UFR 919 - Cycle L3 Informatique

LU3IN009 - Systèmes de Gestion de Bases de Données

Support de TD

2019

Équipe pédagogique

Bernd Amann

Mohamed-Amine Baazizi

Camelia Constantin

Stéphane Gançarski

Hubert Naacke

TD 1 : Rappels

Modèle Entités-Associations et Calcul Relationnel

masquer=1

1. Concepts du modèle Entités-Associations

On considère le schéma relationnel R suivant :

Produit (NumP, NumR*, prix, poids)

Rayon (NumR, designation)

Etagere (NumE, NumR*, hauteur)

Client (NumC, nom, datenais)

ProduitFavori (NumC*, NumP*)

1. On veut déduire le schéma Entités-Associations qui a servi à obtenir le schéma R.
   a) Enumérez les entités du schéma E-A en indiquant leurs attributs.
   b) Indiquez les entités faibles et les entités fortes auxquelles elles sont reliées.
   c) Précisez les attributs identifiants pour chacune des entités.
2. On considère l'instance suivante de la relation Rayon :

Rayon

NumR	designation
10	Légumes
20	Boissons

Indiquez si les instances suivantes de la relation **Etagere** sont valides ou non, et justifiez votre réponse.

Etagere

NumE	NumR	hauteur
1	10	20
2	10	25
1	30	10

Instance 1

Etagere

NumE	NumR	hauteur
4	20	20
3	10	25
4	20	10

Instance 2

Etagere

NumE	NumR	hauteur
4	20	20
3	10	25
3	20	10

Instance 3

3. Un client peut-il avoir plusieurs produits favoris ?

2. Conception d'un schéma E-A : Fédération de cyclisme

Base de données de production

La fédération internationale de cyclisme désire mettre au point une base de données. Celle-ci comporte des informations sur les différents coureurs, les équipes, les résultats obtenus aux différentes courses organisées ainsi que, pour des raisons d'actualité, sur le suivi médical des coureurs.

Les coureurs sont identifiés par leur nom et leur prénom, on connaît leur taille, leur date de naissance et l'équipe à laquelle ils appartiennent. Une équipe est identifiée par son nom, elle possède un budget, un directeur sportif dont on connaît le nom, le prénom et la date de naissance. Elle est financée par des sponsors qui peuvent varier selon les années et dont on connaît le nom, l'adresse et le domaine d'activité.

Une course correspond à un nom de course (ex. « Tour de France »), on en connaît la distance totale à parcourir. Elle peut comporter une ou plusieurs étapes, dont on connaît le numéro d'ordre (ex. « 3è étape »), la date, le type (ex. « Contre la montre individuel »), la ville de départ et celle d'arrivée. Pour chaque coureur ayant participé à une étape d'une course, on connaît le classement qu'il a obtenu lors de cette étape. Pour chaque course, on connaît le vainqueur final et l'équipe à laquelle il appartient.

Pour chaque course, les équipes emploient des soigneurs, dont on connaît le nom, le prénom, la date de naissance et la nationalité. On note aussi, à chaque étape, quelle dose de quel(s) produit(s) a administré un soigneur à un coureur. Un produit est identifié par un numéro de produit, a un nom, une indication (ex. « douleur musculaire »), une contre-indication (ex. « ne pas administrer en dessous de 20 ans ») et une posologie (ex. « 1 comprimé par jour »).

Dans cette base de donnée de production, seules les informations courantes (concernant l'édition en cours) de la course, des coureurs, des équipes, etc. sont stockées.

***Exercice** :*

a) Faire le schéma entité-association correspondant aux besoins de la fédération internationale de cyclisme. Ne pas oublier les cardinalités et les identificateurs.

c) En déduire le schéma relationnel de la base de données correspondante, sans oublier de préciser les clés de chaque relation.

Base de données temporelle

Chaque année, les informations évoluent. Les résultats bien sur, mais aussi la composition des équipes, le directeur technique, le budget des équipes, les sponsors, etc. changent chaque année.

Exercice : Comment modifier la base de donnée précédente pour stocker l'historique des informations selon les différentes éditions ?

3. Calcul Relationnel

Rappel :

Dans le *calcul relationnel à variable n-uplet* (ou *calcul n-uplet*), une requête est représentée par une expression de la forme $\{ t.A_0, t.A_1, \ldots \mid F(t) \}$, qui désigne une projection sur les attributs $A_0, A_1, \ldots$ de l'ensemble des n-uplets **t** satisfaisant le prédicat $F(t)$ de la logique du premier ordre. Dans cette expression, la variable **t** représente un n-uplet d'une seule relation ou une concaténation de n-uplets appartenant à des relations différentes (variables non quantifiées ou *libres* dans F).

Base de données « Tennis »

On considère la base TENNIS de schéma :

JOUEUR (NUJOUEUR, NOM, PRENOM, ANNAISS, NATIONALITE)

RENCONTRE (NUGAGNANT, NUPERDANT, LIEUTOURNOI, ANNEE, SCORE)

GAIN (NUJOUEUR, LIEUTOURNOI, ANNEE, PRIME, SPONSOR).

1. Exprimer dans le formalisme du calcul relationnel à variables n-uplet les requêtes ci-dessous:

   a) Numéro et tournoi d'engagement (défini par le lieu et l'année) des joueurs sponsorisés par Peugeot entre 1990 et 1994 ;

   b) Nom et année de naissance des joueurs ayant participé à Roland Garros en 1994 ;

   c) Nom et nationalité des joueurs ayant participé à la fois au tournoi de Roland Garros et à celui de Wimbledon, en 1992 ;

   d) Nom et nationalité des joueurs ayant été sponsorisés par Peugeot et ayant gagné à Roland Garros au moins un match (avec un sponsor quelconque);

   d-bis) Nom et nationalité des joueurs qui n'ont jamais perdu une rencontre;

   d-ter) Nom et nationalité des joueurs qui n'ont jamais perdu une rencontre à Roland Garros;

   d-quart) Nations qui n'ont jamais participé à Roland Garros;

2. Traduisez les requêtes précédentes en SQL.

TD 2 : ALGÈBRE RELATIONNELLE

masquer=1

1. RAPPELS

Les requêtes sont traduites, dans le modèle de l'algèbre relationnelle, par la combinaison d'opérations portant sur les relations de la base. La représentation de chaque opérateur est donnée ci-dessous.

1. Opérateurs de base :

Projection	$\Pi_{\text{liste-d'attributs}}(R)$
Sélection	$\sigma_{\text{condition}}(R)$
Renommage	$\rho_{\text{nom-attribut} \rightarrow \text{ nouveau\_nom}}(R)$
Union	$R \cup S$
Différence	$R - S$
Produit cartésien	$R \times S$

2. Opérateurs dérivés :

Jointure	$R \infty S$
Division	$R \div S$
Intersection	$R \cap S$

2. BASE DE DONNÉES « TENNIS »

Schéma

Soit la base de schéma (voir TD 1) :

JOUEUR (NUJOUEUR, NOM, PRENOM, ANNAISS, NATIONALITE)

RENCONTRE (NUGAGNANT, NUPERDANT, LIEUTOURNOI, ANNEE, SCORE)

GAIN (NUJOUEUR, LIEUTOURNOI, ANNEE, PRIME, SPONSOR)

Les attributs NUGAGNANT, NUPERDANT et NUJOUEUR y sont définis sur le même domaine. Les clés des relations sont soulignées. On suppose que tous les joueurs, engagés dans un tournoi, touchent une prime.

Requêtes

Exprimer (quand c'est possible) les requêtes suivantes, à l'aide de l'algèbre relationnelle (arbres algébriques et/ou expressions algébriques) :

a) Numéro et tournoi d'engagement (défini par le lieu et l'année) des joueurs sponsorisés par Peugeot entre 1990 et 1994.

b) Nom et année de naissance des joueurs ayant participé à Roland Garros en 1994.

c) Nom et nationalité des joueurs ayant participé à la fois au tournoi de Roland Garros et à celui de Wimbledon, en 1992.

d) Nom et nationalité des joueurs ayant été sponsorisés par Peugeot et ayant gagné à Roland Garros au moins un match (avec un sponsor quelconque).

e) Nom des joueurs ayant toutes leurs primes à Roland Garros supérieures à 1MF.

f) Numéros des joueurs qui ont toujours gagné à Roland Garros.

g) Liste des vainqueurs de tournoi, mentionnant le nom du joueur avec le lieu et l'année du tournoi qu'il a gagné.

h) Nom des joueurs ayant participé à tous les tournois disputés en 1994.

i) Nombre de joueurs ayant participé au tournoi de Wimbledon en 1993.

j) Numéros des joueurs ayant eu au moins deux sponsors.

k) Numéros des joueurs ayant eu exactement deux sponsors.

