



Optimisation de SQL

Fabien Coelho
MINES ParisTech

Composé avec l'EPX, révision 24/98

1

Optimisation des requêtes

algèbre requêtes relationnelles équivalentes

jointures associatives, restriction éventuellement distributive...

méthode calculs équivalents

parcours ou utilisation d'index pour les recherches

choix fonction de coûts approximative

Pour le meilleur ? Au moins pour éviter le pire !

3

Plan de calcul d'une requête

– choix d'arbres d'évaluation d'une requête

requêtes algébriquement équivalentes

différentes opérations élémentaires (selon les index)

– évaluation du coût d'un plan

statistiques sur les valeurs et tailles des données

comparaison avec une exécution effective

– choix du meilleur : problème d'**optimisation** difficile

combinatoire ordre des jointures ? placement des filtres ?

heuristiques des optimiseurs, techniques IA...

EXPLAIN...

ANALYZE

EXPLAIN ANALYZE...

5

Stockage des données : la hiérarchie mémoire

un processeur passe son temps à attendre les données !

type	taille	latence
registre	< 1 Ko	1 ns
cache	100 Ko-4 Mo	10 ns
ram	512 Mo-4 Go	100 ns
disque	100 Go-1 To	10 ms
bande	1 To	1 s-1 h

Écart RAM/HD : ×100,000

7

Besoin d'optimisation des requêtes

modèle relationnel élégant... mais efficace ?

15 ans de recherche et développement pour bonnes implémentations !

types de requête avec des besoins différents

transactionnel accès à quelques tuples seulement

– les trouver vite : utilisation d'index

décisionnel calcul systématique de statistiques

– toutes les données parcourues !

– recherche du meilleur plan de calcul...

– pas d'utilisation d'index

2

Opérations élémentaires de calcul

parcours+filtrage d'une relation

recherche des tuples satisfaisant une condition

tri d'une relation

fixe l'ordre des tuples

jointure+condition de deux relations

mise en correspondance de tuples selon une condition

agrégation d'une relation

regroupement des éléments selon un critère

etc. par exemple pour opérations ensemblistes...

WHERE...

ORDER BY...

JOIN ON...

GROUP BY, DISTINCT...

4

Coût des opérations élémentaires

taille des données : nombre de tuples, d'attributs, leurs tailles

valeur des données ! sélectivité d'une conditions (proba. d'être vrai)

implique souvent la taille des résultats !

WHERE id = 110; -- **clef primaire (1)**

WHERE promo = 110; -- **Fraction des élèves**

WHERE age = 110; -- **age de la personne (0%)**

WHERE age <> 110; -- **(100%)**

ressources disponibles : mémoire cache, charge machine...

statistiques min, max, moyenne, distribution avec **ANALYZE**

6

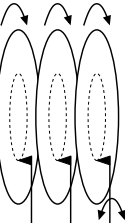


IBM 350 Disk File

- 1956
- HD du 305 RAMAC
- ram account. syst.
- 24 disques
- capacité 4,4 MB !
- 5 M Tbit chars
- location
- \$35,000 par an

8

Technologie des disques

- plateaux, faces, cylindres, secteurs, blocs
- 
- pérenne par rapport à la mémoire vive (panne courant)
 - entrées-sorties limitent les performances *I/O bound*
 - si la base est petite, la charger en mémoire !
 - gestionnaire de mémoire cache spécifique ? cache système ?

9

Mesure de coûts : I/O disques !

- unité d'accès** page postgresql : 8 KB
- accès séquentiel** typiquement 10 à 100 MB/s
- mesure du débit du disque
- variation : bus IDE/SCSI, vitesse rotation, RAID, charge...
- accès aléatoire** typiquement 1 à 10 MB/s
- latence du disque en ms, déplacement du bras et rotations
- un ordre de grandeur du séquentiel!
- calculs et accès mémoire** simplement négligés!
- très rapide par rapport aux accès disques

10

Paramètres de modélisation I/O disques

- B taille d'un bloc de la base en octet (vs bloc FS)
- k lecture séquentielle d'un bloc en seconde
- $K \gg k$ lecture aléatoire d'un bloc en seconde
- n_R nombre de tuples d'une relation R
- t_R taille en octets d'un tuple de R
- $S_R = n_R t_R$ taille de la relation
- s_R taille en octet d'un attribut particulier
- g probabilité de regroupement (coefficient d'aggrégation)
- j probabilité de jointure selon une condition...
- M mémoire disponible (cache, données temporaires)

