6-2. CTE와 SQL 처리과정

홍형경
chariehong@gmail.com
2020.11

1. WITH 절 or CTE (Common Table Expression)

· 서브쿼리의 일종

· WITH 절(clause)이라고도 하고 CTE 라고도 함

· 하나의 서브쿼리를 또 다른 서브쿼리에서 참조하여 재 사용 가능한 구문

· 오라클 11g 까지는 하나의 서브쿼리에서 다른 서브쿼리 참조 못했음

➔ WITH 절 사용

➔ 오라클 12c 부터는 LATERAL 키워드 사용해 가능

1. WITH 절 or CTE

· WITH절 구문

```
WITH  alias1 AS ( SELECT …
                  FROM …  ),
      alias2 AS ( SELECT …
                  FROM …  ),
      ….
      alias_last AS (SELECT …
                  FROM …  )
SELECT …
   FROM alias_last
… ;
```

· WITH 별칭 AS 다음에 서브쿼리 형태

· WITH은 한 번만 명시, 서브쿼리는 여러 개 사용 가능

· 최종 반환 결과는 마지막에 있는 메인 쿼리

· 서브쿼리 내에서 다른 서브쿼리 참조 가능

➔ 서브쿼리 내의 FROM 절에서 다른 서브쿼리 별칭을 기술해 인라인 뷰처럼 사용 가능

· 메인 쿼리에서는 FROM 절에서 서브쿼리 한 개, 혹은 여러 개의 서브쿼리 조인해 결과 조회 가능

2. WITH 절 or CTE 실습

```
WITH dept AS (
SELECT department_id,
       department_name  dept_name
  FROM departments
)
 SELECT a.employee_id
       ,a.first_name || ' ' || a.last_name
  FROM employees a,
       dept b
 WHERE a.department_id = b.department_id
ORDER BY 1;
```

	EMPLOYEE_ID	A.FIRST_NAME||' '||A.LAST_NAME
73	172	Elizabeth Bates
74	173	Sundita Kumar
75	174	Ellen Abel
76	175	Alyssa Hutton
77	176	Jonathon Taylor
78	177	Jack Livingston
79	179	Charles Johnson
80	180	Winston Taylor
81	181	Jean Fleaur
82	182	Martha Sullivan
83	183	Girard Geoni

2. WITH 절 or CTE 실습

```
WITH dept_loc AS (
SELECT a.department_id, a.department_name dept_name,
       b.location_id, b.street_address,   b.city, b.country_id
  FROM departments a,
       locations b
 WHERE a.location_id = b.location_id
),
cont AS (
SELECT b.department_id, b.dept_name,
       b.street_address, b.city, a.country_name
  FROM countries a,
       dept_loc b
 WHERE a.country_id = b.country_id
)
SELECT a.employee_id,  a.first_name || ' ' || a.last_name emp_name,
       b.dept_name, b.street_address,   b.country_name
  FROM employees a,  cont b
 WHERE a.department_id = b.department_id
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPT_NAME	STREET_ADDRESS	COUNTRY_NAME
76	175	Alyssa Hutton	Sales	Magdalen Centre, The ...	United Kingdom
77	176	Jonathon Taylor	Sales	Magdalen Centre, The ...	United Kingdom
78	177	Jack Livingston	Sales	Magdalen Centre, The ...	United Kingdom
79	179	Charles Johnson	Sales	Magdalen Centre, The ...	United Kingdom
80	180	Winston Taylor	Shipping	2011 Interiors Blvd	United States of America
81	181	Jean Fleaur	Shipping	2011 Interiors Blvd	United States of America
82	182	Martha Sullivan	Shipping	2011 Interiors Blvd	United States of America
83	183	Girard Geoni	Shipping	2011 Interiors Blvd	United States of America
84	184	Nandita Sarchand	Shipping	2011 Interiors Blvd	United States of America
85	185	Alexis Bull	Shipping	2011 Interiors Blvd	United States of America
86	186	Julia Dellinger	Shipping	2011 Interiors Blvd	United States of America
87	187	Anthony Cabrio	Shipping	2011 Interiors Blvd	United States of America
88	188	Kelly Chung	Shipping	2011 Interiors Blvd	United States of America

2. WITH 절 or CTE 실습