Requêtes supplémentaires :

1. Quel joueur n'a jamais été sponsorisé par Peugeot ?
2. Quels joueurs ont perdu au premier tour à Roland Garros en 2013 ?
3. Quel joueur a gagné la plus grande prime ?
4. Quel joueur a gagné une plus grande prime que tous les joueurs de sa nationalité
5. ...

TD 3 : Optimisation

masquer=1

1. Taille et cardinalité des opérations algébriques

Soit deux relations R(a1, a2, a3) et S(b1, b2, **a1**) contenant Tr et Ts tuples. L'attribut R.a1 est la clé primaire, l'attribut S.a1 est une clé étrangère vers R. La taille (en octets) des attributs est ta1, ta2, ta3, tb1, tb2. Le nombre de valeurs distinctes des attributs est da1, da2, da3, db1, db2, respectivement.

1.1. Pour les opérations suivantes, donner la cardinalité (nombre de tuples) et la taille (en octets) du résultat :

a) Op1: Projection de R sur a1 et a3 ?

b) Op2: Sélection de R avec le prédicat a2 = 1

c) Op3 : la jointure naturelle de R et S sur l'attribut a1

d) Op4 : le produit cartésien de R et S

1.2. Quelles opérations produisent un résultat de taille supérieure à la taille de leurs opérandes?

2. Arbres algébriques et réécriture

Soit la base de données Vacances qui décrit des stations de vacances, leurs hôtels, les activités sportives ou culturelles que l'on peut y pratiquer, les clients qui ont fait des réservations dans les hôtels et ces réservations.

Station (NumS, Nom, Altitude, Gare, Région)

Hôtel (NumH, NumS, Nom, Adresse, catégorie, nb_Chambre)

Activité (NumS, Type, Description, Adresse_Club)

Client (NumC, Nom, Prénom, Adresse, Tel)

Réservation (NumC, NumH, Date, Durée)

Les caractéristiques physiques des relations stockées dans le SGBD sont indiquées ci-dessous:

Caractéristiques des relations :

Relation	Nbre de tuples	Taille d'un tuple (en octets)
Station	1 000	66
Hôtel	10 000	122
Activité	10 000	129
Client	25 000	133
Réservation	20 000	17

Caractéristiques des attributs :

Relations	Attribut	Nbre de valeurs dist.	Domaine	Longueur(en octet)
Hotel et Réservation	NumH	10 000	-	5
Client et Réservation	NumC	25 000	-	5
Activité	Type	40	-	15
Station, Hôtel et Activité	NumS	1 000	-	4
Client	Nom	25 000	-	40
Station	Altitude	2 000	[0, 2000]	2

2.1. Arbre syntaxique

2.1.1. Donner l'arbre syntaxique des requêtes ci-dessous. Les opérateurs disponibles sont la sélection, la projection, la jointure, l'union et la différence. Les arbres sont construits en respectant l'ordre des prédicats de la clause **WHERE.**

R1 :
```
SELECT  c.Nom
FROM    Client c, Réservation r
WHERE   r.NumH = 536 AND r.NumC = c.NumC
```

R2 :
```
SELECT  c.Nom
FROM    Client c, Réservation r
WHERE   r.NumC = c.NumC AND r.NumH = 536
```

R3:
```
SELECT  r.NumC
FROM    Réservation r, Hôtel h, Activité a, Station s
WHERE   r.NumH = h.NumH AND h.NumS = a.NumS
        AND a.Type = 'tennis' AND h.NumS = s.NumS AND s.Altitude > 1400
```

R4 :
```
SELECT  s.Nom, s.Région
FROM    Station s, Activité a
WHERE   (s.NumS = a.NumS AND a.Type = 'tennis')
        OR (s.NumS = a.NumS AND s.Altitude >1000)
```

2.2. Comparaison de différentes formulations d'une même requête

2.2.1. Pour les requêtes R1 et R2 ci-dessus, calculer le volume (en octets) de l'information manipulée pendant leur exécution, si celle-ci suit l'arbre syntaxique.

2.2.2. Les deux requêtes étant équivalentes (résultat identique), que peut-on en conclure ?

2.3. Restructuration algébrique

Pour réduire la taille des relations manipulées dans un arbre d'opérateur, une règle de transformation est utilisée : traiter les sélections et les projections en priorité avant les jointures.

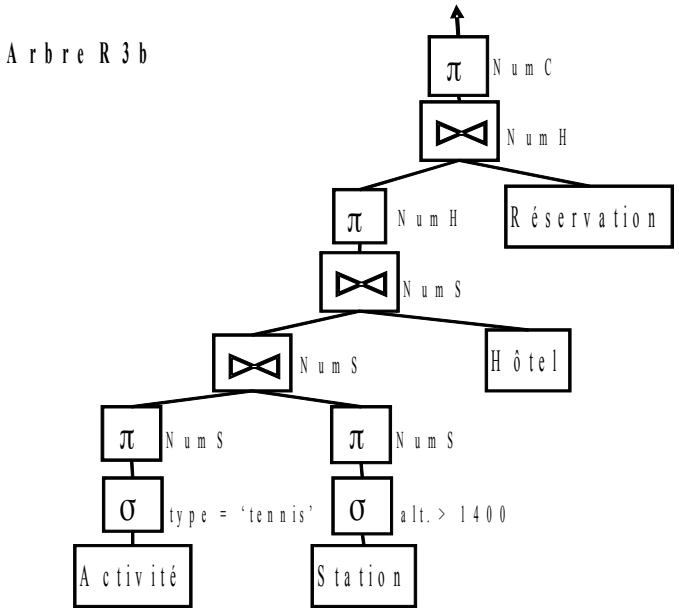
2.3.1. Quelle sont les transformations algébriques à effectuer pour appliquer cette règle ?

2.3.2. Appliquer cette règle aux requêtes R2 et R3. Représenter les arbres *R2a* et *R3a* après restructuration.

2.3.3. Pour la requête R3 :

- Quel est le facteur de sélectivité de la sélection dont le prédicat est : *type='tennis'* ?
- Quel est le facteur de sélectivité de la sélection dont le prédicat est : *altitude>1400* ?

2.3.4. Donner la taille et la cardinalité de chaque opérateur de l'arbre R3 initial, de l'arbre *R3a* obtenu à la question précédente et de l'arbre *R3b* suivant :



2.3.5. (Question facultative) : Donner le nombre d'arbres équivalents pour la requête suivante :

R5: **SELECT** *
FROM Hôtel H, Activité A, Station S
WHERE H.NumS = S.NumS **AND** A.NumS = S.NumS

TD 4 et 5 - Optimisation de requêtes (suite)

Plan d'exécution d'une requête

Préparation des exercices. Explication du plan d'une requête avec le mode *explain*

L'objectif de ce TD est de savoir traduire une requête SQL en un **plan** d'exécution (`explain plan`). Il faut savoir lire un plan d'exécution produit par une requête SQL et comprendre comment l'exécution se déroule. Le plan d'une requête SQL est construit en ajoutant les mots `explain plan for` devant la requête. Il peut ensuite être affiché avec plus ou moins de détails (*cf.* le TME 4).

Représentation d'un plan

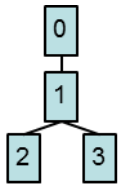
Un plan est représenté par un arbre dont les *n* nœuds sont numérotés de *0* (pour la racine) à *n-1*. La numérotation se fait en largeur, de gauche à droite. L'arbre est binaire et chaque nœud a de 0 (feuille) à 2 fils. Dans la représentation textuelle du plan, chaque nœud correspond à une ligne et la hiérarchie est signalée par l'indentation des lignes (un fils est décalé d'un espace à droite par rapport à son parent).

Par exemple, la requête suivante affiche le nom des joueurs de tennis français avec le lieu du tournoi :

```
SELECT j.nom, g.lieutournoi
FROM Joueur j, Gain g
WHERE j.nujoueur = g.nujoueur AND j.nationalite = 'France' ;
```

et son plan est :

Id	*Operation*	*Name*
0	SELECT STATEMENT	
1	JOIN	
2	TABLE ACCESS FULL	JOUEUR
3	TABLE ACCESS FULL	GAIN



Un nœud a un identifiant (*Id*), un nom (*Operation*) et d'autres informations qu'on expliquera plus tard. Le nom correspond à un opérateur « physique » du SGBD. Les fils d'un nœud (opérateur) sont ses opérandes (les paramètres de l'algorithme qui réalise l'opérateur). Un nœud peut évaluer des prédicats (*Predicate Information*) et des projections (*Projection Information*).

