

TD1 - Index, Arbres B+

Exercice 1 - Index composés

1) $\sigma_{\text{année} = 2016}$ (livre)

chaque index est traversé jusqu'à une feuille, puis
parcours transversal : réponse = - atteindre quelle feuille ?
- quel parcours ?

 I_1 : oui

→ atteindre la clé (2016, min_auteur) avec min_auteur le plus petit auteur pour
mon l'année 2016
racine → 2016 → min_auteur

→ puis parcours latéral jusqu'à (2016, max_auteur) (jusqu'avant année=2017)

I_2 : non, car si on atteint la clé (min_prix, 2016), le parcours des clés jusqu'à (max_prix, 2016)
contient des clés inutiles.

{ nécessiterait un filtrage des clés

{ soit on arrive à identifier une série de clés toutes utiles pour la requête

{ soit filtrer

 I_3 : idem I_2 .

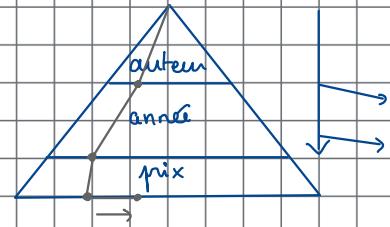
2) (intervalle ouvert sur l'année, point sur l'auteur) année > 2010, auteur = "Hugo"

I_1 : index seulement pour année : racine → 2010 → Hugo
puis parcours latéral et filtre sur auteur = "Hugo"

atteindre (2010, "Hugo") puis parcours (max_annee, "Hugo") contenant des clés inutiles.

 I_2 : non

I_3 : atteindre ("Hugo", 2010, min_prix)
parcours latéral jusqu'à ("Hugo", max_annee, max_prix)



3) année > 2000, prix > 50

I_1 : atteindre (2000, min_auteur)
parcours (max_annee, min_auteur)
lire les tuples de livre
puis filtrer sur prix > 50 (opération se faisant après avoir obtenu les données)

I_2 : atteindre (50, 2000)
parcours (max_prix, max_annee), contenant des clés inutiles (51, 1999)

I_3 : pas de prédictat sur l'auteur, ce qui va créer plein de clés sur tous les auteurs.

4) année = 2016, prix = 20, auteur like "T%"

I_1 : atteindre (2016, min_T%)

parcours (2016, max_T%)
lire les tuples (données)

puis filtrer sur le prix = 20

I_2 : atteindre (20, 2016)

lire les données
filtrer sur l'auteur like "T%"

I_3 : atteindre (min_T%, 2016, 20)

? parcours (max_T%, 2016, 20) contient des clés inutiles (T_x , 2015, 20)

Stratégie : regarder la req

prendre tous les prédictifs

se limiter aux prédictifs conjonctifs

sous-ensemble de prédictifs

écrire par l'index

(diapo 24)

Exercice 1

index non gérant : index "par défaut" - donnée auxiliaire à côté des données

on va regarder le côté courant : est-il possible d'évaluer une requête sans lire les données ?

e) oui : R3 est l'ensemble des valeurs de ville dans l'index I_1 .

f) - utiliser I_1 et I_2 : I_1 contient des entrées (ville, liste de RowID)

- accéder à I_1 : atteindre (Paris)

obtenir L_1 , la liste de RowID.

- accéder à I_2 : atteindre (ville, min_age)

parcourir (ville, max_age)

obtenir L_2 , la liste des couples (RowID, age) pour tous les clés parcourues (union)

- jointure : $L = L_1 \bowtie_{\text{RowID}} L_2$: contient seulement les (RowID, age) tq sport = "ville", ville = "Paris"

- $R_4 = \text{avg}(\Pi_{\text{age}}(L))$

↓ projection sur l'âge

g) - accéder à I_2 : atteindre (18, judo)

obtenir L_2 (RowID)

- accéder à I_4 : atteindre (min_Ted%)

parcourir (max_Ted%)

obtenir L_4 : liste ordonnée de (prénom, RowID)

- $L = L_2 \bowtie_{\text{RowID}} L_4$ (préserver l'ordre)

- $R_5 = \Pi_{\text{prénom}}(L)$ projeter L sur le prénom en préservant l'ordre

Exercise 3

Q1. $\text{card}(R_1) = 200 = \text{card}(\pi_{\text{prix} < 20}(L))$
 $= \text{SF}(\pi_{\text{prix} < 20}(L)) \times \text{card}(L)$
 $200 = \frac{2}{10} \times \text{card}(L)$
 $\Rightarrow \text{card}(L) = \frac{200 \times 10}{2} = 1000$

$\text{card}(R_2) = 60 = \text{card}(\sigma_{\text{annee} > 2013}(L))$
 $= \frac{2}{100} \times \text{card}(L)$

$$\text{card}(L) = 20 \times 100 / 2 = 1000$$

Q2. $\text{coût}(\sigma_{\text{resume} = v}(L)) = 68$ via lecture séquentielle

Q3. $\text{card}(\sigma_{\text{annee} > 2013}(L)) = \text{SF}(\sigma_{\text{annee} > 2013}(L)) \times \text{card}(L)$
 $= \frac{4}{100} \times 1000 = 40$

$$\text{coût}(\sigma_{\text{annee} > 2013}(L)) = \text{card} + \text{Covrid} = 42$$

Q4. $\text{card}(\sigma_{\text{annee} \geq 2000}(L)) = \frac{17}{100} \times 1000 = 170$

$$\text{coût} = \text{card} + \text{Covrid} = 172$$

Q5. $\text{coût}(\pi_{\text{prix} = 10}) = 2 + \text{card} = 2 + \frac{1}{10} \times 1000 = 102$

lecture séquentielle = 68

on n'utilise pas l'index

Q6. $L_1 = []$
 for ligne between 2000 and 2002:
 $L_1.append(\text{line}.annee.getResidu(annee))$

$$L_2 = [] - L_prix(80)$$

$$L = L_1 \cap L_2$$

for rowid in L:

i = ligne + tuple(rowid)
 $L = \text{Affiche}(L)$

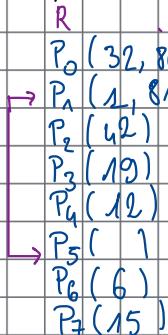
Q7. Select sum(prix) from ligne

on fait le rowid et on somme. Il n'y a pas de rowid qui est multiplié par le prix

TD2 - Hachage extensible

Exercice 1 - Table de hachage extensible

Q1. 1) $PG = 3$. $2^3 = 8$ cases. On inscrit v dans la case $i = nr \bmod 2^3$



S'il prenait un PG de taille $PG = 2$, on ne pourrait pas tout stocker.

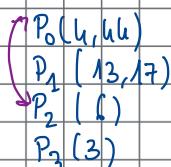
9 -valeurs, ici on peut stocker 16 valeurs. Avec $PG = 2$, on ne pourraient en stocker que 8 .

Peut-on effectuer une fusion pour supprimer P_5 ? Oui car $P_1 = PG$ (pb 3 cours)

R devient $[P_0, P_1, P_2, P_3, P_4, \cancel{P_5}, P_6, P_7]$ et $P_1(1, P_1)$ $PL = 2$.

Qu' se passe-t-il si maintenant on voulait insérer 13? Initialement on l'avait mis dans P_5 . On l'ajoute toujours dans la 5^e case dans P_1 .

2) $PG = PL = 2$



Supprimer 6: $R[P_0, P_1, P_2, P_3]$ et $P_0(4, 44)$ $PL = 1$.

Question 2 $A(0)$
 $B(10)$

a) $R[B, A]$ - $PG = 1$. $9 \bmod 2 = 1$
 $10 \bmod 2 = 0$ d'où $A = P_1$, $B = P_0$.

