

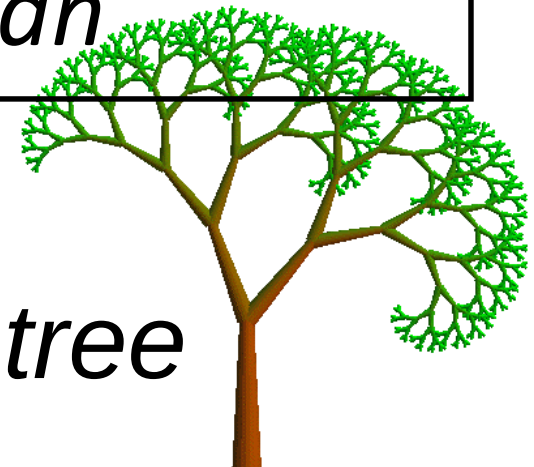


Query Tuning Foundations

Execution Plan

C - Interpret the Plan

Walk through the tree





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

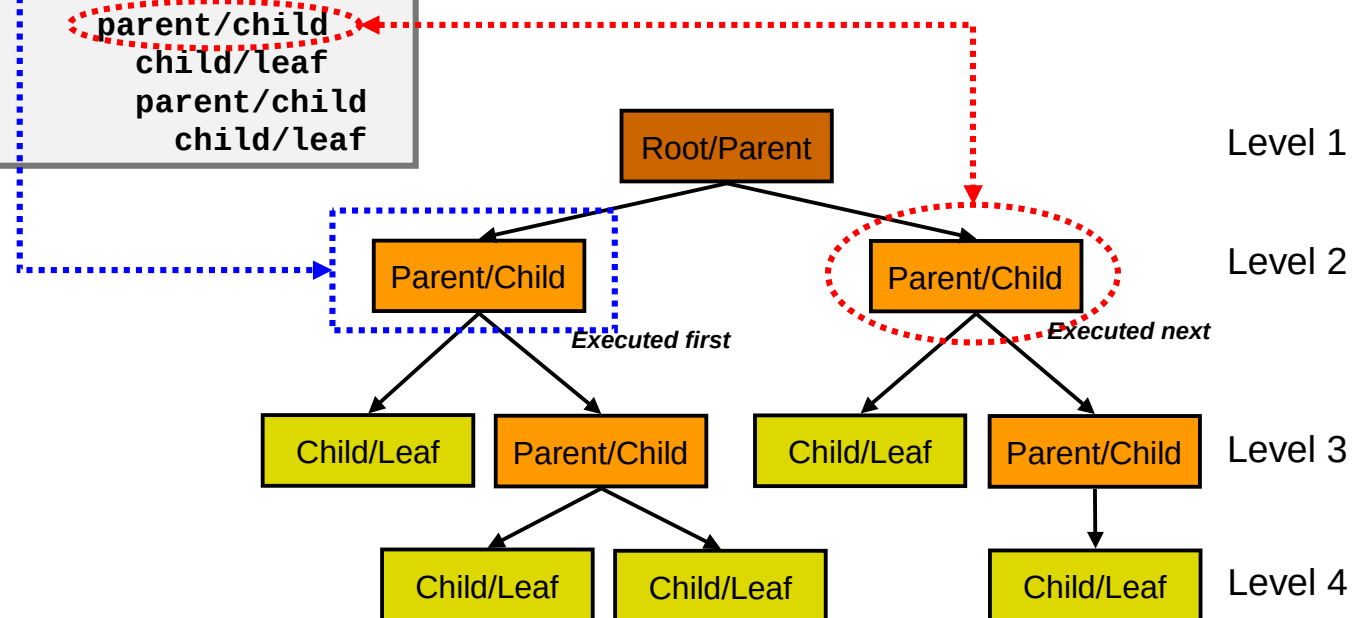
- After completing this lesson, you should be able to do the following:
 - Interpret Execution plan
 - Simplify complex Execution Plan to be able to interpret it

Transform it into a tree

id= 1	(pid=)	root/parent
id= 2	(pid=1) (pos=1)	parent/child
id= 3	(pid=2) (pos=1)	child/leaf
id= 4	(pid=2) (pos=2)	parent/child
id= 5	(pid=4) (pos=1)	child/leaf
id= 6	(pid=4) (pos=2)	child/leaf
id= 7	(pid=1) (pos=2)	parent/child
id= 8	(pid=7) (pos=1)	child/leaf
id= 9	(pid=7) (pos=2)	parent/child
id=10	(pid=9) (pos=1)	child/leaf

PLAN_TABLE is hierarchical

Hierarchy is established with the ID and PARENT_ID (pid) columns.