Un exemple de *Predicate Information* associée au nœud n°2 est :

IdOp	*Predicate*
2	filter(NATIONALITE='France')

Un exemple de *Projection Information* associée au nœud n°1 est :

IdOp	*Column*
1	NOM, LIEUTOURNOI

Remarques : plusieurs nœuds (opérateur physiques) peuvent être nécessaires pour évaluer une opération algébrique telle qu'une jointure ou une sélection, notamment lorsque des index sont utilisés. Inversement, un nœud peut aussi correspondre à une ou plusieurs opérations algébriques, en particulier lorsqu'une projection ou une sélection sont ajoutés à une jointure.

Exécution d'un plan :

Pour expliquer l'exécution d'un plan, on effectue une analyse « bottom-up » en partant des feuilles et en remontant vers la racine. L'analyse démarre avec la feuille située la plus à **gauche** (i.e. celle ayant le plus petit numéro parmi les feuilles). L'étape suivante consiste à identifier l'opération qui correspond au nœud parent. S'il s'agit d'une opération binaire (par exemple, une jointure), on analyse récursivement le 2^{e} sous-arbre (droit) du parent en partant des feuilles. On « remonte » ainsi l'arbre jusqu'à sa racine.

Travail demandé

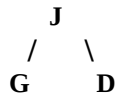
L'explication doit apporter des éléments de réponses aux questions suivantes :

1) Quel **algorithme** est utilisé pour évaluer chaque morceau de la requête ?
2) Dans quel **ordre** les opérations sont-elles évaluées ?
3) Quelle est la **cardinalité** estimée d'une opération (nbre de nuplets produits) ?
4) Est-ce que certaines opérations sont évaluées **plusieurs** fois (imbrication) ?

5) Combien d'opérations unitaires de **lecture** sont effectuées ?

Exemple : exécution d'une jointure

Concernant la question 4) dans le cas d'une jointure représentée par un nœud ayant 2 fils :



L'arbre D est évalué soit une seule fois, soit plusieurs fois selon la nature de l'algorithme utilisé pour la jointure. On détaillera cela plus tard, dans les requêtes de jointure.

Exercice 1. Plan d'une requête

On rappelle le schéma de la base Tennis :

Joueur (NuJoueur, Nom, Prenom, AnNaiss, Nationalite)

Gain (NuJoueur, LieuTournoi, Annee, Prime, Sponsor)

La relation Gain indique la participation d'un joueur à un tournoi identifié par un lieu et une année.

Le SGBD connait les statistiques suivantes (la commande permettant de calculer les statistiques sera vue en TME) :

AnNaiss	Nb joueurs
1952	1
1957	1
1959	1
1963	1
1964	1
1965	2
1966	1
1969	1
1970	2
1972	2
1975	1

Nationalite	Nb joueurs
Allemagne	1
Espagne	1
Etats-Unis	4
France	5
Suede	3

LieuTournoi	Nb gains
Flushing Meadow	6
Roland Garros	24
Wimbledon	22

Annee	Nb gains
1989	10
1990	6
1991	2
1992	18
1993	8
1994	8

Question 1. Expliquer le traitement des requêtes suivantes avec leur plan.

a) Afficher les joueurs.

```
explain plan for select * from Joueur;
```

Plan :

Id	Operation	Name	Rows
0	SELECT STATEMENT		14
1	TABLE ACCESS FULL	JOUEUR	14

Réponse : Le traitement se fait en 2 étapes :

Etape A): opération id= 1 (TABLE ACCESS FULL JOUEUR) parcours séquentiel de la table Joueur.

Cela correspond au SQL : « `Select * From Joueur ».` La cardinalité est: 14 = card(Joueur)

Etape B) : opération id=0 : (SELECT STATEMENT) : afficher le résultat.

Réponse présentée sous la forme d'un tableau à compléter:

Etape	*Id Operation*	*Description*	*Card*
A			
B			

b) Requête avec sélection : Les joueurs nés après 1960

```
explain plan for select * from Joueur j where j.annaiss > 1960;
```

Plan :

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		11
1	TABLE ACCESS FULL	JOUEUR	11

Predicate Information:

Id	*Predicate*
1	filter("J"."ANNAISS">1960)

Réponse présentée sous la forme d'un tableau :

Etape	*Id Operation*	*Description*	*Card*
A			
B			

c) Sélection avec une égalité : Les joueurs de nationalité 'France'

```
explain plan for    select * from Joueur j where j.nationalite='France';
```

Plan :

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		5
1	TABLE ACCESS FULL	JOUEUR	5

Predicate Information:

Id	*Predicate*
1	filter(j.nationalite='France')

Expliquer la cardinalité de la sélection estimée à 5.

d) **Sélection** avec une condition composée :

```
explain plan for     select * from Joueur j
                     where j.annaiss >1960 and j.nationalite='France';
```

Plan:

Id		*Operation*	*Name*	*Rows*
0		SELECT STATEMENT		4
1		TABLE ACCESS FULL	JOUEUR	4

Predicate Information:

IdOp	*Predicate*
1	filter(J.NATIONALITE='France' AND J.ANNAISS>1960

Expliquer la cardinalité de la sélection estimée à 4.

e) Le nom et le prénom des joueurs

```
explain plan for  select nom, prenom from Joueur;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		14

1	TABLE ACCESS FULL	JOUEUR	14

Projection Information:

IdOp	*Column*
1	NOM[VARCHAR2,12], PRENOM[VARCHAR2,14]

Réponse présentée sous la forme d'un tableau :

Etape	*Id Operation*	*Description*	*Card*
A			
B			

f) Requête avec **projection sans double** : les nationalités des joueurs.

```
explain plan for   select distinct nationalite
                   from Joueur;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		5
1	HASH UNIQUE		5
2	TABLE ACCESS FULL	JOUEUR	14

Projection Information:

IdOp	*Column*
1	Nationalite
2	Nationalite

Réponse présentée sous la forme d'un tableau :

Etape	*Id Operation*	*Description*	*Card*
A	2		
B	1		
C	0		

g) **Tri** : Les nom et prénom des joueurs français. Afficher le résultat trié par année croissante et par nom.

```
explain plan for     select annaiss, nom, prenom
                     from Joueur
                     where nationalite='France'
                     order by annaiss;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		5
1	SORT ORDER BY		5
2	TABLE ACCESS FULL	JOUEUR	5

Predicate Information:

IdOp	*Predicate*
2	filter(NATIONALITE='France')

Projection Information:

IdOp	*Column*
1	#keys=1 ANNAISS, PRENOM, NOM
2	NOM, PRENOM, ANNAISS

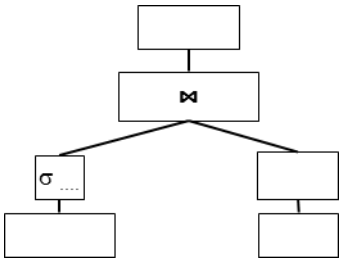
Réponse présentée sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A			
B			
C			

h) **Jointure**. Les joueurs Suédois ayant déjà participé à Rolland Garros. Afficher le nom du joueur et l'année du tournoi.

```
explain plan for  select j.nom, g.annee
                  from Joueur j, Gain g
                  where j.nujoueur = g.nujoueur
                  and j.nationalite = 'Suede'
                  and g.lieutournoi = 'Roland Garros';
```

Compléter l'arbre algébrique de cette requête :



Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		5
1	HASH JOIN		5
2	TABLE ACCESS FULL	JOUEUR	3
3	TABLE ACCESS FULL	GAIN	24

Predicate Information:

IdOp	*Predicate*
1	access(J.NUJOUEUR = G.NUJOUEUR)
2	filter(J.NATIONALITE ='Suede')
3	filter(G.LIEUTOURNOI ='Roland Garros')

Projection Information:

IdOp	*Column*
1	J.NOM, G.ANNEE
2	J.NUJOUEUR, J.NOM
3	G.NUJOUEUR, G.ANNEE

<u>Rappel de l'exécution d'une jointure HASH JOIN :</u>

Expliquer le plan sachant qu'un nœud nommé *« Hash Join »* est exécuté en 2 étapes successives.

