Chapitre 5: Optimisation des instructions SQL

Plan d'exécution

Fadwa FATHI

2025-2026

REQUÊTES SQL

- Une application bien conçue peut se heurter à des problèmes de performances , si les instructions SQL qu'elle utilise sont mal construites.
- La clé dans l'optimisation SQLest réduire au minimum le chemin de recherche que la base utilise pour localiser les données

Optimisation des requêtes :

- Oracle conserve la trace du chemin d'accès d'une requête dans une table appelée « plan_table»
- Tous les graphes des requêtes du LID et LMD sont conservées:

  SELECT, UPDATE , INSERT et DELETE

- Chaque opération sur un objet est notée avec:

  - L'ordre d'exécution
  - L'objet consulté (table, index, cluster, view …)
  - Le type d'opération

Optimisation des requêtes

- Dans le but de fournir l'algorithme d'accès à la base de données pour répondre à une requête exprimée en langage assertionnel, la requête passe par plusieurs étapes allant de l'analyse jusqu'à l'exécution. En comprenant les mécanismes internes, nous pouvons contrôler le temps qu'Oracle passe à évaluer l'ordre de jointure destables, et améliorer les performances des requêtes en général.
- Oracle permet d'analyser les requêtes et de connaître le temps et le plan d'exécution, ces informations permettent de définir ce qui ralentit l'exécution des requêtes afin de les optimiser. Une fois que les requêtes lentes détectées, lancer la commande EXPLAIN permet de comprendre l'exécution et donc connaître ou intervenir pour optimiser.

Préparation de l'instruction SQL

Lorsqu'une instruction SQLentre dansla mémoire cachede la bibliothèque Oracle,
les étapes suivantes se succèdentavant que l'instruction ne soit prête à être exécutée

- **Contrôle de la syntaxe : le système vérifie que l'instruction SQLest correcte en**

termes d'orthographe et d'ordre desmots.

- **Analyse sémantique : Oracle vérifie toutes les tables et les noms de colonnes**

ainsi que les privilèges accordés par rapport au dictionnaire de données.

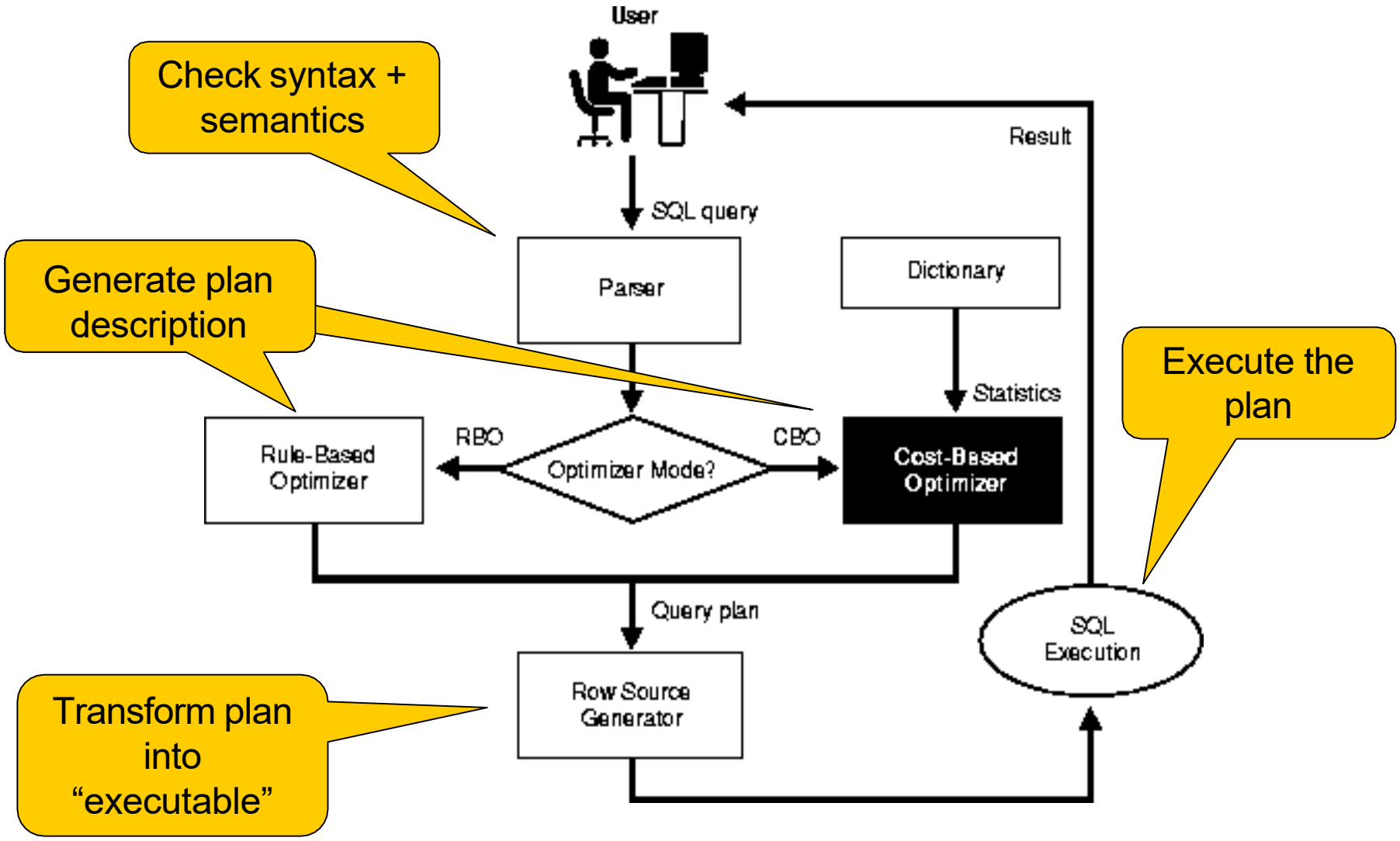
- **Vérification du schéma stocké (stored outline) : Oracle vérifie dans le**