T'_2 : $B(9, 1, 5, 7)$ $PL = 1$
 $A(10, 2, 8, 12)$ $PL = 1$

b) Insérer 13 dans $R[13 \bmod 2] = R[1] = A$ mais A plein. Comme $PL = PG$, on double le répertoire. Donc $R = [B, A, B, A]$ et $PG = 1 + 1 = 2$.

On a bien pour A: $PL < PG$ et on applique le cas 2.

- créer un paquet C
- références C dans R: $R[3] = C$
- incrémenter les PL de A et C $\rightarrow PL = 2$
- redistribuer les valeurs de A

$A(9, 1, 5, 13)$ PL = 2
 $B(10, 2, 8, 12)$ PL = 1
 $C(7)$ PL = 2

insérer 13 dans $R[13 \bmod 4] = R[1] = A$.

c) insérer 41 dans $R[41 \bmod 4] = R[1] = A$ mais A plein et $PL = PG$.

Donc: $R[B, A, B, C, B, A, B, C]$ et $PG = 2+1+3$ (on double la répartition)
Et on a désormais pour A : $PL < PG$ (cas 2). On écrit $R[5] = D$

$A(9, 1, 4, 1)$ PL = 3
 $B(10, 2, 8, 12)$ PL = 1
 $C(7)$ PL = 2
 $D(5, 13)$ PL = 3

et $R = [B, A, B, C, B, D, B, C]$

Q3. H_0 doit indexer 5 valeurs, 2 paquets suffisent (on en peut avoir 4 valeurs par paquet).

$$H_0 : R_0[A_0, B_0] \quad , \quad H_1 : R_1[A_1, B_1]$$

insérer une valeur n dans la table H_i : tq $i = n \bmod 2$ Considérons $n = 15$ pour utiliser les tables

Exemple: insérer 12 dans $H_{12 \bmod 2} = H_0$ puis insérer 12 dans $R_0[12 \bmod 2] = R_0[0] = A_0$.

insérer 10 dans $H_{10 \bmod 2} = H_0$ puis insérer 10 dans $R_0[10 \bmod 2] = R_0[1] = B_0$

insérer 11 dans $H_{11 \bmod 2} = H_1$ puis insérer 11 dans $R_1[11 \bmod 2] = R_1[1] = B_1$

insère 9 dans $H_{9 \bmod 2} = H_1$ puis insérer 9 dans $R_1[9 \bmod 2] = R_1[0] = A_1$

Exercice 9.

Q5. (Hachage linéaire)

Hachage linéaire: pas de répartition mais on connaît la position p du prochain paquet à éclater et le seuil du taux d'occupation: 75%

$$N = 4 \text{ (nombre de paquets au départ)} \quad h_0(n) = n \bmod (N \times 2^0) = n \bmod (4) \\ h_1(n) = n \bmod (N \times 2^1) = n \bmod (8)$$

• $P_0(4, 8)$

$P_1(1)$

$P_2(10, 14)$

$P_3(15)$

On insère 1: $h_0(1) = 1 \% 4 = 1$

$h_0(1) > p$, donc placer 1 dans P_1 .

On insère 15: $h_0(15) = 3 > p$, donc placer 15 dans P_3 .

Après l'insertion de 15, le taux d'occupation est: $t = \frac{6}{8} = 75\%$ donc le seuil n'est pas dépassé. Il faut normalement le calculer à chaque insertion.

Insérer 22: $h_0(22) = 22 \bmod 4 = 2 > p \rightarrow P_2$

Mais ici, P_2 est plein: on procède en 2 temps:

- débordement: $P_2(10, 14) \rightarrow P'_2(22)$

taux d'occupation: $t = \frac{7}{8} = 87,5\% > 75\%$

- déclenche un éclatement du paquet $P_p = P_0$

On incrémente $p = 0 + 1 = 1$ et on redistribue les malwares de P_0 entre P_0 et P_1 .

$P_0(8)$

- $P_1(1)$

$P_2(10, 16) \rightarrow P'_2(22)$

$P_3(15)$

----- éclatement

$P_4(4)$

et taux d'occupation = $\frac{7}{10} = 70\% \leq 75\%$

Placer 4 dans $h_0(4) = 0$ mais $h_0(4) < p$

donc utiliser $h_1(4) = 4 \rightarrow P_4$

Placer 8 dans $h_0(8) = 0 < p$ donc $h_1(8) = 0 \rightarrow P_0$.

TD3 - Optimisation de requêtes

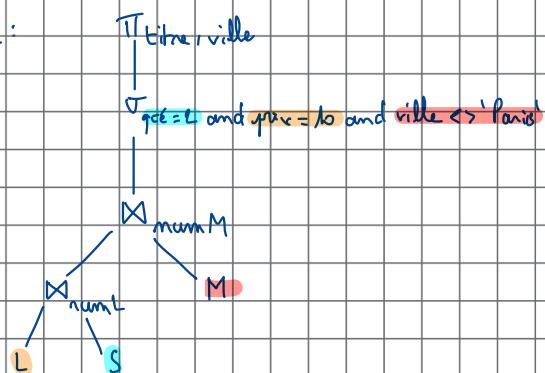
$\hat{c}_\text{out} \neq c_\text{and}$

↓
dansé de
la requête

Exercise 1

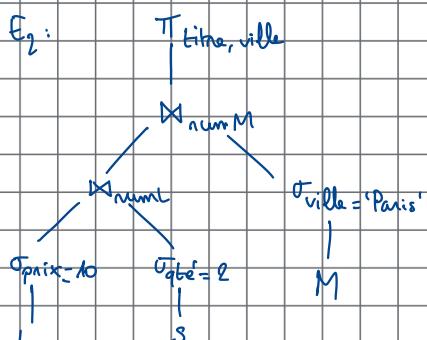
Q1. Expression algébrique de R1.

E_A:



→ idée : n'ordonner les opérations

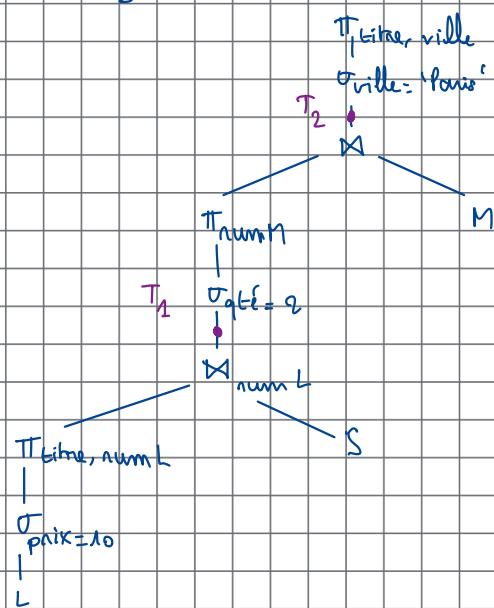
$$E_9 :=$$



- E_2 equivalente à E_1

- E_2 n'est pas linéaire à gauche à cause de σ_{gtr} et σ_{ville} .

