# LIF4 - TD8
# Correction
**Optimisation**

**Exercice 1:**

Supposons un fichier ordonné avec $r = 30,000$ enregistrements stockés sur un disque avec des blocs de taille $B = 1024$ octets. Les enregistrements sont de taille fixe avec une longueur $R = 100$ octets.

1. Calculer le nombre d'enregistrements par bloc

   **Correction:** $epb = \lfloor (B/R) \rfloor = \lfloor 1024/100 \rfloor = 10$ enregistrements

2. Calculer le nombre de blocs nécessaires pour le fichier

   **Correction:** $b = \lceil (r/epb) \rceil = \lceil (30000/10) \rceil = 3000$ blocs

Une recherche binaire (ou dichotomique) dans un fichier ordonné est effectuée de la manière suivante:

- On se donne une variable contenant un intervalle. Tout au long de l'algorithme cet intervalle contient la valeur cherchée si elle est présente dans le fichier.
- Au départ les bornes de l'intervalle sont le premier et le dernier enregistrement du fichier.
- On répète ensuite l'action suivante jusqu'à obtenir une intervalle de taille 1 enregistrement qui contient l'enregistrement cherché s'il est dans le fichier:
  - Diviser l'intervalle en deux parties égales en utilisant l'enregistrement *pivot* situé au milieu des enregistrements "bornes" de l'intervalle courant.
  - Comparer l'enregistrement cherché à *pivot*. S'il est plus petit, changer l'intervalle en prenant le premier sous-intervalle. Sinon changer l'intervalle en prenant le second sous-intervalle.

Donner le nombre d'accès blocs nécessaires pour une recherche binaire dans le fichier ci-dessus.
**Correction:** Approximativement $log_2(b) = log_2(3000) = 12$ accès blocs

**Exercice 2:**

Soit une base de données avec les relations suivantes :

- Employe(Prenom, Nom, NSS, DN, Adresse, Sexe, Salaire, SuperNSS, NumDep)
- Departement(NomD,NumD, NSSResp, DateEntreeResp)
- EmplacementDept(NumD,Emplacement)
- Projet(NomProjet,NProjet,Emplacement,NumD)
- TravailleSur(TNSS, NumProjet, Heures)

Considérons la requête suivante :

Q : SELECT Nom
FROM Employe, TravailleSur, Projet
WHERE NomProjet='Aquarius' AND NProjet=NumProjet
AND TNSS=NSS AND DN > '31-DEC-1955'

1. Donner en langage naturel le résultat de $Q$
2. Donner l'arbre de requête optimal pour $Q$. On traduira d'abord $Q$ en une expression de l'algèbre relationnelle que l'on optimisera ensuite.

**Correction:** Cette requête renvoie les Nom des employés travaillant sur le projet "Aquarius" et qui sont nés après le 31/12/55.

$\pi_{Nom}(\sigma_{NomProjet='Aquarius' \wedge DN>'31/12/1955'}((Projet \bowtie_{NProjet=NumProjet} TravailleSur) \bowtie_{TNSS=NSS} Employe))$

Q2 : SELECT E1.Nom, Projet.NomProjet
FROM Employe E1, Employe E2, Departement, TravailleSur, Projet
WHERE E1.SuperNSS=E2.NSS AND Projet.NProjet=TravailleSur.NumProjet
AND TravailleSur.TNSS=E1.NSS AND E2.NSS=Departement.NSSResp
AND E2.Sexe='M' AND E2.Salaire > 10000
AND TravailleSur.Heures > 20

Mêmes questions pour Q2.

**Correction:** Cette requête renvoie pour chaque employé homme dont le/la supérieure hiérarchique est responsable d'un département et gagne plus de 10000 euros, la liste des projets pour lesquels il a travaillé plus de 20 heures.

**Exercice 3:**

Soit une base de données avec les relations suivantes :

- LIVRE(ISBN, titre, , auteur, éditeur)
- Prêt(NumCarteP, ISBNP, date)
- Lecteur(NumCarte,nom, adresse)

Soit la requête à optimiser: la liste des noms des lecteurs et des titres pour tous les prêts d'avant le 15.3.06.

Donner la requête en SQL puis traduire cette requête en algèbre relationnelle. Donner l'arbre d'optimisation de la requête.

**Correction:** forme de la requête avant optimisation

$\pi_{Nom,titre}(\sigma_{date<15.03.06}(\sigma_{numcarte=numcarteP \wedge ISBN=ISBNP}(PretXlecteurXLivre)$

L'arbre est inchangé par l'étape de descente des projections. L'arbre est inchangé par l'étape regroupement des sélections.

Résultat: $G_1 = (\sigma_{date=15.03.06}Pret) \bowtie_{numcarte=numcarteP)} Lecteur$
$\pi_{nom,titre}(G_1 \bowtie_{ISBN=ISBNP} Livre$