

UFR 919 - Cycle L3 Informatique

UE 3I009 - Systèmes de Gestion de Bases de Données

Support de TD

2015 – 2016

Équipe pédagogique

Bernd Amann

Mohamed-Amine Baazizi

Camelia Constantin

Anne Doucet

Stéphane Gançarski

Hubert Naacke

1. Rappels : E/A, calcul et SQL	p. 1
2. Algèbre relationnelle	p. 4
3. Optimisation 1	p. 6
4. Indexs / Arbre-B+	p. 9
5. Hachage extensible	p. 11
6. Optimisation 2	p. 13
7. Triggers	p. 15
8. Dépendances fonctionnelles	p. 17
9. Décomposition et formes normales	p. 19
10. et 11. Transactions et concurrence	p. 23

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*)

- On veut déduire le schéma Entités-Associations qui a servi à obtenir le schéma R.
 - Enumérez les entités du schéma E-A en indiquant leurs attributs.
 - Indiquez les entités faibles et les entités fortes auxquelles elles sont reliées.
 - Précisez les attributs identifiants pour chacune des entités.
- 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
TD1-révisions_24	30	20
3	10	25
3	20	10

Instance 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. « 3^e 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 $\{ t.A_0, t.A_1, \dots \mid \mathbf{F}(t) \}$, qui désigne une projection sur les attributs A_0, A_1, \dots de l'ensemble des n-uplets t satisfaisant le prédicat $\mathbf{F}(t)$ 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 \mathbf{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	$\Pi_{\text{liste-d'attributs}}(\mathbf{R})$
Sélection	$\sigma_{\text{condition}}(\mathbf{R})$
Renommage	$\rho_{\text{nom-attribut} \rightarrow \text{nouveau_nom}}(\mathbf{R})$
Union	$\mathbf{R} \cup \mathbf{S}$
Différence	$\mathbf{R} - \mathbf{S}$
Produit cartésien	$\mathbf{R} \times \mathbf{S}$

2. Opérateurs dérivés :

Jointure	$\mathbf{R} \bowtie \mathbf{S}$
Division	$\mathbf{R} \div \mathbf{S}$
Intersection	$\mathbf{R} \cap \mathbf{S}$

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.

TD 3 : OPTIMISATION

masquer=1

1. TAILLE ET CARDINALITÉ DES OPÉRATIONS ALGÈBRIQUES

Soit deux relations $R(a_1, a_2, a_3)$ et $S(b_1, b_2, a_1)$ contenant T_r et T_s tuples. L'attribut $R.a_1$ est la clé primaire, l'attribut $S.a_1$ est une clé étrangère vers R . La taille (en octets) des attributs est $ta_1, ta_2, ta_3, tb_1, tb_2$. Le nombre de valeurs distinctes des attributs est $da_1, da_2, da_3, db_1, db_2$, 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 a_1 et a_3 ?
- b) Op2: Sélection de R avec le prédicat $a_2 = 1$
- c) Op3 : la jointure naturelle de R et S sur l'attribut a_1
- 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 ALGÈBRIQUES 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 Réserveation	NumH	10 000	-	5
Client et Réserveation	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, Réserveation r
 WHERE r.NumH = 536 **AND** r.NumC = c.NumC

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

R3: **SELECT** r.NumC
 FROM Réserveation 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.

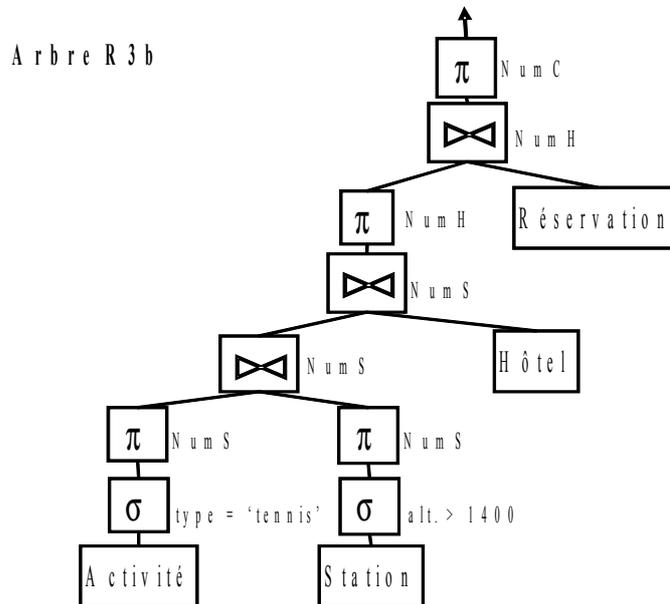
2.3.1. Quelle sont les transformations algébriques à effectuer pour appliquer cette règle ?