E_2 :



$$T_1 = \Pi_{\text{tilde}, \text{numL}}(r_{\text{PAIX} \rightarrow 1}(L)) \otimes S$$

$$T_2 = \text{np.sum}(\text{t}_{\text{abs} - 9}^{\text{abs}}(T_1)) \approx M$$

$$E_3 = \Pi_{\text{ville}, \text{ville}} (\sigma_{\text{ville} \leftrightarrow \text{Paris}}, T_2)$$

On sait que : $\text{card}(L) = 2000$ $P(L) = 2000/10 = 200$
 $\text{card}(T) = 50$ $P(T) = 50/10 = 5$
 $\text{card}(S) = \text{card}(L) \times \text{card}(T) = 100000$
 $P(S) = 100000$

$$\text{card}(S \bowtie L) = \text{card}(S) = 100000$$

$$\text{card}(S \bowtie L \bowtie T) = \text{card}(S) = 100000$$

$$T_3 = \sigma_{qte=2}(S) \quad \text{coût}(T_3) = \text{card}(\sigma_{qte=2}(S)) = \frac{\text{card}(S)}{50} = 2000$$

$$R_3 = T_3 \bowtie L$$

nb qte distincts, supposé uniforme.

$$\text{coût}(R_3) = \text{coût}(T_3 \bowtie S) = \text{coût}(T_3) + \text{card}(T_3) \times \underbrace{\text{card}(\sigma_{numL=v}(S))}_{\text{card}(L) \times \frac{1}{numL}} \quad (v = \text{valeur quelconque})$$

On remarque l'index Livre(numL),
donc on peut appliquer T_1 .

$$\text{card}(L) \times \frac{1}{numL} = 1$$

$$\Rightarrow \text{coût}(R_3) = \text{coût}(T_3) + \text{card}(T_3) = 4000$$

Si on utilise T_4 : jointure par hachage.

$$\text{coût}(T_3 \bowtie L) = \text{coût}(T_3) + \text{coût}(L) = 2000 + P(L) = 2200$$

$$T_4 = \sigma_{prix=20}(L) \quad \text{coût}(T_4) = \text{card}(\sigma_{prix=20}(L)) = \frac{\text{card}(L)}{10} = 200 \quad (\text{on suppose prix uniforme})$$

$$R_4 = T_4 \bowtie S \quad \text{Idem, on remarque l'index Stock(numL).}$$

$$\text{coût}(R_4) = \text{coût}(T_4) + \text{card}(T_4) \times \text{card}(\sigma_{numL=v}(S))$$

$$= 200 + 200 \times \text{card}(S) \times \frac{1}{2000} \quad \text{nb de numL}$$

$$= 200 + 200 \times 50 = 10200$$

Jointure par hachage.

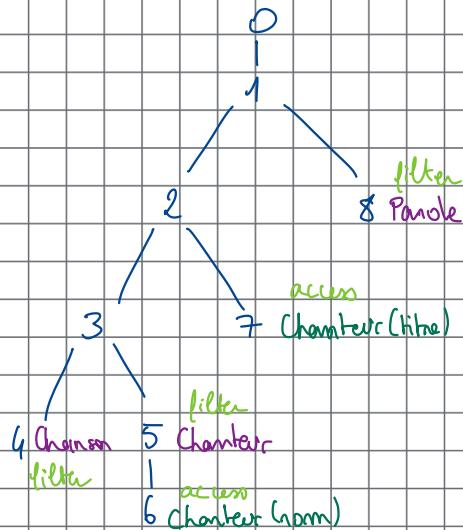
$$\text{coût}(R_4) = \text{coût}(T_4) + \text{coût}(S) = \text{coût}(T_4) + P(S) = 10200$$

Q2. $T = 50$ pages.

a)

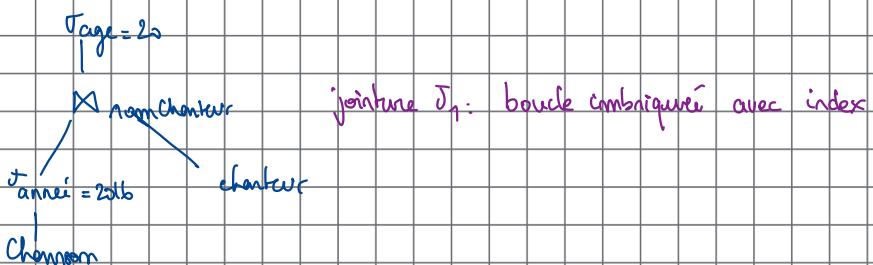
Exercice 2

Q1. Représenter l'arbre de P_1 .



b) Sélection $\text{âge} = 20$ effectuée en 5.

$\Rightarrow \text{dvalue}$ après les prédictats de jointure



TD4 - Optimisation de requêtes

Q1. a) predicates

id

1 filter (region = 'DF' and université in ('S', 'D'))

2 access (diplôme = 'Master Info')

b) on utilise l'index sur diplôme

for rowid in I-forma-diplome.getRowids('Master Info')

f-lineTuple(rowid)

if f.region = 'DF' and f.université in ('S', 'D')

affiche(f.NF, f.université)

Q2. list id1 = I-Etu-nf.getRowids('F1')

list id2 = I-Etu-nf.getRowids('Paris')

liste = liste_id1 ∩ liste_id2

for n in liste:

e = lineTuple(n)

if e.prenom like "A%"

affiche(e.prenom)

Q3. for e in Etudiant_tuples() table access full

id = I-forma-nf.getRowids(e.NF). index unique scan

f-lineTuple(id)

table access by row i

if f.diplôme = 'Licence':

affiche(e.prenom, f.region)

Q4. for n in I-forma-diplome.getRowids('Master') index range scan 1

f-lineTuple(n)

for n' in I-Etu-NF.getRowids(f.NF)

e = lineTuple(n')

index range scan 3

6

affiche(e.prenom, f.région)

Q5. D = {} cle : NF, valeur : liste de prénoms

For n in I-Etu-Réside.getRowids('Aix')

e = lineTuple(n)

if e.NF in D:

D[e.NF].append(e.prenom)

else

D[e.NF] = [e.prenom]

For n in I-Forma-Diplome.getRowids('M2'):

f = lineTuple(n)

for p in D[f.NF]:

affiche(p, f.université)

Q6.

- projection pour l'opération n°2 : jointure entre F et E
 $\rightarrow f.\text{université}, e.\text{prénom}, e.\text{mE}$

- projection pour l'opération n°3 : f.université, f.mF

a) pseudo-code :

dictionnaires

D_1 : clé : nF, valeur : université

D_2 : clé : mE, valeur : (université, prénom)

for f in Formation.tuples() : Table access full

$D_1[f.\text{mF}] = f.\text{université}$

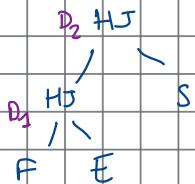
for e in Etudiant.tuples()

$D_2[e.\text{mE}] = (D_1[e.\text{nF}], e.\text{prénom})$

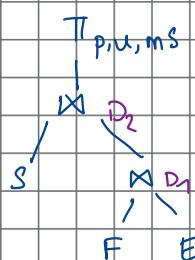
for s in Stage.tuples()

$u, p = D_2[s.\text{mE}]$,

affiche(p, u, s.ms)



b) Plan linéaire à droite : Stage \bowtie (Formation \bowtie Etudiant)



$D_1 : \text{clé} : \text{mF}, \text{valeur} : \text{université}$

$D_2 : \text{clé} : \text{mE}, \text{valeur} : \text{ms}$

for s in Stage.tuples() :

$D_2[s.\text{mE}] = s.\text{ms}$

for f in Formation.tuples() :

$D_1[f.\text{mF}] = \text{université}$

for e in Etudiant.tuples() :

$u = D_1[e.\text{mF}]$

$ms = D_2[e.\text{mE}]$

affiche(e.prénom, u, ms)

$D_1 : \text{clé} : \text{mF}, \text{valeur} : \text{université}$

$D_2 : \text{clé} : \text{mE}, \text{valeur} : \text{liste de ms} \quad (\text{car mE pas unique})$

for s in Stage.tuple() :

if s.mE in D2 :