```
WITH emp_info AS (
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       b.department_id, b.department_name dept_name,
       c.street_address, c.city,
       d.country_name, e.region_name
  FROM employees a,    departments b,
       locations c,   countries d,   regions e
 WHERE a.department_id = b.department_id
   AND b.location_id   = c.location_id
   AND c.country_id    = d.country_id
   AND d.region_id     = e.region_id
)
SELECT *
  FROM emp_info
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME	STREET_ADDRESS	CITY	COUNTRY_NAME	REGION_NAME
73	172	Elizabeth Bates	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
74	173	Sundita Kumar	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
75	174	Ellen Abel	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
76	175	Alyssa Hutton	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
77	176	Jonathon Taylor	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
78	177	Jack Livingston	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
79	179	Charles Johnson	80	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
80	180	Winston Taylor	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
81	181	Jean Fleaur	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
82	182	Martha Sullivan	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
83	183	Girard Geoni	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
84	184	Nandita Sarchand	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
85	185	Alexis Bull	50	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas

2. WITH 절 or CTE 실습

```
WITH coun_sal AS (
SELECT c.country_id, SUM(a.salary) sal_amt
  FROM employees a,
       departments b,
       locations c
 WHERE a.department_id = b.department_id
   AND b.location_id = c.location_id
GROUP BY c.country_id ),
mains AS (
SELECT b.country_name, a.sal_amt
  FROM coun_sal a,
           countries b
 WHERE a.country_id = b.country_id   )
SELECT *
  FROM mains
 ORDER BY 1;
```

	COUNTRY_NAME	SAL_AMT
1	Canada	19000
2	Germany	10000
3	United Kingdom	311000
4	United States of America	344416

3. WITH 절 특징

· WITH 절은 내부적으로 TEMP 테이블 스페이스를 사용함

➔ TEMP 테이블스페이스에 각 서브쿼리 결과를 담아두고 있음

· TEMP 테이블스페이스는 정렬 용도로 사용

· 과도한 WITH 절 사용 시, TEMP 테이블스페이스 공간을 점유해 성능에 좋지 않음

· 일반적인 경우에는 서브쿼리를 사용하고, 서브쿼리 사용이 수월치 않은 경우 WITH 절 사용

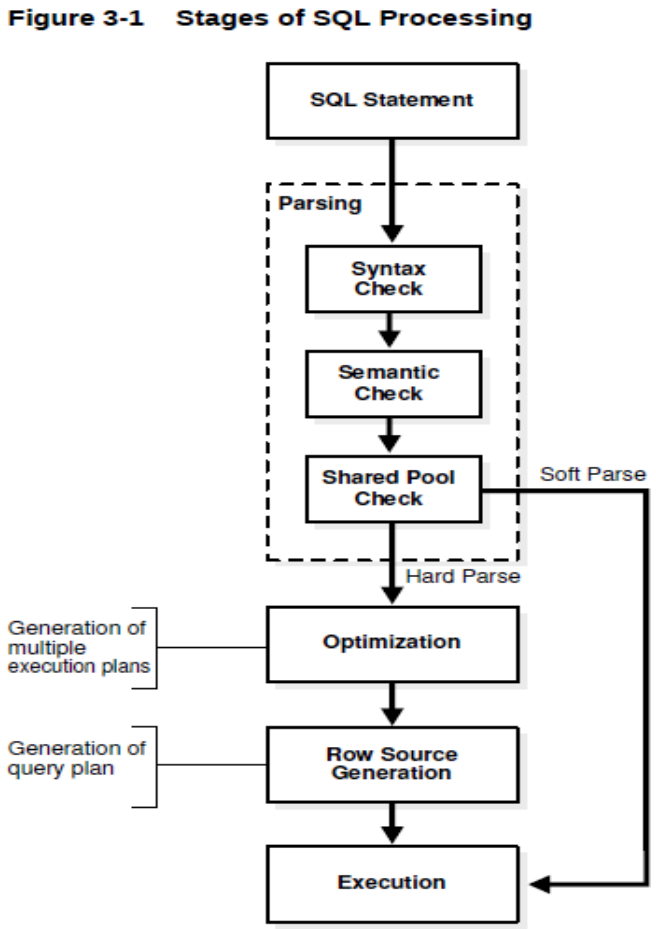
4. SQL 처리과정

· SQL 문장을 작성해 실행하면 오라클 내부에서 어떻게 처리될까?