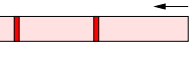
11

Parcours+Filtrage : parcours séquentiel

- sequential scan**
- lecture d'une relation sur disque
- filtrage des tuples au cours du parcours

```
SELECT * FROM oeuvres
WHERE titre LIKE 'A%';

seq scan on oeuvres
filter: titre LIKE 'A%'
```



12

Parcours+Filtrage : parcours Indexé

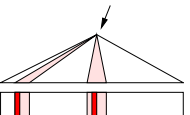
Index scan

accès *direct* à quelques tuples... mais coût de l'utilisation de l'index!

$$K + K \ln(n_s/B)$$

```
SELECT * FROM oeuvres
WHERE id BETWEEN 12 AND 14
AND titre LIKE 'A%';
```

```
index scan using oeuvres_pkey on oeuvres
index cond: id>=12 AND id<=14
filter: titre LIKE 'A%'
index scan using oeuvres_titre_idx on oeuvres
index cond: titre LIKE 'A%'
filter: id>=12 AND id<=14
```



13

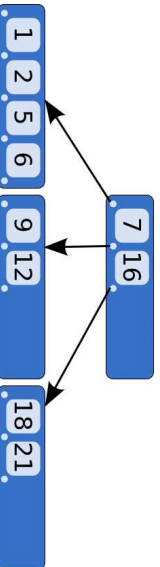
Indexation des attributs d'une relation principes simples, détails compliqués et subtils

- hashage** sur disque
 - partitionnement des données avec fonction de hashage
 - permet des recherches = <>
- dichotomie** nombreuses variantes *B-tree* (*B-balanced*)
 - suppose une relation d'ordre entre les éléments!
 - index denses ? partiels ? unique ? multi-niveau ? ...
 - opérations élémentaires ? conservation de l'équilibrage ?
 - adaptés à des recherches = <> et par intervalle < <= > =>
- autres** adaptation de la dichotomie à des cas plus complexes
- R-tree* (*region tree*) pour les données géographiques

14

B-tree

- arbre *n*-aire, séparation sur valeurs limites
- feuilles référencent les tuples



15

Création d'un index

- unique (pour une clé) ou non
- précise la méthode d'indexation
- la/les attributs/expressions indexés **ensemble(s)**
- création automatique **UNIQUE**, e.g. clés primaires

```
CREATE [ UNIQUE ] INDEX idxname
ON tablename USING [ hash | btree | rtree | gist ]
( { column | (expr) } [, ...] ) [ WHERE predicat ];
```

16

Importance des index

- utilisation **éventuelle** par les requêtes !
- vérification des contraintes **INSERT UPDATE DELETE**
- existence de clefs étrangères, unicité d'une clef... **CLUSTER...**
- forcer l'ordre de stockage
- parcours séquentiel dans l'ordre d'un index particulier

Coût des index

- espace disque utilisé
- maintenance à chaque opération **INSERT UPDATE DELETE**

17

Index sur un attribut

```
SELECT * FROM oeuvre
WHERE titre = 'Triple Jeu';

CREATE INDEX oeuvre-titre
ON oeuvre(titre);
```

18

Index avec prédicats

- indexation partielle : taille et coût réduits
- restreint à une recherche avec ce prédicat
- utile pour contraintes partielles... *si telle condition, tel attribut est unique...*

```
SELECT * FROM answer
WHERE idusr=12 AND istast;
```

```
CREATE INDEX active_user_answer
ON answer (idusr)
WHERE istast;
```

19

Index fonctionnels

- indexation de calculs sur les attributs
- restreint à une recherche avec cette fonction *immuable*

```
SELECT titre FROM oeuvre
WHERE UPPER(titre)='SLALOMS';

CREATE INDEX titre_upper
ON oeuvre
USING BTREE ((UPPER(titre)));
```

20

Tri d'une relation

- tri fusion *merge sort* de bloc triés
- espace mémoire et disque utilisable ?
- meilleures techniques avec parcours séquentiels... $kXn \ln(n)$
- nombreuses variantes...
- demandée par l'utilisateur, ou techniques de calcul

ORDER BY évidemment !

