Master d'informatique- M1

UE 4IN803

SAM : Stockage et Accès aux Mégadonnées

Support de TD - Partie 2

année 2023

Hubert NAACKE

Exercice 1a (ref 1-17): Conception de BD réparties pts

Soit la base de données AutoRoul d'une chaîne de garages automobiles qui contient les tables suivantes :

Personne (<u>idpers</u>, nom, prenom, age, tél) /* un mécanicien ou un client */
Garage (<u>idgarage</u>, nom, ville, jourdefermeture)
Habilite (<u>idgarage, marque</u>) /* le garage peut réparer des véhicules de la marque */
Mecanicien (<u>idpers</u>, idgarage, niveau) /* un mécanicien travaille dans un garage */
Client (<u>idpers</u>, taille)
Possede (<u>immat</u>, marque, modele, idclient) /* marque, modèle et immatriculation du véhicule possédé */
Reparation (<u>idmecanicien, immat, date,</u> intervention) /* immat est un entier entre 1 et 10000 */
Tarif (<u>intervention</u>, prix)

Question 1. Il y a un site informatique par ville. L'allocation des fragments se fait donc selon la ville. Par exemple, Personne_v représente le fragment de la relation Personne stocké sur le site S_v de la ville v.
Définir les fragments de la base de données AutoRoul et leur allocation. Personne_v, Garage_v...

Question 2. La fragmentation que vous proposez est-elle disjointe ? Sinon, quelles sont les tables dont la fragmentation n'est pas disjointe ?

Question 3. On suppose maintenant qu'il y a un site informatique par marque, et non plus par ville. L'allocation des fragments se fait donc selon la marque. Par exemple, Client_m représente le fragment de la relation Client stocké sur le site S_m de la marque m.
Compléter le tableau suivant afin de définir les fragments de la base de données AutoRoul et leur allocation.

Exercice 1b (ref 2-17) : Evaluation de requêtes réparties pts

Soit la requête **R** suivante sur la base AutoRoul de l'exercice 1.

```
Select p.marque
from Reparation r, Tarif t, Possede p
where r.intervention = t.intervention and t.prix < 100 and r.immat = p.immat
and p.immat < 6000;
```

Question 1. Exprimer en français la requête R

Question 2. Donner l'expression algébrique de la requête R telle que les sélections sont effectués avant les opérateurs binaires quand c'est possible (dans tout l'exercice, on ne se préoccupe pas des projections). Joindre d'abord les réparations et les tarifs, puis joindre le résultat avec Possede.

On suppose maintenant que les relations sont réparties sur deux sites, S1 et S2 tel que $\textbf{Table}_{\textbf{Si}}$ est le fragment de Table alloué au site Si. On a :

$\textbf{Possede}_{\textbf{S1}} = \sigma_{\text{immat}<7000}\ \text{Personne}$ et $\textbf{Possede}_{\textbf{S2}} = \text{Personne} - \text{Personne}_{S1}$.

$\textbf{Reparation}_{\textbf{S1}} = \sigma_{\text{immat}<7000}\ \text{Reparation}$ et $\textbf{Reparation}_{\textbf{S2}} = \text{Reparation} - \text{Reparations}_{S1}$.

$\textbf{Tarif}_{\textbf{S1}} = \textbf{Tarif}_{\textbf{S2}} = \textbf{Tarif} = \textbf{Tarif}_{\textbf{S1}\vee\textbf{S2}}$

Question 3. Exprimer la requête R sur les fragments ainsi définis. Pour cela, reprendre l'expression de la question 2, remplacer les relations par leur expression en termes de fragments, puis appliquer les sélections le plus tôt possible.

Question 4. Appliquer la distributivité de la jointure sur l'union puis supprimer à nouveau le(s) sous-arbre(s) inutile(s).

Question 5. On suppose que la requête est posée sur le site S1. Quelles données doivent être transférées sur le réseau pour exécuter le plan d'exécution trouvé à la question 4. Que peut-on en conclure sur l'efficacité des simplifications faites aux questions 3 et 4 ?

Exercice 2a (ref 1-16) Conception de BD réparties pts

Soit la base de données CoifPlus d'une chaîne de salons de coiffure. La chaîne gère plusieurs marques, dans plusieurs villes :

Personne (idpers, nom, prenom, age, couleurcheveux, tél) /* un coiffeur ou un client */
Salon (idsalon, marque, ville, surface, jourdefermeture)
Coiffeur (idpers, idsalon, niveau) /* un coiffeur travaille dans un seul salon */
Client (idpers, coupefavorite)
RDV (idcoiffeur,idclient, jour, heure, coupe) /* rendez-vous passés et futurs */
Tarif (coupe,marque, prixbase)
Promo (heure, coef) /* coef à appliquer au prixbase selon l'heure */

Chaque marque possède un site informatique dans chaque ville, qui doit stocker les données nécessaires aux salons de cette marque dans la ville. L'allocation des fragments se fait donc selon la marque et la ville. Par exemple, $\text{Personne}_{m,v}$ représente le fragment de la relation Personne stocké sur le site $S_{m,v}$ de la marque m dans la ville v.

Question 1. Compléter le tableau suivant afin de définir les fragments de la base de données CoifPlus et leur allocation. On précise qu'un site doit stocker non seulement les coiffeurs qui travaillent dans les salons correspondants, mais aussi tous les coiffeurs de la ville correspondante car ils peuvent être appelés d'urgence pour un remplacement imprévu.

Question 2. La fragmentation que vous proposez est-elle disjointe ? Sinon, quelles sont les tables dont la fragmentation n'est pas disjointe ?

Exercice 2b (ref 2 – 16) Evaluation de requêtes réparties pts

Soit la requête **R** suivante sur la base CoifPlus de l'exercice 1.

```
select coif.nom, coif.prénom,
from Personne coif, RDV r, Personne cli
where coif.idpers=r.idcoiffeur and r.idclient=cli.idpers and coif.age > cli.age
and coif.couleurcheveux = 'brun' and r.heure = 15 and cli.age > 45;
```

Question 1. Exprimer en français la requête R

Question 2. Donner l'expression algébrique de la requête R telle que les sélections sont effectués avant les opérateurs binaires quand c'est possible (dans tout l'exercice, on ne se préoccupe pas des projections). Joindre d'abord les clients et les RDVs, puis joindre le résultat avec les coiffeurs.

On suppose maintenant que la relation Personne est répartie sur deux sites, S1 et S2 tel que $\textbf{Personne}_{\textbf{Si}}$ est le fragment de Personne alloué au site Si. On a :

$\textbf{Personne}_{\textbf{S1}} = \sigma_{age<30}\,\text{Personne}$ et $\textbf{Personne}_{\textbf{S2}} = \text{Personne} - \text{Personne}_{S1}$.

De même, on a $\textbf{RDV}_{\textbf{S1}} = \sigma_{heure<12}\,\text{RDV}$ et $\textbf{RDV}_{\textbf{S2}} = \text{RDV} - \text{RDV}_{S1}$.

Question 3. Exprimer la requête R sur les fragments ainsi définis. Répondre en **dessinant un arbre**

Question 4. Appliquer la distributivité de la jointure sur l'union puis supprimer à nouveau le(s) sous-arbre(s) inutile(s). Répondre en dessinant l'arbre obtenu.

