursor for v_sql;
else
v_sql:=v_sql||'where department_id =:id';
open d_cursor for v_sql using p_dept_id;
end if;
loop
fetch d_cursor into v_empno, v_first_name;
exit when d_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;

```

```
close d_cursor;  
end;  
execute emp_list;  
execute emp_list(30);
```

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

Correction TD3

Exercice 1 :

1. Ecrire un package qui contient 2 fonctions: (1) pour calculer l'aire d'un carré, (2) pour calculer l'aire d'un rectangle

- create or replace package area is
function square_area (p_side number) //header of subprog
return number;
function rectangle_area (p_l number, p_w number)
return number;
end;
- create or replace package body area is
function square_area (p_side number) //header of subprog
return number
is begin
return p_side*p_side;
end;
function rectangle_area (p_l number, p_w number) // LONGUEUR x
LARGEUR
return number;
is begin
return p_l*p_w;
end;
end;
Select area.square_area(4) from dual;
Select area.rectangle_area(4,10) from dual;

2. Soit la table suivante et la séquence « student_req » pour ajouter automatiquement les valeurs de la colonne « student_id »

Table student	
Student_id	Number, primary key

First_name	Varchar2(100)
birthday	date

Create sequence student_req ;

3. Ecrire un package (spécification et body) nommé General_student qui contient :

une procédure « insert_student » pour l'insertion d'un nouveau étudiant

une procédure « delete_student » pour la suppression d'un étudiant

une fonction « get_name » pour retourner le nom de l'étudiant à partir de son numéro

- create or replace package General_student **is**
 procedure insert_student (p_first_name varchar2, p_birthday
 date); //header of subprog
 procedure delete_student (p_student_id number);
 function get_name (p_student_id number)
 return varchar2; // pour retourner le nom de l'etudiant
end;
- create or replace package body General_student **is**
 procedure insert_student (p_first_name varchar2, p_birthday date)
 is
 begin
 insert into student values (student_seq.nextval,p_first_name,
 p_birthday);
 commit;
 exception...
 end;
 procedure delete_student (p_student_id number)
 is begin delete from student where student_id=p_student_id;
 commit;
 end;
 function get_name (p_student_id number)
 return varchar2 is
 v_name student.first_name%type;
 begin
 select first_name into v_name
 from student
 where student_id=p_student_id;
 return v_name;

```
exception when others then return null;

end;
```

4. Ecrire le code correspondant pour exécuter les procédures du ce package

```
/* pour l'exécution
Execute general_student.insert_student ('ahmed','10-may-81');
Execute general_student.delete_student (1);
```

Exercice 2 :

```
Create or replace package proc_rules_calling
Is
Procedure print_emp_details (p_emp_id number);
End
Create or replace package body proc_rules_calling
Is
function get_no_work_days (p_emp_id number)
Return number
Is
V_hiredate date;
Begin
Select hire_date into v_hiredate
From employees
Where employee_id=p_emp_id;
Return round (sysdate-v_hiredate);/* function returns a number rounded
to a certain number
End;
Procedure print_emp_details (p_emp_id number)
Is
V_details employees%rowtype;
Begin
Select * into v_details from employees
Where employee_id=p_emp_id;
Dbms_output.put_line('id:' || v_details.employee_id);
Dbms_output.put_line('fname:' || v_details.first_name);
Dbms_output.put_line('salary:' || v_details.salary);
Dbms_output.put_line('hire date:' || v_details.Hire_date);
Dbms_output.put_line('nbr of days work:' || get_no_work_days(p_emp_id));
End;
```

End;

Execute proc_rules_calling.print_em_details(101);

Exercice 3 :

1. En utilisant SQL dynamique,

écrire une procédure « delete_any_table » qui prend en paramètre le nom du tableau à supprimer et qui affiche combien de ligne a été supprimés dans cette table.

```
Create or replace procedure delete_any_table (p_table_name varchar2)
Is
V_no_rec number;
Begin
Execute immediate 'delete from' || p_table_name;
V_no_rec:=sql%rowcount;
Commit;
Dbms_output.put_line(v_no_rec || 'record(s) deleted from' || p_table_name);
End;

/* execute delete_any_table(emp1);
```

écrire une procédure « create_any_table » qui prend en paramètre le nom du tableau à créer et les détails (les colonnes avec leurs types).

```
Create or replace procedure create_any_table (p_table_name varchar2,
p_details varchar2)
Is
V_details varchar2(30000);
Begin
V_details:='create table' || p_table_name || '(' || p_details || ')';
Dbms_output.put_line(v_details);
Execute immediate v_details;
end;

/*      execute      create_any_table('emp1','emp_id      number,      name
varchar2(100)');
```

Expliquer le rôle de ref cursor et commenter le code suivant:

➔ On peut ouvrir le ref cursor plusieurs fois avec différents sql statements

Declare

```

Type c_emp_dept is ref cursor ;//definition d'un cursor dynamique
D_cursor c_emp_dept;// declarer une variable nommé D_cursor est
de de type c_emp_dept
v_empno employees.employee_id%type;
v_first_name employees.first_name%type;
begin
open D_cursor for select employee_id, first_name//SQL statement
de ce sursor
from employees where department_id=10;
loop
fetch D_cursor into v_empno, v_first_name;
exit when D_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;
close d_cursor;11

```

