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

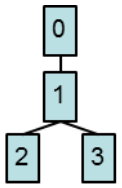
Un plan est représenté par un arbre dont les *n* nœuds sont numérotés de *0* (pour la racine) à *n-1*. La numérotation se fait en largeur, de gauche à droite. L'arbre est binaire et chaque nœud a de 0 (feuille) à 2 fils. Dans la représentation textuelle du plan, chaque nœud correspond à une ligne et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

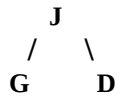
L'explication doit apporter des éléments de réponses aux questions suivantes :

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	Natioanlite

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

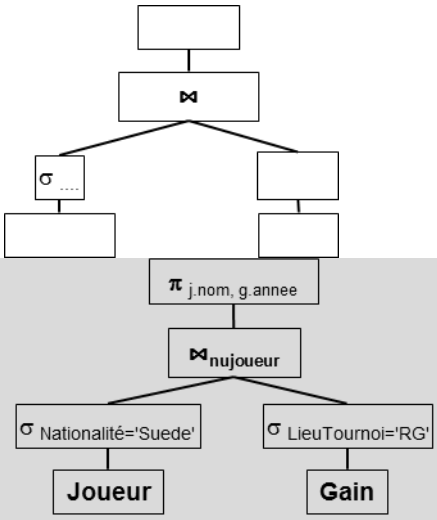
Réponse présentée sous la forme d'un tableau :

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	850	[1 000, 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) contenant **1000** villes.

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 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

```
explain plan for     select a.nom, a.prenom, v.ville
                     from Annuaire a, Ville v
                     where a.cp = v.cp
                     and a.cp between 75000 and 76000;
```

Plan:

Id	*Operation*	*Name*	*Rows*
0	SELECT STATEMENT		22
1	NESTED LOOP		
2	NESTED LOOP		22
3	TABLE ACCESS FULL	Ville	11
4	INDEX RANGE SCAN	IndexCP	2
5	TABLE ACCESS BY INDEX ROWID	Annuaire	1

Predicate Information:

IdOp	*Predicate*
3	Filter(cp>=75000 and cp <=76000)
4	Access(a.cp = v.cp)

Projection Information:

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

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).

Réponse présentée sous la forme d'un tableau :

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

Exercice 3 : 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 : $a \in [1,1000]$, $b \in [1,10]$, $c \in [1,100]$.
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

Si B n'est **pas** une table mais est une expression, il faut rajouter son coût d'évaluation et le coût de la stocker.

$coût(A \bowtie_{A.a=B.a} B) = coût(B) + P(B) + coût(A) + P(A) * P(B)$

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

si *a* est indexé : $coût(\sigma_{a=v} C) = card(\sigma_{a=v} C)$
sinon : $coût(\sigma_{a=v} C) = P(C)$.

Le coût d'une lecture séquentielle de toute la table C est P(C).

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 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, $\pi_A(R1) = \pi_A(R2) = \pi_A(R3)$
  - 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 ?