Question 5. On suppose que la requête est posée sur le site S1. Quelles données doivent être transférées sur le réseau pour exécuter le plan d'exécution trouvé à la question 4. Que peut-on en conclure sur l'efficacité des simplifications faites aux questions 3 et 4 ?

Exercice 3 (ref 1-15) : Conception de bases de données réparties

EDF maintient une base de données pour surveiller et gérer la consommation d'électricité en Île de France. Le schéma est composé de 4 tables (les clés primaires sont soulignées) :

- **Client** (idCli, nom, adresse, departement, idCompt)
- **Consommation** (idCompt, annee, trimestre, consKWh)
- **Tarif** (annee, trimestre, tarifKWH, tarifAbo)
- **Facture** (idFacture, idClient, annee, trimestre, somme)

La table Client contient pour chaque client un identifiant, son nom, son adresse, son département et l'identifiant du compteur installé chez le client (il y a au maximum un client par compteur). L'historique de la consommation de chaque compteur est stocké dans la table Consommation qui contient pour chaque trimestre d'une année la consommation en KWh (trimestre 1 = janvier à mars, trimestre 2= avril à juin, ... trimestre 4= octobre à décembre). Les tarifs d'abonnement et de consommation par KWh valables pour chaque trimestre sont stockés dans la table Tarif. Les factures sont générées et stockées dans la table Facture à la fin de chaque trimestre. Le montant à payer est la somme consKWh*tarifKWh + tarifAbo pour le compteur et le trimestre correspondant.

La base de données est répartie sur 3 sites : l'un (PA) correspond à Paris, c'est-à-dire au département 75, le deuxième (PC) à la petite couronne qui comporte les départements 92, 93 et 94, le troisième (GC) à la grande couronne et comporte les départements 91, 77, 78 et 95.

Donner la définition des différents fragments ainsi que leur allocation, en utilisant les opérateurs de l'algèbre relationnelle. Pour allouer un fragment, il suffit de l'indicer en utilisant la variable *i* dont la valeur est dans {PA, PC, GC}. Justifier vos réponses.
EDF décide d'envoyer des informations publicitaires à ses abonnées par courrier postal. C'est le site PA qui gère ces envois pour Paris, alors que le site PC gère les envois pour tous les autres départements. Quel(s) fragment(s) faut-il ajouter à la fragmentation précédente ? Utiliser les fragments déjà existant pour définir le ou les nouveaux fragments.

Exercice 4 (ER2-19) : Evaluation de requêtes réparties

Soient $\mathbf{R_1}$(A, B), $\mathbf{R_2}$(A, C), $\mathbf{R_3}$(B, D) trois tables.

Les clés des relations sont soulignées, de plus R_2.A est une clé étrangère faisant référence à R_1.A, R_1.B est une clé étrangère faisant référence à R_3.B. Il y a trois sites S_1, S_2, S_3.

R_1 est répliquée sur S_1 et S_2,
R_2 est sur S_2,
R_3 est sur S_3.

Les trois sites ont une mémoire principale de 201 pages. La taille d'une page est de 4000 octets.
Le nombre de pages de chaque relation est le suivant :

$P(R_1) = 10\,000$
$P(R_2) = 100\,000$
$P(R_3) = 10\,000$

La taille de chaque attribut est de 10 octets. On note card(R) le nombre de n-uplets de R. **On considère 2 algorithmes de jointure** : tri-fusion externe et hachage externe

On désire calculer la requête $Q = (R_1 \bowtie R_2) \bowtie R_3$, posée sur S_1. On note $T = R_1 \bowtie R_2$.

On note t_{io} le coût de lecture/écriture d'une page sur le disque et t_s le coût de transfert d'une page sur le réseau.

Question 1. Calculer Card(R_1), puis Card(T) en expliquant vos calculs

Question 2 Calculer P(T), puis P(Q) en expliquant vos calculs

Question 3 Calculer en t_{io} et t_s le coût du plan P1 suivant pour exécuter Q en expliquant vos calculs

1.	Calcul de T sur S2	
2.	Transfert de T sur S3	
3.	Calcul de Q sur S3	
4.	Transfert de Q sur S1	Total P1

Question 4 Calculer en t_{io} et t_s le coût du plan P2 suivant pour exécuter Q en expliquant vos calculs

1.	Lire R2 sur S2 pour le transférer sur S1
2.	Calculer T sur S1
3.	Transférer T sur S3
4.	Calculer Q sur S3
5.	Transférer Q sur S1
Total P2 = .. t_{io} + …… t_s	

Question 5 Calculer en t_{io} et t_s le coût du plan P3 suivant pour exécuter Q en expliquant vos calculs

1.	Lire R2 et le transférer vers S1	
2.	Calculer T sur S1	
3.	Lire R3 sur S3 et le transférer vers S1	
4.	Calculer Q sur S1	Total P3

Exercice 5 (ref 3-15) : Optimisation de requêtes réparties

Soit la base de données répartie sur les sites S1 et S2 telle que :

Sportif (ids, nom) est sur S2,

Performance (ids, comp, classement) est sur S1.

L'attribut *comp* identifie une compétition sportive auquel un sportif obtient un classement. Seuls 10% des sportifs de la base ont déjà participé à une compétition, les autres ne pratiquent le sport qu'en tant que loisir. On suppose que la distribution des valeurs des attributs est uniforme.

Soit t(attr) la taille en octets d'un attribut, on a :

t(ids) = 5, t(nom) = 20, t(comp) = 10, t(classement) = 10

Soit **T**(R) la taille d'une relation R, exprimée nombre de **pages**.

On pose T(Sportif) = N.

La valeur de T(Performance) n'est pas connue.

Soit Card(R) la cardinalité d'une relation, on a Card(Sportif) = 10 000, Card(Performance)=100 000.

Le **coût** d'un plan d'exécution d'une requête est composé d'un coût de transfert (unité t_t) et d'un coût d'accès au disque (unité t_{io}), tous deux en nombre de pages.

Sportif (respectivement Performance) est indexée par un arbre-B+ de hauteur 2 (respectivement 3) sur ids. Le coût d'accès à un n-uplet de Sportif connaissant la valeur de ids est donc de 2 t_{io} (respectivement 3 t_{io}). On suppose que tous les nuplets de Performance correspondant à une valeur de ids tiennent dans une seule page.

Soit la requête **R1** posée sur S3 (le résultat doit être délivré sur S3)

R1 :
```
select s.ids, nom, comp, classement
from Sportif s, Performance p
where p.ids = s.ids ;
```

On considère le plan d'exécution de R1 suivant :

P1 : transférer Sportif sur S1, faire la jointure sur S1 par boucle imbriquée en utilisant l'index puis transférer le résultat sur S3.

a) Que vaut T(Performance) en fonction de N ?

b) Quelle est la taille d'une page en octets, en fonction de N ?

c) Que vaut Card(R1) ? Que vaut T(R1) en fonction de N ? Rappel : T(R1) est exprimée en nombre de pages.

d) Détailler le plan P1 en calculant les coûts (t_t et t_{io}) à chaque étape. Certains coûts peuvent être fonction de N.