```

open D_cursor for select employee_id, first_name//SQL statement
de ce sursor
from employees where department_id=30;
loop
fetch D_cursor into v_empno, v_first_name;
exit when D_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;
close d_cursor;

```

```

create or replace procedure emp_list( p_dept_id number default
null)
is
type c_emp_dept is ref cursor;
d_cursor c_emp_dept;
v_empno employees.employee_id%type;
v_first_name employees.first_name%type;
v_sql varchar2(1000):='select employee_id, first_name from
employees';
begin

```

```

if p_dept_id is null then
open d_cursor for v_sql;
else
v_sql:=v_sql||'where department_id =:id';
open d_cursor for v_sql using p_dept_id;//recupéré à partir du
parameter
end if;
loop
fetch d_cursor into v_empno, v_first_name;
exit when d_cursor%notfound;
dbms_output.put_line(v_empno||' '||v_first_name);
end loop;
close d_cursor;
end;

execute emp_list; //to get all the employees
execute emp_list(30); //to get all the employees in specific dept

```

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

TD4

Exercice 1:

Soit le schéma BD de la société X :

SERVICE (num_service, nom, responsable, site)

PROJET (num_projet, num_service, nom, responsable, budget, durée_estimée, date_de_début)

EMPLOYE (Mat_emp, # num_service, nom, prénom, date_naissance, adresse, téléphone, fonction, Salaire)

La société X est située sur 3 sites différents (L'attribut site de la table Service): Bizerte, Rades et Skhira. Le site de Radès est également utilisé comme siège social de la société.

- L'attribut nom de SERVICE prend une des valeurs suivantes: "commercial", "financier", "Technologie", "maintenance", "recherche & développement", etc.
- Toutes les données sur la rémunération des employés ou les informations personnelles doivent être centralisées au siège de l'entreprise.
- Les employés seront affectés à un seul site, à l'exception des employés du service de maintenance qui auront la possibilité d'intervenir sur tous les sites.

Question 1. Proposez une stratégie de fragmentation en utilisant des opérateurs d'algèbre relationnelle.

Question 2. Pour chacune des décompositions proposées, donnez les requêtes de reconstruction.

Question 3. Donnez la définition d'une fragmentation correcte. Montrez que votre décomposition est correcte.

Exercice 2:

Trois universités X, Y et Z ont décidé de mettre en commun leurs bibliothèques et leur service de prêt, afin de permettre à tous les étudiants des universités concernées d'emprunter des livres dans toutes les bibliothèques. La gestion commune des bibliothèques et des prêts est assurée par une base de données distribuée, dont le schéma global est le suivant :

EMPLOYÉ (Id_pers, nom, adresse, statut, affectation)

ETUDIANT (Id_etu, nom, adresse, université, cours, nb_emprunts)

LIVRES (Id_ouv, titre, éditeur, année, champ, stock, site)

AUTEURS (#Id_ouv, nom_auteur)

PRÊTS (#Id_ouv, #Id_etu, date_emprunt, date_retour)

La gestion de cette application est basée sur les hypothèses suivantes :

- un employé est affecté à un seul site (L'attribut site désigne la bibliothèque qui gère cet ouvrage)
- un étudiant est inscrit dans une seule université, mais peut emprunter dans toutes les bibliothèques.
- un livre emprunté dans une bibliothèque est retourné dans cette même bibliothèque.
- L'attribut nb_emprunts de la relation ETUDIANT est utilisé pour limiter le nombre de livres empruntés simultanément par un étudiant dans toutes les bibliothèques. Il est mis à jour pour chaque emprunt et chaque retour, quelle que soit la bibliothèque emprunteuse.
- L'attribut stock de la relation LIVRES indique le nombre de livres encore disponibles pour le prêt.
- Chaque université gère ses propres étudiants
- Chaque bibliothèque gère son personnel et ses fonds.

Question 1. Donnez la définition des différents fragments en utilisant les opérateurs de l'algèbre relationnelle ainsi que le schéma de répartition des fragments.

Question 2. Montrez que la fragmentation que vous proposez pour la relation ETUDIANT est correcte.

Question 3. Donner les opérations de reconstruction des relations globales.

Exercice 3:

Soient $Q = \{ q_1, q_2, q_3, q_4, q_5 \}$ un ensemble de requêtes, $A = \{ A_1, A_2, A_3, A_4, A_5 \}$ un ensemble d'attributs, et $S = \{ S_1, S_2, S_3 \}$ un ensemble de sites. La matrice (a) ci-dessous décrit l'utilisation des attributs par les requêtes et la matrice (b) décrit la fréquence d'utilisation des requêtes par les sites.

	A_1	A_2	A_3	A_4	A_5
q_1	0	1	1	0	1
q_2	1	1	1	0	1
q_3	1	0	0	1	1
q_4	0	0	1	0	0
q_5	1	1	1	0	0

(a)

	S_1	S_2	S_3
q_1	10	20	0
q_2	5	0	10
q_3	0	35	5
q_4	0	10	0
q_5	0	15	0

(b)

Question 1. Calculer la matrice d'affinité.

Matière : Base de données Avancée

Chargée de cours : Farah Barika Ktata
Chargée de TD : Emna Ammar Elhadjamor

Correction TD4

Exercice 1 :

Question 1. Proposez une stratégie de fragmentation en utilisant des opérateurs d'algèbre relationnelle.

-La fragmentation **horizontale primaire** de la table **SERVICE** selon la valeur de l'attribut site:

```
SERVICE Rades =  $\Pi_{\text{num-serv, nom, responsable}}$  (  $\sigma_{\text{site} = \text{"Rades"}}$  (SERVICE ) )
```

```
SERVICE Skhira =  $\Pi_{\text{num-serv, nom, responsable}}$  (  $\sigma_{\text{site} = \text{"Skhira"}}$  (SERVICE ) )
```

```
SERVICE Bizerte =  $\Pi_{\text{num-serv, nom, responsable}}$  (  $\sigma_{\text{site} = \text{"Bizerte"}}$  (SERVICE ) )
```

- La fragmentation **horizontale dérivée** de la table **PROJET** en fonction du service correspondant:

```
PROJET Rades = PROJET  $\bowtie$  SERVICE Rades
```

```
PROJET Skhira = PROJET  $\bowtie$  SERVICE Skhira
```

```
PROJET Bizerte = PROJET  $\bowtie$  SERVICE Bizerte
```

- **EMPLOYE** sera, par contre, fragmentée en plusieurs étapes:

a) La fragmentation verticale, ici, aura pour but de séparer les informations personnelles et les informations professionnelles.

```
EMPLOYE perso =  $\Pi_{\text{mat-emp, nom, prénom, salaire, date\_naissance, adresse, téléphone}}$ (EMPLOYE)
```

```
EMPLOYE prof =  $\Pi_{\text{mat-emp, num-serv, nom, prénom, fonction}}$  (EMPLOYE)
```

Le fragment EMPLOYE personnelle sera obligatoirement stocké sur le site de Rades.

b) Le 2ème fragment EMPLOYE professionnelle suivra une fragmentation **horizontale dérivée** en fonction des règles suivantes: (1) les employés du service maintenance interviennent dans tous les sites, et donc leurs données doivent être répliquées sur les trois sites, et (2) les autres employés seront gérés par le site où ils sont affectés.

```
EMPLOYE maintenance = EMPLOYE professionnelle  $\bowtie$  (  $\sigma_{\text{nom} = \text{"maintenance"}}$  (SERVICE) )
```