dictionnaire des données s'il existe un schéma stocké pour l'instruction SQL.

- **Génération du plan d'exécution : Oracle utilise des algorithmes et des**

statistiques d'optimisation baséssur le coût dans le dictionnaire de données pour déterminer le plan d'exécution optimal.

- **- Création du code binaire: Oracle génère un exécutable binaire en fonction du** plan d'exécution.


User
Check syntax + semantics
Result
SQL query
Generate plan description
Parser
Dictionary
Statistics
Execute the plan
Rule-Based Optimizer
RBO
Optimizer Mode?
CBO
Cost-Based Optimizer
Query plan
SQL Execution
Row Source Generator
Transform plan into "executable"

Génération d'un plan

- Outils;
  - EXPLAIN
  - AUTOTRACE
  - V$SQL_PLAN

8

1-Génération de plan d'exécution: EXPLAIN PLAN

- Lacommande explainplan détermine le chemin d'accès que la base de données utiliserapour satisfaire à une requête.

- Cette instruction évalue le chemin d'exécution d'une requête et palace les résultantsde sonévaluation dansune table nommée PLAN_TABLE

- Oracle conserve la trace du chemin d'accès d'une requête dans une table appelée « plan_table »

```
Plan d'exécution
----------------------------------------------------------------------
| Id  | Operation             |  Name       | Rows  | Bytes | Cost  |
----------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |    14 |   518 |     2 |
|   1 |  TABLE ACCESS FULL    | ARTISTE     |    14 |   518 |     2 |
----------------------------------------------------------------------

Statistiques
```

- Id : Identifiant de l'opérateur ;
- Operation : type d'opération utilisée
- Name : Nom de la relation utilisée ;
- Cost: coût estimé par oracle;
- Rows : Le nombre de lignes qu'Oracle pense transférer. ,
- Bytes : Nombre d'octets qu'oracle pense transférer.

1- EXPLAIN PLAN:

EXPLAIN PLAN
[SET statement_id=id] (option)
[INTO table_name] (option)
FOR sql_statement

- La première ligne de cette commande indique à la base de données qu'elle va expliquer le pland'exécution pour larequête sansréellement l'exécuter.
- Ladeuxième lignemarque les enregistrements relatifs àcette requête dansla table PLAN_TABLEavec une valeurid dans la colonne Statement_Id.
- Larequête à analysersuit le mot clé for

```
EXPLAIN PLAN FOR <QUERY>;

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
```

Script :

```
CREATETABLE Customers (
        CustomersId int,
        CustomerName  varchar(255),
        City varchar(255),
        Country varchar(255)
    );
INSERTINTO CustomersVALUES(100 , 'Cardinal', 'Stavanger', 'Norway');
```

Affichage du plan d'explication avec la méthode EXPLAIN

```
EXPLAIN PLAN FOR SELECT * FROM Customers;
EXPLAIN PLAN FOR SELECT * FROM Customers WHERE CustomersId = 100;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY());
SELECT * FROM Customers WHERE CustomersId = 100;
SELECT * FROM PLAN_TABLE;
```

2-AUTOTRACE:

```
SET AUTOTRACE ON;
```

```
SET AUTOTRACE ON [EXPLAIN|STATISTICS];
```

```
SET AUTOTRACE TRACE[ONLY] ON [EXPLAIN|STATISTICS];
```

```
SET AUTOTRACE OFF;
```