$D2[s.\text{mE}] . \text{append}(s.\text{ms})$

else :

$D2[s.\text{mE}] = [s.\text{ms}]$

for f in Formation.tuple() :

$D1[f.\text{mF}] = \text{université}$

for e in Etudiants.tuple() :

$u = D1[e.\text{mF}]$

for ms in D2[e.mE] :

affiche(e.prénom, u, ms)

$D2.\text{get}(e.\text{mE}, [])$

Exercice 8

Q1. $SF_1 = \frac{5 \text{ valeurs}}{20} = 0,25$ $S_1 : \text{note} > 12 \wedge \text{note} \leq 17$

$SF_2 = \frac{4 \text{ valeurs}}{100} = 0,04$ $S_2 : \text{lieu im} (\dots)$

$SF_3 = \frac{1}{1000} \times \frac{1}{8} = \frac{1}{8000}$ (1 étudiant, 1 spécialité)

$SF_4 = \frac{(\text{durée} = 6 \text{ or } \text{durée} = 12)}{2 \text{ valeurs parmi 4}} \wedge \text{note} > 5$

$$= \frac{2}{4} + \frac{15}{20} - \frac{2}{4} \times \frac{15}{20}$$

15 valeurs parmi 20

on enlève le million

$$= \frac{25}{20} - \frac{15}{40}$$

$$= \frac{50 - 15}{40} = \frac{35}{40} = \frac{7}{8}$$

Q2. - $\text{card}(R_1) = SF(\tau_{\text{mention}=\beta}(\text{Etudiant})) \times SF(\tau_{\text{région}=\text{idf}_1}(\text{Formation}))$
 $\times \text{card}(\text{Formation})$

$$= \frac{1}{4} \times \frac{1}{5} \times \text{card}(\text{Etu}) \quad (\text{car jointure sur nF})$$

$$= \frac{1}{20} \times 1000 = 50$$

ou $R_1 = T_1 \bowtie T_2$ avec $T_1 = \tau_{\text{mention}=\beta}(\text{Etudiant})$
 $T_2 = \tau_{\text{région}=\text{idf}_1}(\text{Formation})$

$$\text{card}(T_1) = \frac{1}{4} \times 1000 = 250$$

$$\text{card}(T_2) = \frac{1}{5} \times 10 = 10$$

$$\text{card}(T_1 \bowtie T_2) = \text{card}(T_1) \times \text{card}(T_2) \times \frac{1}{\text{nombre de nF}}$$

$$= 250 \times 10 \times \frac{1}{50} = \frac{2500}{50} = 50$$

- R2. $\text{card}(R2) = \text{card}(\text{Stage}) = 2000$

ou $\text{card}(R2) = \text{card}((E \bowtie S) \bowtie F)$

$$= \text{card}(E \bowtie S) \times \text{card}(F) \times \frac{1}{\text{nombre de nF}}$$

$$= \text{card}(E) \times \text{card}(S) \times \frac{1}{m_E} \times \text{card}(F) \times \frac{1}{m_F}$$

$$\text{card}(R) = 1000 \times 2000 \times \frac{1}{1000} \times 50 \times \frac{1}{50}$$

$$= 2000$$

- $\text{card}(R3) = \text{card}(\text{Stage} \times_{nE} \text{Pratique Langue})$

$$= \text{card}(\text{Stage}) \times \text{card}(\text{Pratique Langue}) \times \frac{1}{mb \ nE}$$

$$= 2000 \times 300 \times \frac{1}{1000}$$

$$= 600$$

Q3. R3: Etudiant \times Formation

mf

$$\text{cout}(R3) = \text{cout}(E \times F) \text{ avec jointure NL et index } E, mf$$

$$= \text{cout}(\text{Etudiant}) + \text{card}(\text{Formation}) \times \text{cout}(\sigma_{nf=x}(\text{Formation}))$$

$$= P(E) + 1000 \times \underbrace{\text{card}(\sigma_{nf=x}(F))}_{1}$$

$$= \frac{1000}{10} + 1000 = 1100$$

$$\text{cout}(\sigma_{...}(E \times F)) = \text{cout}(E \times F)$$

Q4. réponses : P1: 21

P2: 600

P3: 120

$$R = \sigma_{e.mention=Tb \text{ and } s.moteur>19 \text{ and } s.lien=bal}(E \times S)$$

$$\sigma_{mention=Tb}$$

a) liaison à gauche



$$T1 = \sigma_{moteur > 19}(\sigma_{lien = bal}(\text{Stage}))$$

$$SF(\sigma_{moteur > 19}(S)) = \frac{1}{20},$$

$$SF(\sigma_{lien = bal}(S)) = \frac{1}{100}$$

$$\sigma_{moteur > 19}$$

$$\sigma_{lien = bal}$$

S

$$\text{card}(T1) = \text{card}(\sigma_{moteur > 19}(T2)) \text{ où } T2 = \sigma_{lien = bal}(S)$$

$$= SF(\sigma_{moteur > 19}(S)) \times \text{card}(T2) = \frac{1}{20} \times SF(\sigma_{lien = bal}(S)) \times \text{card}(S)$$

$$\text{cond}(T_1) = \frac{1}{20} \times \frac{1}{100} \times 2000 = 1$$

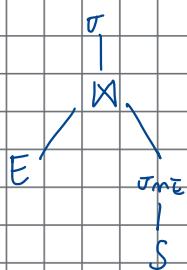
$$\text{cont}(T_1) = \text{cont}(\sigma_{\text{not} >_1} (T_2)) \\ = \text{cont}(T_2) = \text{cont}(\sigma_{\text{new_dot}} (\$))$$

$$\text{index } S(\text{lieu}) \quad d'out \quad \text{cont}(T_2) = \text{card}(T_2) = \frac{1}{100} \times 1000 = 10$$

Ainsi :

$$\begin{aligned}
 \text{cont}(P1) &= \text{cont}(\bigcap_{m \in E} T_1) = \text{cont}(T_1) + \text{card}(T_1) \times \text{cont}(\bigcap_{\substack{m \in E \\ m \neq r}} (E)) \\
 &= \text{cont}(T_1) + \text{card}(T_1) \\
 &= 1 + 20 = 21
 \end{aligned}$$

$$b) P_2 :$$



Nested Loops	
Nested Loops	
Table access full	Student
Index Unique scan	I_Scan
Table Access by RowId	Stage

$$\begin{aligned}
 \text{cont}(P_2) &= \text{cont}\left(E \underset{m}{\times} S\right) = \text{cont}(E) + \text{card}(S) \times \text{cont}\left(\sigma_{m \in m}(S)\right) \\
 &= P(E) + 2000 \times \text{cont}\left(\sigma_{m \in m}(S)\right) \\
 &= 100 + 2000 \times SF(m_E(j)) \times \text{card}(S) \\
 &= 100 + 2000 \times \frac{1}{1000} \times 2000 \\
 &= 100 ???
 \end{aligned}$$

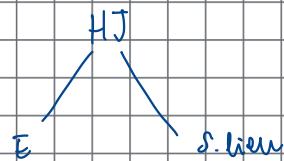
c) P3 : Plan :

Hash Join

Table access by row id
Index Range Scan
Table access full

Stage
I_Stage
Etudiant

access ($e.mE = s.mE$)
filter ($s.mot > 10$)
access ($s.lien = 'Dol'$)
filter ($e.mention = TB$)



$$\begin{aligned}
 \text{cost}(P3) &= \text{cost}(E) + \text{cost}(\sigma_{lien=Dol}(S)) \\
 &= P(E) + \text{card}(\sigma_{lien=Dol}(S)) \\
 &= P(E) + SF(\sigma_{lien=Dol}(S)) \times \text{card}(S) \\
 &= 100 + 2000 \times \frac{1}{100} \\
 &= 120
 \end{aligned}$$

5. $E \underset{mF}{\bowtie} F \underset{mE}{\bowtie} S \underset{mE}{\bowtie} P$

TD6

Exercice 1a - Conception de BD réparties

1. Garage_v = $\sigma_{ville=v}(\text{garage})$

→ fragmenter l'autre table avec cette table et pas joindre

$$\text{Habilite}_v = \text{Habilite} \times_{\text{idgarage}} \text{Garage}_v$$

$$\text{Mecanicien}_v = \text{Mecanicien} \times_{\text{idgarage}} \text{Garage}_v$$

$$\text{PersonneNecav}_v = \text{Personne} \times_{\text{idpers}} \text{Mecanicien}_v$$

(définir aussi Personne Client...)