Exercice 6 (ref 1-01) : Conception de BD réparties

Le but de ce TD est l'étude de la conception d'une base de données répartie. Dans la première partie, une base dont le schéma conceptuel a été défini est distribuée sur plusieurs sites. Le but principal de la distribution est de maximiser les accès locaux par rapport aux accès répartis. Dans la seconde partie, une base préexistante est intégrée au système. Dans ce cas, l'intégration doit être réalisée sans modification de la base locale qui possède ses propres applications locales déjà écrites, sous la forme d'une vue répartie. La troisième partie est une étude quantitative du coût de traitements des requêtes sur une BD répartie. Enfin, la dernière partie est une étude de cas.

Schéma global de la base

La base de données hospitalière de la région Alsace a le schéma suivant :

Service (Snum, nom, hôpital, bât, directeur)

le directeur d'un service est un docteur désigné par son numéro

Salle (Snum, SAnum, surveillant, nbLits)

le numéro de salle est local à un service, i.e., il peut y avoir des salles avec le même numéro dans des services différents d'un même hôpital.

nbLits est le nombre total de lits d'une salle,

un surveillant de salle est un infirmier désigné par son numéro

Employé (Enum, nom, adr, tél)

Docteur (Dnum, spéc) -- *spéc est la spécialité du médecin*

Infirmier (Inum, Snum, rotation, salaire)

Un employé est soit infirmier soit docteur (Inum et Dnum font référence à Enum).

Patient (Pnum, Snum, SAnum, lit, nom, adr, tél, mutuelle, pc)

L'attribut pc est la prise en charge par la mutuelle

Acte (Dnum, Pnum, date, description, coef) -- *coef est le coefficient de l'acte médical*

Question 1

Exprimer en SQL la question suivante: "Donner le nom des cardiologues qui ont traité un ou plusieurs patients hospitalisés dans un service de gérontologie."

Répartition des données

La base est répartie sur trois sites informatiques, "Strasbourg", "Colmar" et "Régional", correspondant aux valeurs "*Ambroise Paré*", "*Colmar*" et "*autre"* de l'attribut *hôpital* de Service.

Question 2

Proposer (et justifier) une bonne décomposition de la base hospitalière sur ces trois sites. On pourra utiliser la fragmentation horizontale et/ou verticale ainsi que la réplication des données, en se basant sur les hypothèses suivantes (H1 à H5) :

• H1: Les sites Strasbourg et Colmar ne gèrent que les hôpitaux correspondants.

• H2 : Les infirmiers sont employés dans un service donné.

• H3 : Les docteurs travaillent le plus souvent sur plusieurs hôpitaux (ou cliniques).

• H4 : La gestion des lits d'hôpitaux est locale à chaque hôpital.

• H5 : On désire regrouper la gestion des frais d'hospitalisation au centre régional.

Pour chaque fragment, on donnera sa définition en algèbre relationnelle à partir du schéma global.

Question 3

Indiquer comment se calcule chaque relation de la base globale à partir de ses fragments.

Question 4

Proposer un plan d'exécution réparti pour la requête SQL vue en Question 1, sachant maintenant que les données sont réparties sur les trois sites selon la décomposition proposée à la Question 2.

Question 5 : Conception de BD fédérée par intégration

On suppose que l'hôpital de Belfort est rattaché à la base de données hospitalière de la région Alsace après son implémentation répartie. L'hôpital de Belfort possède donc son propre site de traitement qui doit être connecté aux autres sites.

Le schéma de la base à Belfort avant l'intégration est le suivant:

B_Service (Snum, Nom, Bâtiment, Directeur)

B_Salle (Snum, SAnum, Surveillant, NbLits)

B_Docteur (Dnum, Nom, Adresse, Téléphone, spécialité)

B_Infirmier (Inum, Nom, Adresse, Téléphone, Snum, Salaire)

B_Patient (Pnum, Snum, SAnum, Lit, Nom, Adresse, Téléphone, Mutuelle, PriseEnCharge)

B_Acte (Dnum, Pnum, Date, Description, Code)

Discuter les problèmes et proposer des solutions pour l'intégration de la base Belfort au système réparti déjà défini. L'intégration devra se faire sans transfert d'information et sans modification de des bases existantes, mais uniquement par définition de vues.

Question 6

Définir le nouveau schéma global intégrant la base Belfort.

Chaque relation du schéma global (Service2, Salle2, …Acte2) est définie en fonction des fragments sur les 4 sites.

Question 7

L'hypothèse H5 est-t-elle toujours respectée après l'intégration de la base Belfort ?

Si non, quelles sont les modifications de schéma nécessaires pour respecter H5 ?

Question 8

Proposer une décomposition et un plan d'exécution pour la question SQL précédente après l'intégration de la base "Belfort".

Exercice 7 (ref 1-02) : Evaluation de requêtes réparties

On considère la base de données répartie de schéma suivant **:**

Employés (#emp, #serv, salaire)

Service (#serv, #dir, budget)

L'attribut #dir représente l'identificateur de l'employé dirigeant le service.

La relation Employés est stockée à Naples, la relation Service est stockée à Berlin.

La taille des n-uplets de ces deux relations est de 20 octets. La taille de **#emp** et de **#dir** est de 10 octets. Les attributs **salaire** et **budget** contiennent des valeurs uniformément réparties dans l'intervalle [0, 1 000 000]. La

relation **Employés** comprend 100 000 pages, la relation **Service** comprend 5000 pages. Chaque processeur a **400** pages de buffer. La taille d'une page (du buffer et du disque) est de 4000 octets. Le coût d'entrée/sortie d'une page est t_{IO}, le coût de transfert d'une page d'un site à un autre est t_T. L'unité de transfert est la page. On suppose qu'il n'y a pas d'index.

On considère la requête suivante :

```
SELECT *
FROM Employés E, Service S
WHERE E.#emp = S.#dir
```

La requête est envoyée de Londres, et on sait que 1% des employés sont des directeurs.

Question 1.

a) Calculer la taille d'un tuple du résultat.

b) Calculer la cardinalité du résultat.

c) Calculer la taille (en nombre de pages) du résultat de cette requête.

Question 2.

On suppose que toutes les jointures sont faites en utilisant l'algorithme de tri-fusion, dont le coût dépend du nombre de pages (noté *p()*) des relations participant à la jointure. On a :

Par exemple, pour une jointure entre 2 relations locales non triées et lorsque le tri nécessite de stocker temporairement les morceaux de R et S triés, on a :

$$\text{Coût}(R \bowtie_a S) = 3 * (p(R) + p(S)) * t_{IO}$$

Pour une jointure entre 2 relations locales déjà triées, on a :

$$\text{Coût}(R \bowtie_a S) = (p(R) + p(S)) * t_{IO}$$

Si R est déjà trié et est transmise depuis un site distant. Si S est locale non triée et que son tri nécessite de stocker temporairement les morceaux de S triés, on a

$$\text{Coût}(R \bowtie_a S) = \text{coût}(R) + (3*p(S)) * t_{IO}$$

On demande de calculer le coût de l'exécution de cette requête pour les différents plans d'exécution suivants :

Plan P1: On calcule la requête à Naples en envoyant la relation Service à Naples ; le résultat est envoyé à Londres.