```
EMPLOYE autre = EMPLOYE professionnelle  $\bowtie$  (  $\sigma_{\text{nom} \neq \text{"maintenance"}}$  (SERVICE) )
```


--- sur le site Rades, on aura, en plus du fragment EMPLOYE personnelle, le fragment suivant:

$EMPLOYE_{prof-Rades} = EMPLOYE_{maintenance} \cup (EMPLOYE_{autre} \bowtie SERVICE_{Rades})$

--- sur le site Skhira, on aura le fragment suivant:

$EMPLOYE_{prof-Skhira} = EMPLOYE_{maintenance} \cup (EMPLOYE_{autre} \bowtie SERVICE_{Skhira})$

--- enfin, sur le site de Bizerte, on aura le fragment suivant:

$EMPLOYE_{prof-Bizerte} = EMPLOYE_{maintenance} \cup (EMPLOYE_{autre} \bowtie SERVICE_{Bizerte})$

Remarque: Une autre solution consiste à stocker la totalité de la relation EMPLOYE sur le site de Rades. Pour les autres sites, on suivra la même fragmentation donnée ci-dessus.

Question 2. Pour chacune des décompositions proposées, donnez les requêtes de reconstruction.

Pour cette question, on aura besoin de 3 relations temporaires:

$T_{rades}(\text{site})$ qui contient un n-uplet ayant la valeur centre.

$T_{skhira}(\text{site})$ qui contient un n-uplet ayant la valeur sud.

$T_{bizerte}(\text{site})$ qui contient un n-uplet ayant la valeur nord.

$Service = \bigcup_i (Service_i) \times T_i$, avec $i = \{rades, skhira ; bizerte\}$ (\times : produit cartésien)

On a eu recours au produit cartésien pour pouvoir récupérer l'attribut « site ».

$PROJET = \bigcup_i PROJET_i$

$EMPLOYE_{prof} = \bigcup_i EMPLOYE_{prof-i}$, avec $i = \{rades, skhira ; bizerte\}$

Remarque : l'union élimine les employés de maintenance en double provenant de la réplication

$EMPLOYE = EMPLOYE_{prof} \bowtie EMPLOYE_{perso}$ (Jointure)

Question 3. Donnez la définition d'une fragmentation correcte. Montrez que votre décomposition est correcte.

Une fragmentation est correcte si i) tous les n-uplets sont stockés et ii) la reconstruction permet de retrouver exactement tous les n-uplets de la relation d'origine.

SERVICE : tous les n-uplets pourront être retrouvés (décomposition horizontale complète : l'attribut site prend 3 valeurs, la décomposition suit ces trois valeurs) Ensuite on procède avec l'union.

PROJET : idem

EMPLOYE : les deux premiers sont des unions, suite à des décompositions horizontales ou horizontale dérivée. Le dernier est une jointure sur la clé (mat-emp), et tous les attributs seront ainsi présents.

Exercice 2 :

Question 1. Donnez la définition des différents fragments en utilisant les opérateurs de l'algèbre relationnelle ainsi que le schéma de répartition des fragments.

❖ **EMPLOYE**

Fragmentation horizontale de EMPLOYE en fonction de la valeur de l'attribut affectation:

$$\text{EMPLOYE}_x = \Pi_{\text{Id_pers, nom, adresse, statut}} (\sigma_{\text{affectation} = "x"} (\text{EMPLOYE}))$$

$$\text{EMPLOYE}_y = \Pi_{\text{Id_pers, nom, adresse, statut}} (\sigma_{\text{affectation} = "y"} (\text{EMPLOYE}))$$

$$\text{EMPLOYE}_z = \Pi_{\text{Id_pers, nom, adresse, statut}} (\sigma_{\text{affectation} = "z"} (\text{EMPLOYE}))$$

Allocation : Un fragment EMPLOYE_i est sur le site i.

❖ **ETUDIANT**

a) Fragmentation verticale pour séparer les données concernant la gestion des emprunts de celles concernant la gestion des étudiants.

$$\text{ETUDIANT}_{\text{bibli}} = \Pi_{\text{Id_etu, nb_emprunts}} (\text{ETUDIANT})$$

Allocation : ETUDIANT_{bibli} est dupliqué sur les trois sites. Chaque mäj est répercutée sur les répliques, de façon à toujours avoir le nb_emprunts global.

$$\text{ETUDIANT}_{\text{univ}} = \Pi_{\text{Id_etu, nom, adresse, université, cursus}} (\text{ETUDIANT})$$

b) Fragmentation horizontale de ETUDIANT_{univ} en fonction de l'attribut université.