1 - Interpret Plan - Example 1

```
select /*+ RULE */ ename,job,sal,dname from emp,dept
where dept.deptno = emp.deptno
      and not exists (select *
                      from salgrade
                      where emp.sal between losal and hisal);
```

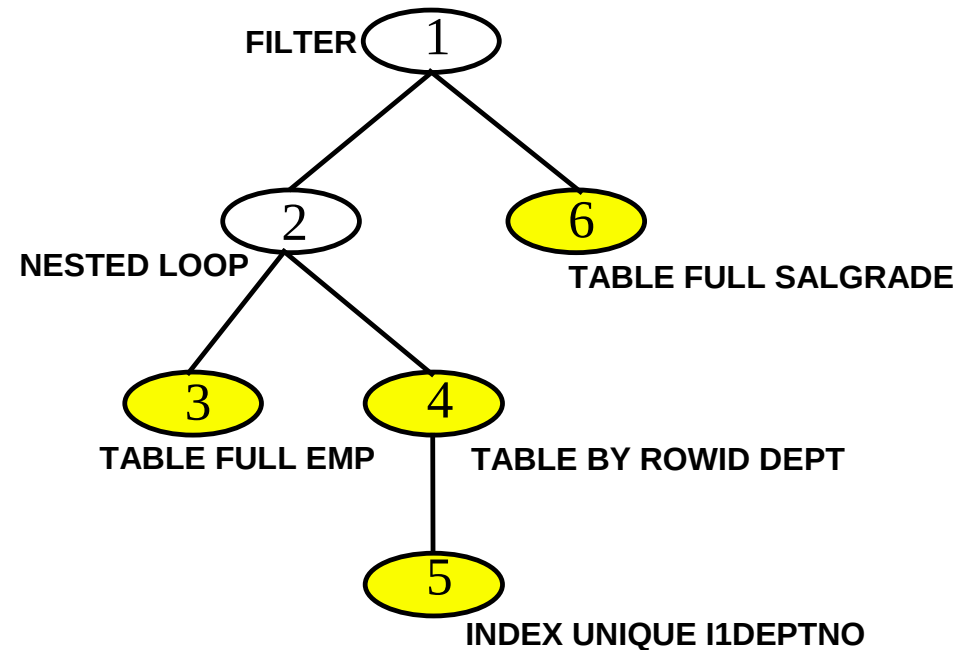
Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	EMP
4	TABLE ACCESS BY INDEX ROWID	DEPT
* 5	INDEX UNIQUE SCAN	PK_DEPT
* 6	TABLE ACCESS FULL	SALGRADE

Predicate Information (identified by operation id):

1 - filter(IS NULL)
5 - access("DEPT"."DEPTNO"="EMP"."DEPTNO")
6 - filter(("HISAL">=:B1 AND "LOSAL"<=:B2))

Note

- rule based optimizer used (consider using cbo)



Execution Order - Example 1

```
select /*+ RULE */ ename,job,sal,dname from emp,dept
where dept.deptno = emp.deptno
and not exists (select *
                 from salgrade
                 where emp.sal between losal and hisal);
```

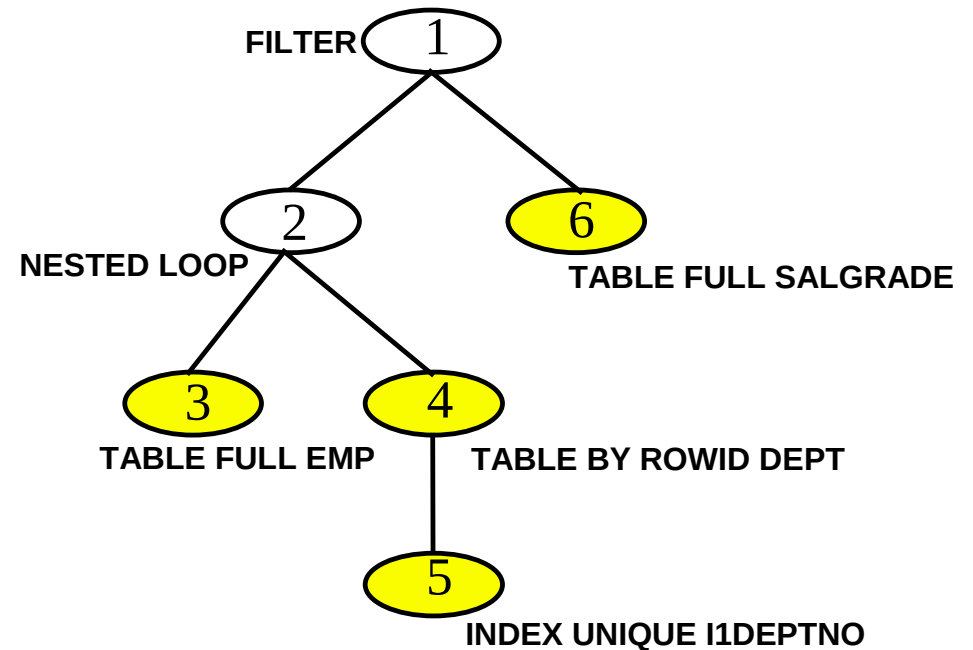
Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	EMP
4	TABLE ACCESS BY INDEX ROWID	DEPT
* 5	INDEX UNIQUE SCAN	PK_DEPT
* 6	TABLE ACCESS FULL	SALGRADE