Plan P2: On calcule la requête à Berlin en envoyant la relation Employés à Berlin ; le résultat est envoyé à Londres

Plan P3: On calcule la requête à Londres, en envoyant les deux relations à Londres.

Plan P4: On calcule la requête à Berlin puis à Naples, en utilisant une semi-jointure à Berlin ; on envoie le résultat final à Londres. (attention, la semi-jointure peut amener à créer des relations temporaires qu'il faudra intégrer dans le coût)

Plan P5: On calcule la requête à Naples puis à Berlin en utilisant une semi-jointure à Naples ; on envoie le résultat à Londres.

Question 3.

D'après vos réponses, quel est le plan qui minimise les transferts ? Est-ce forcément le plan le plus intéressant ? Quel est le meilleur plan ?

Exercice 8 : (ref 3-2004) Requêtes réparties pts

On considère la relation

Employé (E, nom, D, salaire)

Un n-uplet représente un employé identifié par son numéro *E*. Le numéro *D* fait référence au directeur de l'employé. Le directeur est aussi un employé.

Soit la requête R1 :

```
select a.nom
from Employé a, Employé b
where a.D = b.E
and a.salaire > b.salaire
```

Question 1. Traduire la requête R1 en une phrase, en français :

Question 2. Donner l'expression algébrique de la requête R1 en fonction de la relation globale *Employé*, et telle que les opérations les plus réductrices sont traitées le plus tôt possible.

Question 3. La relation *Employé* est fragmentée en 2 fragments *E1* et *E2* tels que:

E1 contient tous les employés dont le salaire est inférieur ou égal à 1000,

E2 contient tous les employés dont le salaire est supérieur à 1000.

Donner l'expression algébrique de la requête *R1*, en fonction des fragments *E1* et *E2*, et telle que l'expression soit de la forme :

$$R1 \Leftrightarrow \pi_{\text{nom}} (\text{T1} \cup \text{T2} \cup \ldots \cup \text{Tn})$$

où les sous expressions *Ti* ne contiennent pas d'union et peuvent ne pas être vides,

les opérations les plus réductrices sont traitées le plus tôt possible.

Préciser combien il y a de sous expressions Ti.

Exercice 9 (ref R 21.8a): Requêtes réparties pts

On considère le schéma relationnel suivant :

```
EMPLOYES (#emp, #service, salaire)
SERVICES (#service, #chef, budget)
```

Les valeurs de l'attribut `#chef` sont des valeurs de `#emp`. La relation `EMPLOYES` contient 100 000 pages, la relation `SERVICES` contient 5 000 pages.Les n-uplets des deux relations ont 20 octets. Les valeurs des attributs salaire et budget contiennent des valeurs entières uniformément réparties entre 0 et 1M (on suppose 1M de valeurs distinctes). La taille des pages est de 4 000 octets.

Les relations sont réparties sur 10 sites, de la façon suivante : la relation SERVICES est partitionnée horizontalement sur les 10 sites par #service, avec le même nombre de n-uplets sur chaque site. La relation EMPLOYES est partitionnée horizontalement en fonction du salaire. Les n-uplets dont le salaire est <= 100 000 sont stockés sur le site 1, ceux dont le salaire est compris entre 100 000 et 200 000 sont sur le site 2, etc. La partition des n-uplets dont le salaire est inférieur à 100 000 est fréquemment lue, et très peu souvent mise à jour. Elle est donc répliquée sur tous les sites. Aucune autre partition de la relation EMPLOYE n'est dupliquée.

Décrivez le plan d'exécution des requêtes suivantes, et donnez leur coût, sachant que le coût de transfert d'une page est Ct, et le coût d'un accès disque est Cio.

On dispose de 1 000 pages en mémoire pour trier des données.

Question 1. Calculer la jointure naturelle entre EMPLOYES et SERVICES, en utilisant la stratégie qui consiste à envoyer tous les fragments de la plus petite relation vers les sites contenant les n-uplets de la plus grande relation. L'algorithme de jointure utilisé est le tri-fusion.

Question 2. Quel est l'employé le mieux payé ? (en cas de dupliqués, la requête doit renvoyer tous les n-uplets répondant au critère).

Question 3.Quel est le chef le mieux payé ?

Exercice 10 (ref R 21-3): Requêtes parallèles et réparties

On considère un SGBD parallèle dans lequel chaque relation est stockée par partitionnement horizontal des n-uplets sur tous les disques. On a la relation suivante :

Joueurs (*joueur-id* : integer**,** *eq-id* : integer**,** *salaire* : real**)**

On suppose que l'attribut salaire contient des valeurs dans l'intervalle [0, 1 500 000] uniformément distribuées. La relation a 150 000 pages. Une page a une taille de 4K octets, et comprend 200 n-uplets (un n-uplet a une taille de 20 octets). Le coût de lecture d'une page du disque (ou d'écriture sur le disque) est t_d, et le coût d'envoi d'une page d'un processeur vers un autre est t_s. On suppose que la relation a été partitionnée suivant l'algorithme round-robin, et qu'il y a **15** processeurs. La mémoire d'un processeur est de 201 pages. La taille de l'attribut eq-id est de 10 octets. On suppose que l'attribut salaire est uniformément réparti sur la relation.

Question 1. Décrivez brièvement le plan d'évaluation de la requête R1, et calculez son coût en termes de temps d'accès (t_d) et temps d'envoi (t_s). On donnera le coût total de la requête, ainsi que le coût en temps écoulé (si plusieurs opérations sont effectuées en parallèle, le temps écoulé est le maximum du temps pris par chacun des processeurs pour faire son travail).

R1 . Quel est le joueur le mieux payé ?

Question 2. La relation Joueurs est maintenant stockée dans un SGBD parallèle comprenant 15 sites. Elle est partitionnée horizontalement sur les 15 sites, selon les valeurs de l'attribut *salaire,* en stockant les n-uplets vérifiant la condition *salaire* <=100000 sur le premier site, les n-uplets vérifiant la condition 100000 <*salaire*<=200000 sur le second site, et ainsi de suite. La base est complétée par la relation **Equipes** suivante :

Equipes (*eq-id* : integer, *cap-id* : integer, *pays*: string)

Le champ *cap-id* de la relation Equipes est le *joueur-id* du capitaine de l'équipe. La relation Equipes comprend 4500 pages, et est répartie horizontalement selon l'attribut eq-id. On suppose que la répartition est uniforme (on a le même nombre de n-uplets sur chaque site), et aléatoire (il n'y a pas de critère pour répartir les n-uplets sur les sites). Les n-uplets de la relation Equipes ont une taille de 20 octets.

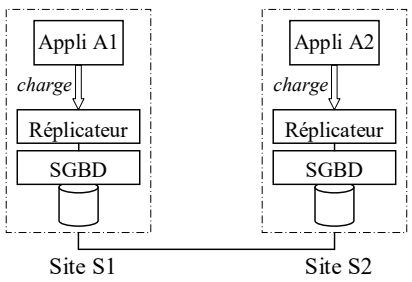
Décrivez brièvement les plans d'exécution des requêtes R1 et R2, et donnez leur coût en fonction de t_d et t_s.

R1. Quel est le joueur le mieux payé ?

