

**Département d’informatique et mathématiques**

**8TRD157 – Base de données avancées**

Travail pratique 4 : Optimisation de requêtes et analyse de la performance

|  |  |
| --- | --- |
| **Cours** | 8TRD157 – Base de données avancées |
| **Étudiant 1 (nom et CP)** |  |
| **Étudiant 2 (nom et CP)** |  |

|  |  |
| --- | --- |
| **Rapport** | **(100 pts)** |
| **Code SQL pour le schéma interne**  **(création d'index et CLUSTER)** | **/ 15** |
| **Reformulation des requêtes avec indices**  **(et nouvelles tables)** | **/ 15** |
| **Méthodologie expérimentale (plans d'exécution et statistiques)** | **/ 15** |
| **Analyse des résultats** | **/ 40** |
| **Choix de la configuration (jointure double et sélection) et justification** | **/ 15** |
|  |  |
| **Total** | **/ 100** |

# Opérations de sélection

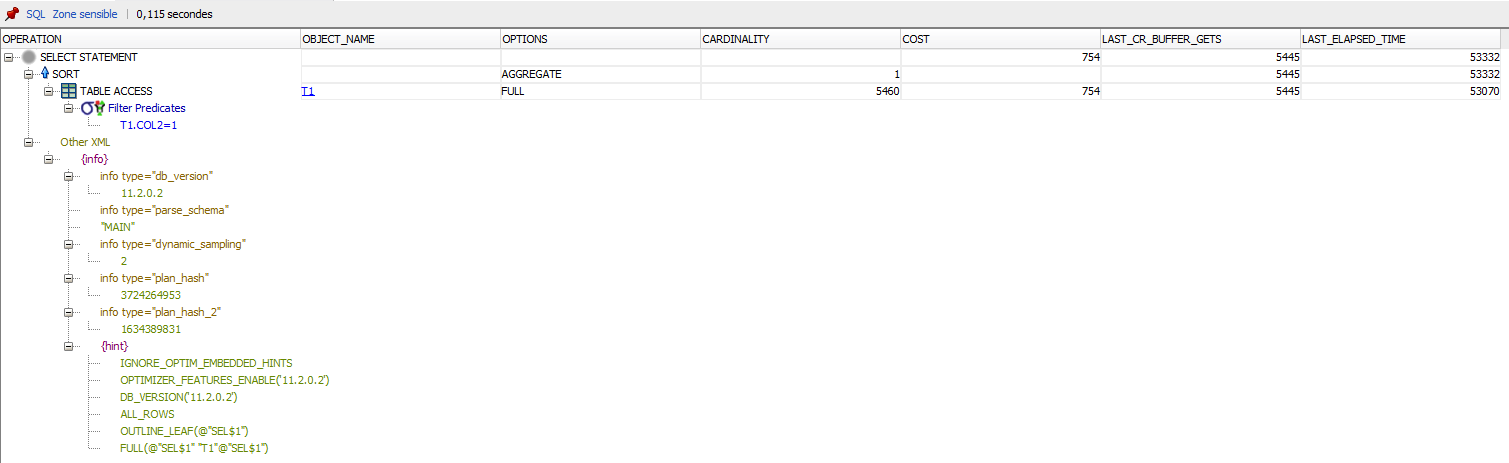
## Sélectivité petite

### Évaluation des optimisations

[Pour chacune des configurations suivantes, donnez 1) le code SQL employé pour créer les indexes ou CLUSTER requis, 2) la requête testée comprenant les indices et, possiblement, les nouvelles tables insérées dans les CLUSTER, 3) le plan d'exécution et les statistiques obtenues lors de l'exécution de la requête]