Predicate Information (identified by operation id):

1 - filter(IS NULL)
5 - access("DEPT"."DEPTNO"="EMP"."DEPTNO")
6 - filter(("HISAL">=:B1 AND "LOSAL"<=:B2))

Note

- rule based optimizer used (consider using cbo)



Build the tree - Example 1

```
select /*+ RULE */ ename,job,sal,dname from emp,dept
where dept.deptno = emp.deptno
      and not exists (select *
                      from salgrade
                      where emp.sal between losal and hisal);
```

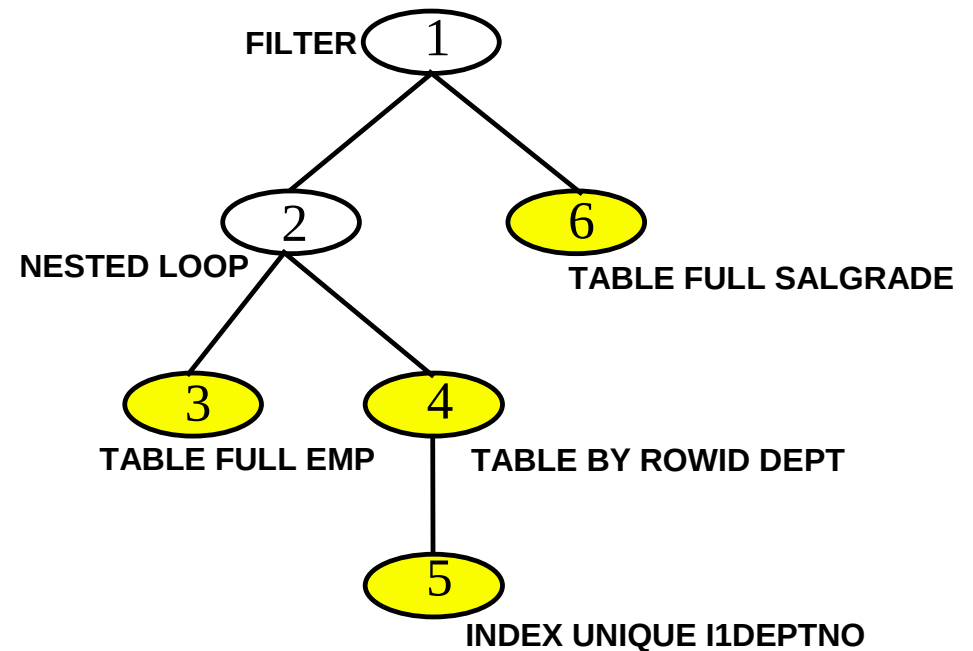
Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	EMP
4	TABLE ACCESS BY INDEX ROWID	DEPT
* 5	INDEX UNIQUE SCAN	PK_DEPT
* 6	TABLE ACCESS FULL	SALGRADE

Predicate Information (identified by operation id):

1 - filter(IS NULL)
5 - access("DEPT"."DEPTNO"="EMP"."DEPTNO")
6 - filter(("HISAL">=:B1 AND "LOSAL"<=:B2))

Note

- rule based optimizer used (consider using cbo)



Interpretation - Example 1

```
select /*+ RULE */ ename,job,sal,dname from emp,dept
where dept.deptno = emp.deptno
and not exists (select *
                 from salgrade
                 where emp.sal between losal and hisal);
```