R2. Faire la jointure naturelle des relations Joueurs et Equipes en utilisant la stratégie qui consiste à envoyer tous les fragments de la plus petite relation sur chaque site contenant les n-uplets de la plus grande relation.

Exercice 11 (ref 2-05) Allocation de fragments, réplication pts

On considère l'architecture répartie suivante :



Chaque site héberge un SGBD et une application. Une application produit une charge constituée de lectures et d'écritures. Pour maintenir les sites à jour, on ajoute un module de gestion des répliques (intergiciel appelé réplicateur). Si une donnée est répliquée sur plusieurs sites, une écriture doit être traitée sur **toutes** les répliques. Une lecture est traitée localement si possible.

Soit un fragment F et deux sites S1, S2. Le coût pour lire (resp. écrire) un nuplet vaut 1 (resp. 2). Le coût pour transférer un nuplet vaut 3. Ainsi, on a le modèle de coût suivant :

Lecture locale : LL = 1
Ecriture locale : EL = 2
Lecture distante, depuis Si, d'un nuplet de Sj (avec i≠j) : LD = 4
Ecriture distante, depuis Si, d'un nuplet de Sj (avec i≠j) : ED = 5

On souhaite allouer F sur un ou plusieurs sites de manière à minimiser le coût de traitement des applications.

Question 1

Une application A_i produit, sur le site S_i , une charge valant *6*n* lectures et *n* écritures de F (toutes les applications produisent la même charge). On veut calculer, en fonction de *n*, la quantité de lectures et d'écritures traitées par les SGBD des sites S1 et S2.

1.1) On suppose que F est seulement sur S1. Quel est le coût des traitements sur chaque site ?
1.2) On suppose maintenant que F est répliqué sur S1 et S2. Quel est le coût des traitements sur chaque site ?

Question 2

On complète l'architecture initiale avec un troisième site S3 et son application A3. Les applications produisent la charge suivante : A2 produit 3 fois plus de traitements que A1, A3 produit 4 fois plus de traitements que A1. Chaque application produit 10 fois plus de lectures que d'écritures. A1 produit *n* écritures et *10*n* lectures.

On veut savoir s'il est intéressant d'allouer F sur S3 ; plus précisément, dans le cas où F ne serait pas alloué sur S3, quelle serait l'augmentation (ou la diminution) du coût que cela induirait pour les traitements de A3.

2.1.a) Quelle est, en fonction des 3 variables n, L (lecture) et E (écritures) la charge produite par chaque application.
2.1.b) Est-il nécessaire d'allouer F sur S3 pour minimiser globalement le coût des traitements ? Justifier brièvement.
2.1.c) Le fait d'allouer F sur S3 a-t-il un impact sur le coût de traitement des lectures de A3 ? Si oui, quelle est la différence de coût pour traiter les lectures de A3 en comparaison avec une configuration où F n'est pas sur S3 ? Donner une réponse en fonction de n.
2.1.d) Quelle est l'allocation pour laquelle F n'est pas sur S3 et les écritures sont minimisées ? Dans ce cas, quel est en fonction de n, le coût de traitement des écritures produites par A3 ?
2.1.e) Quelle est l'allocation pour laquelle F est sur S3 et les écritures sont maximisées ? Dans ce cas, quel est en fonction de n, le coût de traitement des écritures produites par A3 ?
2.1.f) Finalement, quelle est, en fonction de n, la plus petite augmentation (ou la plus forte diminution) de coût apportée par une configuration où F n'est pas sur S3.
2.2) Quel est le coût des traitements si F est seulement sur S3 ?
2.3) Quel est le coût des traitements si F est répliquée sur S2 et S3 ?
2.4) Quel est le coût des traitements si F est répliquée sur S1, S2 et S3 ?
2.5) Quelle allocation minimise globalement le coût des traitements ?

Exercice 12 (ref 5-16): Requêtes parallèles — pts

On considère le fichier *rdv.txt* des rendez-vous de clients chez des coiffeurs. Chaque ligne a 5 valeurs séparées par un espace.

idCoiffeur idClient année mois jour

Le fichier est découpé pour être stocké sur les machines M_1 à M_{12} en fonction du mois. Les rendez-vous de janvier sont sur M_1, ceux de février sur M_2 et ainsi de suite.

a) On considère la requête : « En 2015, quel est le mois avec le plus de rendez-vous ? » Le résultat doit s'afficher sur M1. Décrire les étapes du traitement de cette requête : quelle machine fait quelle tâche ? Quelles sont les tâches effectuées en parallèle ?

b) Chaque machine M_i dispose de 4 processeurs, expliquer comment calculer la liste des identifiants distincts de clients, en parallélisant au mieux le calcul. Le résultat ne doit pas avoir de doubles et doit s'afficher sur M1.

JDBC

Exercice 1 ((ER2-19). Requêtes réparties avec JDBC

On considère 4 tables réparties sur 3 sites. On a :

sur S_1 : **Etu** (nE, nom, résidence), **Fait** (nE*, nS*, note, validé)

// validé vaut 'oui' ou 'non' et ne sert que dans l'exercice 4

sur S_2 : **Stage** (nS, lieu, durée) *// la durée vaut 3, 6, 9 ou 12 mois.*

sur S_3 : **Visite**(nE*, ville) *// un étudiant a visité des villes*

Les clés sont soulignées et sont de type String. Les attributs marqués d'une étoile font référence à une clé. Sauf indication contraire, on traite les requêtes avec JDBC en faisant des jointures par boucles imbriquées et en commençant par pousser les sélections sur les sites concernés. On demande d'utiliser des requêtes paramétrées dès que possible. Les programmes JDBC s'exécutent sur le site S_0. Une variable c_i est une connexion à S_i. Une variable s_i est un Statement, p_i est un PreparedStatement, res_i est un ResultSet. On suppose que les c_i sont créées et les s_i , p_i et res_i sont déjà déclarées.

Question 1. Calculer A1(nom, lieu). Pour les étudiants qui font un stage avec une note > 10, afficher le nom d'étudiant et le lieu de stage. On traite A_1 avec une requête sur S_1 et une requête paramétrée sur S_2. Compléter le traitement en suivant la structure proposée.

```
s1 = ..........
res1 = s1. ..........  ( …)
p2 = c2. ..............( )
while(                  ) {
   …..
}
```

Question 2. A2(lieu, nbEtu) : On suppose que le lieu d'un stage et la résidence d'un étudiant sont des noms de ville. Pour chaque lieu de stage parmi les stages durant 6 mois, afficher le nombre d'étudiants *nbEtu* résidant dans ce lieu seulement si plus de 10 étudiants y résident. Afficher le résultat dans l'ordre croissant des lieux.

Question 3. A3(durée, moy) : Pour chaque durée parmi celles des stages ayant lieu à Paris, afficher la note moyenne *moy* des stages ayant cette durée. Répondre en 2 parties :

- Définir d'abord les requêtes (complètes et paramétrées) servant à calculer A3,
- Préciser l'algorithme en français. La syntaxe JDBC n'est **pas** demandée ici.

Question 4.. A4(ne). Afficher le numéro *ne* des étudiants ayant fait au moins un stage de 3 mois et visité Aix. On sait que la ville Aix a très peu de visites, donc on commence par une requête sur Visite. Ne pas afficher les *ne* en double. Pour répondre, définir d'abord les requêtes (complètes et paramétrées) utilisées, puis préciser l'algorithme en français. La syntaxe JDBC n'est pas demandée ici.