```
      HASH JOIN
      /       \
     G         D
```

- On évalue d'abord **G entièrement** (pour créer une table de hachage utilisée à l'étape suivante).
- Puis **on itère sur D** : pour chaque nuplet *x* de D, on recherche dans la table de hachage les nuplets qui correspondent avec *x*.

En conséquence, les arbres G et D sont exécutés **une seule fois**.

Présenter la réponse sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A			

B			
C			
D			

i) Nom et nationalité des joueurs ayant participé à la fois au tournoi de Roland Garros et à celui de Wimbledon, en 1992 ;

```
explain plan for  select j.Nom, j.Nationalite
                  from Joueur j, Gain g1, Gain g2
                  where j.NuJoueur = g1.NuJoueur AND g1.LieuTournoi = 'Roland
      Garros' AND G1.Annee = 1992
                  AND   (j.NuJoueur = g2.NuJoueur AND g2.LieuTournoi =
      'Wimbledon' AND g2.Annee = 1992);
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		4
1	HASH JOIN		4
2	HASH JOIN		7
3	TABLE ACCESS FULL	GAIN	7
4	TABLE ACCESS FULL	JOUEUR	14
5	TABLE ACCESS FULL	GAIN	8

Predicate Information:

IdOp	*Predicate*
1	Access(J.NUJOUEUR = G1.NUJOUEUR)
2	Access(J.NUJOUEUR = G2.NUJOUEUR)
3	filter(G2.ANNEE=1992 AND G2.LIEUTOURNOI='Wimbledon')
5	filter(G1.ANNEE=1992 AND G1.LIEUTOURNOI='Roland Garros')

Projection Information:

IdOp	*Column*
1	J.NOM, J.NATIONALITE
2	J.NUJOUEUR, J.NOM, J.NATIONALITE
3	g2.NUJOUEUR
4	J.NUJOUEUR, J.NOM, J.NATIONALITE
5	g1.NUJOUEUR

Réponse présentée sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A			
B			
C			
D			
E			
F			

Exercice 2 : Plan d'une requête utilisant un index

On a les relations :

Annuaire (nom, prenom, age, cp, tel) contenant **2000** personnes.

On connait les statistiques suivantes sur les personnes :

Attribut	Nb valeurs distinctes	Domaine	Rmq
nom	2 000		Le nom est unique
cp	1000	[1 000, 100 900]	(1 000, 1 100, 1 200, ..., 100 800, 100 900) En moyenne, il y a 2,3 personnes par code postal
age	100	[1, 100]	On connait aussi le nbre de personnes pour chaque âge

et **Ville** (cp, ville, population) contenant **1000** villes.

avec la population dans [2000, 1 000 000]

Tous les cp de l'annuaire existent dans la relation Ville.

Les index définis sur la table Annuaire sont :

Create index **IndexAge** on Annuaire(age);

Create index **IndexCp** on Annuaire(cp);

Les index permettent un accès ciblé aux données et exprimé dans le plan par une suite de deux opérations :

- `INDEX RANGE SCAN INDEXAGE` avec le prédicat d'accès noté `Access(age=18)`
  Traverser l'index âge pour atteindre la feuille contenant 18. Lire les rowid des personnes ayant 18 ans. Cette opération ne lit **aucun** nuplet de la table Annuaire.
- `TABLE ACCESS BY INDEX ROWID ANNUAIRE.` Pour chaque rowid lire le nuplet de la personne correspondante dans la table Annuaire

Expliquer l'exécution des requêtes suivantes :

a) Le nom et prénom des personnes de l'annuaire ayant 18 ans

```
explain plan for     select a.nom, a.prenom
                     from Annuaire a
                     where a.age=18;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		20
1	TABLE ACCESS BY INDEX ROWID	Annuaire	20
2	INDEX RANGE SCAN	IndexAge	20

Predicate Information:

IdOp	*Predicate*
2	Access(a.age=18)

Projection Information:

IdOp	*Column*
1	a.nom, a.prenom
2	a.rowid

Réponse présentée sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A			
B			
C			

Préciser le nombre de lectures de pages de nuplets d'Annuaire.

b) Le nom et prénom des personnes âgées de 20 à 26 ans.

```
explain plan for        select a.nom, a.prenom
                        from Annuaire a
                        where a.age between 20 and 26;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		124
1	TABLE ACCESS BY INDEX ROWID	Annuaire	124
2	INDEX RANGE SCAN	IndexAge	124

Predicate Information:

IdOp	*Predicate*
2	Access(age>=20 AND age<=26)

c) Le nom et prénom des personnes de moins de 70 ans et dont le code postal de résidence est 93000 ou 75000.

```
explain plan for     select a.nom, a.prenom
                     from Annuaire a
                     where a.age < 70 and (a.cp = 93000 or a.cp = 75000)
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		3
1	TABLE ACCESS BY INDEX ROWID	Annuaire	3
2	INDEX RANGE SCAN	IndexCP	5

Predicate Information:

IdOp	*Predicate*
1	Filter(age < 70)
2	Access(cp=75000 or cp = 93000)

Projection Information:

IdOp	*Column*
1	a.nom, a.prenom
2	a.rowid

Réponse présentée sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A	2		
B	1		
C	0		

d) Le nom et prénom des personnes de 20 ans et dont le code postal est 13000.

```
explain plan for     select a.nom, a.prenom
                     from Annuaire a
                     where a.age = 20 and a.cp = 13000 and a.nom like 'T%';
```

Quelles seraient les étapes d'un plan qui exécute cette requête en utilisant les 2 index ?

e) Jointure avec Ville

```
explain plan for     select a.nom, a.prenom, v.ville
                     from Annuaire a, Ville v
                     where a.cp = v.cp
                     and a.age=18;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		20
1	HASH JOIN		20
2	TABLE ACCESS BY INDEX ROWID	Annuaire	20
3	INDEX RANGE SCAN	IndexAGE	20
4	TABLE ACCESS FULL	VILLE	1000

Predicate Information:

IdOp	*Predicate*
1	Access(a.cp = v.cp)
3	Access(a.age=18)

Projection Information:

IdOp	*Column*
1	a.nom, a.prenom, v.ville
2	a.nom, a.prenom, a.cp
3	a.rowid
4	v.ville, v.cp

Réponse présentée sous la forme d'un tableau :

Etape	*Id Op*	*Description*	*Card*
A			
B			
C			
D			

f) Jointure avec Ville

Rappel de l'exécution d'une jointure NESTED LOOP

Expliquer le plan sachant qu'un nœud de jointure *« Nested Loop »* est exécuté en **imbriquant D dans** l'itération de G.

- On itère sur G : : **pour chaque nuplet de G**, on évalue la jointure **par une itération sur D**.

En conséquence, G est évalué une seule fois mais D est évalué **plusieurs fois** (i.e. autant de fois qu'il y a de nuplets produits par G).

Décrire les plans suivants en rédigeant le pseudo-code correspondant au plan. Mettre en évidence les itérations *foreach* en détaillant sur quel ensemble se fait l'itération et le contenu d'une itération.

```
explain plan for     select a.nom, a.prenom, v.ville
                     from Annuaire a, Ville v
                     where a.cp = v.cp
                    and v.population >= 985000;
```

```
| Id      | Operation                          | Name                | Rows |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |  3647 |
|   1 |  NESTED LOOPS                      |                     |       |
|   2 |   NESTED LOOPS                     |                     |  3647 |
|*  3 |    TABLE ACCESS FULL               | VILLE               |    17 |
```

```
|*  4 |    INDEX RANGE SCAN               | INDEXCP         |   220 |
|   5 |   TABLE ACCESS BY INDEX ROWID     | BIGANNUAIRE     |   220 |
-------------------------------------------------------------------
```

Predicate Information:

IdOp	*Predicate*
3	`filter(V.POPULATION>=985000)`
4	`access(A.CP=V.CP)`

Projection Information:

IdOp	*Column*
1	`a.nom, a.prenom, v.ville`
2	`v.ville, a.rowid`
3	`v.ville, v.cp`
4	`a.rowid`
5	`a.nom, a.prenom`

Exercice 3: OPTIONNEL Coût en nombre de pages.

Exercice déplacé

Exercice 4: INTRO pour le TME 6 Jointures

Soit le schéma relationnel :

Joueur (licence: integer, cnum : integer, salaire: integer, sport: char(20))

Club (cnum: integer, nom: char(20), division: integer, ville : char(10))

Finance (cnum: integer, budget: real, dépense: real, recette: real)

On considère la requête :

```
SELECT C.nom, F.budget
FROM Joueur J, Club C, Finance F
WHERE J.cnum = C.cnum AND C.cnum = F.cnum
AND C.division = 1 AND J.salaire > 59000 AND J.sport = 'aviron'
```

1) Déterminer un arbre d'opérateurs de l'algèbre relationnel qui reflète l'ordre des opérations qu'un optimiseur de requête peut choisir. Combien y a-t-il de possibilités équivalentes pour ordonner les jointures ?