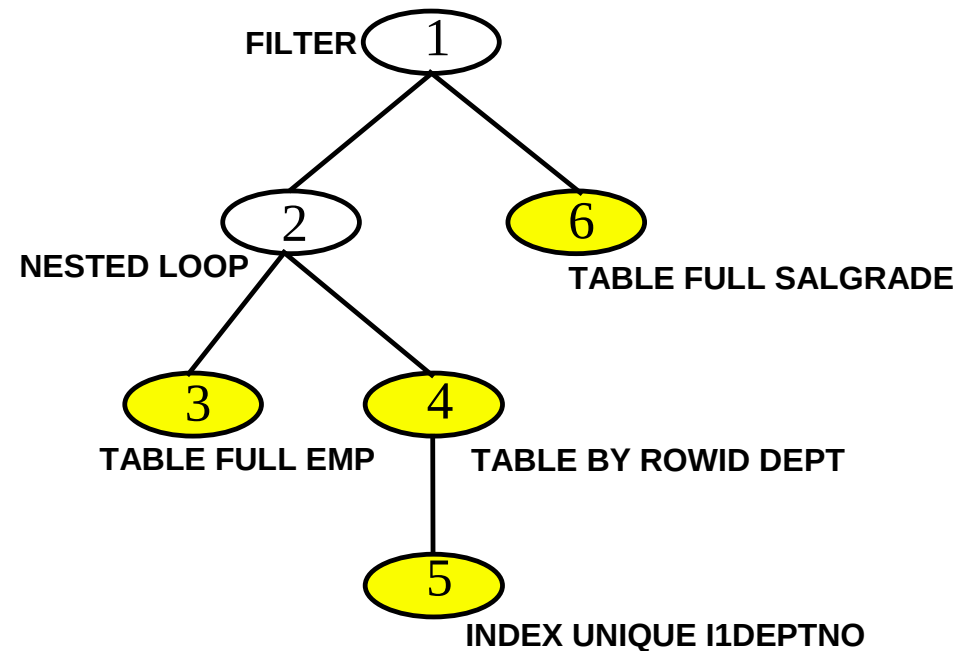
Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	EMP
4	TABLE ACCESS BY INDEX ROWID	DEPT
* 5	INDEX UNIQUE SCAN	PK_DEPT
* 6	TABLE ACCESS FULL	SALGRADE

Predicate Information (identified by operation id):

1 - filter(IS NULL)
5 - access("DEPT"."DEPTNO"="EMP"."DEPTNO")
6 - filter(("HISAL">=:B1 AND "LOSAL"<=:B2))

Note

- rule based optimizer used (consider using cbo)