2.

fragmentation horizontale dérivée disjointe
car Garage_v est une fragmentation disjointe et idGarage clé de Garage, idpers clé de Mecanicien.

$$\text{Reparation}_v = \text{Reparation} \times_{\text{idmecanicien} = \text{idpers}} \text{Mecanicien}_v \rightarrow \text{disjointe car idPers clé de Mecanicien}$$

termes 1 à 1 disjoint

Fragmenter la relation Possède ?

semi-jointure sur "manque" ? non si $\Pi_{\text{manque}}(\text{Habilite}_v) = \Pi_{\text{manque}}(\text{Habilite})$

$$\text{Possede}_v = \text{Possede} \times_{\text{immatt}} \text{Reparation}_v \rightarrow \text{pas disjointe car immat n'est pas une clé de Reparation.}$$

Possede_v n'est pas complet, il manque le véhicule jamais réparé.

$$\text{PossedeNeuf} = \text{Possede} - \cup_v \text{Possede}_v$$

$$\text{Client}_v = \text{Client} \times_{\text{idpers} = \text{idclient}} \text{Possede}_v \rightarrow \text{non disjoint car Possede, non disjoint}$$

$$\text{ClientNeuf} = \text{Client} - \cup_v \text{Client}_v$$

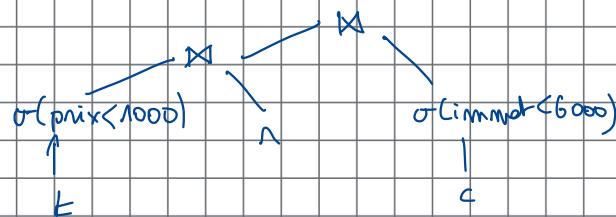
$$\text{PersonneClient}_v = \text{Personne} \times_{\text{v}} \text{Client}_v$$

$$\text{PersonneClientNeuf} = \text{Personne} - \cup_v \text{PersonneClient}_v - \cup_v \text{PersonneMecav}_v$$

$$\text{Tarif}_v = \text{Tarif} (\text{applique dans chaque ville})$$

Exercice 1b - Évaluation des requêtes réparties

2. $\pi_{\text{intervention}} \sigma_{prix < 100} (t) \times_{immat} \sigma_{immat < 6000} (p)$



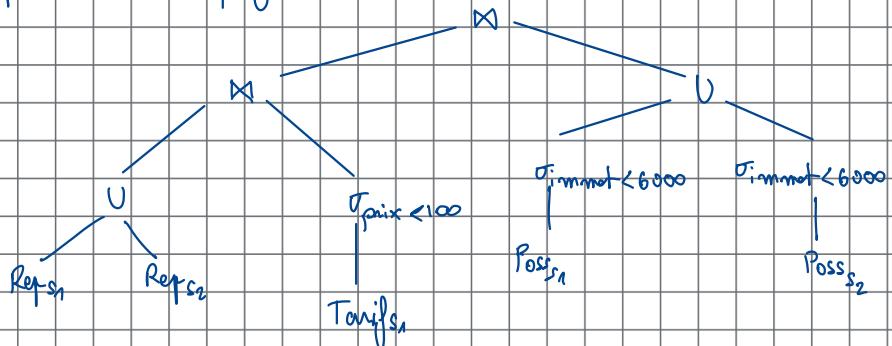
3. À partir des fragments on a

$$\text{Réparation}_0 = \text{Réparation}_{S_1} \cup \text{Réparation}_{S_2}$$

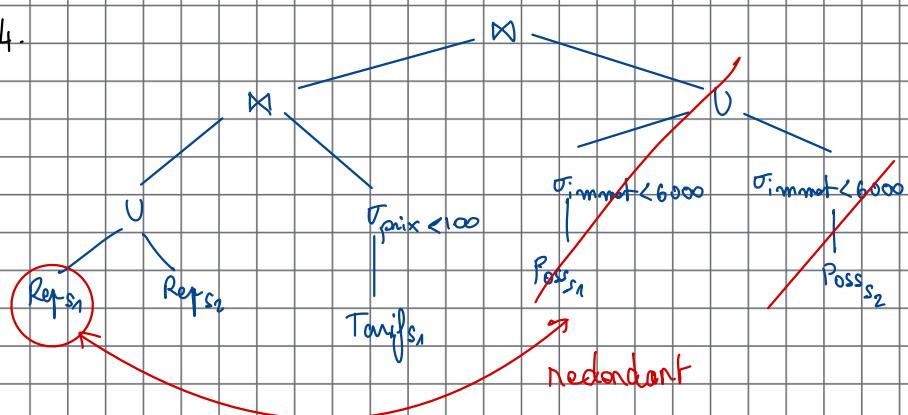
$$\text{Possède} = \text{Possède}_{S_1} \cup \text{Possède}_{S_2}$$

$$\text{Tarif} = \text{Tarif}_{S_1} - \text{Tarif}_{S_2}$$

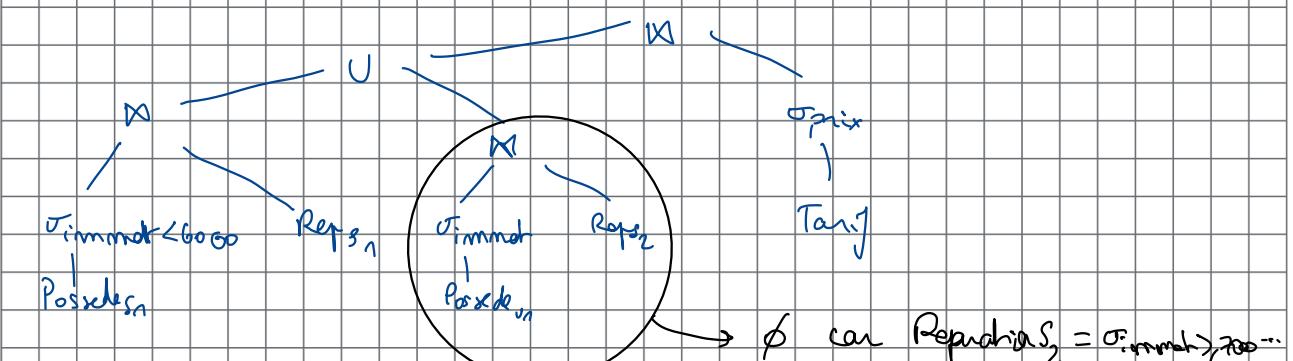
R exprimée sur les fragments



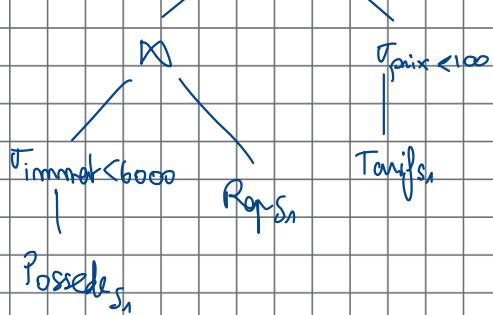
4.