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 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, N_i pour les nœuds intermédiaires et F_i pour les feuilles. On peut utiliser la syntaxe suivante pour représenter le contenu d'un nœud : $N(v_1, v_2, \dots)$ 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

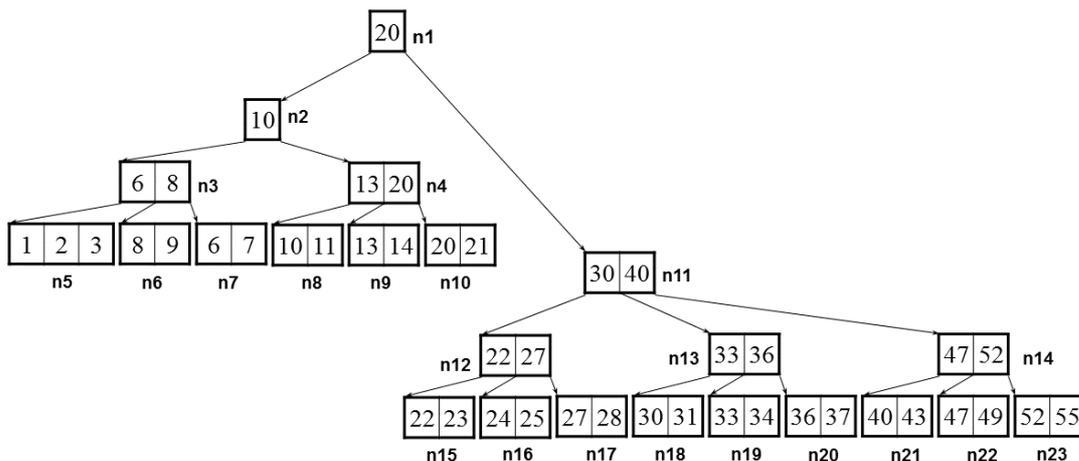
Dans cet exercice, les arbres sont d'ordre 1 (i.e., il y a 1 ou 2 valeurs par nœud).

1) Soit l'arbre A_0 composé d'une racine $N_1(10)$, et de deux feuilles $F_1(5)$ et $F_2(20)$. A_0 a 2 niveaux. On insère successivement dans A_0 les nombres entiers 8 puis 30 puis 15. On obtient A_1 . Dessiner A_1 .

2) On insère dans A_1 le nombre 10. On obtient A_2 . Dessiner A_2 .

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 Index – Hachage extensible

Notations et conventions

Dans toutes les tables de hachage considérées, un paquet ne peut pas contenir plus de 4 valeurs. Lors d'une suppression, si un paquet devient vide, on tente de le fusionner avec un autre paquet, si ce n'est pas possible, il reste vide.

On utilise la notation suivante.

Le répertoire est noté $\mathbf{R}[P_0, P_1, P_2, \dots, P_k]$ $\mathbf{PG}=m$ avec

P_i ... les noms d'un paquet,

m la profondeur globale.

Rmq : le répertoire contient k cases avec $k = 2^m = 2^{\mathbf{PG}}$

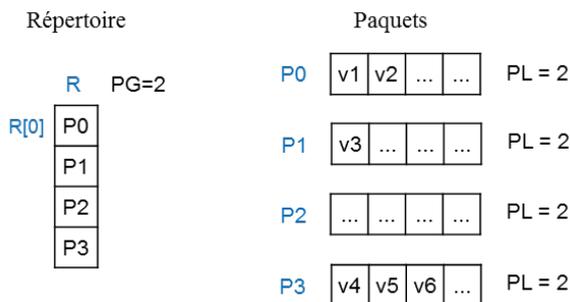
Un paquet est noté $P_i(v_j, \dots, \dots)$ $\mathbf{PL}=n$ avec

P_i le nom du paquet, par exemple A,B, ... ,

V_j les valeurs que contient le paquet,

n la profondeur locale.