GROUP BY et regroupement des données contiguës

DISTINCT similaire au précédent

JOIN ON jointure par fusion

21

Jointures de relations

- relation (= ou autre...) entre attribut(s), clefs ou non...
- tables jointes de cardinalité n_R et n_P , résultat $j_{n_R n_P}$
- évaluation de j selon les données ?

n_{nom}	d_{pR}	d_{nom}
Avon	77	13
Briare	45	Bouches du Rhône
Calors	46	Indre
Figeac	46	Loiret
Fontainebleau	77	Lot
Glen	45	Paris
Hérivy	77	Savoie
Marseille	13	Seine et Marne

22

Jointure par fusion triée

- tri des relations selon le critère de jointure !
- puis parcours parallèle et fusion au vol des relations $\frac{k}{2}(n_R \ln(n_R) + n_P) + t_P(Y n_P \ln(n_P) + n_P) + t_{RP} j_{n_R n_P}$
- écriture ou exploitation au vol du résultat ?

n_{nom}	d_{pR}	d_{nom}
Marseille	13	Bouches du Rhône
Briare	45	Indre
Orléans	45	Loiret
Glen	45	Lot
Calors	46	Savoie
Figeac	46	Paris
Paris	75	Seine et Marne
Villaines	77	

Jointure par hash

- relation mise en **mémoire**, indexée par critère de jointure
- puis parcours séquentiel de l'autre relation $\frac{k}{2}(n_R n_P + k n_P + t_{RP} j_{n_R n_P})$
- la mémoire doit être suffisante... risque de *swap* ?

Autres techniques de calculs de jointure

- **boucles** différentes variantes, avec index...
- **filtre** produit cartésien, puis filtrage...

23

24

Agrégations

- regroupements si attributs égaux : `GROUP BY, DISTINCT`
- évaluation du taux de regroupement $g?$

Département	Ville
Seine et Marne	Avon
Loiret	Bièvre
Lot	Cahors
Lot	Figeac
Seine et Marne	Fonainebeau
Loiret	Gien
Seine et Marne	Héricy
Bouches du Rhône	Marseille

25

Agrégation par fusion triée

- tri de la relation selon le critère d'agrégation
- groupement ou agrégation des éléments contigus

$$\frac{k}{B} (X_n \ln(n) + tn + gtn)$$

Département	Ville	Département	Nb villes
Bouches du Rhône	Marseille	Bouches du Rhône	1
Loiret	Bièvre	Loiret	3
Loiret	Orléans	Loiret	2
Lot	Gien	Paris	1
Lot	Cahors	Seine et Marne	7
Paris	Figeac	Paris	
Seine et Marne	Paris	Valaines	

26

Agrégation par hash

- allocation en mémoire du résultat $gtn!$
- remplissage lors du parcours, puis écriture ou exploitation

$$\frac{k}{B}(tn + gtn)$$

27

Informations statistiques

1. statistiques sur données pour optimisation *analyze*
2. historique requêtes et performances *log*
3. état courant (connexions, verrous, requêtes) *systeme*
4. opérations tables/indexes *stat rel (row)*
5. comptages accès disque/cache *stat io (block)*

29

Collecte d'Informations : requêtes, temps (attention au coût)

log.connection connexions à la base

log.statement requête exécutée

log.duration durée d'exécution

```
[15611] LOG: query:
SELECT DISTINCT domaine, libmin
FROM thes, state1
WHERE state1.code_dipl=thes.domaine
AND domaine BETWEEN 50000 AND 59999
ORDER BY libmin
[15611] LOG: duration: 7.228555 sec
```

31

Informations sur les volumes

- pg.column.size** volume d'une donnée
- pg.database.size** volume d'un catalogue
- pg.relation.size** volume d'une table, d'un index
- pg.total_relation.size** complet avec index et *toasted*
- pg.tablespace.size** volume d'un espace (partition ?)
- pg.size.pretty** interprétation avec unités (KB, M...)

```
SELECT pg_size_pretty(pg_database_size('test'));
```

pg_size_pretty
8949 kb

28

Table pg_statistic

- statistiques sur les valeurs de chaque colonne de chaque table
 - mise à jour par la commande `ANALYZE` (siron, vague défaut)
 - utilisé par l'optimiseur pour évaluer les tailles de relations
- filtrage** nombreux opérateurs
- regroupement**
- ```
WHERE ...
GROUP BY ...
JOIN ... ON ...
```
- jointure**
- selon le type analysé : min, max, moyenne, distribution...
  - problème de droits d'accès : vue `pg_stats`

30

### Configuration de la collection de statistiques

- stats\_start\_collector** collecteur démarré
- processus séparé de collecte, pas temps-réel
- stats\_command\_string** requête en cours
- stats\_row\_level** statistiques sur les tuples
- insert, update, fetch via indexes* ou scans
- stats\_block\_level** accès blocs cache ou disque

32

### Consultation des informations collectées

- nombreuses tables systèmes spéciales pg\_stat\*
- niveau database, table et index, table, index ...
- opération scan avec ou sans index, ins/upd/del...
- **cache read** (disque) vs **hit** (mémoire)
- **obsolete fetch** vs **read**
- Informations différées ... pas trop de charge de collecte
- détection des indexes inutilisés, des tables scannées...