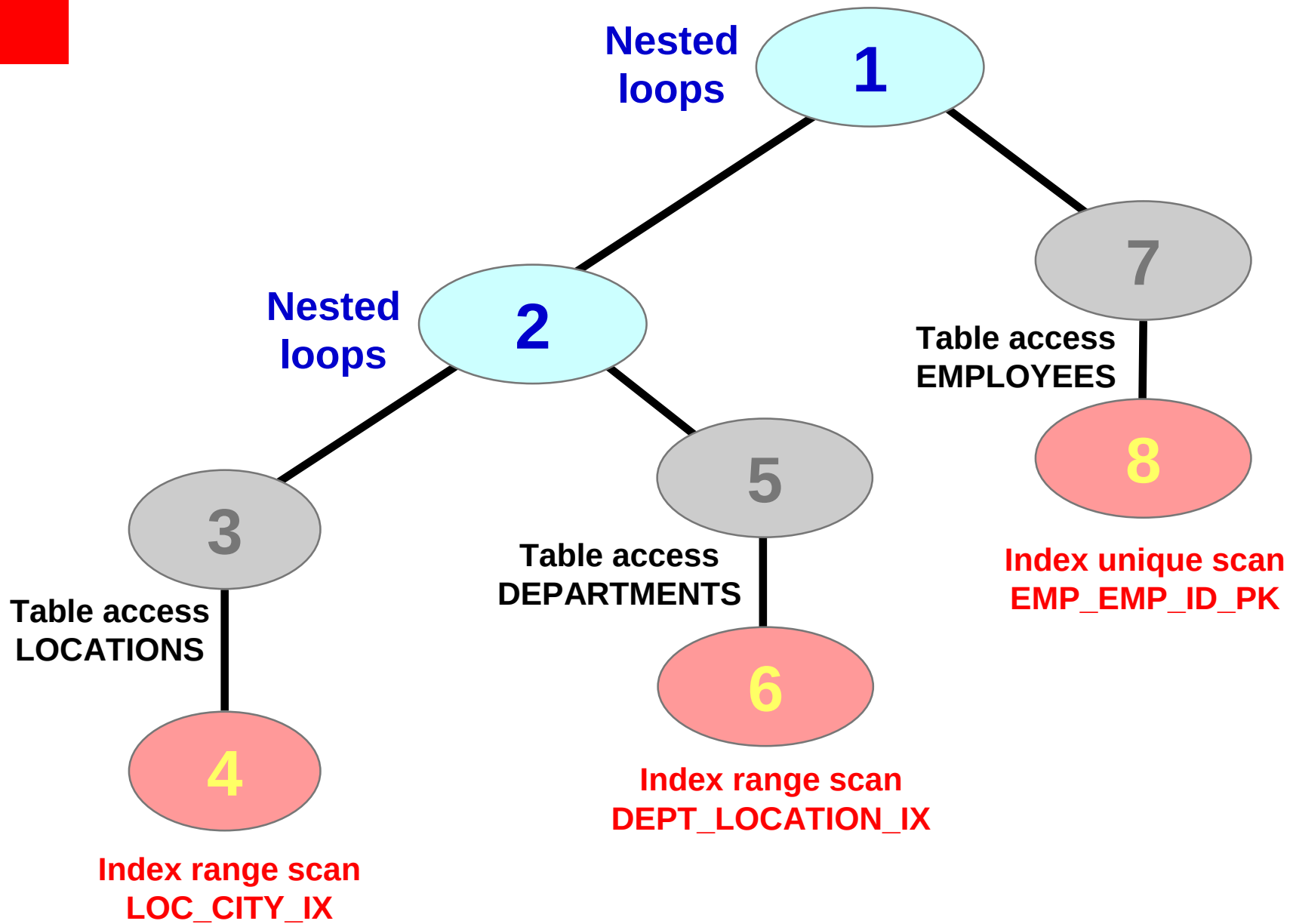


2 - Interpret Plan example 2

```
SQL> select m.last_name as dept_manager
2      ,      d.department_name
3      ,      l.street_address, l.postal_code, l.city
4 from    employees m
5          join departments d on (d.manager_id = m.employee_id)
6          join locations l on (l.location_id = d.location_id)
7 where   l.city = 'Seattle';
```

Execution Plan

```
-----
0      SELECT STATEMENT Optimizer=CHOOSE
1 0      NESTED LOOPS
2 1          NESTED LOOPS
3 2              TABLE ACCESS (BY INDEX ROWID) OF 'LOCATIONS'
4 3                  INDEX (RANGE SCAN) OF 'LOC_CITY_IX'
5 2              TABLE ACCESS (BY INDEX ROWID) OF 'DEPARTMENTS'
6 5                  INDEX (RANGE SCAN) OF 'DEPT_LOCATION_IX'
7 1              TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES'
8 7                  INDEX (UNIQUE SCAN) OF 'EMP_EMP_ID_PK'
```



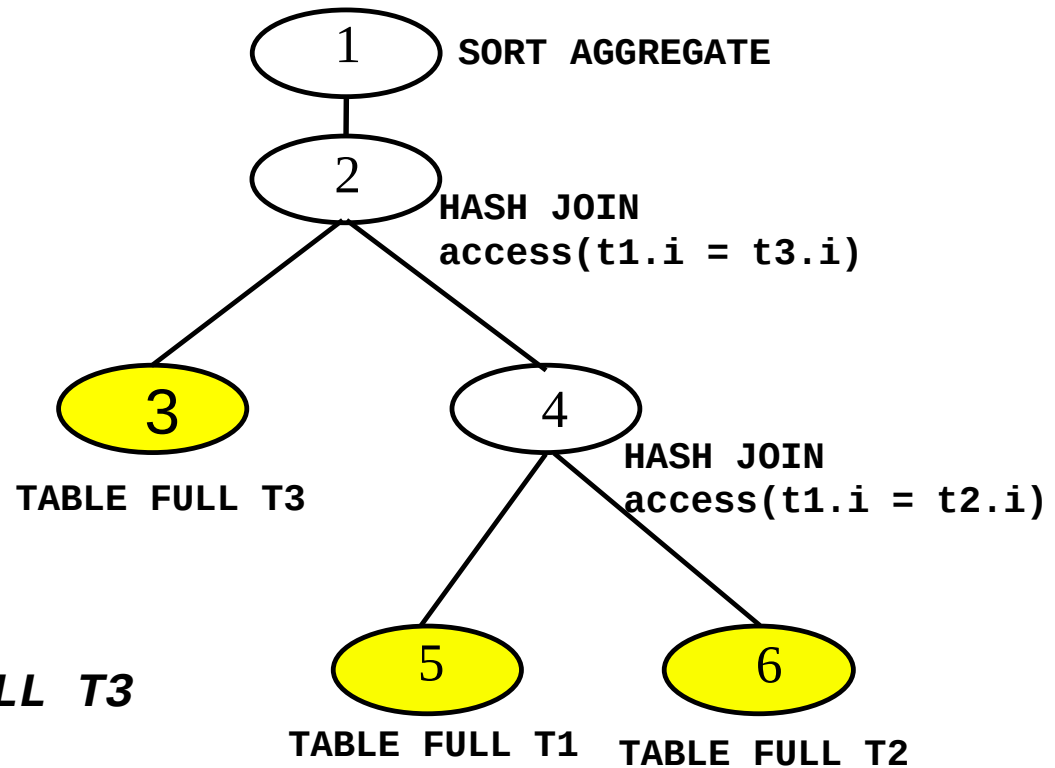
3 – Interpret Plan Example 3

Operations order & Joins order

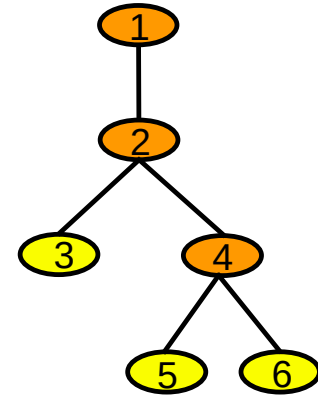
```
select max(t1.i)
  from t1,t2,t3
 where t1.i = t2.i
    and t1.i = t3.i
```

