Bases de Données Relationnelles, Cours 1

Anne-Cécile Caron, Anne Étien, Mikaël Monet, Sylvain Salvati

Plan

Objectifs du cours

Comprendre une partie du fonctionnement interne des Systèmes de Gestion de Bases de Données (SGBD) relationnels pour :

- savoir comment il est possible de gérer de très grandes quantités de données,
- utiliser les SGBD au mieux de leurs capacités,
- découvrir de nouveaux algorithmes.

Objectifs du cours

Comprendre une partie du fonctionnement interne des Systèmes de Gestion de Bases de Données (SGBD) relationnels pour :

- savoir comment il est possible de gérer de très grandes quantités de données,
- utiliser les SGBD au mieux de leurs capacités,
- découvrir de nouveaux algorithmes.

Le cours se fera en six séances :

- Introduction, algorithmes de jointure (1h cours, 3h TD)
- Indexation et plans d'exécution (2h cours, 2h TD)
- Procédures stockées et triggers (2h cours, 2h TD)
- Modélisation, dénormalisation et optimisation (2h cours, 2h TD)
- Transactions et contraintes (2h cours, 2h TD)
- Administration (2h cours)

\+ 2h d'évaluation par un TP

Objectifs du cours

Comprendre une partie du fonctionnement interne des Systèmes de Gestion de Bases de Données (SGBD) relationnels pour :

- savoir comment il est possible de gérer de très grandes quantités de données,
- utiliser les SGBD au mieux de leurs capacités,
- découvrir de nouveaux algorithmes.

Le cours se fera en six séances :

- Introduction, algorithmes de jointure (1h cours, 3h TD)
- Indexation et plans d'exécution (2h cours, 2h TD)
- Procédures stockées et triggers (2h cours, 2h TD)
- Modélisation, dénormalisation et optimisation (2h cours, 2h TD)
- Transactions et contraintes (2h cours, 2h TD)
- Administration (2h cours)

\+ 2h d'évaluation par un TP

NB : par SGBD, nous entendrons toujours SGBD relationnel.

Remerciements

Ce cours doit énormément aux cours suivants :

- Pierre Senellart : `http://pierre.senellart.com/enseignement/2016-2017/bd/`,
- Nicolas Travers : `https://chewbii.com/education/optimisation-de-bases-de-donnees/`
- Philippe Rigaux : `http://www.bdpedia.fr/systemes-relationnels/`

Plan

Gestion de données

En grande majorité, les applications modernes (sites web, app android, ...) génèrent, collectent et utilisent des données. La gestion de données est un composant essentiel de ces applications. Cela implique :

- de structurer les données utiles de l'application,
- de conserver les données de façon persistante (même lorsque le logiciel ne tourne pas, que la machine est éteinte,...),
- de rechercher efficacement les données même lorsqu'elles sont très volumineuses,
- de mettre à jour les données tout en conservant leur structure et leur cohérence,
- de permettre l'accès à plusieurs utilisateurs de façon concurrente,
- parfois, il est souhaitable (open data, ...) que différentes applications/ordinateurs/terminaux puissent accéder aux mêmes données.

Exemple : agence de voyage

Accès à des propositions de voyage depuis un site web dédié, depuis des agrégateurs sur internet, depuis un logiciel d'agence, depuis un logiciel de comptabilité... Les fonctionnalités attendues sont :

- données structurées qui représentent des clients, des réservations, des voyages (itinéraires, hôtels, activités,...)
- persistance des données lorsque l'on change les ordinateurs, qu'il y a une panne de courant, lors de mises à jours...
- retrouver instantanément les voyages disponibles, les promotions en cours, les voyages prévus pour un client, les clients qui doivent payer, les billets à imprimer pour la semaine,...
- il doit être aisé de réserver un voyage tout en s'assurant qu'il y encore des places dans l'avion et l'hôtel prévus,
- Les différents acteurs (client, comptable, vendeur, commercial...) n'ont pas accès aux mêmes informations (confidentialité, prérogatives différentes...) ; on ne peut pas réserver un même voyage (même place d'avion, chambre d'hôtel,...) en même temps.

