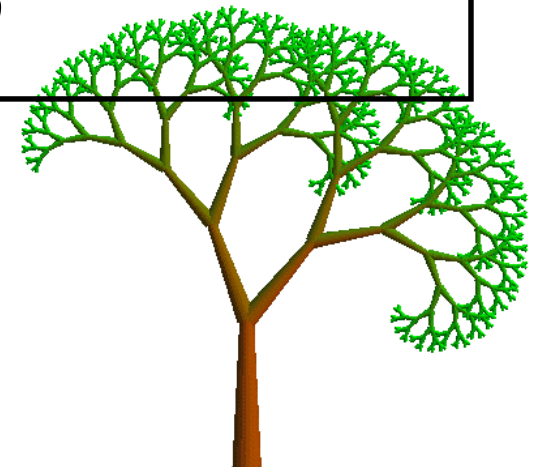




Query Tuning Foundations

Execution Plan

B - Find the Plan





Execution Plan

- The execution plan of a SQL statement is composed of small building blocks called “row sources.”
- The combination of row sources for a statement is called the execution plan.
- By using parent-child relationships, the explain plan can be displayed in a tree (Text or Graphical)



Objectives

- After completing this lesson, you should be able to do the following:
 - Gather/Display Execution plan
 - Understand differences between Plans available
 - Understand purpose of information available with Plan



Where to find Execution Plan

- 1 - PLAN_TABLE (explain plan for ...)
- 2 - V\$SQL_PLAN (Library Cache)
- 3 - DBA_HIST_SQL_PLAN (AWR)
- 4 - STATS\$SQL_PLAN (Statspack)
- 5 - Event 10046 trace file & tkprof
- 6 - Event 10132 trace file
- 7 - Event 10053 trace file
- 8 - Process state dump tracefile in 10g



1 - PLAN_TABLE – Theoretical plan

- Explain plan for ...
- Generates optimizer execution plan
- Stores plan in a PLAN table

```
EXPLAIN PLAN FOR
  SELECT e.ename, d.dname
  FROM emp e, dept d
  WHERE e.deptno = d.deptno;
Explained.
```

- Advantage : SQL is not Execute
- Disadvantage : Maybe NOT actual Plan, Bind not Peeked

Plan_table Output ('TYPICAL')

```
Explain plan for select * from EMP where ename = 'KING';
```

```
select plan_table_output from table(DBMS_XPLAN.DISPLAY());
```

Plan hash value: 351108634

Id	Operation	Name	Rows	Bytes	Cost	

0	SELECT STATEMENT		1	27	2	
* 1	TABLE ACCESS FULL	EMP	1	27	2	

Predicate Information (identified by operation id):

1 - filter("EMP"."ENAME"='KING')

Note

- outline "01" used for this statement

Plan_table Output ('ALL')

```
explain plan for
select * from emp@DD where ename= 'KING'
union all
select * from emp where ename= 'KING';
```

```
select plan_table_output from table(DBMS_XPLAN.DISPLAY(null,null,'ALL'))
```

Id	Operation	Name	Rows	Bytes	Cost	Inst	IN-OUT
0	SELECT STATEMENT		4	232	4		
1	UNION-ALL						
2	REMOTE		1	37	2	DD	R->S
* 3	TABLE ACCESS FULL	EMP	3	195	2		

Predicate Information (identified by operation id):

3 - filter("EMP"."ENAME"='KING')

Remote SQL Information (identified by operation id):

2 - SELECT "EMPNO","ENAME","JOB","SAL","COMM","DEPTNO"
FROM "EMP" "EMP"
WHERE "ENAME"='KING' (accessing 'DD.FR.ORACLE.COM')

Plan_table Output('ADVANCED')

```
var A number
exec :A := 0;
explain plan for select avg(T.i) from T where T.j= :A;
```

```
select plan_table_output from table(dbms_xplan.display('PLAN_TABLE',null, 'ADVANCED'));
```

```
-----
| Id | Operation                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----
|  0 | SELECT STATEMENT          |      |      1 |      7 |    56   (2)| 00:00:01 |
|  1 |   SORT AGGREGATE          |      |      1 |      7 |           |          |
|*  2 |    TABLE ACCESS FULL    | T     |   5000 | 35000 |    56   (2)| 00:00:01 |
-----
```

...