$$\text{ETUDIANT}_x = \Pi_{\text{Id_etu, nom, adresse, cursus}} (\sigma_{\text{université} = "x"} (\text{ETUDIANT}_{\text{univ}}))$$

$$\text{ETUDIANT}_y = \Pi_{\text{Id_etu, nom, adresse, cursus}} (\sigma_{\text{université} = "y"} (\text{ETUDIANT}_{\text{univ}}))$$

$$\text{ETUDIANT}_z = \Pi_{\text{Id_etu, nom, adresse, cursus}} (\sigma_{\text{université} = "z"} (\text{ETUDIANT}_{\text{univ}}))$$

Allocation : Un fragment ETUDIANT_i est sur le site i.

❖ **OUVRAGES est fragmenté horizontalement sur l'attribut site.**

$$\text{OUVRAGES}_x = \Pi_{\text{Id_ouv, titre, éditeur, année, domaine, stock}} (\sigma_{\text{site} = "x"} (\text{OUVRAGES}))$$

$$\text{OUVRAGES}_y = \Pi_{\text{Id_ouv, titre, éditeur, année, domaine, stock}} (\sigma_{\text{site} = "y"} (\text{OUVRAGES}))$$

$$\text{OUVRAGES}_z = \Pi_{\text{Id_ouv, titre, éditeur, année, domaine, stock}} (\sigma_{\text{site} = "z"} (\text{OUVRAGES}))$$

Allocation : Un fragment OUVRAGES_i est sur le site i.

❖ **AUTEURS est une fragmentation horizontale de OUVRAGES_i**

$$\text{AUTEURS}_x = \Pi_{\text{Id_ouv, nom_auteur}} (\text{AUTEURS} \bowtie \text{OUVRAGES}_x)$$

$$\text{AUTEURS}_y = \Pi_{\text{Id_ouv, nom_auteur}} (\text{AUTEURS} \bowtie \text{OUVRAGES}_y)$$

$$\text{AUTEURS}_z = \Pi_{\text{Id_ouv, nom_auteur}} (\text{AUTEURS} \bowtie \text{OUVRAGES}_z)$$

Allocation : Un fragment AUTEURS_i est sur le site i.

❖ **PRETS est une fragmentation horizontale des OUVRAGES_i**

$$\text{PRETS}_x = \Pi_{\text{Id_ouv, Id_etu, date_emprunt, date_retour}} (\text{PRETS} \bowtie \text{OUVRAGES}_x)$$

$$\text{PRETS}_y = \prod_{\text{Id_ouv}, \text{Id_etu}, \text{date_emprunt}, \text{date_retour}} (\text{PRETS} \bowtie \text{OUVRAGES } y)$$

$$\text{PRETS } z = \prod_{\text{Id_ouv}, \text{Id_etu}, \text{date_emprunt}, \text{date_retour}} (\text{PRETS} \bowtie \text{OUVRAGES } z)$$

Allocation : Un fragment PRETS_i est sur le site i.

Question 2. Montrez que la fragmentation que vous proposez pour la relation ETUDIANT est correcte.

Pour ETUDIANT :

La 1ère étape de la fragmentation consiste en une fragmentation verticale. La clé se retrouve dans les deux fragments, et l'union des attributs projetés englobe tous les attributs de la relation d'origine.

Cette 1ère étape est correcte.

La 2ème étape consiste en une fragmentation horizontale. Le critère de répartition est la valeur de l'attribut université. Dans la relation d'origine, cet attribut ne prend que les valeurs "x", "y" et "z". La distribution se fait en fonction de ces trois valeurs, tous les tuples de la relation sont donc affectés à l'une des trois relations. La répartition est donc correcte.

Question 3. Donner les opérations de reconstruction des relations globales.

Restructuration de la relation EMPLOYE

Soit T_i une relation ayant un seul attribut, l'attribut affectation. La valeur de cet attribut est i.

$$\text{EMPLOYE} = U (\text{EMPLOYEE}_i \times T_i)$$

Restructuration de la relation ETUDIANT

Se fait en plusieurs étapes :

Soit R_i une relation ayant un seul attribut, l'attribut université. La valeur de cet attribut est i.

$$\text{ETUDIANT}_{\text{univ}} = U (\text{ETUDIANT}_i \times R_i)$$

$$\text{ETUDIANT} = \text{ETUDIANT}_{\text{bibli}} \bowtie \text{ETUDIANT}_{\text{univ}}$$

Restructuration de la relation OUVRAGES

Soit S_i une relation ayant un seul attribut, l'attribut site. La valeur de cet attribut est i.

$$\text{OUVRAGES} = U (\text{OUVRAGES}_i \times S_i)$$

Restructuration de la relation AUTEURS

$$\text{AUTEURS} = U \text{ AUTEURS}_i$$

Restructuration de la relation PRETS

$$\text{PRETS} = U \text{ PRETS}_i$$

Exercice 3 :

Question 1. Calculer la matrice d'affinité.

On suppose ici que la clé de la table A est A1 et $\text{ref } i(q_k) = 1$; pour tout q_k et S_i.