2) Pour réduire l'espace de recherche exploré pendant l'optimisation, on considère seulement les arbres de jointure qui n'ont pas de produit cartésien et qui sont linéaires à gauche. Donner la liste de tous les arbres de jointure construits. Expliquer comment vous obtenez cette liste.

Les informations suivantes sont extraites du catalogue du SGBD.
Les attributs Joueur.cnum, Joueur.salaire, Club.division, Club.cnum et Finance.cnum sont indexé par un arbre B+.
Le salaire d'un joueur est compris entre 10.000 et 60.000 EUR.
Les joueurs peuvent pratiquer 200 sports différents. Un club est en division 1 ou 2.
La BD contient au total 50000 joueurs et 5000 clubs. Il y a un nuplet d'information financière par club.

3) Pour chaque relation (Joueur, Club et Finance) estimer le nombre de nuplets qui sont sélectionnés après avoir traité les prédicats de sélection et avant de traiter les jointures.

4) D'après la réponse à la question précédente, quel est l'arbre de jointure de coût minimum que l'optimiseur construit ?

Exercice 5 : Jointure entre 3 relations (facultatif)

L'objectif de cet exercice est de comparer deux méthodes pour le choix du plan d'exécution d'une requête. La première méthode est la transformation de plan basée sur des heuristiques. La 2ème méthode est la génération exhaustive de l'espace de recherche associée au choix du plan candidat de moindre coût.

Soit le schéma **R1**(A,B, …), **R2**(A,B, …) et **R3**(A, B, …)

Soit la requête : select R1.A, R3.B

from R1, R2, R3

where R1.A=R2.A and R2.A=R3.A and R1.B=1 and R2.B=2

1) Donner l'arbre algébrique, nommé P1, correspondant en respectant l'ordre des prédicats donnés dans la clause *where*.
2) Donner un arbre équivalent, nommé P2, en appliquant les opérations les plus réductrices (sélection puis projection) d'abord.
3) Soit le modèle de coût simplifié suivant, où l'unité de coût est l'accès à un nuplet.

- Pour toute relation R, si card(R) est le nombre de tuples de R, le coût d'une *sélection* sur égalité est :

  coût ($\sigma_{a.=valeur}$ (R)) = card($\sigma_{a.=valeur}$ (R)) s'il y a un index sur l'attribut *a*

  = card(R) sinon.

- Le coût d'une lecture séquentielle est le coût d'une sélection sans index.
- Pour toutes relations R ayant card(R) nuplets et S ayant card(S) nuplets, le coût d'une *equi-jointure* est :

  coût (R $\bowtie_{a.}$ S) = cout(R) + card(R) s'il y a un index sur l'attribut *a* de *S*

  = cout(R) + card(R) * card(S) sinon.

- On suppose que R1, R2, R3 ont les caractéristiques suivantes :
  - il y a un **index** sur l'attribut B de R1, A est clé primaire de R1, R2 et R3 (il existe un index pour chaque clé primaire)
  - card(R1) = card(R2) = card(R3) = 1000, π_A(R1) = π_A(R2) = π_A(R3)
  - il y a 10 valeurs possibles pour B, uniformément réparties dans R1 et aussi dans R2 et dans R3.

Questions :

a) Quelle est la cardinalité du résultat de la requête ?

b) Pour simplifier, on ignore les projections (car leur coût est nul). Donner l'expression algébrique de P1' (resp P2') correspondant à P1 (resp. P2) sans aucune projection.

c) Donner le coût de P1' et P2'. Préciser votre réponse en détaillant le coût et la cardinalité des résultats intermédiaires.

d) Quel est l'arbre de coût minimal pour évaluer la requête ? Quel est son coût ?

Exercice (3) : OPTIONNEL Coût d'un plan en nombre de pages à lire

Notations et conventions

Dans cet exercice, on suppose que la distribution des attributs est uniforme. Les attributs sont tous indépendants les uns des autres.

On s'intéresse au coût des opérations en termes de quantité d'accès aux données. L'unité de coût est la lecture d'une page.

Une opération (sélection, projection, jointure) qui accède à une *table* stockée dans le SGBD a un coût.

Une opération de sélection (ou projection) qui accède au *résultat* d'une requête R ne rajoute pas de coût supplémentaire car on peut évaluer la sélection (ou la projection) sur chaque nuplet résultant de R, sans accéder à aucune autre donnée de la base.

Le coût de la jointure notée A $\bowtie_{A.a=B.a}$ B représente la quantité d'accès à la relation B nécessaire pour évaluer la jointure. Si B n'est pas une table de la base mais une requête, alors on stocke B comme une *table temporaire* dans le SGBD avant d'effectuer la jointure.
Soit les relations :

R (a, b) et **S** (c),

Le domaine des attributs est : a∈[1,1000], b∈[1,10], c∈[1,100].
Les cardinalités des relations sont : card(R) = 1000, card(S)= 100
La taille en nb de pages des relations est : P(R) = 500, P(S) = 10

Le modèle de coût est :
Le coût d'une équi-jointure entre les relations A et B, avec le prédicat A.a = B.a est (avec v étant une valeur quelconque):
Si B est une table :

coût(A $\bowtie_{A.a=B.a}$ B) = coût(A) + card(A) * card($\sigma_{a=v}$ B) si B.a est indexé

coût(A $\bowtie_{A.a=B.a}$ B) = coût(A) + P(A) * P(B) si B.a n'est pas indexé

TD6 Index – Arbre B+

Notations et conventions

Nombre de valeurs dans un nœuds. Pour un arbre B+ d'ordre d, le nombre de valeurs qu'un nœud peut contenir est :

dans l'intervalle [1, 2.d] pour la racine,

dans l'intervalle [d, 2.d] pour les nœuds intermédiaires et les feuilles.

Certains exercices n'indiquent pas l'ordre de l'arbre mais directement le nombre de valeurs dans un nœud.

Dessin d'un arbre. Pour repérer facilement les nœuds d'un arbre quand on le dessine, on peut attribuer un nom à chaque nœud : R pour la racine, Ni pour les nœuds intermédiaires et Fi pour les feuilles. On peut utiliser la syntaxe suivante pour représenter le contenu d'un nœud : N(v_1, v_2, …) où N est le nom du nœud et les v_i sont les valeurs.

Insertion d'une valeur en cas de débordement : quand la feuille *F* déborde, on garde les *d+1* plus petites valeurs dans *F*, les *d* autres valeurs vont dans une nouvelle feuille. Quand le nœud intermédiaire *N* déborde on garde les *d* plus petites valeurs dans *N*, les *d* plus grandes valeurs vont dans un nouveau nœud. La valeur restante est insérée dans le nœud père.

Suppression d'une valeur. Si le nœud ne contient que *d* valeurs avant la suppression, alors on considère d'abord la redistribution avec le nœud voisin (de même père) situé à gauche, puis avec celui situé à droite. Si aucune redistribution n'est possible, on considère la fusion avec le voisin (de même père) situé à gauche, puis avec celui situé à droite. Une suppression avec redistribution nécessite d'ajuster le contenu du nœud père. Une suppression avec fusion nécessite de supprimer une valeur dans le nœud père.

Décompte du nombre de nœuds lus et écrits. Une opération d'insertion ou de suppression peut nécessiter de lire, modifier ou créer des nœuds. On appelle *L* le nombre de nœuds lus pendant une opération, et respectivement *E* le nombre de nœuds écrits ou créés.

Exercice 1 : Insertions successives

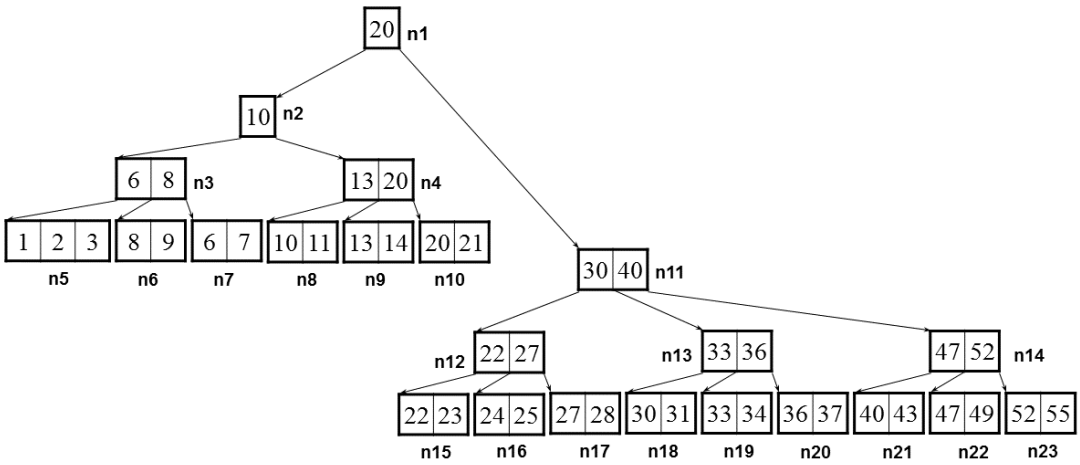
Dans cet exercice, les arbres sont d'ordre 1 (i.e., il y a 1 ou 2 valeurs par nœud).