$$\sigma_{\text{immobil} < 6000} (\text{Possède}_{S_2}) = \emptyset \text{ car } \text{Possède}_{S_2} = \sigma_{\text{immobil} > 7000} (P)$$

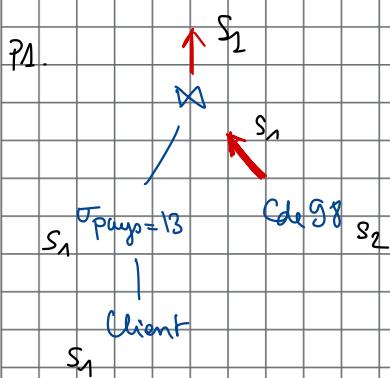


$$\sigma_{\text{immobil} < 6000} (\text{Possède}_{S_1}) \Rightarrow \text{Réparation}_{S_1} \Rightarrow \text{Tprix} < 100 \text{ (Tarif)}_S$$

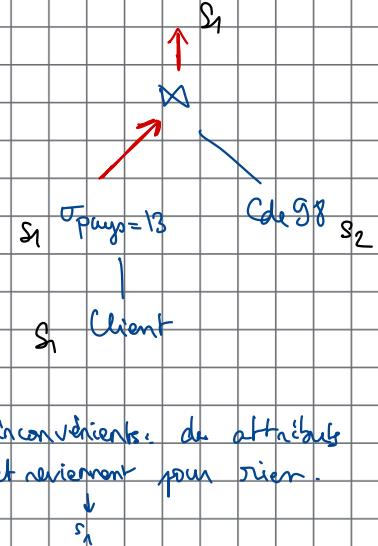


5. R posci sur S1 traité de la même façon de transfert.

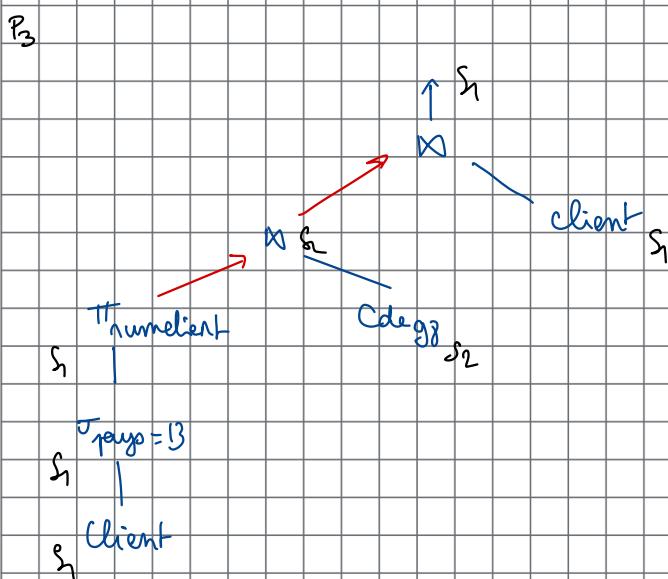
TP excl



inconvénient : des commandes inutiles sont transmises



inconvénient : des attributs de Client partent et reviennent pour rien.



Exercice 4 - Evaluation de requêtes réparties

$S_1 : R_1$

$S_2 : R_1, R_2$

$S_3 : R_3$

1. Taille d'un tuple de R_1 : $t(R_1) = 2 \times 10 = 20$ octets (2 attributs)

$$\text{card}(R_1) = P(R_1) \times \frac{4000}{t(R_1)} = 10^6 \times \frac{4000}{20} = 2 \cdot 10^5$$

$$\begin{aligned} \text{card}(T) &= \text{card}(R_1 \bowtie R_2) = \cancel{\text{card}(R_1)} \times \frac{1}{\cancel{20}} \times \text{card}(R_2) \\ &= \text{card}(R_2) \\ &= P(R_2) \times \frac{4000}{t(R_2)} = 10^5 \times 2 \cdot 10^2 \\ &= 2 \cdot 10^7 \end{aligned}$$

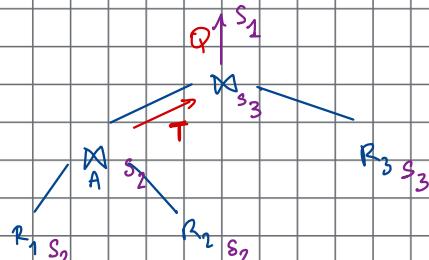
2. Taille d'un tuple de T : $t(T) = 30$ ($T(A, B, C)$)

$$\begin{aligned} \text{et } P(T) &= \text{card}(T) \times t(T) \\ &= 2 \cdot 10^7 \times \frac{4000}{1000} \\ &= \frac{3 \cdot 10^5}{2} = 150000 \end{aligned}$$

$$\begin{aligned} - \text{card}(Q) &= \text{card}(T \bowtie R_3) \\ &\quad \text{B} \\ &= \text{card}(T) = 2 \cdot 10^7 \end{aligned}$$

$$\begin{aligned} \text{et } P(Q) &= \text{card}(Q) \times \frac{4000}{6000} \quad \text{où } 60 = t(Q) \quad \text{car } Q(A, B, C, D) \\ &= 2 \cdot 10^5 = 200000 \end{aligned}$$

3.



Coût local, unité : t_{lo}
coût de transport : t_g

- coût(T) ?

par hachage : mi R_1 , mi R_2 tiennent en mémoire.

Line des blocs de 200 pages : on a $\frac{10000}{200} = 50$ blocs.

on veut répartir les données de R_1 en 50 paquets de 200 pages en utilisant une fonction de hachage h

$\rightarrow P(R_1)$ écriture pour répartir R_1

$$\text{coût}(T) = 9 \lfloor \log_{201} (P(R_2)) \rfloor (P(R_1) + P(R_2)) + P(R_1) + P(R_2)$$

Line R_2 par blocs de 200 pages pour la répartition en fonction de R
Rmq les paquets de R_2 font $\frac{100\ 000}{50} = 2000$ pages
 $\Rightarrow P(R_2)$ écritas

Jointure entre les parties de paquets ayant le même numéro de paquet.
 $\Rightarrow P(R_1) + P(R_2)$

Total : coût (T) = $(3(P(R_1) + P(R_2))) t_{io}$

- par tri-fusion

- Line et trier R_1 en 50 blocs de 200 pages : $2 \times P(R_1)$
- Line et trier R_2 en $\frac{100\ 000}{200} = 500$ blocs de 200 pages : $2 \times P(R_2)$

on $50 + 500 > 200$ donc il faut fusionner R_2 avant de commencer la jointure par fusion.