QUERY_PLAN

SELECT STATEMENT
1 **SORT AGGREGATE**
2 1 **HASH JOIN**
3 2 **TABLE ACCESS FULL T3**
4 2 **HASH JOIN**
5 4 **TABLE ACCESS FULL T1**
6 4 **TABLE ACCESS FULL T2**

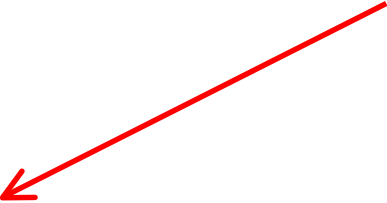


Example 3 (continued)



This is same example as previous slide

- Order in which tables are accessed :Order column
- Join order is t1 - t2 - t3

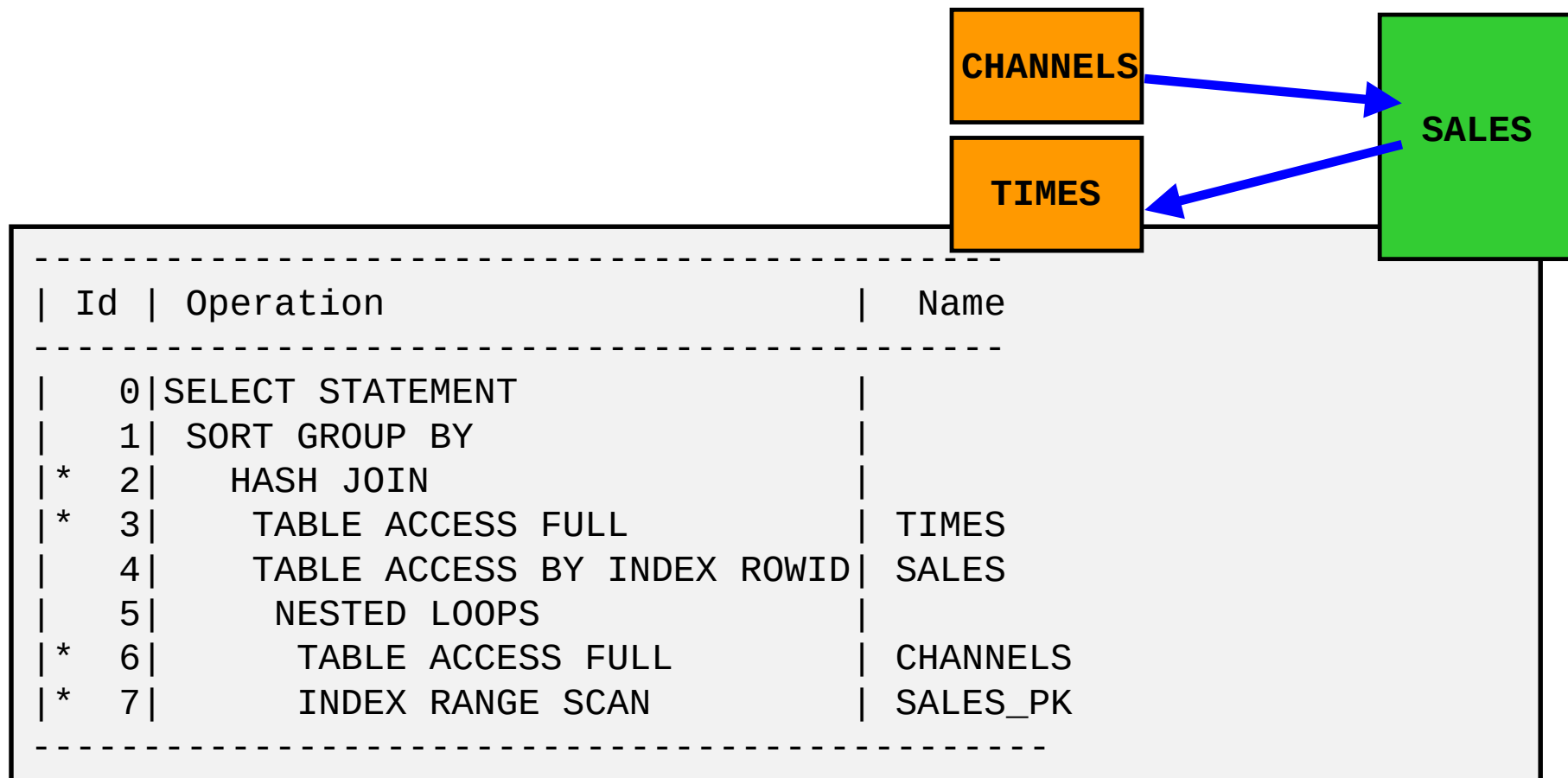


Operation	Line ID	Object	Object Type	Order	Rows	Size (KB)	Cost	Time (sec)	CPU Cost	I/O Cost
SELECT STATEMENT	0			7	1	0.009	121	2	18133014	117
SORT AGGREGATE	1			6	1	0.009				
HASH JOIN	2			5	10000	87.891	121	2	18133014	117