33

### .activity connexions et requêtes en cours (si longue)

| datid | datname   | procpid | username  | current_query                 | query_start   |
|-------|-----------|---------|-----------|-------------------------------|---------------|
| 19007 | corrector | 563     | corrector | SELECT COUNT(*) FROM answer2; | 2005-04-12... |
| 19007 | corrector | 549     | postgres  | <IDLE>                        | 2005-04-12... |
| 19209 | comics    | 676     | coelho    | <IDLE>                        | 2005-04-12... |

34

### -database par base de données

- nombre de connexions en cours
- nombre de transactions confirmées commit/annulées rollback
- blocks lus sur disque, lus dans le cache

| datid | datname   | numbackends | xact_commit | xact_rollback | blks_read | blks_hit |
|-------|-----------|-------------|-------------|---------------|-----------|----------|
| 17232 | coelho    | 0           | 112         | 7             | 149       | 9077     |
| 19007 | corrector | 2           | 87633       | 7             | 1085      | 9681100  |
| 19209 | comics    | 0           | 7765        | 2533          | 716       | 1047755  |
| 19499 | comest    | 0           | 326         | 171           | 347       | 82055    |
| ...   | ...       | ...         | ...         | ...           | ...       | ...      |

35

### -user.tables utilisation logique des tables (all user sys)

- nb parcours séquentiels et nb tuples lus (selon taille table)
- nb parcours indexés et nb tuples lus (comparables ?)
- nb d'insertions, mises à jour et effacement

| relname    | sq_scan | sq_lup_read | idx_scan | idx_lup_fetch | n_lup_ins | n_lup_upd | n_lup_del |
|------------|---------|-------------|----------|---------------|-----------|-----------|-----------|
| a.ecrit    | 75      | 31935       | 0        | 0             | 492       | 0         | 0         |
| collection | 81      | 3535        | 0        | 0             | 87        | 0         | 0         |
| banque     | 44      | 48          | 0        | 0             | 2         | 0         | 0         |
| exemplaire | 82      | 20766       | 0        | 0             | 279       | 0         | 0         |
| editeur    | 106     | 2586        | 0        | 0             | 67        | 0         | 0         |
| auteur     | 204     | 15064       | 0        | 0             | 151       | 0         | 16        |
| oeuvre     | 91      | 2387        | 119      | 116           | 298       | 13        | 0         |

36

### Io.user.tables utilisation disque/cache des tables all user sys

- nombre de blocks table lus sur disque/cache
- index de blocks des indexes de la table lus sur disque/cache
- idem pour données toastées (attributs volumineux)...

| relname    | heap_read | heap_hit | idx_read | idx_hit |
|------------|-----------|----------|----------|---------|
| oeuvre     | 3         | 1048     | 6        | 505     |
| exemplaire | 4         | 506      | 2        | 50      |
| editeur    | 2         | 271      | 4        | 168     |
| a.ecrit    | 2         | 894      | 3        | 198     |
| auteur     | 2         | 605      | 2        | 154     |
| langue     | 2         | 85       | 0        | 0       |
| collection | 2         | 287      | 4        | 172     |

37

### -user.indexes analyse par index all user sys

- nb parcours avec l'index
- nb tuples lus par l'index
- nb tuples récupérés par l'index
- différent nb tuples lus pour tuples expriés

| relname    | indexrelname        | idx_scan | idx_lup_read | idx_lup_fetch |
|------------|---------------------|----------|--------------|---------------|
| auteur     | auteurs_pkey        | 36       | 2414         | 2414          |
| collection | collections_nom_key | 2        | 86           | 86            |
| collection | collections_pkey    | 1        | 43           | 43            |
| exemplaire | exemplaires_pkey    | 0        | 0            | 0             |
| oeuvre     | oeuvres_pkey        | 1106     | 9935         | 9935          |
| oeuvre     | oeuvres_livre_key   | 15       | 3069         | 3069          |
| a.ecrit    | a.ecrit_pkey        | 127      | 53975        | 53975         |
| ...        | ...                 | ...      | ...          | ...           |