Implémentation d'un tel système

- définition des structures de données dans le langage de programmation pour manipuler des données,
- définition d'un format de stockage sur le disque, protocole de synchronisation de la mémoire et du disque, mécanisme de récupération après panne,
- stockage en mémoire des données et structures sous-jacentes (arbres de recherche, tables de hachage, ...) et algorithmes de recherche rapide (tri, agrégation, ...),
- mise à jour des données avec vérification des contraintes d'intégrité des données,
- gestion des utilisateurs, des droits d'utilisation des données et gestion de la concurrence (thread, verrous, sémaphores, ...),
- implémentation des protocoles de communication réseau, ...

Implémentation d'un tel système

- définition des structures de données dans le langage de programmation pour manipuler des données,
- définition d'un format de stockage sur le disque, protocole de synchronisation de la mémoire et du disque, mécanisme de récupération après panne,
- stockage en mémoire des données et structures sous-jacentes (arbres de recherche, tables de hachage, ...) et algorithmes de recherche rapide (tri, agrégation, ...),
- mise à jour des données avec vérification des contraintes d'intégrité des données,
- gestion des utilisateurs, des droits d'utilisation des données et gestion de la concurrence (thread, verrous, sémaphores, ...),
- implémentation des protocoles de communication réseau, ...

Travail énorme, requiert énormément de compétences, n'est pas réutilisable.

Rôles des SGBD/DBMS

Systèmes de Gestion de Bases de Données
(DataBase Management Systems –DBMS–)

- interagir (créer, consulter, mettre à jour, effacer) *efficacement* avec des données,
- garantir leur cohérence,
- garantir leur pérennité.

Et tout cela avec des volumes de données qui peuvent être considérables (parfois plusieurs To).

Fonctionnalités d'un SGBD

Indépendance physique : pour se servir d'un SGBD, il n'est pas nécessaire de savoir comment les données sont stockées (dans un fichier, sur plusieurs machines, sur disque, ...). La méthode de stockage peut changer au cours de la vie de la base de données.

Fonctionnalités d'un SGBD

Indépendance physique : pour se servir d'un SGBD, il n'est pas nécessaire de savoir comment les données sont stockées (dans un fichier, sur plusieurs machines, sur disque, ...). La méthode de stockage peut changer au cours de la vie de la base de données.

Indépendance logique : les mécanismes de vue permettent de présenter aux utilisateurs des informations structurées qui peuvent être très éloignées du schéma des données. Les utilisateurs n'ont accès qu'aux données qui les concernent et dans une forme simple d'utilisation.

Fonctionnalités d'un SGBD

Indépendance physique : pour se servir d'un SGBD, il n'est pas nécessaire de savoir comment les données sont stockées (dans un fichier, sur plusieurs machines, sur disque, ...). La méthode de stockage peut changer au cours de la vie de la base de données.

Indépendance logique : les mécanismes de vue permettent de présenter aux utilisateurs des informations structurées qui peuvent être très éloignées du schéma des données. Les utilisateurs n'ont accès qu'aux données qui les concernent et dans une forme simple d'utilisation.

Accès aisé aux données : utilisation d'un *langage déclaratif* qui décrit les données à chercher ou à modifier plutôt qu'un algorithme de recherche ou de modification des données.

Fonctionnalités d'un SGBD

Indépendance physique : pour se servir d'un SGBD, il n'est pas nécessaire de savoir comment les données sont stockées (dans un fichier, sur plusieurs machines, sur disque, ...). La méthode de stockage peut changer au cours de la vie de la base de données.

Indépendance logique : les mécanismes de vue permettent de présenter aux utilisateurs des informations structurées qui peuvent être très éloignées du schéma des données. Les utilisateurs n'ont accès qu'aux données qui les concernent et dans une forme simple d'utilisation.

Accès aisé aux données : utilisation d'un *langage déclaratif* qui décrit les données à chercher ou à modifier plutôt qu'un algorithme de recherche ou de modification des données.

Optimisation de requête : les requêtes posées au SGBD sont optimisées avant d'être exécutées afin d'avoir des temps de réponse les plus rapides possibles.

Garanties apportées par un SGBD

Intégrité logique : le SGBD permet d'imposer des contraintes sur la structure des données. Toute modification des données qui violerait ces contraintes est rejetée.

Garanties apportées par un SGBD

Intégrité logique : le SGBD permet d'imposer des contraintes sur la structure des données. Toute modification des données qui violerait ces contraintes est rejetée.