· SQL 내부 처리 프로세스

- SQL Syntax Check
- SQL Semantic Check
- 가능한 여러 개의 실행계획(Execution Plan) 수립
- 최적의 실행계획을 선택 해 SQL 실행
- 실행 결과 반환

4. SQL 처리과정



· Syntax Check : SQL 문장 검사 (오타 등)

예) select * **form** employees;

ORA-00923: FROM keyword not found where expected

· Semantic Check : 의미 검사, 객체 권한 검사

예) select * from **hong**;

ORA-00942: table or view does not exist

· Shared Pool Check : SQL문장에 ID 부여 등

· Optimization

- SQL 문장을 최적화 해 재작성
- 여러 개의 실행계획 생성

· Row Source Generation : 최적의 실행계획 선정

· Execution : 실행

출처: Oracle Database SQL Tuning Guide (Oracle 18c Manual)

4. SQL 처리과정

· 최적의 실행계획이란 ?

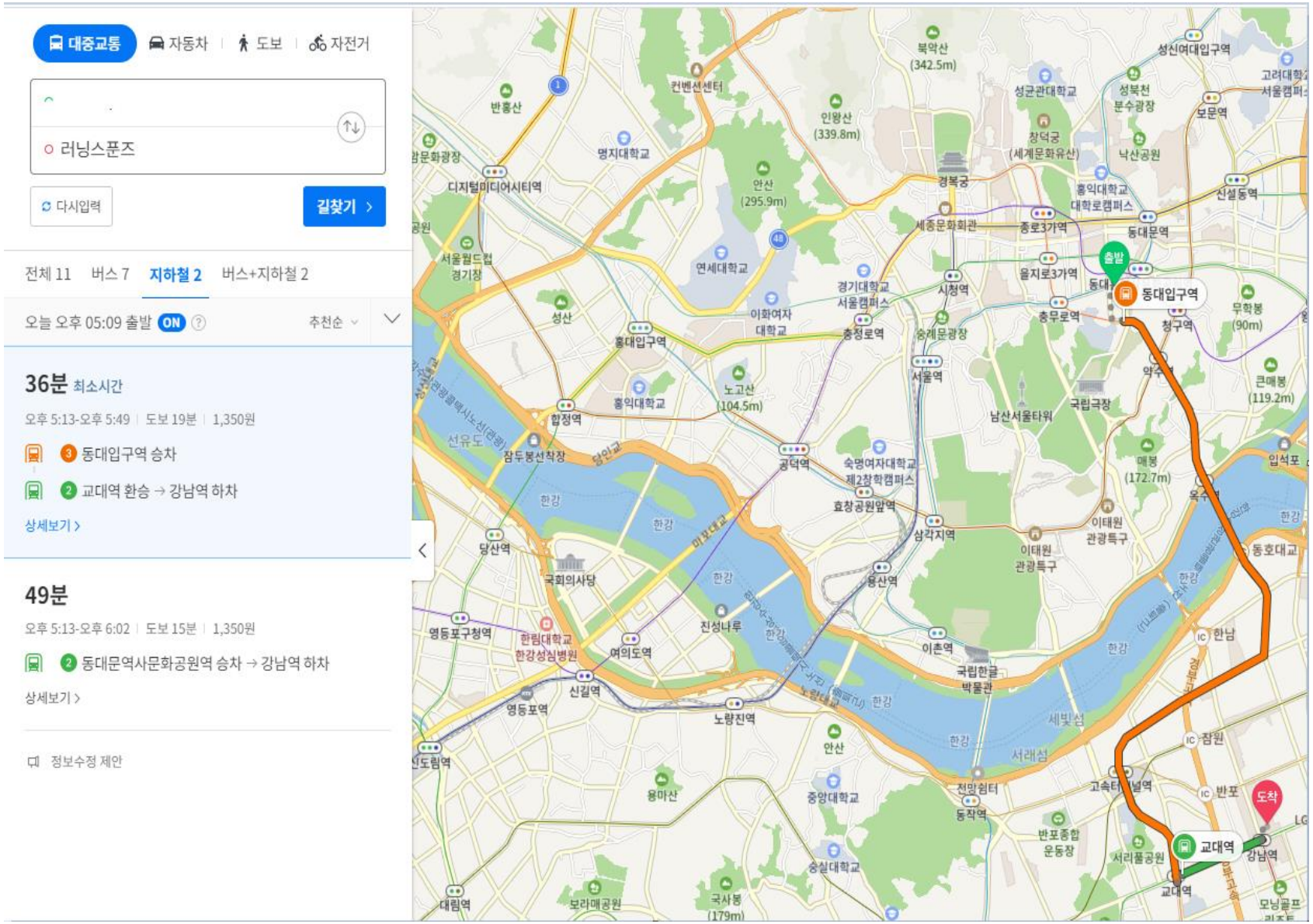
· 내비게이션 시스템과 비슷함

· 여러 개의 실행계획을 세우고 그 중 가장 비용(Cost)이 낮은 계획을 선택해 실행

➔ 가장 빨리 결과를 내는 실행계획을 선택

4. SQL 처리과정

· 회사 → 러닝스푼즈



4. SQL 처리과정

· 최적의 실행계획을 위해서는 테이블의 통계정보를 최신으로 갱신

· 통계정보 : 테이블의 로우 수, 블록 수 등 실행계획을 세우기 위한 기초 정보

· 같은 테이블에 100건, 10000건, 백만 건 있을 경우에 따라 통계정보 달라짐
  - 조인 시, 어떤 테이블을 먼저 읽느냐에 따라 성능 차이 발생

· 내비게이션에서 현재 교통상황을 반영하면 경로가 달라지는 것과 유사

4. SQL 처리과정

· 오라클 버전이 올라갈수록 실행계획을 잘 세우고 있음

· 실행계획을 잘 못 세웠을 경우, SQL 실행 계획을 조정해 성능을 향상

➔ SQL 튜닝 (힌트 사용)

· 대부분의 경우, 튜닝 시 조인 방식과 순서를 변경

4. SQL 처리과정

· 실행했던 SQL 이력 조회 (ORAUSER로 접속해 실행)

```
SELECT *
    FROM V$SQL;
```


4. SQL 처리과정

```
WITH coun_sal AS ( /*+ gather_plan_statistics */
SELECT c.country_id, SUM(a.salary) sal_amt
  FROM employees a,
       departments b,
       locations c
 WHERE a.department_id = b.department_id
   AND b.location_id = c.location_id
GROUP BY c.country_id ),
mains AS (
SELECT b.country_name, a.sal_amt
  FROM coun_sal a,
       countries b
 WHERE a.country_id = b.country_id
)
SELECT *
  FROM mains
 ORDER BY 1;
```

```
SQL_ID  dr3g074gz5bw7, child number 0
-------------------------------------
WITH coun_sal AS ( /*+ gather_plan_statistics */ SELECT c.country_id,
SUM(a.salary) sal_amt   FROM employees a, /* HONG */        departments
b,        locations c  WHERE a.department_id = b.department_id    AND
b.location_id = c.location_id GROUP BY c.country_id ), mains AS (
SELECT b.country_name, a.sal_amt   FROM coun_sal a,        countries b
WHERE a.country_id = b.country_id ) SELECT *   FROM mains  ORDER BY 1

Plan hash value: 2024698841

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | E-Rows |E-Bytes| Cost (%CPU)| E-Time   |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |        |       |     8 (100)|          |       |       |          |