38

### Io.user.indexes nb blocs lus sur le disque/cache all user sys

| relname    | indexrelname        | idx_blks_read | idx_blks_hit |
|------------|---------------------|---------------|--------------|
| auteur     | auteurs_pkey        | 2             | 73           |
| collection | collections_nom_key | 2             | 5            |
| collection | collections_pkey    | 2             | 3            |
| editeur    | editeurs_nom_key    | 2             | 1            |
| editeur    | editeurs_pkey       | 2             | 1            |
| exemplaire | exemplaires_pkey    | 2             | 1            |
| langue     | langues_nom_key     | 2             | 1            |
| langue     | langues_pkey        | 2             | 1            |
| oeuvre     | oeuvres_livre_key   | 2             | 2213         |
| oeuvre     | oeuvres_pkey        | 4             | 60           |
| a.ecrit    | a.ecrit_pkey        | 4             | 511          |

39

### Io.user.sequences nb accès disque/cache all user sys

| relid | schemaname | relname                  | blks_read | blks_hit |
|-------|------------|--------------------------|-----------|----------|
| 19008 | public     | connection_idcom_seq     | 1         | 3        |
| 19019 | public     | users_idusr_seq          | 1         | 19       |
| 19034 | public     | class_idcls_seq          | 1         | 2        |
| 19121 | public     | correctiontype_idcor_seq | 1         | 13       |
| 19133 | public     | question_idque_seq       | 1         | 75       |
| 19062 | public     | exercice_idexe_seq       | 1         | 1        |
| 19163 | public     | answer_idans_seq         | 1         | 659      |
| 19078 | public     | session_idses_seq        | 1         | 3        |

40

### Optimisation matérielle des entrées-sorties

- vitesse des disques 5000 à 10000 tr/min
- types de disques IDE/ATA/RAID...
- séparation des flux sur des disques différents
- wal** écriture des transactions
- tablespace** localisation des données
- logs** syslog sans fsync ?
- os** exécutable, etc.

41

### Conclusion sur les performances

- optimisation** difficile : selon valeurs, statistiques nécessaires
- contrôle** explicite ?
- algorithme générique si trop gros (10 tables ?)
  - forcer les jointures : `join_collapsed_limit=1`
- indexation** accélère, mais coût en maintenance
- choix des index et types pour les requêtes fréquentes
  - collecte statistiques, utilisation des logs...
- caches** données souvent sollicitées gardées en mémoire...

43

### Évolutions des *Solid-State Drive* : SSD

- mémoire flash permanente** (appareils photos, MP3)
- cache de disques magnétiques ? disques complets ?
- performances** des SSD : R/W, débit/lance, aléa/séq
- lecture  $\approx$  écriture (100 MB/s)
  - aléatoire  $\approx$  séquentiel
  - cycle W page/E block 10K-100K, éc
  - **effets d'usure** : si plein écriture alé
- impact** coûts  $\times 30$ , capacité  $\times \frac{1}{4}$
- performances & optimisations ? vitesse



42

- influence** du matériel à l'application
- matériel disponible, dédié ou partagé à l'application, duplication
  - ruptures technologiques : SSD vs HD
  - configuration de l'OS, de la base de donnée...
  - maintenance DB `VACUUM FULL ANALYZE...`
  - application : requêtes inutiles, relations mal conçues...

44

### List of Slides

- 1 Optimisation de SQL
- 2 Besoin d'optimisation des requêtes
- 3 Optimisation des requêtes
- 4 Opérations élémentaires de calcul
- 5 Plan de calcul d'une requête
- 6 Coût des opérations élémentaires
- 7 Stockage des données : la hiérarchie mémoire
- 8 IBM 350 Disk File
- 9 Technologie des disques
- 10 Mesure de coûts : I/O disques I
- 11 Paramètres de modélisation I/O disques

- 12 Parcours+Filtrage : parcours séquentiel
- 13 Parcours+Filtrage : parcours indexé
- 14 Indexation des attributs d'une relation
- 15 B-tree
- 16 Création d'un index
- 17 Importance des index
- 17 Coût des index
- 18 Index sur un attribut
- 19 Index avec prédicats
- 20 Index fonctionnels
- 21 Tri d'une relation
- 22 Jointures de relations
- 23 Jointure par fusion triée

- 24 Jointure par hash
- 24 Autres techniques de calculs de jointure
- 25 Agrégations
- 26 Agrégation par fusion triée
- 27 Agrégation par hash
- 28 Informations sur les volumes
- 29 Informations statistiques
- 30 Table `pg_statistic`
- 31 Collecte d'informations : requêtes, temps (attention au coût)
- 32 Configuration de la collection de statistiques
- 33 Consultation des informations collectées
- 41 Optimisation matérielle des entrées-sorties
- 42 Évolutions des *Solid-State Drive* : SSD

43 Conclusion sur les performances