TABLE ACCESS FULL	3	T3	TABLE	1	10000	29.297	40	1	2703523	39
HASH JOIN	4			4	10000	58.594	80	1	10418269	78
TABLE ACCESS FULL	5	T1	TABLE	2	10000	29.297	40	1	2703523	39
TABLE ACCESS FULL	6	T2	TABLE	3	10000	29.297	40	1	2703523	39

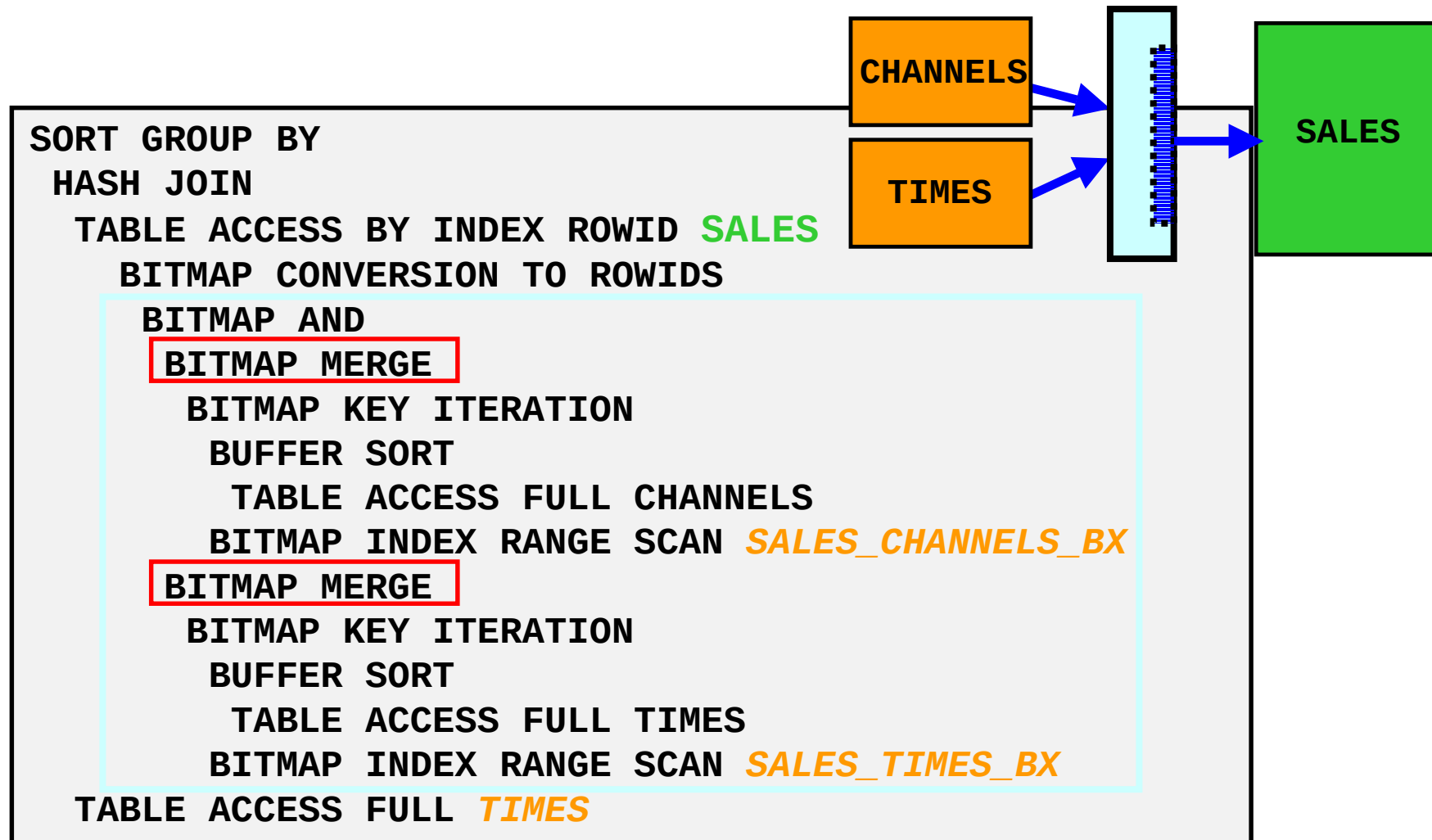
4 - Interpret Plan example 4 Star transformation plan

```
SELECT
    t.calendar_quarter_desc,
    SUM(s.amount_sold) sales_amount
FROM sales s, times t, channels ch
WHERE s.time_id = t.time_id AND
      s.channel_id = ch.channel_id AND
      ch.channel_desc IN ('Internet', 'Catalog') AND
      t.calendar_quarter_desc IN ('1999-Q1', '1999-Q2')
GROUP BY t.calendar_quarter_desc;
```