Outline Data

/*+

```
  BEGIN_OUTLINE_DATA
  FULL(@"SEL$1" "T"@"SEL$1")
  OUTLINE_LEAF(@"SEL$1")
  ALL_ROWS
  OPTIMIZER_FEATURES_ENABLE('10.2.0.3')
  IGNORE_OPTIM_EMBEDDED_HINTS
  END_OUTLINE_DATA
```

*/



Plan_table – AUTOTRACE

- Sql*Plus facility
- Introduced with Oracle7.3
- Needs a PLAN_TABLE
- Needs the PLUSTRACE role to retrieve statistics from some V\$ views
- Produces execution plan and/or statistics after running the query
- Maybe NOT the ACTUAL plan when Bind peeking (recursive EXPLAIN PLAN FOR)

Plan_table Output - (AUTOTRACE)

```
var A number
exec :A := 0;
set autotrace on
select avg(T.i) from T where T.j= :A;
      AVG(T.I)
-----
      9999.5
```

Execution Plan

```
-----
0 SELECT STATEMENT Optimizer=ALL_ROWS (Cost=56 Card=1 Bytes=7)
1 0 SORT (AGGREGATE)
2 1 TABLE ACCESS (FULL) OF 'T' (TABLE) (Cost=56 Card=5000 Bytes=35000)
```

Statistics

```
-----
0 recursive calls
0 db block gets
3 consistent gets
0 physical reads
0 redo size
414 bytes sent via SQL*Net to client
504 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
1 rows processed
```



2 - V\$SQL_PLAN - Actual plan

- New in 9i
- Shows the "actual" execution plan
 - Explain Plan gives a theoretical plan
- Contains the execution plan of every cursor in the library cache (including childs)
- Links to V\$SQL
 - address, hash_value, CHILD_NUMBER (9i)
 - SQL_ID, CHILD_NUMBER (10g)
- Enabling execution trace (10046) is another way to display actual execution plan.
 - Plan are the same, information available different

9i v\$sql_plan – Method 1

```
SQL> SeLeCt * from dept d WHERE d.deptno= 10;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

```
select plan_table_output from TABLE( dbms_xplan.display (
    'DYN_PLAN'
    , (select rawtohex(address)||'_'||child_number x
        from v$sql where sql_text like 'SeLeCt%')
    , 'ALL')) ;
```

OR

```
select plan_table_output from TABLE( dbms_xplan.display (
    'DYN_PLAN'
    , (select rawtohex(address)||'_'||child_number x
        from v$sql where hash_value=&HV and child_number=&CH)
    , 'ALL')) ;
```

9i v\$sql_plan Output - Method 1

```
select plan_table_output from TABLE( dbms_xplan.display
('DYN_PLAN'
, (select rawtohex(address)||'_'||child_number x
    from v$sql
    where hash_value=793380943
    and child_number=0)
, 'ALL')) ;
```

```
-----
|Id|Operation                               | Name | Rows | Cost |
-----
| 0|SELECT STATEMENT                       |      |      |      |
| 1| TABLE ACCESS BY INDEX ROWID          | DEPT  |    1 |    1 |
|*2|  INDEX UNIQUE SCAN                     | PK_DEPT|    1 |      |
-----
```

```
2 - access("DEPTNO"=10)
```



9i v\$sql_plan – Method 2

```
select address,hash_value,sql_text from v$sqltext
  where sql_text like '%--%';

create table P1 as select * from v$sql_plan
  where hash_value=&hash_value and address = '&address';

alter table P1 add (STATEMENT_ID VARCHAR2(30));
alter table P1 add (timestamp date);

select distinct child_number from P1;
-- For each child, dump plan like below
update P1 set timestamp=sysdate where child_number = &CHLD;
commit;
select plan_table_output from
  table(dbms_xplan.display('P1',null,'ALL'));
```



9i v\$sql_plan with Row Source Execution Statistics

```
SQL> SELEct * from dept d WHERE d.deptno= 10;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

```
HASH_VALUE CHI SQL_TEXT
```

```
2007160847 - 0 SELEct * from dept d WHERE d.deptno= 10
```

Plan Table

Operation	Name	Starts	E-Rows	A-Rows	Buffers
TABLE ACCESS BY INDEX ROWID	DEPT	1	1	2	1
INDEX UNIQUE SCAN	PK_DEPT	1	1	1	1



10g v\$sql_plan DBMS_XPLAN Output

DBMS_XPLAN.DISPLAY_CURSOR

Last executed statement from v\$sql_plan

```
select * from dept where deptno=10;
```

```
select plan_table_output from  
       table(dbms_xplan.display_cursor()));
```

Id	Operation	Name	Rows	Cost
0	SELECT STATEMENT		1	1
1	TABLE ACCESS BY INDEX ROWID	DEPT	1	1
* 2	INDEX UNIQUE SCAN	PK_DEPT	1	

Predicate Information (identified by operation id):

2 - access("DEPT"."DEPTNO"=10)