Exercice 2: (ER2-18) Requêtes réparties avec JDBC pts

On considère 3 tables réparties sur 3 sites. On a :

sur S_1 : **TA** (numA, a1, a2)

sur S_2 : **TB** (numB, numA*, b3, b4)

sur S_3 : **TC** (numC, numB*, c5, c6)

Les clés sont soulignées et sont de type String. Les attributs marqués d'une étoile font référence à une clé. Les autres attributs sont des nombres entiers.

Sauf indication contraire, on traite les requêtes avec JDBC en faisant des jointures par boucles imbriquées et en commençant par pousser les sélections sur les sites concernés. On demande d'utiliser des requêtes paramétrées (PreparedStatement) dès que possible. Les programmes JDBC s'exécutent sur le site S_0.

Question 1.

```
SELECT x.a1
FROM TA x, TB y
WHERE x.numA = y.numA AND x.a2 < 10 AND y.b3 = 9
```

L'algorithme pour traiter la requête est défini ainsi :

```
Sur S1 poser une requête R1 puis
   Pour chaque tuple obtenu poser une requête R2 sur S2.
            Pour chaque tuple obtenu, afficher un tuple du résultat.
```

Quelle(s) requête(s) paramétrée et non paramétrée(s) faut-il exécuter ? Pour chacune, préciser le site, le code SQL et si elle est paramétrée ou non.

Question 2.

```
SELECT b.b4, c.c5
FROM TB b, TC c
WHERE b.numB = c.numB
AND b3 < c5  AND  b.b4 NOT IN (      SELECT a.a2  FROM TA  )
```

Décrire l'algorithme pour traiter cette requête : détailler les itérations (*pour chaque tuple ...*) sur le résultat des requêtes posées sur S2, S3, puis S1. Rmq : vous pouvez faire référence aux requêtes détaillées par la suite.

Question 3. Soit la requête globale :

```
SELECT a.numA, a1, b3
FROM TA a, TB b
WHERE a.numA = b.numA   AND   a2=0   AND   b4=0
```

En vous inspirant de la méthode de traitement par semi-jointure, proposer une solution pour calculer le résultat sans **jamais** transférer vers le programme JDBC une valeur de l'attribut *a1* ou de *b3* qui ne soit pas dans le résultat. Par exemple, si on a :

TA = {(na1, 500, 0), (na2, 600, 0), (na3, 10, 1)}

et TB = {(nb1, na2, 10, 0), (nb2, na2, 20, 0) (nb3, na3, 30, 0)},

Le résultat contient seulement {(na2, 600, 10), (na2, 600, 20)}.

Ne pas poser de requête sur TA dont un tuple du résultat contiendrait (na1, 500), ni de requête sur TB dont un tuple du résultat contiendrait (na3, 30). Rappel : l'instruction next() transfère vers le programme JDBC un tuple du résultat en entier.

Décrire l'algorithme pour traiter cette requête : détailler les itérations (*pour chaque tuple ...*) et faire référence aux requêtes détaillées par la suite.

Question 4 (bonus). On veut écrire en JDBC un programme qui affiche le résultat de la requête suivante.

```
SELECT b.numB as X,   c.numB as Y
FROM TB b, TC c
WHERE b4 = c6
```

Rmq, la jointure n'est **pas** sur numB mais est définie par b4=c6

On considère une solution qui traite cette requête par un algorithme de **tri fusion** (*cf.* le TME JDBC). Répondre en décrivant brièvement l'algorithme et en détaillant les requêtes posées. Pour simplifier on suppose que b4 est unique et que toutes les valeurs de c6 existent dans le domaine de b4.

Exercice 3 (4-17): Requêtes réparties avec JDBC pts

Soit la base de données répartie sur les sites S_1 à S_4 :
Sur **S1**
Mécanicien (idpers, idgarage, niveau) // un mécanicien travaille dans un garage
Garage (idgarage, nom, ville, jourdefermeture)

Sur **S2** :
Réparation (idmecanicien, immat, date, intervention) // le mécanicien a réparé le véhicule immat
Habilité (idgarage, marque) // le garage peut réparer des véhicules de la marque

Sur **S3** :
Personne (idpers, nom, prenom, age, tél) // un mécanicien ou un client
Possède (immat, marque, modele, idclient) // le véhicule immat (avec sa marque et son modèle) appartient à idclient

Les autres tables (Tarif et Client) sont sur S4. Tous les programmes JDBC s'exécutent sur S0 et peuvent se connecter au site S_i avec l'URL url_i.

Question 1 : Soit la requête globale **R1**:

select v.marque, m.nom, m.prénom
from Personne m, Réparation r, Possède v
where m.idPers = r.idMecanicien **and** v.immat = r.immat
and r.intervention = 'remplacement moteur' **and** r.date = '16/5/2017'

On traite cette requête avec JDBC en faisant des jointures par boucles imbriquées et en **commençant** par traiter les **sélections** sur les sites concernés.

a) Quelle(s) requête(s) **non** paramétrée(s) faut-il exécuter ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.
b) Quelle(s) requête(s) paramétrée(s) faut-il exécuter ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.

Question 2 : Soit la requête globale **R2** affichant le jour de fermeture du Garage central d'Aix s'il est habilité pour la marque du modèle Medusa. Le résultat ne contient **aucun** doublon.

select distinct g.idgarage, g.jourdefermeture
from Garage g, Habilité h, Possède v
where g.idgarage = h.idgarage **and** h.marque = v.marque
and v.modèle = 'Medusa'
and g.nom = 'Garage central'
and g.ville = 'Aix'

On traite cette requête avec JDBC en faisant des jointures par boucles imbriquées et en commençant par pousser les sélections sur les sites concernés.

a) Quelle(s) requête(s) **non** paramétrée(s) faut-il exécuter ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.
b) Quelle(s) requête(s) paramétrée(s) faut-il exécuter ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.

Question 3 : Soit la requête globale **R3** : « les garages (idgarage, nom, ville) qui ne sont **PAS** habilités pour réparer la marque 'Carapid' ». Ecrire la procédure J3 pour qu'elle affiche le résultat de R3 en transférant le **moins** de données possible vers S0.

```
1  void J3() {
2    s1 = DriverManager.getConnection(url1); // site S1
3    (…) idem pour les connexions s2, s3 et s4
4    Statement S = ..... .createStatement();
```

```
5    PreparedStatement P = …
```

Question 4. Soit la requête globale **R4**:

Le résultat affiche les couples (idgarage, nbMeca) où nbNeca est le nombre de mécaniciens. Le résultat est trié par ordre croissant de idgarage. La requête R4 est :

select m.idgarage, **count**(**distinct** m.idPers) **as** nbMeca
from Mecanicien m, Reparation r, Possede v
where m.idpers = r.idmecanicien **and** r.immat = v.immat
and r.idmecanicien = v.idclient
group by idgarage **order by** idgarage;

Proposer une solution pour traiter R4 de telle sorte que le nombre de nuplets transférées vers S0 soit minimal. Préciser les requêtes à traiter (les nommer A, B, C, ...). Préciser si elles sont paramétrées ou non, préciser le site traitant chaque requête. Préciser les itérations « pour chaque nuplet de la requête A .... fin pour». Indenter les itérations pour montrer les imbrications.