Intégrité physique : la base reste dans un état cohérent de façon durable et ce même en cas de panne matériel.

Garanties apportées par un SGBD

Intégrité logique : le SGBD permet d'imposer des contraintes sur la structure des données. Toute modification des données qui violerait ces contraintes est rejetée.

Intégrité physique : la base reste dans un état cohérent de façon durable et ce même en cas de panne matériel.

Partage des données : les données sont accessibles à plusieurs utilisateurs avec des droits d'accès différents, de manière concurrente et sans que l'intégrité physique et logique de la base ne soient mises en péril.

Garanties apportées par un SGBD

Intégrité logique : le SGBD permet d'imposer des contraintes sur la structure des données. Toute modification des données qui violerait ces contraintes est rejetée.

Intégrité physique : la base reste dans un état cohérent de façon durable et ce même en cas de panne matériel.

Partage des données : les données sont accessibles à plusieurs utilisateurs avec des droits d'accès différents, de manière concurrente et sans que l'intégrité physique et logique de la base ne soient mises en péril.

Normalisation : les SGBD respectent la norme SQL, de sorte qu'il est possible (dans une certaine mesure) de remplacer un SGBD par un autre si nécessaire sans pour autant avoir à changer substantiellement le code des applications clientes.

Interagir avec de grands volumes de données

Supposons que nous ayons deux tables :

Clients	
`Client`	`ID`
Entreprise1	1309809
Entreprise2	2342980
...	...

Commandes	
`Prod`	`ID_Client`
P1230	124090
P409248	3092
...	...

Interagir avec de grands volumes de données

Supposons que nous ayons deux tables :

Clients	
`Client`	`ID`
Entreprise1	1309809
Entreprise2	2342980
...	...

Commandes	
`Prod`	`ID_Client`
P1230	124090
P409248	3092
...	...

On suppose que la table **Clients** contient 10^6 lignes et que la table **Commandes** en contient 10^7.

Interagir avec de grands volumes de données

Supposons que nous ayons deux tables :

Clients	
`Client`	`ID`
Entreprise1	1309809
Entreprise2	2342980
...	...

Commandes	
`Prod`	`ID_Client`
P1230	124090
P409248	3092
...	...

On suppose que la table **Clients** contient 10^6 lignes et que la table **Commandes** en contient 10^7.

On cherche à répondre à la requête

```
select (Client,Prod) from Clients Commandes where Clients.ID=Commandes.ID_Client
```

Interagir avec de grands volumes de données

On cherche à répondre à la requête

```
select (Client,Prod) from Clients Commandes where Clients.ID=Commandes.ID_Client
```

Une méthode naïve consisterait à chercher pour chaque client chacune de ses commandes en parcourant entièrement la table **Commandes**. Cela nous ferait effectuer :

$10^6 \times 10^7 = 10^{13}$ lectures de lignes de la table de commandes

Si chaque lecture prend 10^{-9} secondes, cela représente à peu près 2h45 pour obtenir les 10^7 éléments de la réponse...

Interagir avec de grands volumes de données

On cherche à répondre à la requête

```
select (Client,Prod) from Clients Commandes where Clients.ID=Commandes.ID_Client
```

Une méthode naïve consisterait à chercher pour chaque client chacune de ses commandes en parcourant entièrement la table **Commandes**. Cela nous ferait effectuer :

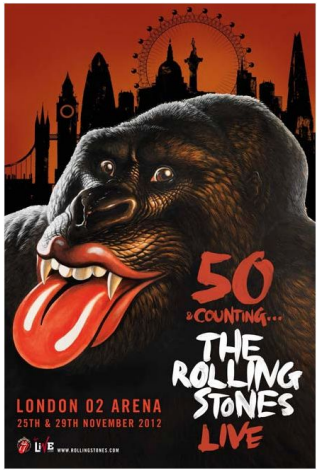
$10^6 \times 10^7 = 10^{13}$ lectures de lignes de la table de commandes

Si chaque lecture prend 10^{-9} secondes, cela représente à peu près 2h45 pour obtenir les 10^7 éléments de la réponse...

Il faut quelques secondes à un SGBD pour répondre à une telle requête avec des données beaucoup plus grandes.

Interagir avec des données dans un contexte de concurrence

Que se passe-t-il lorsque l'on met en vente des billets pour les deux concerts des Rolling Stones à Londres en 2012 (les 32000 billets ont été vendus en sept minutes) ?