- Fusion de blocs de R_2 en $\frac{500}{3} = 3$ blocs : $2 \times P(R_2)$

Nombre de blocs restants : $R_1 : 50$

$R_2 : 3$

$53 < 200$ donc on peut faire la jointure

- jointure par fusion : $P(R_1) + P(R_2)$

Total : $(3P(R_1) + 5P(R_2)) t_{io}$

- Transférer T sur S_3 : $P(T) \times t_s = 150\ 000 t_s$

- coût (Q) ? par hachage externe

line R_3 et répartition en $\frac{100\ 000}{200} = 50$ blocs = $2P(R_3)$

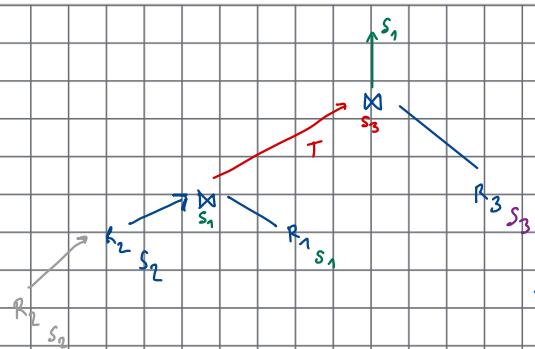
répartition T (provenant du site S_2) en 50 blocs : $P(T)$

jointure : $P(R_3) + P(T)$

Total : $(3P(R_3) + P(T)) t_{io}$

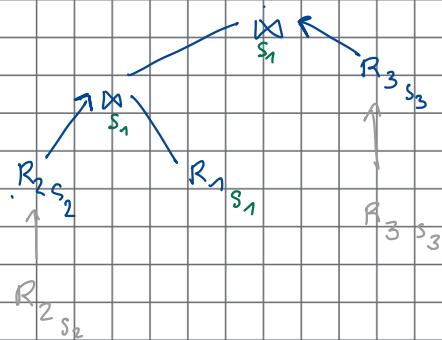
- transférer Q : $P(Q)t_s = 200\ 000 t_s$

4.



$$\text{Transferj: } P(R_2) + P(T) + P(S_3)$$

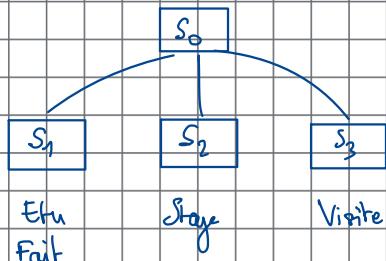
5.



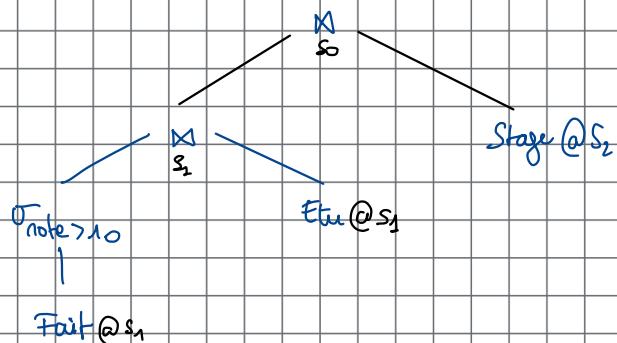
$$\text{Transferj: } P(R_2) + P(Q_3)$$

JDBC

Exercice 1 : Requêtes réparties avec JDBC



Q1. $A_1 : \exists \text{nom, lieu} (\exists \text{note} > 10 (\text{Fait} \bowtie_{\text{ME}} \text{Etu} \bowtie_{\text{NS}} \text{Stage}))$



$s_1 = c_1. createStatement();$

$res1 = s1. executeQuery (" select c.nom, f.mS
from Fait f, Etude e
where f.mE = e.mE
and f.note > 10 ");$

$p2 = c2. prepareStatement (" select lieu
from Stage
where mS = ? ");$

```

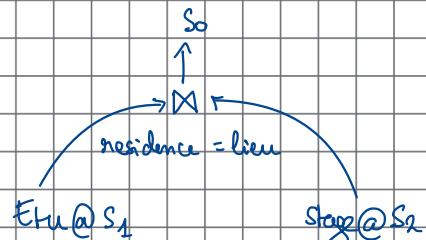
while ( res1. next() ) {
    p2. setInt (1, res1. getString ("f.mS"));
    res2 = p2. executeQuery();
    res2. next() // résultat unique
    System. out. print ( res1. getString ("c.nom") +
        res1. getString ("lieu"));
}
  
```

}

Q2. Select lieu, COUNT(*) as nbEtu
 From FaIt, Stage
 Where f.residence = s.lieu
 and s.duree = 6
 GROUP BY lieu
 HAVING COUNT(*) > 10
 ORDER BY lieu ASC

S_1 : Etu (mE, residence)
 FaIt (mE, ms)

S_2 : Stage (ms, lieu, duree)



n1 = c1.prepareStatement ("select count(*) as nbEtu
 from Etu e
 where residence = ?")

n2 = c2.createStatement();

res2 = n2.executeQuery("select * from Stage where duree = 6 order by lieu");

while (res2.next())

n1.setString(1, res2.getString("lieu"));

res1 = g1.executeQuery();

res1.next();

if (res1.getInt("nbEtu") > 10){

System.out.println(res2.getString("lieu") + " " + res1.getInt("nbEtu"));

}

}

sinon : n1 = ... "select *
 from (select count(*) as nbEtu
 from Etu
 where residence = ?)
 where nbEtu > 10"

test sur le if {}

on envoie (si on commence par S_1):

"select e.residence, count(*) as nbEtu
 from Etu
 group by residence
 having count(*) > 10
 order by residence"

"select 1
 from Stage
 where lieu = ? and duree = 6"

Q2. Requête globale :

```
Select duree, avg(mote)
from Fait fr Stage s
where f.mS = s.mS
and s.lien = "Paris"
group by duree
```

- site S_2 : obtenir les durées distinctes

```
"select distinct duree
from Stage
where lieu = "Paris"
```

- site S_1 : obtenir les ms qui ont la bonne durée

```
"select ms
from Stage
where duree = ?"
```

- site S_1 :

```
"select sum(mote) as S, count(*) as C
from Fait
where mS = ?"
```

- calculer la moyenne dans l'application ($\frac{S}{C}$ = moyenne pondérée)

Algo :

for r_1 in R_1 :

$T_S = 0$

$T_C = 0$

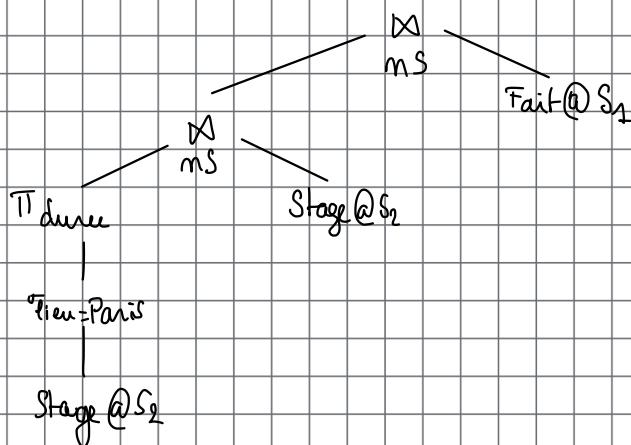
for r_2 in $R_2(r_1.duree)$:

$s, c = R_2(r_2.ms)$

$T_S += s$

$T_C += c$

print($r_1.duree, T_S/T_C$)



Q4. select distinct f.mE
 from Stage s, Fait f, Visite v
 where e.mE = nr.mE
 and s.ms = f.ms
 and s.duree = 3
 and v.ville = 'Aix'

$r_3 = c_3.createStatement$

$r_3 = r_3.executeQuery("Select mE from Visite where ville = 'Aix'")$

$r_1 = c_1.prepareStatement("select distinct mE, ms from Fait where mE = ?")$
 $r_1.setString(1, r_3.getString("mE"))$
 $r_1.executeUpdate()$

$r_2 = c_2.prepareStatement("select 1 from Stage where ns = ? and duree = 3")$
 $r_2.setString(1, r_1.getString("ms"))$
 $r_2.executeUpdate()$

Site 3 : obtenir les mE qui ont visité 'Aix'
 "select mE" ← unique, pas besoin de distinct
 R_3
 from Visite
 where ville = 'Aix')

Site 1 : obtenir les ms qui ont fait des stages

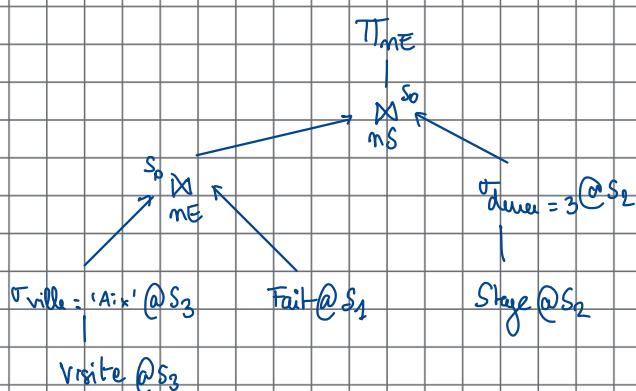
"select ms
 from Fait
 where mE = ?"