|   1 |  SORT GROUP BY               |                   |    106 |  3392 |     8  (13)| 00:00:01 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                  |                   |    106 |  3392 |     7   (0)| 00:00:01 | 1236K| 1236K| 1619K (0)|
|*  3 |    HASH JOIN                 |                   |     29 |   725 |     4   (0)| 00:00:01 | 1476K| 1476K| 1592K (0)|
|   4 |     NESTED LOOPS             |                   |     23 |   414 |     2   (0)| 00:00:01 |       |       |          |
|   5 |      VIEW                    | index$_join$_003  |     23 |   138 |     2   (0)| 00:00:01 |       |       |          |
|*  6 |       HASH JOIN              |                   |        |       |            |          | 1610K| 1610K| 1606K (0)|
|   7 |        INDEX FAST FULL SCAN| LOC_COUNTRY_IX    |     23 |   138 |     1   (0)| 00:00:01 |       |       |          |
|   8 |        INDEX FAST FULL SCAN| LOC_ID_PK         |     23 |   138 |     1   (0)| 00:00:01 |       |       |          |
|*  9 |       INDEX UNIQUE SCAN      | COUNTRY_C_ID_PK   |      1 |    12 |     0   (0)|          |       |       |          |
|  10 |      VIEW                    | index$_join$_002  |     29 |   203 |     2   (0)| 00:00:01 |       |       |          |
|* 11 |       HASH JOIN              |                   |        |       |            |          | 1610K|  1610K| 1506K (0)|
|  12 |        INDEX FAST FULL SCAN | DEPT_ID_PK        |     29 |   203 |     1   (0)| 00:00:01 |       |       |          |
|  13 |        INDEX FAST FULL SCAN | DEPT_LOCATION_IX  |     29 |   203 |     1   (0)| 00:00:01 |       |       |          |
|  14 |     TABLE ACCESS FULL        | EMPLOYEES         |    107 |   749 |     3   (0)| 00:00:01 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$DA08B5C9
   5 - SEL$0EDF14EB / C@SEL$1
   6 - SEL$0EDF14EB
   7 - SEL$0EDF14EB / indexjoin$_alias$_001@SEL$0EDF14EB
   8 - SEL$0EDF14EB / indexjoin$_alias$_002@SEL$0EDF14EB
   9 - SEL$DA08B5C9 / B@SEL$2
  10 - SEL$FE1F385F / B@SEL$1
  11 - SEL$FE1F385F
  12 - SEL$FE1F385F / indexjoin$_alias$_001@SEL$FE1F385F
  13 - SEL$FE1F385F / indexjoin$_alias$_002@SEL$FE1F385F
  14 - SEL$DA08B5C9 / A@SEL$1
```