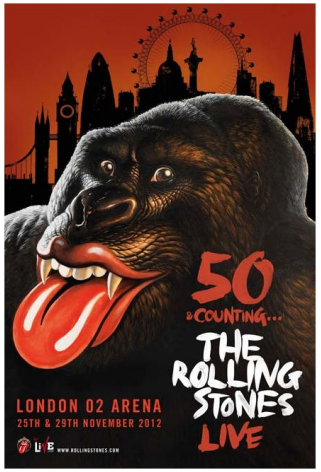
Interagir avec des données dans un contexte de concurrence

Que se passe-t-il lorsque l'on met en vente des billets pour les deux concerts des Rolling Stones à Londres en 2012 (les 32000 billets ont été vendus en sept minutes) ?

- Des milliers de connexions à la base de données gérant les places se déroulent en même temps,

Interagir avec des données dans un contexte de concurrence

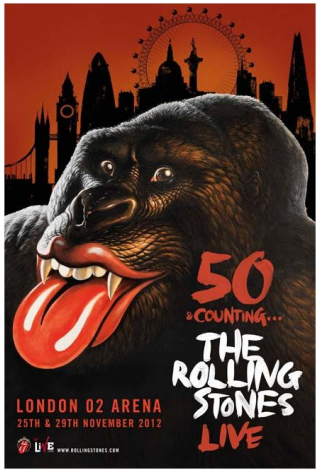
Que se passe-t-il lorsque l'on met en vente des billets pour les deux concerts des Rolling Stones à Londres en 2012 (les 32000 billets ont été vendus en sept minutes) ?



- Des milliers de connexions à la base de données gérant les places se déroulent en même temps,
- forcément certaines places sont demandées au même moment,

Interagir avec des données dans un contexte de concurrence

Que se passe-t-il lorsque l'on met en vente des billets pour les deux concerts des Rolling Stones à Londres en 2012 (les 32000 billets ont été vendus en sept minutes) ?



- Des milliers de connexions à la base de données gérant les places se déroulent en même temps,
- forcément certaines places sont demandées au même moment,
- on veut garantir que chaque place n'est achetée qu'une seule fois.

Garantir la durabilité des données

Dans une entreprise, les bases de données font partie du *code hérité* (legacy code). Une fois construites, elles doivent fonctionner pendant des dizaines d'années. Cela implique plusieurs choses :

- les données doivent persister si les machines qui les stockent doivent être éteintes ou mises à jour. Elles doivent être sur un support permanent (typiquement des disques durs),
- il faut pouvoir les transférer d'une machine à une autre à mesure que la technologie évolue et que le matériel s'use,
- les données doivent survivre et rester cohérentes même lorsque l'infrastructure qui les contient vient à tomber en panne.

Garantir la durabilité des données

Dans une entreprise, les bases de données font partie du *code hérité* (legacy code). Une fois construites, elles doivent fonctionner pendant des dizaines d'années. Cela implique plusieurs choses :

- les données doivent persister si les machines qui les stockent doivent être éteintes ou mises à jour. Elles doivent être sur un support permanent (typiquement des disques durs),
- il faut pouvoir les transférer d'une machine à une autre à mesure que la technologie évolue et que le matériel s'use,
- les données doivent survivre et rester cohérentes même lorsque l'infrastructure qui les contient vient à tomber en panne.

Que se passe-t-il lorsqu'une panne survient lors d'une utilisation intensive de la base ? Des requêtes sur la base de données peuvent être interrompues par la panne, il faut cependant être capable de retrouver un état cohérent lors du redémarrage du système.

Plan

Le modèle relationnel

- La seule structure manipulée par un SGBD est celle de la (multi)-relation, représentée par des tables : il s'agit de (multi-)ensembles de tuples, composés d'éléments de type simple.

Le modèle relationnel

- La seule structure manipulée par un SGBD est celle de la (multi)-relation, représentée par des tables : il s'agit de (multi-)ensembles de tuples, composés d'éléments de type simple.
- Les relations représentées par une base de données sont données par un schéma :
  - noms des relations,
  - liste des noms et des types des attributs de chaque relation,
  - contraintes logiques sur les attributs.

