Optimisation des Entrepôts de Données

Optimisation des requêtes

- **Caractéristiques des entrepôts**
  - Grand volume de données (TB)
  - Requêtes décisionnelles complexes
  - Peu de mises à jour (ajout seulement!)
- **Exigence des décideurs**
  - Temps de réponse raisonnable
  - Respect des délais

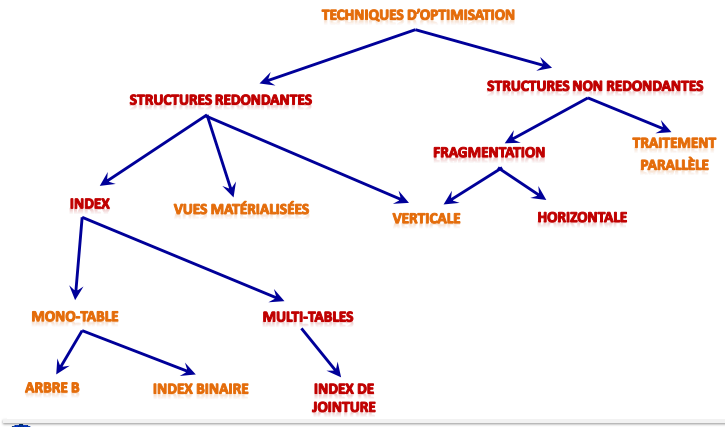
Objectif Exécution rapide des requêtes ∀ la complexité des requêtes ou des données

➡ **Nécessité de techniques d'optimisation**

Techniques utilisées

- **Vues matérialisées**
  - Une vue est une requête nommée
  - Une vue matérialisée : les données résultant de sa requête sont stockées et maintenues
- **Index**
  - Structures permettant d'associer à une clé d'un n-uplet l'adresse relative de cet n-uplet
- **Traitement parallèle**
- **Groupement**
- **Fragmentation**

Classification



Vues matérialisées

- **Exigence de ressources:**
  - Espace disque
  - Coût de maintenance (rafraîchissement des données)
  - Coût de calcul

➡ Impossibilité de matérialiser toutes les vues

- **Problème de Sélection des Vues (PSV):**

  - Sélectionner un ensemble de vues afin de maximiser ou minimiser une fonction objectif sous une contrainte

  - Fonction « objectif »:

    Optimiser le coût d'exécution des requêtes ou Optimiser le coût de maintenance

Exemple

```
CREATE MATERIALIZED VIEW VUE1
ENABLE QUERY REWRITE
AS
SELECT  *
FROM Vente, Client, Article
 WHERE Vente.noClient = Client.noClient AND
Vente.noArticle = Article.noArticle
```

Maintenance des vues matérialisées

- Vues matérialisées sont calculées à partir des sources (tables)
- Mise à jour des sources implique la mise à jour des vues
- Deux méthodes de maintenance
  - Statique
  - Incrémentale

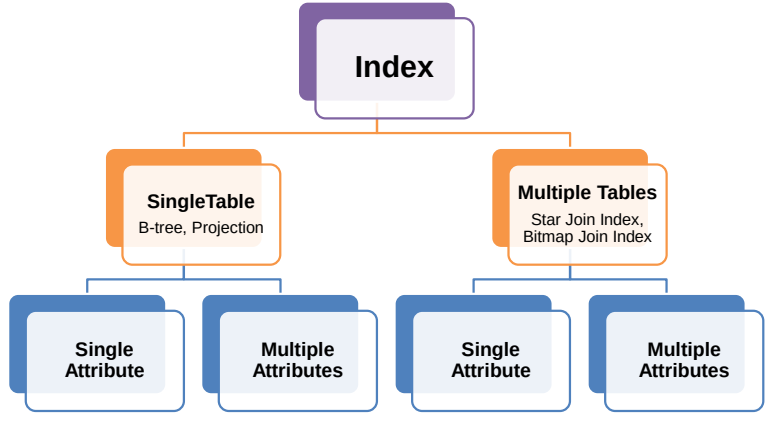
Réécriture des requêtes

- Processus de réécriture:
  - Après le processus de sélection des vues, toutes les requêtes définies sur l'entrepôt doivent être réécrites en fonction des vues
- Sélectionner la meilleure réécriture pour une requête est une tâche difficile.
- Processus supporté dans la plupart des SGBD multidimensionnels (Oracle)