Les accès suivants (3 sites):

$$\text{Acc1}(q_1) = 10 \quad \text{Acc2}(q_1) = 20 \quad \text{Acc3}(q_1) = 0$$

$$\text{Acc1}(q_2)=5 \text{ Acc2}(q_2)=0 \text{ Acc3}(q_2)=10$$

$$\text{Acc1}(q_3)=0 \text{ Acc2}(q_3)=35 \text{ Acc3}(q_3)=5$$

$$\text{Acc1}(q_4)=0 \text{ Acc2}(q_4)=10 \text{ Acc3}(q_4)=0$$

$$\text{Acc1}(q_5)=0 \text{ Acc2}(q_5)=15 \text{ Acc3}(q_5)=0$$

$$\text{Aff}(\mathbf{A1}, \mathbf{A2}) = \sum_{k=2} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = \text{Acc}_1(q_2) + \text{Acc}_2(q_2) + \text{Acc}_3(q_2) + \text{Acc}_1(q_5) + \text{Acc}_2(q_5) + \text{Acc}_3(q_5) = 5 + 0 + 10 + 0 + 15 + 0 = 15 + 15 = 30$$

$$\text{Aff}(\mathbf{A1}, \mathbf{A3}) = \sum_{k=2} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = 15 + 15 = 30$$

$$\text{Aff}(\mathbf{A1}, \mathbf{A4}) = \sum_{k=3} \sum_s \text{Acc}_s(q_k) = \text{Acc}_1(q_3) + \text{Acc}_2(q_3) + \text{Acc}_3(q_3) = 0 + 35 + 5 = 40$$

$$\text{Aff}(\mathbf{A1}, \mathbf{A5}) = \sum_{k=3, k=2} \sum_s \text{Acc}_s(q_k) = 40 + 15 = 55$$

$$\text{Aff}(\mathbf{A2}, \mathbf{A3}) = \sum_{k=1} \sum_{k=2} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = 30 + 15 + 15 = 60$$

$$\text{Aff}(\mathbf{A2}, \mathbf{A5}) = \sum_{k=1} \sum_{k=2} \sum_s \text{Acc}_s(q_k) = 30 + 15 = 45$$

$$\text{Aff}(\mathbf{A3}, \mathbf{A5}) = \sum_{k=1} \sum_{k=2} \sum_s \text{Acc}_s(q_k) = 15 + 30 = 45$$

$$\text{Aff}(\mathbf{A4}, \mathbf{A5}) = \sum_{k=3} \sum_s \text{Acc}_s(q_k) = \text{Acc}_1(q_3) + \text{Acc}_2(q_3) + \text{Acc}_3(q_3) = 0 + 35 + 5 = 40$$

$$\text{Aff}(\mathbf{A1}, \mathbf{A1}) = \sum_{k=2} \sum_{k=3} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = \text{Acc}_1(q_2) + \text{Acc}_2(q_2) + \text{Acc}_3(q_2) + \text{Acc}_1(q_3) + \text{Acc}_2(q_3) + \text{Acc}_3(q_3) + \text{Acc}_1(q_5) + \text{Acc}_2(q_5) + \text{Acc}_3(q_5) = 15 + 15 + 40 = 70$$

$$\text{Aff}(\mathbf{A2}, \mathbf{A2}) = \sum_{k=1} \sum_{k=2} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = 30 + 15 + 15 = 60$$

$$\text{Aff}(\mathbf{A3}, \mathbf{A3}) = \sum_{k=1} \sum_{k=2} \sum_{k=4} \sum_{k=5} \sum_s \text{Acc}_s(q_k) = 30 + 15 + 10 + 15 = 70$$

$$\text{Aff}(\mathbf{A4}, \mathbf{A4}) = \sum_{k=3} \sum_s \text{Acc}_s(q_k) = 35 + 5 = 40$$

$$\text{Aff}(\mathbf{A5}, \mathbf{A5}) = \sum_{k=1} \sum_{k=2} \sum_{k=3} \sum_s \text{Acc}_s(q_k) = 30 + 15 + 40 = 85$$

	A ₁	A ₂	A ₃	A ₄	A ₅
A ₁	70	30	30	40	55
A ₂	30	60	60	0	45
A ₃	30	60	70	0	45
A ₄	40	0	0	40	40
A ₅	55	45	45	40	85

Question 2. Utilisez bond Energy Algorithm (BEA) pour la génération de la matrice d'affinité groupée

Une fois qu'on a obtenu la matrice d'affinité correspondante, on peut passer à l'algorithme BEA. On commence par fixer les 2 premières colonnes de la matrice d'affinité groupée (AG) comme suit :

	A1	A2			
A1	70	30			
A2	30	60			
A3	30	60			
A4	40	0			
A5	55	45			

Puis on va étudier l'ordre de A3. 3 ordres sont possible : (a) à gauche de A1, (b) entre A1 et A2, et enfin à la droite de A2.

$$cont(A_i, A_k, A_j) = 2 \cdot \{bond(A_i, A_k) + bond(A_k, A_j) - bond(A_i, A_j)\}$$

$$bond(A_x, A_y) = \sum_{z=1}^n aff(A_z, A_x) \cdot aff(A_z, A_y)$$

a) $Cont(A0, A3, A1) = 2 \cdot (bond(A0, A3) + bond(A3, A1) - bond(A0, A1)) = 2 \cdot (0 + 8475 - 0) = 16950$

$bond(A0, A3) = aff(A1, A0) \cdot aff(A1, A3) + aff(A2, A0) \cdot aff(A2, A3) + aff(A3, A0) \cdot aff(A3, A3) + aff(A4, A0) \cdot aff(A4, A3) + aff(A5, A0) \cdot aff(A5, A3) = 0$

$bond(A3, A1) = aff(A1, A3) \cdot aff(A1, A1) + aff(A2, A3) \cdot aff(A2, A1) + aff(A3, A3) \cdot aff(A3, A1) + aff(A4, A3) \cdot aff(A4, A1) + aff(A5, A3) \cdot aff(A5, A1) = 30 \cdot 70 + 60 \cdot 30 + 70 \cdot 30 + 0 + 45 \cdot 55$

$bond(A0, A1) = 0$