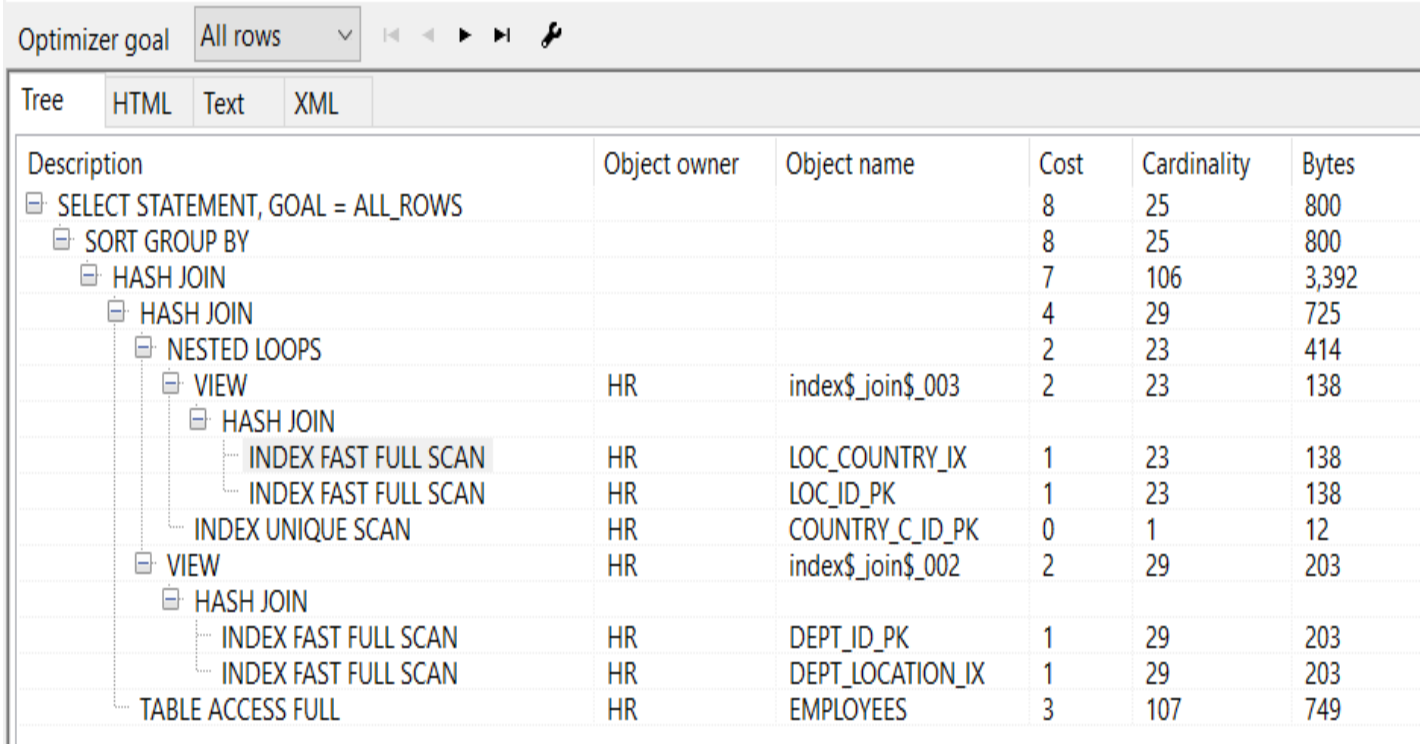

4. SQL 처리과정

```
WITH coun_sal AS ( /*+ gather_plan_statistics */
SELECT c.country_id, SUM(a.salary) sal_amt
  FROM employees a,
       departments b,
       locations c
 WHERE a.department_id = b.department_id
   AND b.location_id = c.location_id
GROUP BY c.country_id ),
mains AS (
SELECT b.country_name, a.sal_amt
  FROM coun_sal a,
       countries b
 WHERE a.country_id = b.country_id
)
SELECT *
  FROM mains
 ORDER BY 1;
```

```
SQL_ID  dr3g074gz5bw7, child number 0
-------------------------------------
WITH coun_sal AS ( /*+ gather_plan_statistics */ SELECT c.country_id,
SUM(a.salary) sal_amt   FROM employees a, /* HONG */        departments
b,        locations c  WHERE a.department_id = b.department_id    AND
b.location_id = c.location_id GROUP BY c.country_id ), mains AS (
SELECT b.country_name, a.sal_amt   FROM coun_sal a,        countries b
WHERE a.country_id = b.country_id ) SELECT *   FROM mains  ORDER BY 1

Plan hash value: 2024698841

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | E-Rows |E-Bytes| Cost (%CPU)| E-Time   |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |        |       |     8 (100)|          |       |       |          |
|   1 |  SORT GROUP BY               |                   |    106 |  3392 |     8  (13)| 00:00:01 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                  |                   |    106 |  3392 |     7   (0)| 00:00:01 |  1236K|  1236K| 1619K (0)|
|*  3 |    HASH JOIN                 |                   |     29 |   725 |     4   (0)| 00:00:01 |  1476K|  1476K| 1592K (0)|
|   4 |     NESTED LOOPS             |                   |     23 |   414 |     2   (0)| 00:00:01 |       |       |          |