On peut aussi préciser le contenu d'une case particulière du répertoire avec $\mathbf{R}[i]=L$ (avec $\mathbf{R}[0]$ étant la 1^{ère} case).



Exercice 1 : Hachage extensible

Une structure de hachage extensible contient des entrées x dont les valeurs de hachage $h(x)$ sont les suivantes : 1, 4, 5, 7, 10, 12, 15, 16, 19, 21, 32.

- 1) Avec au plus 4 valeurs par paquets, combien faut-il de paquets pour avoir la structure la plus compacte possible ?
- 2) Dessiner la table de hachage, nommée T1, pour ces valeurs. Représenter le répertoire avec le nom des paquets qu'il contient et sa profondeur globale. Représenter chaque paquet avec les valeurs qu'il contient et sa profondeur locale.
- 3) Expliquer pas à pas comment retrouver le paquet contenant la valeur 7.
- 4) Insérer 13
- 5) Insérer 20
- 6) Insérer 29
- 7) On considère la table obtenue. Donner un exemple de plus petit ensemble de valeurs à supprimer pour obtenir une division par 2 du répertoire.

Exercice 2

On considère une structure de hachage extensible de profondeur globale 3. Dans l'état initial, nommé $\mathbf{T0}$, les paquets du répertoire sont les suivants : $\mathbf{R}[A, B, C, D, A2, B, C, D]$ (i.e., $\mathbf{R}[0]=A, \dots, \mathbf{R}[7]=D$), et $\mathbf{PG}=3$

Les valeurs (tous paquets confondus) sont les suivantes : 1, 4, 5, 7, 10, 12, 15, 16, 20, 21, 36, 51, 64.

- 1) Représenter $\mathbf{T0}$.
- 2) Insérer 68 dans $\mathbf{T0}$.
- 3) On considère la table $\mathbf{T0}$ initiale. Insérer 17 et 69 dans $\mathbf{T0}$.
- 4) Supprimer 21 de $\mathbf{T0}$.
- 5) Supprimer 10, puis 64 et 16 de $\mathbf{T0}$.

Exercice 3

- 1) La table de hachage T_0 contient seulement les valeurs 1, 4, 5, 6, 9, 14, 25. De combien de paquets a-t-on besoin au **minimum**? Préciser leur contenu, leur PL, le contenu du répertoire.
- 2) On insère 7 dans T_0 . On obtient T_1 . Représenter T_1 . Préciser ce que contient chaque case du répertoire R
- 3) On insère 13 dans T_1 obtenu à la question précédente. On obtient T_2 . Représenter T_2 .
- 4) On considère la table T_3 avec les paquets A(10), B(16), C(17,31), D(30), E(36). Représenter T_3 . Rmq : vous devez préciser le contenu de chaque case du répertoire ($R[0]$ est différent de A, la taille du répertoire est une puissance de 2).
- 5) On supprime 10 dans T_3 . On obtient T_4 . Représenter T_4
- 6) On supprime 30 dans T_4 obtenu à la question précédente. On obtient T_5 . Représenter T_5 .

TD Optimisation de requêtes (2)

Notations et conventions

Dans tous les exercices, 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 soit la page (exercice 1), soit le nuplet (ex. 2 et 3).

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. Evaluer la jointure consiste à parcourir les nuplets résultant de A afin d'accéder itérativement aux nuplets de B qui satisfont le prédicat de jointure (cf. cours : algorithmes de jointure). Cela suppose que A soit le *résultat* d'une requête et que B soit une *table* stockée dans le SGBD. Le coût de A est calculé de manière séparée et vient s'ajouter à celui de 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.

Exercice 1 : Jointure entre 2 relations

Soit les relations :

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

Le domaine des attributs est : $a \in [1, 1000]$, $b \in [1, 10]$, $c \in [1, 100]$.