Site 2 : obtenir les mE qui ont fait un stage de 3 mois

"select "oui"
 R_2
 from Stage
 where ms = ? and duree = 3"

Algo : for mE in R_3 :

```
for ms in  $R_1[mE]$ :
    if  $R_2[ms] = "oui"$ :
        affiche(mE)
        break
```

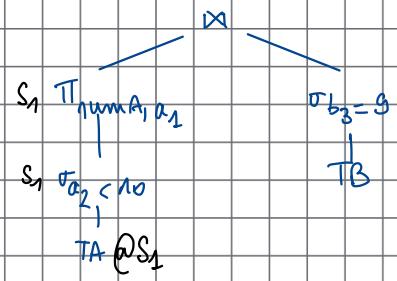


Ex 2 p12

Q1. requête sur S1 : "Select numA, an
From TA
Where a2 < 10"

requête sur S2 : "select "ok"
From TB
Where numA = ? and
b3 = ?"

Q2. requête sur S1 : "select a.a2 from TA"



requête sur S2 : - "select b4, numbB
From TB
where b4 < ?"

- "select b4, numbB
From TB
where b3 < ?"

"select b4, numbB
From TB
where b4 < ?
b3 < ?"

je sais pas si c'est une bonne idée de décomposer... jsp si bl not im? marche
mais je pense plus

requête sur S3 : "select c5
from TC
where numbB = ?"

```
while (n1.next())
    n2.setInt(1, R1["a2"])
    while (n2.next())
        n3.setInt(1, R2["numB"])
        while (n3.next())
            n2.setInt(2, R3["c5"])
```

R1: select "ok"
from TA
where a2 = ?

for n2 in R2:
 if R1(n2.b4) != "ok":
 for n2 in R3(n2, numbB, n2.b3)
 if n3(c5)

R2: Select numbB, b3, b4
From TB

je suis perplexe

R3: select c5
from TC
where numbB = ? and c5 > ?

when I close my eyes, I am an aquanaut

Q3. solution non réursive :

R_1 : select numA, a1 from TA where a2 = 0

R_2 : select b3 from TB where b1 = 0 and numA = ?

for $n1$ in R_1 :

 for $n2$ in R_2 ($n1$.numA):

 affiche ($n1$.numA, $n1$.a1, $n2$.b3)

inconvénient : l'application lit tous les tuples $n1$ y compris ceux qui ne sont pas dans le résultat.

Réponse : R_1 : select numA from TA where a2 = 0

R_2 : idem

R_3 : select a1 from TA where numA = ?

for numA in R_1 :

 for $n2$ in R_2 (numA)

 afficher (numA, R_3 (numA), $n2$.b3)

Q4 $R_1 @ S_2$: select b4, numB

from TB

order by b4

$R_2 @ S_3$: select c6, numB

from TC

order by c6

for $n1$ in R_1 :

 while ($n1.b4$) $n2.c6$:

$n2.next()$

 while ($n1.b4$ == $n2.c6$)

 afficher()

$n2.next()$

Exercice 1 - Transactions à large échelle avec Calvin

Q1. Transactions locales: T_1, T_5, T_6

Transactions globales: T_2, T_3, T_4, T_7

inépendant de la machine qui reçoit la transaction

Q2. a) $M_1: T_5, T_6, T_3$

$M_2: T_2, T_7$

$M_3: T_4, T_2$

$M_4: T_7, T_1, T_3, T_5$

On considère les transactions reçues sur M_1 puis M_2, M_3, M_4

→ sur M_1 , T_4 sera traité par Π_3 et Π_4

T_5 sera traité par Π_1

→ sur M_2 , T_2 sera traitée par Π_2 et Π_3

T_6 _____ Π_1

→ sur M_3 , T_4 _____ Π_4

T_7 _____ Π_2 et Π_4

→ sur M_4 , T_3 _____ Π_1 et Π_4

b) N'importe pas l'ordre des transactions.

Pour T_7 sur M_4 : lire G et l'envoyer à Π_2 , mais inutile de traiter T_7 sur Π_4 .

Q3. On suppose que pour T_7 : Lect(D), Ecr(D)

Lect(G)

pour les autres transactions: les données sont lues et écrites.

sur M_1 : [exécuter T_5

[exécuter T_6

[envoyer B à M_4

T_3 [recevoir G de Π_4

[exécuter T_3

sur M_2 : [envoyer C à Π_3

T_2 [recevoir E de Π_3

[exécuter T_3

T_7 [ne PAS envoyer D sur Π_4

[recevoir F de Π_4

[exécuter T_7

sur Π_3 : [envoyer E à Π_4

T_4 [recevoir G de M_4

[exécuter T_4

[envoyer F sur Π_2

T_2 [recevoir de Π_2

[exécuter T_2

sur Π_4 : [envoyer G à M_2

T_4 [recevoir E de Π_3

[exécuter T_4

T_2 [exécuter T_4

T_7 [envoyer G à M_2

inutile de recevoir D ou d'exécuter T_7

[envoyer G à Π_3

[recevoir B de Π_3

[exécuter T_3

Q4 (supplémentaire). T_2 : Ecr(C)

Lec(E) Ecr(F)

La donnée C n'est pas lue, donc inutile pour M_2 d'envoyer C vers M_3 .

M_2 : recevoir E de Π_3
exécuter E de Π_3

M_3 : envoyer E sur Π_2
exécuter T_2

Autre exemple :

T_7 Lect(D) En(D)
Lect(G)

T_7'

$M_1: T_5, T_6, T_3$
 $M_2: T_2, T_7, T_7'$
 $M_3: T_4, T_2$
 $M_4: T_4, T_1, T_7, T_3, T_7'$

sur M_1 :
 $T_4: [$ envoyer G à M_2
recevoir E de M_3
exécuter T_4
 T_2 exécuter T_1
 $T_7: [$ envoyer G à M_2 initial de recevoir D en
exécuter T_7
 $T_3: [$ envoyer G à M_2
recevoir B de M_1
exécuter T_3

sur M_2 :
 $T_2: [$ envoyer C à M_3
recevoir E de M_3
exécuter T_3
 $T_7: [$ me PAS envoyer D sur M_1
recevoir G de M_4
exécuter T_7
exécuter T_7'

Exercice 10 p10

Le joueurs sont répartis sur 15 machines en "round robin".
à tour de rôle

(Q1) joueur le mieux payé?

• sur M_i : select *
from J_i
where salaire = (select max(salaire) from J_i)

envoyer le résultat à M_0

• sur M_0 : union des résultats reçus de M_i s et sélectionner le joueur avec max(salaire).

(Q2) Fragmentation horizon par intervalle de salaire sur M_{15} : m requêtes