***Sans optimisation***

SELECT /\*+ FULL\*/ COUNT(\*) FROM T1 WHERE T1.col2=1

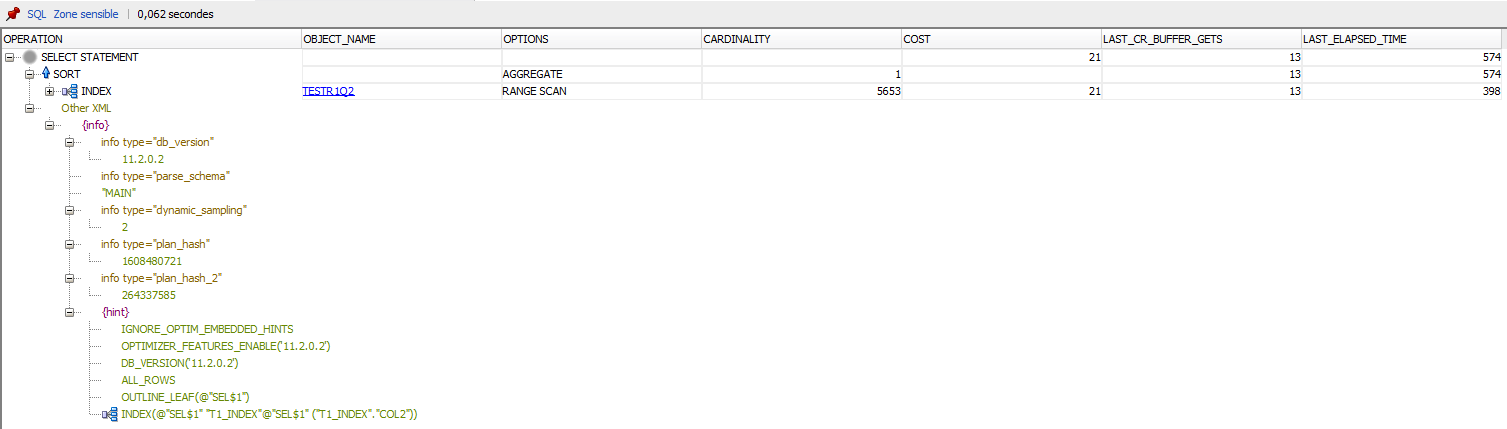


***Avec index secondaire et indice INDEX ou INDEX\_FFS (précisez lequel a été choisi)***

CREATE TABLE T1\_INDEX AS SELECT \* FROM T1;

CREATE INDEX testR1Q2 ON T1\_INDEX(col2);

SELECT /\*+ INDEX(T1\_INDEX col2)\*/ COUNT(\*) FROM T1\_INDEX WHERE T1\_INDEX.col2=1



***Avec HASH CLUSTER***

CREATE CLUSTER Grappe1 (col2 INTEGER)

SIZE 160 HASHKEYS 100;

CREATE TABLE T1\_CLUSTER CLUSTER Grappe1(col2) AS SELECT \* FROM T1;

SELECT /\*+ HASH(T1\_CLUSTER col2)\*/ COUNT(\*) FROM T1\_CLUSTER WHERE T1\_CLUSTER.col2=1

### Analyse globale

[Faites une courte analyse (10-15 lignes) expliquant les résultats obtenus à la lumière de la matière vue en classe]

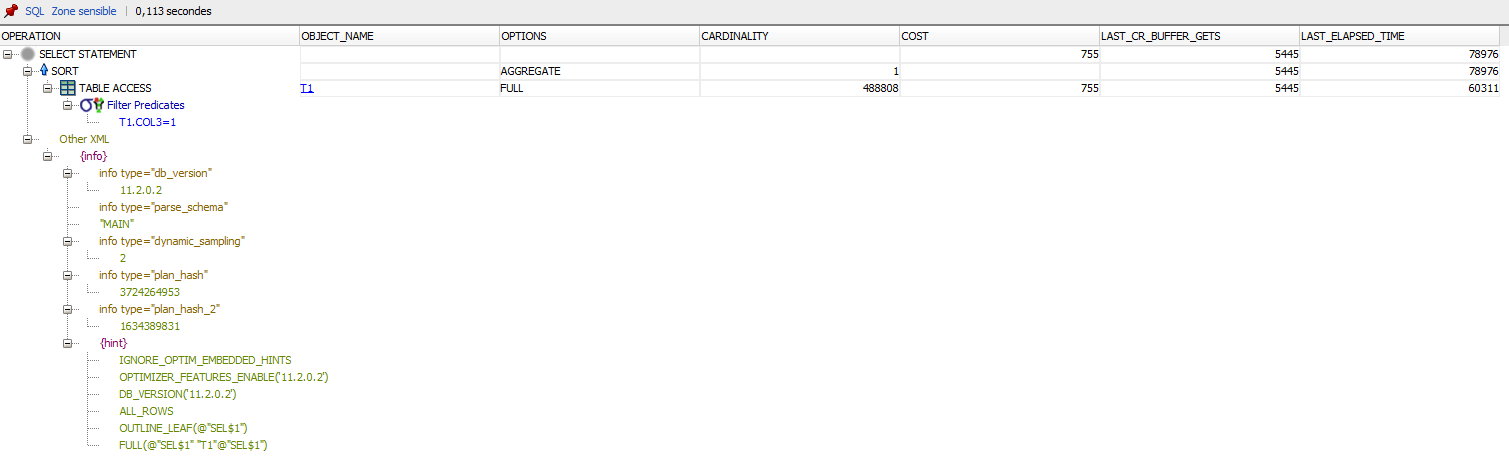
## Sélectivité grande

### Évaluation des optimisations

[Pour chacune des configurations suivantes, donnez 1) le code SQL employé pour créer les indexes ou CLUSTER requis, 2) la requête testée comprenant les indices et, possiblement, les nouvelles tables insérées dans les CLUSTER, 3) le plan d'exécution et les statistiques obtenues lors de l'exécution de la requête]