Les cardinalités des relations sont : $\text{card}(R) = 1000$, $\text{card}(S) = 100$

La taille en nb de pages des relations est : $P(R) = 500$, $P(S) = 10$

Le modèle de coût simplifié est :

Le coût d'une équi-jointure entre les relations A et B, avec le prédicat $A.a = B.a$ est :

$$\text{coût}(A \bowtie_{A.a=B.a} B) = \text{card}(A) * \text{coût}(\sigma_{a=v} B) \quad \text{où } v \text{ est une valeur quelconque.}$$

Le coût d'une sélection sur la relation A avec le prédicat $a=v$ est :

si *a* est indexé : $\text{coût}(\sigma_{a=v} A) = \text{card}(\sigma_{a=v} A)$

sinon : $\text{coût}(\sigma_{a=v} A) = P(A)$.

Le coût d'une lecture séquentielle est le coût d'une sélection sans index.

Soit la requête :

```
select *
from R, S
where a = c and b = 1
```

Pour chaque question, énumérer tous les plans équivalents (notés P1 à P4) et calculer leur coût. Pour cela, représenter un plan d'exécution sous la forme d'un arbre d'opérateurs. Donner le coût et la cardinalité de chaque opérateur de l'arbre, donner le coût total de l'arbre.

- 1) Il y a seulement un index sur R.a et un index sur R.b,
- 2) Il y a seulement un index sur S.c
- 3) Il y a seulement un index sur R.b et un index sur S.c.

Exercice 2 : Jointure entre 3 relations

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 2^{ème} 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 $\text{card}(R)$ est le nombre de tuples de R, le coût d’une *sélection* sur égalité est :

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

$$= \text{card}(R) \text{ sinon.}$$
 - Le coût d’une lecture séquentielle est le coût d’une sélection sans index.
 - Pour toutes relations R ayant $\text{card}(R)$ nuplets et S ayant $\text{card}(S)$ nuplets, le coût d’une *equi-jointure* est :

$$\text{coût}(R \bowtie_a S) = \text{card}(R) \text{ s’il y a un index sur l’attribut } a \text{ de } S$$

$$= \text{card}(R) * \text{card}(S) \text{ sinon.}$$
 - 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)
 - $\text{card}(R1) = \text{card}(R2) = \text{card}(R3) = 1000$, $\pi_A(R1) = \pi_A(R2) = \pi_A(R3)$
 - 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: Club de joueurs - A faire ou à continuer en TME

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 ?

TD 7 : TRIGGERS

masquer=1

1. RAPPELS TRIGGERS

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 implémenté 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 :

TD 8 : DÉPENDANCES FONCTIONNELLES

masquer=1

1. FERMETURE D'ATTRIBUTS ET CLÉ D'UN SCHÉMA DE RELATION

Exercice

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

- 1.1 Quelle est la fermeture $[A]^+_F$ de A ?
- 1.2 Donnez les clés de R par rapport à F ?

Exercice

Soient $R(A, B, C, D, E, F, G, H)$ et $G = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$.

- 1.3 Donnez les clés de R par rapport à G .

2. FERMETURE D'UN ENSEMBLE DE DÉPENDANCES FONCTIONNELLES

Soient $R(A, B, C, D, E, F, G, H)$ et $F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$.

- 2.1 Donnez les fermetures $[AB]^+_G$ et $[BG]^+_G$.
- 2.2 Est-ce que $AB \rightarrow E$, $BG \rightarrow C$ et $AB \rightarrow G$ font partie de la fermeture de F ?

3. ENSEMBLE MINIMAL

Soient $R(A, B, C, E, H)$ et les deux ensembles de dépendances fonctionnelles

$$F = \{A \rightarrow B; CE \rightarrow H; C \rightarrow E; A \rightarrow CH\} \text{ et}$$

$$G = \{A \rightarrow BC; C \rightarrow EH; AE \rightarrow H\}$$

- 3.1 L'ensemble F est-il minimale ?
- 3.1 L'ensemble G est-il minimale ?
- 3.2 (facultatif) Les deux ensembles de dépendances fonctionnelles F et G sont-ils équivalents ?

Exercice facultatif :

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

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

- 4.1 Établir $[C]^+_F$, $[BD]^+_F$
- 4.2 En déduire des dépendances non triviales $BD \rightarrow X$ (DF triviale = obtenue par réflexivité) de F^+ qui n'appartiennent pas à F .