b) $Cont(A1, A3, A2) = 2 \cdot (bond(A1, A3) + bond(A3, A2) - bond(A1, A2)) = 2 \cdot (8475 + 10725 - 8175) =$

22050 nous renvoie la plus grande valeur → [A1, A3, A2] sera le meilleur ordre à choisir.

$bond(A1, A3) = aff(A1, A1) \cdot aff(A1, A3) + aff(A2, A1) \cdot aff(A2, A3) + aff(A3, A1) \cdot aff(A3, A3) + aff(A4, A1) \cdot aff(A4, A3) + aff(A5, A1) \cdot aff(A5, A3) = 70 \cdot 30 + 30 \cdot 60 + 30 \cdot 70 + 40 \cdot 0 + 55 \cdot 45 = 8475$

$bond(A3, A2) = aff(A1, A3) \cdot aff(A1, A2) + aff(A2, A3) \cdot aff(A2, A2) + aff(A3, A3) \cdot aff(A3, A2) + aff(A4, A3) \cdot aff(A4, A2) + aff(A5, A3) \cdot aff(A5, A2) = 30 \cdot 30 + 60 \cdot 60 + 70 \cdot 60 + 0 + 45 \cdot 45 = 10725$

$bond(A1, A2) = aff(A1, A1) \cdot aff(A1, A2) + aff(A2, A1) \cdot aff(A2, A2) + aff(A3, A1) \cdot aff(A3, A2) + aff(A4, A1) \cdot aff(A4, A2) + aff(A5, A1) \cdot aff(A5, A2) = 70 \cdot 30 + 30 \cdot 60 + 30 \cdot 60 + 0 + 55 \cdot 45 = 8175$

c) $Cont(A2, A3, A4) = 2 \cdot (bond(A2, A3) + bond(A3, A4) - bond(A2, A4)) = 2 \cdot (10725 + 3000 - 3000) = 21450$

$bond(A2, A3) = 10725$

$bond(A3, A4) = 3000$

$bond(A2, A4) = 3000$

$Cont(A1, A2, A3) = 2 \cdot (bond(A1, A2) + bond(A2, A3) - bond(A1, A3)) = 2 \cdot (8175 + 10725 - 8475) = 20850$

Ainsi, le meilleur ordre est A1, A3, A2 comme le montre la matrice suivante.

	A1	A3	A2		
A1	70	30	30		
A3	30	70	60		
A2	30	60	60		
A4	40	0	0		
A5	55	45	45		

/* Les lignes doivent être organisées dans le même ordre que les colonnes

On refait pareil pour A4 :

Puis on va étudier l'ordre de A4 : 4 ordres sont possible : (a) à gauche de A1, (b) entre A1 et A3, (c) entre A3 et A2 et enfin à la droite de A2.

- a) $\text{Cont}(\text{A0}, \text{A4}, \text{A1}) = 2 * (\text{bond}(\text{A0}, \text{A4}) + \text{bond}(\text{A4}, \text{A1}) - \text{bond}(\text{A0}, \text{A1})) = 2 * 6600 = 8800$ nous renvoie la plus grande valeur → [A0, A4, A1] sera le meilleur ordre à choisir.

$\text{bond}(\text{A0}, \text{A4}) = 0$

$\text{bond}(\text{A4}, \text{A1}) = \text{aff}(\text{A1}, \text{A4}) * \text{aff}(\text{A1}, \text{A1}) + \text{aff}(\text{A2}, \text{A4}) * \text{aff}(\text{A2}, \text{A1}) + \text{aff}(\text{A3}, \text{A4}) * \text{aff}(\text{A3}, \text{A1}) + \text{aff}(\text{A4}, \text{A4}) * \text{aff}(\text{A4}, \text{A1}) + \text{aff}(\text{A5}, \text{A4}) * \text{aff}(\text{A5}, \text{A1}) = 40 * 70 + 0 + 0 + 40 * 40 + 40 * 55 = 6600$

$\text{bond}(\text{A0}, \text{A1}) = 0$

- b) $\text{Cont}(\text{A1}, \text{A4}, \text{A3}) = 2 * (\text{bond}(\text{A1}, \text{A4}) + \text{bond}(\text{A4}, \text{A3}) - \text{bond}(\text{A1}, \text{A3})) = 2 * (6600 + 3000 - 8475) = 2250$

$\text{bond}(\text{A1}, \text{A4}) = 6600$

$\text{bond}(\text{A4}, \text{A3}) = \text{aff}(\text{A1}, \text{A4}) * \text{aff}(\text{A1}, \text{A3}) + \text{aff}(\text{A2}, \text{A4}) * \text{aff}(\text{A2}, \text{A3}) + \text{aff}(\text{A3}, \text{A4}) * \text{aff}(\text{A3}, \text{A3}) + \text{aff}(\text{A4}, \text{A4}) * \text{aff}(\text{A4}, \text{A3}) + \text{aff}(\text{A5}, \text{A4}) * \text{aff}(\text{A5}, \text{A3}) = 40 * 30 + 0 + 0 + 0 + 40 * 45 = 3000$

$\text{bond}(\text{A1}, \text{A3}) = 8475$

- c) $\text{Cont}(\text{A3}, \text{A4}, \text{A2}) = 2 * (\text{bond}(\text{A3}, \text{A4}) + \text{bond}(\text{A4}, \text{A2}) - \text{bond}(\text{A3}, \text{A2})) = 2 * (3000 + 3000 - 10725) = -9450$

$\text{bond}(\text{A3}, \text{A4}) = 3000$

$\text{bond}(\text{A4}, \text{A2}) = \text{aff}(\text{A1}, \text{A4}) * \text{aff}(\text{A1}, \text{A2}) + \text{aff}(\text{A2}, \text{A4}) * \text{aff}(\text{A2}, \text{A2}) + \text{aff}(\text{A3}, \text{A4}) * \text{aff}(\text{A3}, \text{A2}) + \text{aff}(\text{A4}, \text{A4}) * \text{aff}(\text{A4}, \text{A2}) + \text{aff}(\text{A5}, \text{A4}) * \text{aff}(\text{A5}, \text{A2}) = 40 * 30 + 0 + 0 + 0 + 40 * 45 = 3000$