Script :

- set autotrace on;
- set autotrace traceonly explain;
- set autotrace traceonly statistics;
- set autotrace traceonly;
- show autotrace;
- Set autotrace off;

3-V$SQL_PLAN:


V$SQLAREA
V$SQL_WORKAREA
V$SQL
V$SQL_PLAN
V$SQL_PLAN_STATISTICS
V$SQL_PLAN_STATISTICS_ALL
V$SQL_PLAN

Type d'opération:

- Chaque opération sur un objet est notée avec :
  - L'ordre d'exécution (ordre des opérateurs)
  - Le chemin d'accès aux objets consultés (table, index, cluster, view, …)
  - Le type d'opération (opérateurs physiques)

Structure de PLAN_TABLE:

STATEMENT_ID	VARCHAR2(30)	id choisi
TIMESTAMP	DATE	date d'exécution
REMARKS	VARCHAR2(80)	
OPERATION	VARCHAR2(30)	opération (plus loin)
OPTIONS	VARCHAR2(30)	option de l'opération
OBJECT_NODE	VARCHAR2(128)	pour le réparti
OBJECT_OWNER	VARCHAR2(30)	propriétaire
OBJECT_NAME	VARCHAR2(30)	nom objet (table, index, ….
OBJECT_INSTANCE	NUMBER(38)	
OBJECT_TYPE	VARCHAR2(30)	(unique, …)
OPTIMIZER	VARCHAR2(255)	
SEARCH_COLUMNS	NUMBER	
ID	NUMBER(38)	n° nœud courant
PARENT_ID	NUMBER(38)	n° nœud parent
POSITION	NUMBER(38)	1 ou 2 (gauche ou droite)
COST	NUMBER(38)	
CARDINALITY	NUMBER(38)	
BYTES	NUMBER(38)	
OTHER_TAG	VARCHAR2(255)	
PARTITION_START	VARCHAR2(255)	
PARTITION_STOP	VARCHAR2(255)	
PARTITION_ID	NUMBER(38)	
OTHER	LONG	

Opérations / Options (1)

Operation	Option	Description
AND-EQUAL		Operation accepting multiple sets of rowids, returning the intersection of the sets, eliminating duplicates. Used for the single-column indexes access path.
	CONVER-SION	`TO ROWIDS` converts bitmap representations to actual rowids that can be used to access the table. `FROM ROWIDS` converts the rowids to a bitmap representation `COUNT` returns the number of rowids if the actual values are not needed
	INDEX	`SINGLE VALUE` looks up the bitmap for a single key value in the index. `RANGE SCAN` retrieves bitmaps for a key value range. `FULL SCAN` performs a full scan of a bitmap index if there is no start or stop key.
	MERGE	Merges several bitmaps resulting from a range scan into one bitmap
	MINUS	Subtracts bits of one bitmap from another. Row source is used for negated predicates. Can be used only if there are nonnegated predicates yielding a bitmap from which the subtraction can take place
	OR	Computes the bitwise `OR` of two bitmaps
CONNECT BY		Retrieves rows in hierarchical order for a query containing a `CONNECT BY` clause

Opérations / Options (2)

Operation	Option	Description
CONCATENATION		Operation accepting multiple sets of rows returning the union-all of the sets
COUNT		Operation counting the number of rows selected from a table
FILTER		Operation accepting a set of rows, eliminates some of them, and returns the rest
FIRST ROW		Retrieval on only the first row selected by a query
FOR UPDATE		Operation retrieving and locking the rows selected by a query containing a `FOR UPDATE` clause
HASH JOIN		Operation joining two sets of rows and returning the result
	SEMI	Hash semi-join

Opérations / Options (3)

Operation	Option	Description
INDEX	**UNIQUE SCAN**	Retrieval of a single rowid from an index
	RANGE SCAN	Retrieval of one or more rowids from an index. Indexed values are scanned in ascending order
	RANGE SCAN DESCENDING	Retrieval of one or more rowids from an index. Indexed values are scanned in descending order
INLIST ITERATOR		Iterates over the operation below it for each value in the `IN`-list predicate
INTERSECTION		Operation accepting two sets of rows and returning the intersection of the sets, eliminating duplicates
MERGE JOIN		Operation accepting two sets of rows, each sorted by a specific value, combining each row from one set with the matching rows from the other, and returning the result
	OUTER	Merge join operation to perform an outer join statement

Opérations / Options (4)