Exercice 4: (ref 4-16) Requêtes réparties avec JDBC pts

Soit la base de données répartie sur les sites S1 et S2 telle que :

Pers (idPers, nom, âge) est sur **S1**. // certaines personnes n'ont pas de rendez-vous.
Coiffeur (idPers, idSalon, niveau) est sur **S1**
Salon (idSalon, ville) est sur **S2**
RDV (idCoiffeur, idClient, jour, heure, coupe) est sur **S2**

Les programmes java, proposés par la suite, sont des extraits de programmes complets, suffisants pour comprendre ce que fait le programme. Tous les programmes JDBC s'exécutent sur S0 et peuvent se connecter au site S_i avec l'URL url_i.

Question 1 : Qu'affiche la procédure A1 ?

```
1   void A1(int X, int Y) {
2     s1 = DriverManager.getConnection(url1); // site S1
3     s2 = DriverManager.getConnection(url2); // site S2
4
5     PreparedStatement q1 = s1.prepareStatement("select idPers, nom, age,
6   from Pers where age < ?");
7     PreparedStatement q2 = s2.prepareStatement("select count(*) as n
8   from RDV where idClient = ?");
9     q1.setInt(1, X);
10    ResultSet r1 = q1.executeQuery();
11    while(r1.next()) {
12      q2.setInt(1, r1.getInt(1);
13      ResultSet r2 = q2.executeQuery();
14        while(r2.next()) {
15          if(r2.getInt(1) >= Y)
16              out.println(r1.getString(2) + " " + r1.getInt(3));
17        }
18    }
19  }
```

Question 2 : Soit la requête globale **R1** affichant l'identifiant, le nom et le niveau des coiffeurs ayant des rendez-vous à 8h du matin :

select distinct p.idPers, p.nom, c.niveau,
from Coiffeur c, RDV r, Pers p
where r.idCoiffeur = c.idPers **and** c.idPers = p.idPers
and r.heure=8 ;

a) On suppose que R1 contient 10 coiffeurs parmi les 1000 coiffeurs existants. Compléter la procédure J1 pour qu'elle affiche le résultat de R1 en transférant le moins de données possible vers S0.

```
1   void J1() {
2     s1 = DriverManager.getConnection(url1); // site S1
3     s2 = DriverManager.getConnection(url2); // site S2
4
5     Statement S = ________.createStatement();
6
7     PreparedStatement P =
```

b) On veut traiter R1 en faisant une jointure par **fusion** des résultats d'une requête posées sur S1 et d'une autre posée sur S2. Ecrire en SQL les requêtes que le programme JDBC pose sur chaque site.

Question 3 : Soit la requête globale **R2**:

select * **from** RDV r
where r.idCoiffeur **in** (**select** c.idPers
from Coiffeur c, Salon s
where c.idSalon = s.idSalon **and** s.ville = 'Aix')

On traite cette requête avec JDBC en faisant des jointures par boucles imbriquées et en poussant les sélections sur les sites S1 ou S2.

a) Combien de requêtes **non** paramétrées sont nécessaires ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.

b) Combien de requêtes paramétrées sont nécessaires ? Pour chacune, donner son code SQL et préciser sur quel site elle s'exécute.

Exercice 5 (ref 5-15) : Requêtes réparties avec JDBC

Soit la base de données répartie sur les sites S1 et S2 telle que :

Perf (ids, comp, rang) est sur **S1**.
// le sportif *ids* est classé à la position *rang* pour la compétition *comp*.
Si un sportif participe à une compétition sans être classé, son rang vaut 0.
Tous les sportifs ne font pas de compétition.

Sportif (ids, nom) est sur **S2**.
// ids est le numéro d'identification du sportif (nombre entier)

Les programmes java, proposés par la suite, sont des extraits de programmes complets, suffisants pour comprendre ce que fait le programme. Ils s'exécutent sur S0 et accèdent aux sites S1 et S2. Seul le site S0 peut accéder à S1 et S2 par JDBC (connexion avec l'url1 pour S1 et l'url2 pour S2).

Question 1 : Qu'affiche la procédure A1 ?

```
1   void A1(int X, int Y) {
2     s1 = DriverManager.getConnection(url1); // site S1
3
4     PreparedStatement q1 = s1.prepareStatement("select comp, rang from
5   Perf where ids = ? order by comp, rang");
6     q1.setInt(1, X);
7     ResultSet r1 = q1.executeQuery();
8
9     while(r1.next()) {
10        if(r1.getInt(2) < Y)
11              out.println(r1.getString(1) + " " + r1.getInt(2));
12    }
13    q1.close(); s1.close();
14  }
```

Question 2 : Qu'affiche la procédure A2 ?

```
1   void A2(int X) {
2     s1 = DriverManager.getConnection(url1); // site S1
3
4     PreparedStatement q1 = s1.prepareStatement("select ids from Perf
```

```
5   group by ids having count(*) >= ? ");
6     q1.setInt(1, X);
7     ResultSet r1 = q1.executeQuery();
8     int c=0;
9     while(r1.next()) { c = c+1; } // fin du while
10    out.println(c);
11    q1.close(); s1.close();
12  }
```

Question 3 : Définir A3(int X) qui affiche le numéro des sportifs qui ont été classés au moins 2 fois au rang X (dans 2 compétitions différentes).

Question 4 : Compléter la procédure J1 pour qu'elle affiche le résultat de la jointure :

J1 = Perf $\bowtie_{ids}$ Sportif.

Le schéma du résultat de J1 est (ids, comp, rang, nom)

Question 5 : Compléter la procédure J2 pour qu'elle affiche le résultat de la requête :

```
SELECT p.ids, s.nom, p.comp
FROM Sport s, Perf p
WHERE s.ids = p.ids
AND p.rang >10
ORDER BY s.nom
```

Le traitement utilise des semi-jointures.
On utilise les tables auxilliaires T1(a int, b varchar) et T2(a int). T1 et T2 sont vides initialement.
Lorsqu'un *PreparedStatement* représente une insertion SQL, alors il est exécuté avec la méthode *execute()*.

Transactions

Exercice 1 (BDLE 2019). Transactions à large échelle avec Calvin pts

On considère un système de gestion de transactions à large échelle fonctionnant comme Calvin (*cf.* cours) et déployé sur les machines M1 à M4, chacune stockant une partie des données (sans réplication) et pouvant recevoir des transactions. On rappelle que les demandes reçues pendant une période P_i sont ordonnées ainsi :

Si deux transactions T_i posée sur M_a et T_j posée sur M_b accèdent à au moins une donnée commune, on a :

si a < b, alors T_i précède T_j si a = b et i < j, alors Ti précède Tj

Une transaction est traitée sur chaque machine stockant au moins une donnée écrite par la transaction.

Les données sont placées sur les machines comme suit :

M1 : A, B **M2** : C,D **M3** : E,F **M4** : G.