1) Soit l'arbre A0 composé d'une racine N1(10), et de deux feuilles F1(5) et F2(20). A0 a 2 niveaux. On insère successivement dans A0 les nombres entiers 8 puis 30 puis 15. On obtient A1. Dessiner A1.

2) On insère dans A1 le nombre 10. On obtient A2. Dessiner A2.

Exercice 2 : Chercher l'erreur

On considère l'arbre B+ d'ordre 2 suivant :



1) Cet arbre est-il équilibré ? Pourquoi ?

2) Trouver les erreurs dans cet arbre. Indiquer quel nœud n_i est erroné et expliquer brièvement l'erreur. S'il est possible de corriger l'erreur sans restructurer l'arbre, mais en modifiant seulement des valeurs de clés, alors suggérer une correction.

Exercice 3 : Insertion, suppression, perte d'un niveau

On considère un arbre B+ d'ordre d=2. La racine de l'arbre **A1** contient la valeur 50. Un seul niveau intermédiaire contient (tous nœuds confondus) les valeurs 8, 18, 32, 40 73, 85. Les feuilles contiennent (toutes feuilles confondues) les valeurs 1, 2, 5, 6, 8, 10, 18, 27, 32, 39, 41, 45, 52, 58, 73, 80, 91, 99.

1) Dessinez l'arbre A1.
2) Dessiner l'arbre A2 après l'insertion de la valeur 9 dans A1. Combien valent L et E ?
3) Dessiner l'arbre A3 après l'insertion de la valeur 3 dans A1. Combien valent L et E ?
4) a) Dessiner l'arbre A4 après suppression de la valeur 8 dans A1. Si nécessaire, on envisage une redistribution à gauche. Combien valent L et E ?

   b) Même question mais en considérant uniquement la redistribution à droite si possible, sinon fusionner 2 feuilles.
5) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de l'insertion de la clé 46 suivie de la suppression de la clé 52
6) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de la suppression successive des clés 32, 39, 41, 45 et 73.

Exercice 4 : Effet d'une séquence insertion-suppression

On étudie l'effet de l'insertion d'une valeur *v* suivie de la suppression immédiate de *v*. Cela ne modifie pas l'ensemble des valeurs indexées. Mais est-ce que cela modifie la structure de l'arbre ? Pour répondre à cette question, on considère un arbre B+ d'ordre 2, nommé A1. La racine de A1 contient les clés 13, 17, 24, 30. Ses feuilles contiennent (toutes feuilles confondues) les valeurs 2, 3, 5, 7, 14, 16, 19, 20, 22, 24, 27, 29, 33, 34, 38, 39.

1) Dessiner A1.
2) Donner 4 valeurs de clé telles que leur insertion successive puis leur suppression dans l'ordre inverse résulte dans un état identique à l'état initial
3) Donner une valeur de clé dont l'insertion suivie de la suppression résulte dans un état différent de l'état initial.
4) Montrer l'état de l'arbre résultant, à partir de l'arbre A1, de l'insertion de la clé 30.

TD 7 : CONCURRENCE

masquer=1

1. ISOLATION ET SÉRIALISABILITÉ

1.1 Soit transfert la procédure de transfert(X,Y,Z) entre comptes bancaires suivante :

procedure *transfert* (CompteDébité, CompteCrédité, montant)

(1) variable := ***Lire****(CompteDébité);*

(2) ***Ecrire*** *(CompteDébité, variable – montant);*

(3) variable := ***Lire*** *(CompteCrédité);*

(4) ***Ecrire(CompteCrédité, variable + montant);***

On lance simultanément les programmes transfert(A, B, 150) et transfert(B, C, 70), ce qui génère les transaction T1 et T2. Chacun des processus (i.e., chacune des transactions de transfert) utilise une variable locale à son propre espace mémoire, soient v1 et v2. Soient A_0, B_0, C_0 les valeurs initiales des trois comptes concernés.

1.1.1 Quel est l'état final des comptes A, B et C dans chacun des cas ci-dessous, où ABi représente l'ième instruction du transfert de A vers B et BCj la jème instruction du transfert de B vers C?

cas (a) = exécution complète de *transfert*(B,C,70), puis exécution de *transfert*(A,B,150)

L2(B), E2(B), L2(C), E2(C),L1(A),E1(A), L1(B), E1(B)

cas (b) = exécution des instructions dans l'ordre

AB1 BC1 BC2 BC3 AB2 AB3 AB4 BC4, soit

L1(A), L2(B),E2(B), L2(C),E1(A),L1(B), E1(B),E2(C)

cas (c) = exécution des instructions dans l'ordre

AB1 AB2 AB3 BC1 BC2 BC3 AB4 BC4, soit

L1(A), E1(A), L1(B), L2(B), E2(B), L2(C),E1(B),E2(C)

1.1.2 Quelle invariant n'est pas respecté par l'une des exécutions précédentes ?

1.1.3 Dans chacun des deux derniers cas, peut-on obtenir une exécution séquentielle des deux programmes de transfert, par permutation d'opérations successives non conflictuelles ?

1.1.4 Donnez dans chacun des cas (a), (b) et (c) de l'exercice précédent le *graphe de précédence* du transfert de A vers B, et du transfert de B vers C.

2. VERROUILLAGE EN DEUX PHASES

2.1 Soit la procédure *transfert* de l'exercice précédent dans laquelle on a inséré des instructions de verrouillage et de déverrouillage selon le protocole de *verrouillage en deux phases strict*.

2.1.1. Écrire la série d'instructions obtenue, en supposant que deux comptes différents appartiennent à des granules différents.

2.1.2. Les exécutions (b) et (c), décrites dans l'exercice précédent, sont-elles alors possibles ? Si non, donner une exécution possible ayant le même ordonnancement initial jusqu'à la première instruction non réalisable.

2.2 Soit la procédure *somme* qui affiche la somme des soldes de deux comptes bancaires :

procédure *somme* (Compte1, Compte2);

*(1) var1 := **Lire** (Compte1);*

*(2) var2 := **Lire** (Compte2);*

(3) Imprimer (var1 + var2);

où la procédure *Lire* recopie le solde du compte dans une variable en mémoire.

2.2.1. Écrire en SQL la série d'instructions (1) à (3).

2.2.2. *transfert* étant la procédure de l'exercice 1.2, on considère une exécution concurrente de *transfert(A,B,150)* et *somme(A,B)*, au cours de laquelle somme commence son exécution après les deux premières instructions de transfert. Dans le cas d'un verrouillage en deux phases, la procédure *somme* peut-elle s'exécuter en entier avant la reprise de *transfert* ?

2.2.3. Dans le cas d'un protocole qui lèverait le verrou posé par une transaction sur un granule dès que cette transaction aurait fini d'accéder au granule, la procédure somme pourrait-elle alors s'exécuter en entier avant la reprise de transfert ? Quel serait, dans ce cas, le résultat de l'exécution des deux transactions ?

2.3 Soit une base composée de quatre granules A, B, C, D et une exécution de six transactions T1 à T6 avec les accès suivants sur les granules :

A: E2(A) E3(A) L5(A)

B: L2(B) L4(B) L1(B)

C: E5(C) L1(C) L3(C) E4(C)

D: L6(D) L2(D) E3(D)

Ei(X) et Li(X) représentent une opération d'écriture (respectivement de lecture) du granule X par la transaction i.

2.3.1. Donner le graphe de précédence de cette exécution.

2.3.2. On suppose que le contrôle de concurrence est effectué par du verrouillage en deux phases strict, et que les demandesd'accès se font dans l'ordre suivant :

E2(A) L2(B) L6(D) E5(C) E3(A) L5(A) L1(C) L2(D) L3(C) E4(C) E3(D) L4(B) L1(B).