Operation	Option	Description
MINUS		Operation accepting two sets of rows and returning rows appearing in the first set but not in the second, eliminating duplicates
NESTED LOOPS (These are join operations)		Operation accepting two sets of rows, an outer set and an inner set. Oracle compares each row of the outer set with each row of the inner set, returning rows that satisfy a condition.
SORT	**AGGREGATE**	Retrieval of a single row that is the result of applying a group function to a group of selected rows
	UNIQUE	Operation sorting a set of rows to eliminate duplicates
	GROUP BY	Operation sorting a set of rows into groups for a query with a `GROUP BY` clause
	JOIN	Operation sorting a set of rows before a merge-join
	ORDER BY	Operation sorting a set of rows for a query with an `ORDER BY` clause

Opérations / Options (5)

Operation	Option	Description
TABLE ACCESS	**FULL**	Retrieval of all rows from a table
	CLUSTER	Retrieval of rows from a table based on a value of an indexed cluster key
	HASH	Retrieval of rows from table based on hash cluster key value
	BY ROWID	Retrieval of a row from a table based on its rowid
	BY INDEX ROWID	If the table is nonpartitioned and rows are located using index(es)
UNION		Operation accepting two sets of rows and returns the union of the sets, eliminating duplicates
VIEW		Operation performing a view's query and then returning the resulting rows to another operation

Visualisation du graphe
autres exemples (1)

```
select * from clients
where num_client NOT IN
       (select num_client from comptes);
```

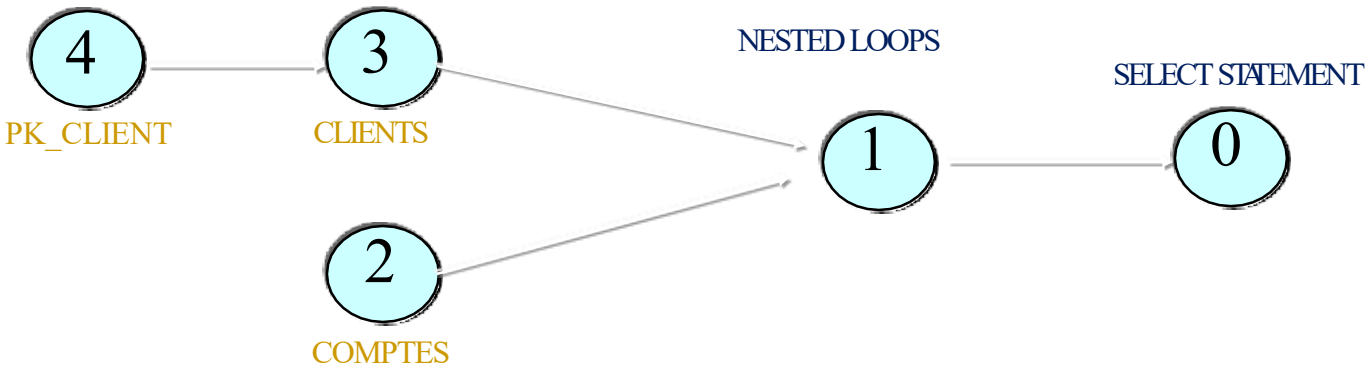
OPERATION	OPTIONS	OBJET	ID	PARENT_ID	POSITION
SELECT STATEMENT			0		
FILTER			1	0	1
TABLE ACCESS	FULL	CLIENTS	2	1	1
TABLE ACCESS	FULL	COMPTES	3	1	2

Visualisation du graphe Mise en œuvre (2)

- Contenu de PLAN_TABLE

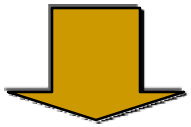
OPERATION	OPTIONS	OBJET	ID	PARE NT_ID	POSITION
SELECT STATEMENT			0		
NESTED LOOPS			1	0	1
TABLE ACCESS	FULL	COMPTES	2	1	1
TABLE ACCESS	BY INDEX ROWID	CLIENTS	3	1	2
INDEX	UNIQUE SC AN	PK_CLIENT	4	3	1

- Graphe correspondant



Visualisation du graphe
autres exemples (2)

```
select * from clients
where num_client   IN
(select num_client from clients
  minus
select num_client from comptes);
```



OPERATION	OPTIONS	OBJET	ID	PARENT_ID	POSITION
SELECT STATEMENT			0		
NESTED LOOPS			1	0	1
VIEW		VW_NSO_1	2	1	1
MINUS			3	2	1
SORT	UNIQUE		4	3	1
TABLE ACCESS	FULL	CLIENTS	5	4	1
SORT	UNIQUE		6	3	2
TABLE ACCESS	FULL	COMPTES	7	6	1
TABLE ACCESS	BY INDEX ROWID	CLIENTS	8	1	2
INDEX	UNIQUE SCAN	PK_CLIENT	9	8	1