***Sans optimisation***

SELECT /\*+ FULL \*/COUNT(\*) FROM T1 WHERE T1.col3=1

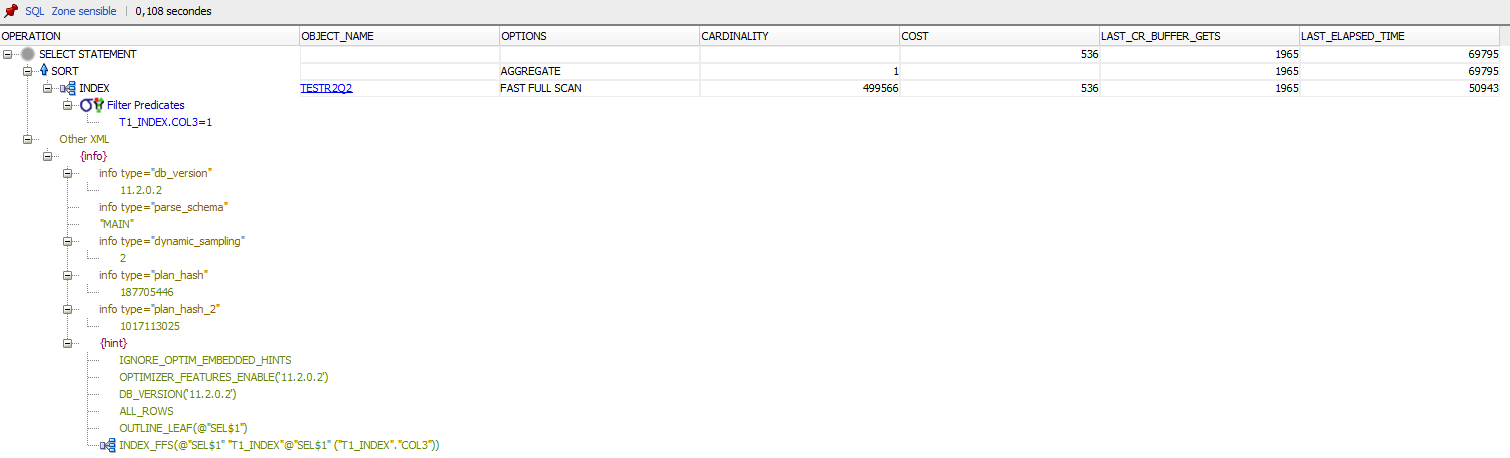


***Avec index secondaire et indice INDEX***

CREATE TABLE T1\_INDEX AS SELECT \* FROM T1;

CREATE INDEX testR2Q2 ON T1\_INDEX(col3);

SELECT /\*+ INDEX(T1\_INDEX col3)\*/ COUNT(\*) FROM T1\_INDEX WHERE T1\_INDEX.col3=1



***Avec HASH CLUSTER***

DROP TABLE T1\_CLUSTER;

DROP CLUSTER Grappe2;

CREATE CLUSTER Grappe2 (col3 INTEGER)

SIZE 160 HASHKEYS 100;

CREATE TABLE T1\_CLUSTER CLUSTER Grappe2(col3) AS SELECT \* FROM T1;

SELECT /\*+ HASH(T1\_CLUSTER col3)\*/ COUNT(\*) FROM T1\_CLUSTER WHERE T1\_CLUSTER.col3=1

***Avec BITMAP INDEX***

Impossible.

### Analyse globale

[Faites une courte analyse (10-15 lignes) expliquant les résultats obtenus en lien avec la matière vue en classe]

# Opérations de jointure

## Jointure simple

### Évaluation des optimisations

[Pour chacune des configurations suivantes, donnez 1) le code SQL employé pour créer les indexes ou CLUSTER requis, 2) la requête testée comprenant les indices et, possiblement, les nouvelles tables insérées dans les CLUSTER, 3) le plan d'exécution et les statistiques obtenues lors de l'exécution de la requête]

***Sans optimisation***

***Avec indexes secondaires et indice USE\_NL\_WITH\_INDEX ou USE\_HASH ou USE\_MERGE (pécisez lequel a été choisi)***

***Avec un HASH CLUSTER contenant les 2 tables (hétérogène)***

***Avec deux INDEX CLUSTER, chacun contenant une des tables (homogène)***

### Analyse globale

[Faites une courte analyse (10-15 lignes) expliquant les résultats obtenus]

## Jointure double avec sélection

### Choix de la configuration

[Décrivez et justifiez birèvement le choix de la configuration retenue pour l'optimisation de cette requête.]

### Analyse des résultats avec différents ordres de jointure

[Faites une courte analyse (10-15 lignes) expliquant les résultats obtenus en changeant l'ordre des jointures]

Note :

An index fast full scan reads the ENTIRE index, unsorted, as it exists on disk. It is basically using the index as a "skinny" version of the table. The query in question would only be accessing attributes in the index (we are not using the index as a way to get to the table, we are using the index INSTEAD of the table) We use multiblock IO and read all of the leaf, branch and the root block. We ignore the branch and root blocks and just process the (unordered) data on the leaf blocks.  
  
An index full scan is when we read the index a block at a time - from start to finish. We'll read the root block, navigate down the left hand side of the index (or right if we are doing a descending full scan) and then when we hit the leaf block - we'll read across the entire bottom of the index - a block at a time - in sorted order. We use single block IO, not multiblock IO for this operation.