- Indiquer pour chaque granule les verrous en cours, et les demandes en file d'attente.
- Donner le graphe d'attentes et conclure.

2.4 Soient cinq transactions T1, T2, T3, T4 et T5 et quatre granules X, Y, Z, T. On considère l'exécution suivante :

L4(Z) L5(X) L3(Y) L4(Y) E5(X) E3(Y) L4(T) L3(Y) L3(Z) L4(Z) E4(T) L5(T) E1(T) E1(X) L2(Y) L4(Z) L3(X) E2(T)

2.4.1. Construire le graphe de précédence de l'exécution. Est-elle sérialisable ? Si oui, donner la ou les exécutions en série équivalente(s).

2.4.2. Cette exécution peut-elle être obtenue par un mécanisme de verrouillage en deux phases ?

2.5 Soient cinq transactions T1, T2, T3, T4 et T5 et quatre granules X, Y, Z, T. On considère l'exécution suivante :

L2(T) L3(Y) L3(X) L3(T) E2(T) E2(X) E3(Y) C3 E5(X) C5 L2(Y) L2(Z) C2 E4(Z) E1(X) L1(Y) E1(Y) L4(T) L1(Z) C1 E4(T), C4

2.5.1. En supposant que chaque transaction effectue une demande de verrou sur un granule juste avant d'essayer d'y accéder et que toutes les transactions respectent le protocole de verrouillage en deux phases, insérer les commandes de verrouillage et déverrouillage dans chaque transaction.

2.5.2. On suppose que les demandes d'accès arrivent dans l'ordre de l'exécution (ce qui ne veut pas dire que ces demandes seront satisfaites dans le même ordre). Indiquer comment se passe l'exécution des cinq transactions.

3. ESTAMPILLAGE

Soient les granules a, b, c et les transactions T1, T2, T3. On considère les exécutions suivantes:

a) L1(a), L1(b), E1(a), L2(b), E3(b), C3, E1(a), C1, L2(b), C2

b) L1(a), E2(a), C2, E1(a), C1, L3(a), C3

c) E1(a), L2(c), L2(a), C2, E1(a), L3(a), L1(c), C1, E3(b), C3

d) E1(a), L1(a), L3(b), E3(a), C3, L2(c), E2(a), E1(c), C1, L2(a), C2

On utilise un protocole de gestion de concurrence basé sur l'estampillage (l'estampille temporelle pour la transaction Ti est i). Pour chaque séquence montrer si l'estampillage sans la règle de Thomas et avec la règle de Thomas (lorsqu'elle est applicable) permet l'exécution de la séquence dans l'ordre donné. Montrer l'ordre d'exécution des opérations et l'ordre en série équivalent.

4. CLICHÉS MULTI-VERSIONS (FACULTATIF)

On suppose maintenant que le contrôle de concurrence utilise des clichés multi-versions. Cela permet de traiter les lectures sans demander de verrou partagé. Définir l'ordre d'exécution des opérations et l'ordre en série équivalent.

a)L1(a), L1(b), E1(a), L2(b), E3(b), C3, E1(a), C1, L2(b), C2

b) L1(a), E2(a), C2, E1(a), C1, L3(a), C3

c) E1a, L2c, L2a, C2, E1a, L3a, L1c, C1, E3b, C3

d) E1(a), L1(a), L3(b), E3(a), C3, L2(c), E2(a), E1(c), C1, L2(a), C2

TD 8 : Dépendances Fonctionnelles

masquer=1

1. Fermeture d'attributs et clé d'un schéma de relation

Exercice 1.1

Soient R (A,B,C,D) et F = { A → B; B → C }.

1.1.1 Quelle est la fermeture $[A]^+_F$ de A?

1.1.2 Donnez les clés de R par rapport à F ?

Exercice 1.2

Soient R(A, B,C, D, E, F, G, H) et *Df*= {AB → C, B→ D, CD → E, CE → GH, G → A}.

1.3 Donnez les clés de R par rapport à *Df*.

2. Fermeture d'un ensemble de dépendances fonctionnelles

Soient R(A, B,C, D, E, F, G, H} et *Df*= {AB → C, B→ D, CD → E, CE → GH, G → A}.

2.1 Donnez les fermetures $[AB]^+_{Df}$ et $[BG]^+_{Df}$

2.2 Est-ce que AB → E, BG → C et AB → G font partie de la fermeture de *Df* ?

3. Ensemble minimal

Soient R{A, B,C, E, H} et les deux ensembles de dépendances fonctionnelles

F = {A → B ; CE → H; C → E ; A → CH} et G = {A → BC; C → EH ; AE → H}

3.1 *L'ensemble F* est-il minimal ? *L'ensemble G* est-il minimal ?

3.2 Les deux ensembles de dépendances fonctionnelles F et G sont-ils équivalents ?

4. Calcul de couverture minimale

Soient R = *(A,B,C,D,E,G) et*

F= {AB→C, C → A, BC → D , ACD → B , D → EG , BE → C , CG → BD , CE → AG }

4.1 Mettre les dépendances de F sous forme canonique.

4.2 Trouver les éventuelles DF non redondantes et redondantes (distinguer les DF redondantes qu'on peut retrouver uniquement à partir de DF non redondantes).

4.5 En déduire les couvertures minimales de F.

TD 9 : DÉCOMPOSITION ET FORMES NORMALES

masquer=1

1 DÉCOMPOSITION SPI ET SPD

Exercice

On considère le schéma de relation R(A,B,C) et la dépendance fonctionnelle A→C.

2.1. Donnez une instance R du schéma R(A,B,C) qui respecte la DF et telle que

$$\Pi_{A,B} R \ |><| \ \Pi_{B,C} R \neq R$$

2.2. Montrez formellement que la décomposition (R1(A,B), R2(B,C)) n'est pas Sans Perte d'Information (SPI)

Exercice

Soit *S* le schéma de base de données relationnelle suivant, sur lequel on a défini un ensemble *F* de dépendances fonctionnelles.

$$S = \{ R(A,B,C,D) \} \qquad F = \{ BC \rightarrow D , D \rightarrow C , C \rightarrow A \}$$

2.3. Quelles sont les clés minimales de R ? Montrez comment vous les obtenez ;

2.4. Le schéma S est-il en forme normale de Boyce-Codd ? S est-il en 3ème forme normale ?

2.5. Quelles sont les dépendances projetées sur R1 et sur R2 dans la décomposition de S en un nouveau schéma S'={ R1(A,D) , R2(B,C,D) } ? La décomposition de S en S' est-elle sans perte de dépendances ?

Exercice

On considère le schéma de relation R(A,B,C,D,E) et les dépendances fonctionnelles suivantes:

$$F=\{A \rightarrow C, \quad B \rightarrow C, \quad C \rightarrow D, \quad DE \rightarrow C, CE \rightarrow A\}$$

2.6. Déterminer, en utilisant l'algorithme du tableau, si la décomposition suivante est Sans Perte d'Information (SPI):

$\Delta 1$ = R1(A,D) R2(A,B) R3(B,E) R4(C,D,E) R5(A,E)

2.7. Donnez une instance r du schéma R(A,B,C,D,E) telle que

$$\Pi_{A,D} r \ |><| \ \Pi_{A,B} r \ |><| \ \Pi_{B,E} r |><| \ \Pi_{C,D,E} r |><| \ \Pi_{A,E} r \neq r$$

2.8. (facultatif) Est-ce que la décomposition $\Delta 1$ est Sans Perte de Dépendances (SPD) ?

2.9. Déterminer, en utilisant l'algorithme du tableau, si la décomposition suivante est Sans Perte d'Information (SPI):

$\Delta 2$ = R1(A,D) R2(A,B) R3(B,E) R4(C,D)
R5(D,E)

2.10. (plus difficile) Donnez une instance R du schéma R(A,B,C,D,E) telle que

$$\Pi_{A,D} R \text{ |><| } \Pi_{A,B} R \text{ |><| } \Pi_{B,E} R \text{ |><| } \Pi_{C,D} R \text{ |><| } \Pi_{A,E} R \neq R$$

2 FORMES NORMALES

Soit la relation UFR, de schéma :

UFR (N°TD, SALLE, JOUR, HEURE, N°ENSEIGNANT, NOM-ENSEIGNANT, PRENOM-ENSEIGNANT, COD-MOD, DIPLOME, MATIERE, N°ETUDIANT, NOM-ETUDIANT, PRENOM-ETUDIANT, ADRESSE, DATE-INSCRIPTION)

Les hypothèses sont les suivantes :

- Un code module précise à la fois un diplôme et une matière.
- Les TDs sont annuels et il y a un TD par semaine dans chaque module.
- Un TD est assuré par un seul enseignant.
- Un N° de TD est relatif à un module.
- Un enseignant peut assurer plusieurs TDs
- Un étudiant peut être inscrit dans plusieurs modules, mais dans un seul TD par module.
- Date-Inscription est la date d'inscription d'un étudiant à un module.