Index avancés

- Index populaires dans les entrepôts:
  - Index simple: sur une seule table ou une seule vue (arbre B, index binaire, index de projection)
  - Index de jointure: sur plusieurs tables dans un schéma en étoile
    - index de jointure en étoile
    - Index de jointure binaire
- Problème de sélection d'index
- Algorithmes de sélection d'index

Classification des Index



Techniques d'indexation

❑Index bitmap (index binaire)

❑ Populaire dans les produits OLAP

- Microsoft SQL server, Sysbase IQ, Oracle
- Un vecteur de bits pour chaque valeur d'attribut

❑ Opération bit à bit pour l'exécution de requêtes

- Comparaison
- Jointure
- Agrégation
- Plus compact que les B+ arbres (compression)

Principe de l'index bitmap

- Soit un attribut A, ayant prenant n valeurs possibles $[v_1, \ldots, v_n]$ (domaine)
- Création d'un index bitmap sur l'attribut A:
  - On crée n tableaux de bit, un pour chaque valeur v_i
  - Le tableau contient un bit pour chaque tuple t
  - Le bit d'un tuple t est à 1 si: $t.A = v_i$, à 0 sinon

Table Employé

ROWID(RID)	N°E	Nom	Fonction
00055 :000 :0023	1	Karim	Fraiseur
00234 :020 :8922	2	Hichem	Soudeur
19000 :328 :6200	3	Salim	Tourneur
21088 :120 :1002	4	Ali	Sableur

Index binaire sur l'attribut Fonction

ROWID(RID)	Soudeur	Fraiseur	Sableur	Tourneur
00055 :000 :0023	0	1	0	0
00234 :020 :8922	1	0	0	0
19000 :328 :6200	0	0	0	1
21088 :120 :1002	0	0	1	0

Index de jointure

- Pré-calculent une opération de jointure
- Utilisés avec les schémas en étoile
- Maintient des relations entre
  - Une clé étrangère (identifiant des tables de dimensions)
  - Les clés primaires de la table des faits

Index de jointure

IdLigneVente	idLigneClient	idLigneArticle
V1	C1	A1
V2	C2	A2
V3	C3	A1
…		

Table Client

IdLigne	noClient	nomVille
C1	1	Poitiers
C2	2	Nantes
C3	3	Poitiers
C4	4	Paris

Table Ventes

IdLigne	noClient	noArticle	DateVente	Montant
V1	1	10	10/01/2008	100
V2	2	20	10/01/2008	200
V3	3	10	10/01/2008	500
…				

Table Article

IdLigne	noArticle	catégorie
A1	10	Beauté
A2	20	Beauté
A3	40	Beauté
A4	50	Jouet
A5	60	Jouet

Index de jointure Bitmap : Exemple

Client

RID^C	CID	Nom	Ville
6	616	Gilles	Poitiers
5	515	Yves	Paris
4	414	Patrick	Nantes
3	313	Didier	Nantes
2	212	Eric	Poitiers
1	111	Pascal	Poitiers

Produit

RID^P	PID	Nom	Catégorie
6	106	Sonoflore	Beauté
5	105	Clarins	Beauté
4	104	WebCam	Multimédia
3	103	Barbie	Jouet
2	102	Manure	Jardinage
1	101	SlimForm	Fitness

Temps

RID^T	TID	Mois	Année
6	11	Janvier	2003
5	22	Février	2003
4	33	Mars	2003
3	44	Avril	2003
2	55	Mai	2003
1	66	Juin	2003

Ventes

RID^S	CID	PID	TID	Montant
1	616	106	11	25
2	616	106	66	28
3	616	104	33	50
4	515	104	11	10
5	414	105	66	14
6	212	106	55	14
7	111	101	44	20
8	111	101	33	27
9	212	101	11	100
10	313	102	11	200
11	414	102	11	102
12	414	102	55	103
13	515	102	66	100
14	515	103	55	17
15	212	103	44	45
16	111	105	66	44
17	212	104	66	40
18	515	104	22	20
19	616	104	22	20
20	616	104	55	20
21	212	105	11	10
22	212	105	44	10
23	212	105	55	18
24	212	106	11	18
25	313	105	66	19
26	313	105	22	17
27	313	106	11	15