Exemple :

```
CREATE TABLE Artiste  (idArtiste INTEGER NOT NULL,
          nom VARCHAR (30) NOT NULL,
          prénom VARCHAR (30) NOT NULL,
          annéeNaiss INTEGER,
          PRIMARY KEY (idArtiste),
          UNIQUE (nom, prénom));
```

L'algèbre relationnelle

Alors que la logique ou SQL permettent de décrire les tuples que l'on recherche dans une base de données, l'algèbre relationnelle est d'une nature plus algorithmique.

L'algèbre relationnelle

Alors que la logique ou SQL permettent de décrire les tuples que l'on recherche dans une base de données, l'algèbre relationnelle est d'une nature plus algorithmique.
Les relations sont manipulées par des opérations :

L'algèbre relationnelle

Alors que la logique ou SQL permettent de décrire les tuples que l'on recherche dans une base de données, l'algèbre relationnelle est d'une nature plus algorithmique.
Les relations sont manipulées par des opérations :

- Opérateurs algébriques :
  - une ou plusieurs relations en entrée,
  - une et une seule relation en sortie.

L'algèbre relationnelle

Alors que la logique ou SQL permettent de décrire les tuples que l'on recherche dans une base de données, l'algèbre relationnelle est d'une nature plus algorithmique.
Les relations sont manipulées par des opérations :

- Opérateurs algébriques :
  - une ou plusieurs relations en entrée,
  - une et une seule relation en sortie.
- Requêtes algébriques :
  - termes construits à partir des opérateurs,
  - les feuilles de ces termes sont les relations de la base interrogée.

Opérations de l'algèbre relationnelle

- Sélection (σ) : filtre les tuples suivant une propriété,
- Projection (π) : élimine des colonnes de la relation,
- Produit ($\times$) : produit cartésien de relations
- Jointure ($\bowtie$) : combine des relations en assurant des propriétés sur des colonnes,
- Différence ($-$) : $R - T$ contient les tuples de R qui ne sont pas dans T,
- Union ($\cup$) : $R \cup T$ contient tous les tuples de R et de T.

Exemple sélection

Relation R :

A	B	C
1	2	5
3	4	6
1	2	7
1	2	7

$\sigma_{C \geq 6 \wedge B < 3}(R)$

Exemple sélection

Relation R :

A	B	C
1	2	5
3	4	6
1	2	7
1	2	7

$\sigma_{C \geq 6 \wedge B < 3}(R)$

A	B	C
1	2	7
1	2	7

Exemple projection

Relation R :

A	B	C
1	2	5
3	4	6
1	2	7
1	2	7

$\pi_{A,B}(R)$

Exemple projection

Relation R :

A	B	C
1	2	5
3	4	6
1	2	7
1	2	7

$\pi_{A,B}(R)$

A	B
1	2
3	4
1	2
1	2

Exemple produit

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$R \times S$

Exemple produit

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$R \times S$

A	R.B	S.B	C
1	2	2	3
1	2	2	3
1	2	4	5
1	2	4	5
1	2	4	5
1	2	4	5

Exemple jointure

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$R \bowtie S$

Exemple jointure

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$R \bowtie S$

A	B	C
1	2	3
1	2	3

Exemple jointure-θ

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$$R \bowtie_{R.B<S.B} S$$

Exemple jointure-θ

Relation R :

A	B
1	2
1	2

Relation S :

B	C
2	3
4	5
4	5

$$R \bowtie_{R.B<S.B} S$$

A	R.B	S.B	C
1	2	4	5
1	2	4	5
1	2	4	5
1	2	4	5

Opérateurs étendus de l'algèbre relationnelle

- **Élimination des doublons** (δ) : élimine toute redondance de la relation,
- **Opérateurs d'aggrégation** (sum, count, avg, min, max) : somme, comptage, moyenne, minimum, maximum,
- **Regroupement** : partitionne une relation suivant les valeurs d'attributs,
- ...

Opérateurs étendus de l'algèbre relationnelle

- **Élimination des doublons** (δ) : élimine toute redondance de la relation,
- **Opérateurs d'aggrégation** (sum, count, avg, min, max) : somme, comptage, moyenne, minimum, maximum,
- **Regroupement** : partitionne une relation suivant les valeurs d'attributs,
- ...

Les extensions de l'algèbre relationnelle permettent de capturer l'ensemble de SQL. Certains opérateurs posent des problèmes algorithmiques intéressants.