$\text{bond}(\text{A3}, \text{A2}) = 10725$

- d) $\text{Cont}(\text{A2}, \text{A4}, \text{A5}) = 2 * (\text{bond}(\text{A2}, \text{A4}) + \text{bond}(\text{A4}, \text{A5}) - \text{bond}(\text{A2}, \text{A5})) = 2 * (3000 + 5680 - 10875) = -4390$

$\text{bond}(\text{A2}, \text{A4}) = 3000$

$\text{bond}(\text{A4}, \text{A5}) = 5680$

$\text{bond}(\text{A2}, \text{A5}) = 10875$

- a) $\text{Cont}(\text{A3}, \text{A2}, \text{A4}) = 2 * (\text{bond}(\text{A3}, \text{A2}) + \text{bond}(\text{A2}, \text{A4}) - \text{bond}(\text{A3}, \text{A4}))$

	A4	A1	A3	A2	
A4	40	40	0	0	
A1	40	70	30	30	
A3	0	30	70	60	
A2	0	30	60	60	
A5	40	55	45	45	

➤ On refait pareil pour A5 :

Puis on va étudier l'ordre de A5 : 5 ordres sont possible : (a) à gauche de A4, (b) entre A4 et A1, (c) entre A1 et A3, (d) entre A3 et A2 enfin à la droite de A2.

- a) $\text{Cont}(\text{A0}, \text{A5}, \text{A4}) = 2 * (\text{bond}(\text{A0}, \text{A5}) + \text{bond}(\text{A5}, \text{A4}) - \text{bond}(\text{A0}, \text{A4})) = 2 * 5680 = 11360$

$\text{bond}(\text{A0}, \text{A4}) = 0$

$$\text{bond}(A5, A4) = \text{aff}(A1, A5) * \text{aff}(A1, A4) + \text{aff}(A2, A5) * \text{aff}(A2, A4) + \text{aff}(A3, A5) * \text{aff}(A3, A4) + \text{aff}(A4, A5) * \text{aff}(A4, A4) + \text{aff}(A5, A5) * \text{aff}(A5, A4) = 55 * 40 + 0 + 0 + 40 + 40 + 85 * 40 = 5680$$

$$\text{bond}(A0, A5) = 0$$

$$\text{b) Cont}(A4, A5, A1) = 2 * (\text{bond}(A4, A5) + \text{bond}(A5, A1) - \text{bond}(A4, A1)) = 2 * (5680 + 12825 - 6600) = 23810$$

$$\text{bond}(A4, A5) = \text{bond}(A5, A4) = 5680$$

$$\text{bond}(A5, A1) = \text{aff}(A1, A5) * \text{aff}(A1, A1) + \text{aff}(A2, A5) * \text{aff}(A2, A1) + \text{aff}(A3, A5) * \text{aff}(A3, A1) + \text{aff}(A4, A5) * \text{aff}(A4, A1) + \text{aff}(A5, A5) * \text{aff}(A5, A1) = 55 * 70 + 45 * 30 + 45 * 30 + 40 * 40 + 85 * 55 = 12825$$

$$\text{bond}(A1, A4) = 6600$$

$$\text{c) Cont}(A1, A5, A3) = 2 * (\text{bond}(A1, A5) + \text{bond}(A5, A3) - \text{bond}(A1, A3)) = 2 * (12825 + 11325 - 8475) = 31350$$

nous renvoie la plus grande valeur → [A1, A5, A3] sera le meilleur ordre à choisir.

$$\text{bond}(A5, A1) = \text{aff}(A1, A5) * \text{aff}(A1, A1) + \text{aff}(A2, A5) * \text{aff}(A2, A1) + \text{aff}(A3, A5) * \text{aff}(A3, A1) + \text{aff}(A4, A5) * \text{aff}(A4, A1) + \text{aff}(A5, A5) * \text{aff}(A5, A1) = 55 * 70 + 45 * 30 + 45 * 30 + 40 * 40 + 85 * 55 = 12825$$

$$\text{bond}(A5, A3) = \text{aff}(A1, A5) * \text{aff}(A1, A3) + \text{aff}(A2, A5) * \text{aff}(A2, A3) + \text{aff}(A3, A5) * \text{aff}(A3, A3) + \text{aff}(A4, A5) * \text{aff}(A4, A3) + \text{aff}(A5, A5) * \text{aff}(A5, A3) = 55 * 30 + 45 * 60 + 45 * 70 + 0 + 85 * 45 = 11325$$

$$\text{bond}(A1, A3) = 8475$$

$$\text{d) Cont}(A3, A5, A2) = 2 * (\text{bond}(A3, A5) + \text{bond}(A5, A2) - \text{bond}(A3, A2)) = 2 * (11325 + 10875 - 10725) = 22950$$

$$\text{bond}(A5, A3) = 11325$$

$$\text{bond}(A5, A2) = \text{aff}(A1, A5) * \text{aff}(A1, A2) + \text{aff}(A2, A5) * \text{aff}(A2, A2) + \text{aff}(A3, A5) * \text{aff}(A3, A2) + \text{aff}(A4, A5) * \text{aff}(A4, A2) + \text{aff}(A5, A5) * \text{aff}(A5, A2) = 55 * 30 + 45 * 60 + 45 * 60 + 0 + 85 * 45 = 10875$$

$$\text{bond}(A3, A2) = 10725$$

$$\text{e) Cont}(A3, A2, A5) = 2 * (\text{bond}(A3, A2) + \text{bond}(A2, A5) - \text{bond}(A3, A5)) = 2 * (10725 + 10875 - 11325) = 20550$$

$$\text{bond}(A3, A2) = 10725$$

$$\text{bond}(A5, A2) = 10875$$

$$\text{bond}(A3, A5) = 11325$$

	A4	A1	A5	A3	A2
A4	40	40	40	0	0
A1	40	70	55	30	30
A5	40	55	85	45	45
A3	0	30	45	70	60
A2	0	30	45	60	60