4.3 Mettre les dépendances de F sous forme canonique.

4.4 Trouver les éventuelles DF non redodantes et 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 $A \rightarrow C$.

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

$$\Pi_{A,D} R \not\subseteq \Pi_{B,C} R \neq R$$

2.2. *Montrez formellement que la décomposition $R_1(A,D)$, $R_2(B,C)$ 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.

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

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 R_1 et sur R_2 dans la décomposition de S en un nouveau schéma $S' = \{ R_1(A,D), R_2(B,C,D) \}$? 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:

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

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

$$\Delta_1 = R_1(A,D) \quad R_2(A,B) \quad R_3(B,E) \quad R_4(C,D,E) \quad R_5(A,E)$$

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

$$\Pi_{A,D} R \not\subseteq \Pi_{A,B} R \not\subseteq \Pi_{B,E} R \not\subseteq \Pi_{C,D,E} R \not\subseteq \Pi_{A,E} R \neq R$$

2.8. *Est-ce que la décomposition Δ_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):*

$$\Delta_2 = R_1(A,D) \quad R_2(A,B) \quad R_3(B,E) \quad R_4(C,D) \quad R_5(D,E)$$

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

$$\Pi_{A,D} R \bowtie \Pi_{A,B} R \bowtie \Pi_{B,E} R \bowtie \Pi_{C,D} R \bowtie \Pi_{A,E} R \neq R$$

2.11. Est-ce que la décomposition $\Delta 2$ est Sans Perte de Dépendances (SPD) ?

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 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 $COD-MOD, N°TD \rightarrow SALLE$ sont redondantes dans cet ensemble de dépendances.
- 2.2. Donner la clef 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. *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 3^{ème} forme normale ?*

Exercice : Décomposition 2

On décompose maintenant la relation INSCRIPTION en quatre relations, et on obtient la décomposition suivante :

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

ETUDIANT (N°ETUDIANT, NOM-ETUDIANT, PRENOM-ETUDIANT, ADRESSE)

INSCRIPTION (N°ETUDIANT, COD-MOD, DATE-INSCRIPTION, N°TD)

MODULE (COD-MOD, DIPLOME, MATIERE)

- 2.9. *Cette décomposition est-elle sans perte d'informations et conserve-t-elle les dépendances ?*
- 2.10. *Ces quatre relations sont-elles en deuxième forme normale ?*
- 2.11. *Ces quatre relations sont-elles en troisième forme normale ?*

Exercice : Décomposition 3 :

Soit la décomposition de la table INSCRIPTION 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)

2.12. Cette décomposition préserve-t-elle les dépendances fonctionnelles ?

2.13. 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.

3. Exercice décomposition annale d'examen

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

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

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

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

7.3 Quelles sont les dépendances projetées sur R_1 et sur R_2 dans la décomposition de S en un nouveau schéma $S' = \{ R_1(A,D), R_2(B,C,D) \}$? La décomposition de S en S' est-elle sans perte de dépendances ?

7.4 La décomposition de S en S' est-elle sans perte d'information ? Si oui prouvez le, si non montrez un contre-exemple. Idem pour $S'' = \{ R_1(A,B), R_2(B,C,D) \}$

TD : CONCURRENCE

masquer=1

1. ISOLATION ET SÉRIALISABILITÉ

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

procédure *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). Chacun des processus (i.e., chacune des transactions de transfert) utilise une variable locale à son propre espace mémoire. 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 AB_i représente l' i ème instruction du transfert de A vers B et BC_j la j ème instruction du transfert de B vers C?

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

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

AB1 BC1 BC2 BC3 AB2 AB3 AB4 BC4

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

AB1 AB2 AB3 BC1 BC2 BC3 AB4 BC4

1.1.2 Quelle contrainte d'intégrité n'est pas respectée 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 précedence* 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)

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

2.3.1. Donner le graphe de précedence de cette exécution.

2.3.2. On suppose que le contrôle de concurrence est effectué par du verrouillage en deux phases, et que les demandes de verrous se font dans l'ordre global 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) E5(X) L2(Y) L2(Z) E4(Z) E1(X) L1(Y) E1(Y) L4(T)
L1(Z) E4(T)

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 T_i 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

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