Cycle de vie d'une requête SQL - lien avec l'algèbre

- Analyse syntaxique et sémantique : construction de l'arbre représentant l'expression algébrique de la requête
- La représentation algébrique est appelée *plan logique* de la requête. Il est réécrit afin de trouver un plan logique équivalent qui devrait prendre moins de temps à l'exécution
- A partir du plan logique, on génère un plan physique d'exécution. Là encore, il faut trouver le plan le moins coûteux (estimation !)
- Exécution du plan physique par le moteur d'exécution des requêtes.

la semaine prochaine nous verrons comment accéder au plan d'exécution d'une requête, et comment l'interpréter.

Plan logique

- Il existe certaines règles syntaxiques pour transformer le plan logique en un plan logique équivalent moins coûteux
- Le coût est estimé selon la taille des résultats (relations) intermédiaires

Exemple :

Tables
```
StarsIn(title, year, starName)
Movie(title, year, length, studioName, ...)
```

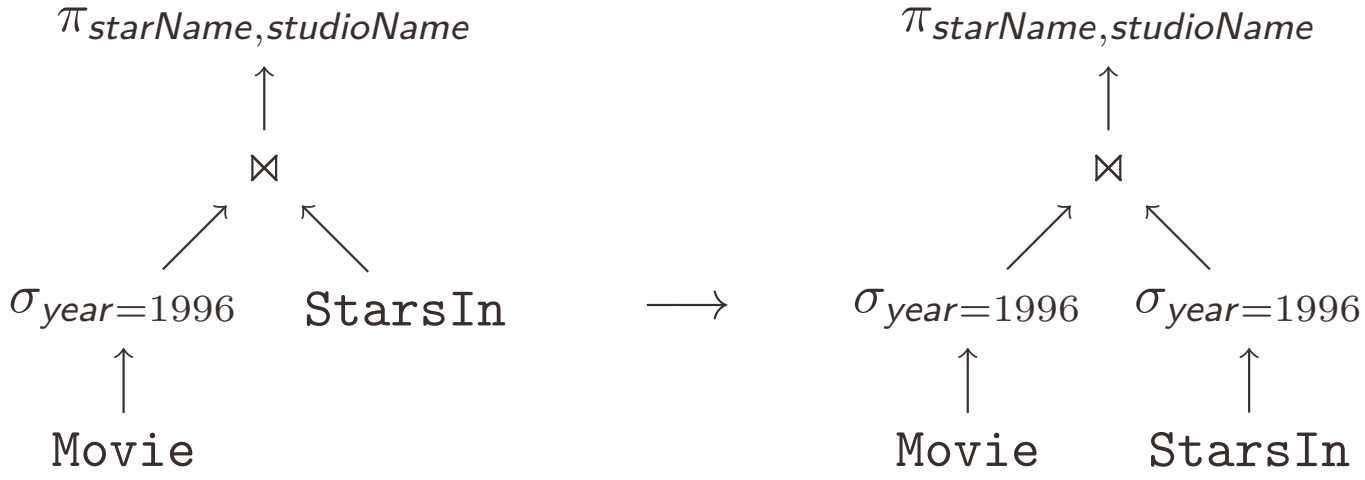
Vue
```
create view MovieOf1996 as
select * from Movie
where year = 1996
```

$\sigma_{year=1996}(Movie)$

Exemple

Requête :

```
select starName, studioName
from MovieOf1996 natural join StarsIn
```



Plan physique d'exécution

- A partir du plan logique choisi, on construit un plan physique d'exécution.
- Il existe différents plans physiques :
  - pour les opérations associatives-commutatives ($\bowtie$, $\cup$, $\cap$), il faut choisir un ordre d'exécution, éventuellement grouper les opérations
  - Trouver un algorithme pour chaque opérateur du plan logique
  - Ajouter des opérateurs supplémentaires (tris, ...) qui n'apparaissent pas dans le plan logique
  - Définir comment on va passer les arguments d'un opérateur à l'autre (mémoire centrale, écritures disques intermédiaires, ...)
- Parmi tous ces plans, on choisit celui qui semble le moins coûteux ; le coût est essentiellement le nombre de lectures/écritures disque.

Cycle de vie d'une requête SQL - vue d'ensemble