	Ville		
RID	P	Pr	N
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	0	0	1
11	0	0	1
12	0	0	1
13	0	1	0
14	0	1	0
15	1	0	0
16	1	0	0
17	1	0	0
18	0	1	0
19	1	0	0
20	1	0	0
21	1	0	0
22	1	0	0
23	1	0	0
24	1	0	0
25	0	0	1
26	0	0	1
27	0	0	1

```
SELECT    count(*)
FROM  Ventes V, Client C
WHERE  V.CID = C.CID
AND  C.Ville = 'Poitiers'
```

```
CREATE BITMAP INDEX Ventes_Pictaviens_bjix ON Ventes(Client.Ville) FROM Ventes, Client WHERE Ventes.CID = Client.CID
```

Deux opérations :

1. Lire dans le bitmap la colonne P
2. Compter le nombre de 1.

➔ Pas de lecture dans la table Ventes

Caractéristiques des Bitmaps

- Les opérations de comparaison, jointure et d'agrégation sont réduites à une arithmétique sur les bits, d'où un traitement plus efficace
- La réponse à une requête multidimensionnelle va consister à faire l'intersection des vecteurs de bits des diverses dimensions
- Réduction significative en espace et nombre d'accès en mémoire secondaire

Intérêt des index de jointure binaire

```
CREATE BITMAP INDEX EMPDEPT_IDX ON
 EMP1(DEPT1.DEPTNO)
FROM EMP1, DEPT1
WHERE EMP1.DEPTNO = DEPT1.DEPTNO
```

```
SELECT /*+ INDEX(EMP1 EMPDEPT_IDX) */ COUNT(*)
FROM EMP1, DEPT1
WHERE EMP1.DEPTNO = DEPT1.DEPTNO;
```

```
COUNT(*)
----------
        14
Elapsed: 00:00:00.67
```

Bitmap Join Index Example 2

Deux tables EMP5/EMP6 ayant 2 million d'instances chacune, empno est la clé étrangère

```
create bitmap index emp5_j6 on
emp6(emp5.empno) from emp5, emp6
where emp5.empno=emp6.empno;
```

Index created.

Elapsed: 00:02:29.91

Bitmap Join Index Example 2

```
select count(*)
from emp5, emp6
where emp5.empno=emp6.empno
```

COUNT(*)

2005007

Elapsed: 00:01:07.18

Bitmap Join Index Example 2

Plan d'exécution

```
0       SELECT STATEMENT Optimizer=CHOOSE
1   0   SORT (AGGREGATE)
2   1     NESTED LOOPS
3   2       TABLE ACCESS (FULL) OF 'EMP6'
4   2       INDEX (RANGE SCAN) OF 'EMP5I_EMPNO' (NON-UNIQUE)
```


Bitmap Join Index Example 2

FORCER L'UTILISATION DE L'INDEX

```
select /*+ index(emp6 emp5_j6) */ count(*)
from emp5, emp6
where emp5.empno=emp6.empno
```

```
COUNT(*)
--------
2005007
```

Elapsed: 00:00:00.87! Comme avec les tables de petites tailles!

Bitmap Join Index - 10,000 plus rapide

Plan d'exécution

0		SELECT STATEMENT Optimizer=CHOOSE
1	0	SORT (AGGREGATE)
2	**1**	**BITMAP CONVERSION (COUNT)**
3	**2**	**BITMAP INDEX (FULL SCAN) OF 'EMP5_J6'**

Selection of BJI : Formalization

INPUT

- Set of dimension tables $D=\{D_1, D_2, ..., D_d\}$
- Fact Table F
- Workload of frequently queries Q
- Storage Space S

OUTPUT

- Set S_BJI of BJI

OBJECTIVES

- Reduce execution cost of Q
- Size (S_BJI) $\leq$ S

Selection of BJI : Complexity

- $A=\{A_1, A_2, \ldots, A_K\}$ is a set of indexable attributes
- A potential BJI is defined on a subset of A
- N the number of potential BJI

Select One BJI

$$N = \binom{k}{1}+\binom{k}{2}+\ldots\ldots+\binom{k}{k} = 2^k - 1$$

Select More Than One BJI

$$N = \binom{2^k-1}{1}+\binom{2^k-1}{2}+\ldots\ldots+\binom{2^k-1}{2^k-1} = 2^{2^k-1}$$