Example 4 – No star transformation



Example 4 – Star Transformation





Let's Read a more complex Plan.

Here is a common query against DBA_EXTENT :

```
SELECT owner , segment_name , segment_type  
FROM dba_extents  
WHERE file_id = &AFN  
      AND &BLOCKNO BETWEEN block_id  
                        AND block_id + blocks -1;
```

Where do I start to look at plan ... ?

Rows	Row Source Operation
1	VIEW (cr=987610 pr=92643 tm=617760587 us)
1	UNION-ALL (cr=987610 pr=92643 tm=617760575 us)
0	NESTED LOOPS (cr=2363 tm=12999 us)
0	NESTED LOOPS (cr=2363 tm=12993 us)
1	TABLE ACCESS BY INDEX ROWID FILE\$ (cr=2 tm=54 us)
1	INDEX UNIQUE SCAN I_FILE1 (cr=1 tm=31 us)
0	TABLE ACCESS FULL UET\$ (cr=2361 tm=12935 us)
0	VIEW (cr=0 tm=0 us)
0	UNION ALL PUSHED PREDICATE (cr=0 tm=0 us)
0	NESTED LOOPS (cr=0 tm=0 us)
0	NESTED LOOPS (cr=0 tm=0 us)
0	TABLE ACCESS CLUSTER USER\$ (cr=0 tm=0 us)
0	MERGE JOIN CARTESIAN (cr=0 tm=0 us)
0	NESTED LOOPS (cr=0 tm=0 us)
0	BUFFER SORT (cr=0 tm=0 us)
0	NESTED LOOPS (cr=0 tm=0 us)
0	MERGE JOIN CARTESIAN (cr=0 tm=0 us)
0	TABLE ACCESS CLUSTER USER\$ (cr=0 tm=0 us)
1	NESTED LOOPS (cr=985247 pr=92643 tm=617747538 us)
13	NESTED LOOPS (cr=985231 pr=92643 tm=1870047923 us)
94668	VIEW (cr=593549 pr=310 tm=10641190 us)
94668	UNION-ALL (cr=593549 pr=310 tm=10546516 us)
94421	NESTED LOOPS (cr=589417 pr=310 tm=10149552 us)
247	NESTED LOOPS (cr=1741 tm=18151 us)
0	HASH JOIN (cr=2391 tm=104482 us)
13	FIXED TAB FIXED INDEX X\$KTFBUE(ind:1)(cr=391682 pr=92333 tm=1879347583)
1	TABLE ACCESS BY INDEX ROWID FILE\$ (cr=16 tm=303 us)
13	INDEX UNIQUE SCAN I_FILE1 (cr=3 tm=129 us)

Where time is being spent
?

Rows	Row Source Operation
1	VIEW (cr=987610 pr=92643 tm=617760587 us)
1	UNION-ALL (cr=987610 pr=92643 tm=617760575 us)
0	NESTED LOOPS (cr=2363 tm=12999 us)
0	NESTED LOOPS (cr=2363 tm=12993 us)
0	VIEW SYS_DBA_SEGS (cr=0 tm=0 us)
1	NESTED LOOPS (cr=985247 pr=92643 tm=617747538 us)
13	NESTED LOOPS (cr=985231 pr=92643 tm=1870047923 us)
94668	VIEW SYS_DBA_SEGS (cr=593549 pr=310 tm=10641190 us)
13	FIXED TAB FIXED INDEX X\$KTFBUE(ind:1)(cr=391682 pr=92333 tm=1879347583)
1	TABLE ACCESS BY INDEX ROWID FILE\$ (cr=16 tm=303 us)
13	INDEX UNIQUE SCAN I_FILE1 (cr=3 tm=129 us)

Simplify