Un jeu de dépendances fonctionnelles F vous est fourni :

1	N°ETUDIANT	→	NOM-ETUDIANT, PRENOM-ETUDIANT, ADRESSE
2	N°ENSEIGNANT	→	NOM-ENSEIGNANT, PRENOM-ENSEIGNANT
3	COD-MOD	→	DIPLOME, MATIERE
4	DIPLOME, MATIERE	→	COD-MOD
5	SALLE, JOUR, HEURE	→	N°TD, COD-MOD
6	COD-MOD, N°TD	→	SALLE, JOUR, HEURE, N°ENSEIGNANT
7	COD-MOD, N°ETUDIANT	→	N°TD, DATE-INSCRIPTION
8	N°ENSEIGNANT, JOUR, HEURE	→	SALLE
9	N°ETUDIANT, JOUR, HEURE	→	SALLE
10	SALLE, JOUR, HEURE	→	N°ENSEIGNANT
11	N°ETUDIANT, COD-MOD, N°TD	→	SALLE, JOUR, HEURE

Exercice : Formes normales et anomalies

2.1. Montrer que les dépendances 10, 11 et COD-MOD, N°TD → SALLE sont redondantes dans cet ensemble de dépendances.Donner une couverture minimale F' de F

2.2. Donner les clefs de cette relation.

2.3. A l'aide d'exemples, montrer quelles redondances et anomalies sont impliquées par ce schéma.

Exercice : Décomposition 1

On considère maintenant la décomposition suivante, de la relation UFR :

ENSEIGNEMENT (N°TD, COD-MOD, JOUR, HEURE, SALLE, N°ENSEIGNANT, NOM-ENSEIGNANT, PRENOM-ENSEIGNANT)

INSCRIPTION (N°ETUDIANT, NOM-ETUDIANT, PRENOM-ETUDIANT, ADRESSE, COD-MOD, DIPLOME, MATIERE, DATE-INSCRIPTION, N°TD)

2.4. Donnez l'ensemble des DF projetées pour chacune des deux relations.

2.5 (facultatif) Cette décomposition préserve-t-elle les dépendances fonctionnelles ? Démontrez le.

2.5. Donnez l'ensemble des DF (projection) pour chacune des deux relations.

2.6. Donnez les clés de ces deux relations ENSEIGNEMENT et INSCRIPTION,

2.7. Montrer que cette décomposition est sans perte d'information.

2.8. Les deux relations sont-elles en 3ème forme normale ?

Exercice : Décomposition 2

Soit la décomposition de la table UFR suivante:

ENS_PLANNING(N°ENSEIGNANT, NOM-ENSEIGNANT, PRENOM-ENSEIGNANT, JOUR, HEURE,N°TD,CODE-MOD)

ENS_SALLE(N°ENSEIGNANT, JOUR, HEURE, SALLE)

ETU_INSCRIPTION(N°ETUDIANT, NOM-ETUDIANT, PRENOM-ETUDIANT, ADRESSE, N°TD, CODE-MOD, DATE-INSCRIPTION, DIPLOME, MATIERE)

ETU_PLANNING (N°ETUDIANT, JOUR, HEURE, CODE-MOD)

2.9. (facultatif) Cette décomposition préserve-t'elle les dépendances fonctionnelles ?

2.10. Proposer une nouvelle décomposition de la relation UFR telle que toutes les relations soient en troisième forme normale. Cette décomposition doit être sans perte d'information et préserver les dépendances fonctionnelles.

TD 10 : TRIGGERS

masquer=1

1. RAPPELS TRIGERS

La syntaxe d'une expression de création de trigger en SQL3 est la suivante :

```
1. CREATE TRIGGER <nom-trigger>
2.         BEFORE | AFTER | INSTEAD OF
3.         INSERT | DELETE | UPDATE OF <liste_attributs>
4.        ON <nom-table>
5.         [ORDER <valeur de priorité>]
6.         [REFERENCING NEW | OLD AS <nom-variable>]
7.         FOR EACH ROW | STATEMENT
8.         [WHEN (<conditionSQL>)]
9.  BEGIN <actionSQL> END ;
```

Les expressions entre […] sont optionnelles. Le symbol '|' sépare les options :

Ligne 1: <nom-trigger> indique le nom du trigger.
Ligne 2: indique si le trigger est déclenché avant (BEFORE), après (AFTER) ou à la place (INSTEAD OF) d'un événement
Ligne 3: indique le type de l'événement
Ligne 4: indique la table concernée par l'événement
Ligne 5: la clause ORDER est optionnel et sert à gérer les priorités entre des triggers en conflit
Ligne 6: la clause REFERENCING sert nommer une ou plusieurs variables temporaires utilisables dans la partie condition et dans la partie action. NEW et OLD désigne respectivement la valeur du dernier n-uplet modifié par l'événement après et avant modification si la granularité de déclenchement est ROW (ligne 7), l'ensemble des nuplets touchés par l'événement, respectivement après et avant l'événement si la granularité est STATEMENT (ligne 7). Le nome de la variable est `:nom_variable` (avec un ' :' ajouté au début dans le reste du trigger. Si la clause REFERENCING est omise, alors les variables s'appellent par défaut `:new` et `:old` dans le reste du trigger.
Ligne 7: la clause FOR EACH détermine la granularité du déclenchement:
- ROW : la règle est déclenchée à chaque n-uplet touché par un événement.
- STATEMENT : la règle est déclenchée qu'une seule fois pour l'événement.

Ligne 8: permet d'indiquer une condition SQL qui doit être vraie pour déclencher le trigger
Ligne 9: indique les actions à effectuer (en PLSQL) – voir TME7 pour pus d'explications.

Il est évident qu'à un événement de type INSERT (resp. DELETE) ne peut pas correspondre une delta-structure OLD (resp. NEW). Une règle AFTER ne devrait en principe pas modifier la valeur de la variable temporaire déclarée par NEW (cette valeur est déjà écrite dans la base), une règle BEFORE ne devrait pas en principe modifier la base par une commande INSERT, UPDATE ou DELETE sur les relations de la base (puisque l'événement n'a pas encore eu lieu réellement), elle ne peut modifier que les variables temporaires.

2. BASE DE DONNÉES « ENTREPRISE »

La base de données d'une entreprise contient les trois tables suivantes :

```
CREATE TABLE  EMPLOYE (
       ID_EMP NUMBER(8) PRIMARY KEY,
       NOM VARCHAR(32) NOT NULL,
       PRENOM VARCHAR(32) NOT NULL,
       FONCTION VARCHAR(32),
       SALAIRE NUMBER(7,2) NOT NULL)) ;
CREATE TABLE PROJET (
       ID_PROJ NUMBER(8) PRIMARY KEY,
       NOM VARCHAR(32) NOT NULL,
       ID_CHEF_PROJET NUMBER(8) REFERENCES EMPLOYE
              ON DELETE SET NULL
              ON UPDATE CASCADE) ;
CREATE TABLE PARTICIPE (
       ID_EMP NUMBER(8) REFERENCES EMPLOYE
              ON DELETE CASCADE
              ON UPDATE CASCADE,
       ID_PROJET  NUMBER(8) REFERENCES PROJET
              ON DELETE RESTRICT
              ON UPDATE RESTRICT,
       PRIMARY KEY (ID_EMP, ID_PROJET));
```

La table EMPLOYE contient pour chaque employé son identifiant, son nom et prénom, sa fonction (optionnel), et son salaire. La table PROJET contient les identifiants et noms des projets et une référence vers l'employé qui dirige le projet. La table PARTICIPE stocke les employés avec les projets auxquels ils participent.

Nb : dans Oracle ON UPDATE CASCADE n'est pas implanté et ON DELETE RESTRICT est par défaut.

2.1 Écrire un trigger BEFORE qui évite qu'un salaire ne puisse diminuer

2.2 Idem avec un trigger AFTER

2.3 Écrire un trigger qui empêche qu'on supprime plus de 50 n-uplets à la fois dans la relation EMPLOYE

2.4 Écrire les triggers AFTER qui simulent les actions ON DELETE | UPDATE associés aux clés étrangères de la table PROJET :

2.5 Écrire les triggers AFTER qui simulent les actions ON DELETE | UPDATE associés aux clés étrangères de la table PARTICIPE :