V\$sql_plan Output ('ADVANCED')

```
var A number
exec :A := 0;
select avg(T.i) from T where T.j= :A;
```

```
select plan_table_output from table(dbms_xplan.display_cursor(null, null, 'ADVANCED'));
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				2 (100)	
1	SORT AGGREGATE		1	7		
2	TABLE ACCESS BY INDEX ROWID	T	2	14	2 (0)	00:00:01
* 3	INDEX RANGE SCAN	J	2		1 (0)	00:00:01

...

Outline Data

```
-----
/*+
  BEGIN_OUTLINE_DATA
  IGNORE_OPTIM_EMBEDDED_HINTS
  OPTIMIZER_FEATURES_ENABLE('10.2.0.1')
  ALL_ROWS
  OUTLINE_LEAF(@"SEL$1")
  INDEX(@"SEL$1" "T"@"SEL$1" ("T"."J"))
  END_OUTLINE_DATA
*/
```

Peeked Binds (identified by position):

```
-----
1 - :A (NUMBER): 0
```



V\$sql_plan Output with *rowsource* execution statistics

- query ran with statistics_level = ALL
- Format Parameters in DBMS_XPLAN
 - ALLSTATS with/without LAST
 - IOSTATS & MEMSTATS with/without LAST
 - RUNSTAT & RUNSTAT_LAST (for V10.1 compatibility)
- Columns Output
 - CBO Estimation Columns
 - Cost / E-Rows / E-Time / E-Bytes
 - 0Mem / 1Mem
 - Row source execution statistics Columns
 - Starts / A-Rows / A-Time / Buffers
 - Reads / Writes
 - Used-Mem & 0/1/M / Used-Tmp & Max-Tmp



3 - DBA_HIST_SQL_PLAN DBMS_XPLAN output

DBMS_XPLAN.DISPLAY_AWR

- execution plan of all children associated to the SQL ID 'atfwcg8anrykp':

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY_AWR('atfwcg8anrykp'));
```

- **DBA_HIST_SQLSTAT** provides information about execution statistics (SQL_ID, PLAN_HASH_VALUE)
- execution plan of all stored SQL containing the string 'SUPPORT':

```
SELECT tf.*  
FROM DBA_HIST_SQLTEXT ht, table  
(DBMS_XPLAN.DISPLAY_AWR(ht.sql_id  
                        ,null,null, 'ALL' )) tf  
WHERE ht.sql_text like '%SUPPORT%';
```



DBA_HIST_SQL_PLAN – AWR awrsqrpt report

- **Report is available using awrsqrpt.sql script**
 - -> choose begin snapshot
 - -> choose end snapshot
 - -> choose hash value
 - -> get report
- On 10gR2, **DBMS_XPLAN.DISPLAY_AWR** can be used



4- Statspack sprepsql report

- **exec statspack.snap(i_snap_level => 6)**

-> then Plan from Top SQLs are recorded in '**stats\$sql_plan**'

- **Report is available using sprepsql.sql script**

- -> choose begin snapshot
- -> choose end snapshot
- -> choose hash bvalue
- -> get report

- On 10gR2, **DBMS_XPLAN.DISPLAY** can be used
**select * from table(dbms_xplan.display('stats\$sql_plan', null,
 'all', 'hash_value=1054118158 and snap_id=4'));**



5 - Event 10046 trace file & tkprof

- Row source operations (ACTUAL) in trc file
- Execution plan (Theoretical) can be generated with Tkprof using
EXPLAIN = U/P
- Segment statistics are available since 9.2
 - These are cumulative statistics
 - They help to read complex execution plan
- There is no predicate information in trace file



6 - Event 10132 trace file

- Dump plan at the end of compilation
- Plan dumped when HARD parse occurs
- Not available on 8i
- Useful to combine it to 10053 (on 9i)
 - Help to recognize chosen PLAN
- Useful to combine to 10046
 - Provide Cost information
 - predicate information section on 10gR2



7 - 10053 trace file

- 10053 is used to report CBO computations to generate plan
- To have the plan helps to read that trace file
- Since 10gR2, plan is dumped at the end
- Before 10gR2, event 10132 can be used to get it (we have plan dump at exactly same time as CBO computations)



8 - Process state Trace file

- This is new in 10gR2
- During its execution, plan is included in process state dump



Other views with info about SQL execution

- v\$sqlarea
- v\$sqlarea_plan_hash
- v\$sql
- v\$sqlstats
- v\$sql_plan_statistics
- v\$sql_plan_statistics_all
- DBA_HIST_SQLSTAT
- DBA_HIST_SQLTEXT
- DBA_HIST_SQL_SUMMARY
- DBA_HIST_SQL_PLAN