Les transactions reçues pendant la période P_1 sont :

M1 : T4(E, G), T5(A) **M3** : T1(G), T7(D, G)

M2 : T2(C,E), T6(B,A) **M4** : T3(G, B)

Question 1. Donner les numéros des transactions qui seront traitées une seule fois (sur une seule machine) et celles qui seront traitées deux fois sur deux machines différentes.

Question 2.

a) Après la transmission (par les séquenceurs) des demandes aux machines concernées, préciser pour chaque machine quelles sont les transactions traitées et quel est l'ordre de traitement. Répondre en utilisant la syntaxe Ti→ Tj pour indiquer que Ti précède Tj.

b) Si on supposait que T7(D,G) lit et écrit D mais ne fait que lire G sans l'écrire. Quelles seraient les conséquences sur le traitement des transactions ?

Question 3. On suppose qu'une transaction lit et écrit toutes les données qu'elle manipule.
Détailler l'exécution des transactions traitées par la machine **M1** à l'aide des actions suivantes :

envoyer *X* **vers** *Mi* **recevoir** *X* **de** *Mi* **traiter** *Ti*

Exercice 2 : (ref 1-01) Verrouillage réparti de données répliquées

1. Quel est le problème posé lorsqu'on veut verrouiller une donnée répliquée ?
2. Qu'est ce qu'un verrou global , un verrou local ?
3. On suppose qu'un gestionnaire de verrou centralise les demandes de verrou et d'accès aux données. Expliquer son fonctionnement en cas de données répliquées. Quels sont les avantages et défauts d'une telle approche ?

Pour accéder à une réplique, il faut avoir le verrou global sur la donnée. Il faut donc s'adresser au gestionnaire de verrou global. L'avantage est la simplicité : dès que le gestionnaire global me donne le verrou, j'ai automatiquement les verrous sur les répliques. Le défaut est lié à la centralisation : il faut mettre en œuvre ce gestionnaire et y accéder sans cesse, risque de goulot d'étranglement et risque en cas de panne de ce site.

4. On cherche maintenant à ne pas mettre en place de gestionnaire centralisé, mais de tirer parti des gestionnaire de verrous et d'accès locaux. Pour cela, les transactions doivent respecter le protocole suivant :

lorsqu'elles ont réussi à verrouiller un certain nombre de répliques d'une donnée, les transactions peuvent déduire qu'elle

ont accès à la donnée. On dit qu'elle obtiennent un *verrou logique* sur la donnée grâce aux *verrous physiques* qu'elles ont obtenus sur les répliques. On suppose qu'une donnée *d* est répliquée sur *n* sites.

a. On suppose que n=3. Montrer qu'en obtenant au moins 2 verrous physiques exclusifs, une transaction peut avoir un verrou logique exclusif sur *d* et donc accès en écriture à *d*, donc à toutes ses répliques.

b. Soit k (resp. k') le nombre de verrous physiques exclusifs (resp. partagés) à obtenir pour en déduire un verrou logique exclusif (resp. partagé). Donner la valeur minimale de k (resp. k') en fonction de n que le protocole doit respecter pour que le verrouillage logique fonctionne.

Exercice 3 : (ref 2-01) Sérialisation globale et locale

On suppose les objets x et y stockés sur le site S1, et les objets z et w sur le site S2. Déterminer pour chacune des exécutions suivantes de deux transactions Ti et Tj les graphes de précédence locaux et global et dire si elle est sérialisable ou non.

1) Exécution 1 :

   S1 : Li(x), Lj(x), Ej(x), Ei(x)

   S2: Li(w), Lj(z), Ej(w), Ei(w)

2) Exécution 2 :

   S1 : Li(x), Ei(x), Lj(x), Ej(x), Li(y), Ej(y)

   S2 : Lj(z), Ej(z), Li(z), Ei(w)

3) Exécution 3 :

   S1 : Li(y), Lj(x), Ej(x)

   S2 : Ei(z), Li(w), Lj(w), Ej(w)

Exercice 4 : (ref 3-01) détection d'interblocage

A et B sont stockés sur S1, C et D sur S2. On utilise un verrouillage en deux phases. On exécute les transactions T1 à T9. Les demandes d'accès sur les différents granules sont les suivants :

S1 A : L7, E3, E2, L1

   B: L3, L4, E6, L9

S2 C: L9, E8

   D : E8, L7, L5

1) Quelles sont les transactions globales ?
2) Décrire l'évolution de la table des verrous dans le cas où une demande de verrou partagé sera toujours satisfaite si possible. Représenter la table avant la première libération de verrou possible. Rappeler quel est l'inconvénient d'une telle gestion.
3) Proposer une autre politique permettant de résoudre le problème précédent et montrer l'état des tables de verrous dans ce cas.
4) Construire les graphe d'attente locaux et le graphe d'attente global. Est-il nécessaire d'utiliser les premiers en entier pour construire le dernier ? Que concluez-vous du résultat obtenu ?
5) On centralise la détection d'interblocages sur S1. Comparer deux solutions : dans la première, chaque modification du graphe local d'attente de S2 est communiqué à S1. Dans la seconde, la copie complète du graphe local de S2 est communiqué périodiquement à S1.

Exercice 5 (ref 3-17) Transactions réparties 4 pts

Soit une base de données répartie telle que :

- les granules **x** et **z** sont sur le site **S1**
- les granules **y** et **t** sont sur le site **S2**

Soit la séquence *Seq* suivante des transactions T1, T2, T3, T4. Li(g) signifie lecture du granule g par la transaction Ti, Ei(g) signifie écriture par la transaction Ti du granule g, Vi signifie validation de la transaction Ti.

Seq : L1(y), L3(z), L2(z), E3(z), L3(t), L1(t), L1(z), E4(t), L2(t), L3(x), L4(y), E1(x)

Question 1 : On suppose un contrôle de concurrence optimiste où le graphe de précédence global de l'exécution est construit sur **S1**, progressivement, à chaque nouvelle précédence générée sur l'un ou l'autre des sites. Les messages sont de la forme (Ti, Tj, g) si Ti précède Tj sur le granule g.

Représenter ce graphe à la fin de *Seq*. L'exécution sera-t-elle validée ? Justifiez votre réponse. Cette décision aurait-elle pu être prise en ne considérant que les graphes de précédences locaux ? Quels messages ont-été envoyés par S2 jusqu'à ce moment ?

Question 2 : On suppose maintenant un contrôle de concurrence par verrouillage deux-phases strict sur la séquence Seq : une demande de verrou se fait juste avant l'opération correspondante, le relâchement des verrous se fait juste après la dernière opération de la transaction. Les transactions lectrices obtiennent toujours le verrou partagé sur un granule si c'est possible, même si des transactions écrivaines sont en attente sur le granule.

Assiste-t-on à un interblocage ? Si oui, montrez l'état de la table des verrous à l'instant où se produit l'interblocage (en indiquant quelle est l'opération de Seq qui génère l'interblocage). Sinon, montrez l'ordre produit par le contrôleur de concurrence (noter Vi pour la validation de la transaction Ti).

Remarque : la promotion de verrou est possible. Rappel : lorsqu'une transaction T demande un verrou exclusif sur un granule g, qu'elle possède déjà un verrou partagé sur g, et qu'aucune autre transaction ne verrouille g, alors le verrou partagé est promu en verrou exclusif afin de satisfaire la demande de T.