|   5 |      VIEW                    | index$_join$_003  |     23 |   138 |     2   (0)| 00:00:01 |       |       |          |
|*  6 |       HASH JOIN              |                   |        |       |            |          |  1610K|  1610K| 1606K (0)|
|   7 |        INDEX FAST FULL SCAN  | LOC_COUNTRY_IX    |     23 |   138 |     1   (0)| 00:00:01 |       |       |          |
|   8 |        INDEX FAST FULL SCAN  | LOC_ID_PK         |     23 |   138 |     1   (0)| 00:00:01 |       |       |          |
|*  9 |       INDEX UNIQUE SCAN      | COUNTRY_C_ID_PK   |      1 |    12 |     0   (0)|          |       |       |          |
|  10 |      VIEW                    | index$_join$_002  |     29 |   203 |     2   (0)| 00:00:01 |       |       |          |
|* 11 |       HASH JOIN              |                   |        |       |            |          |  1610K|  1610K| 1506K (0)|
|  12 |        INDEX FAST FULL SCAN  | DEPT_ID_PK        |     29 |   203 |     1   (0)| 00:00:01 |       |       |          |
|  13 |        INDEX FAST FULL SCAN  | DEPT_LOCATION_IX  |     29 |   203 |     1   (0)| 00:00:01 |       |       |          |
|  14 |     TABLE ACCESS FULL        | EMPLOYEES         |    107 |   749 |     3   (0)| 00:00:01 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$DA08B5C9
   5 - SEL$0EDF14EB / C@SEL$1
   6 - SEL$0EDF14EB
   7 - SEL$0EDF14EB / indexjoin$_alias$_001@SEL$0EDF14EB
   8 - SEL$0EDF14EB / indexjoin$_alias$_002@SEL$0EDF14EB
   9 - SEL$DA08B5C9 / B@SEL$2
  10 - SEL$FE1F385F / B@SEL$1
  11 - SEL$FE1F385F
  12 - SEL$FE1F385F / indexjoin$_alias$_001@SEL$FE1F385F
  13 - SEL$FE1F385F / indexjoin$_alias$_002@SEL$FE1F385F
  14 - SEL$DA08B5C9 / A@SEL$1
```


4. SQL 처리과정

```
SELECT /*+ gather_plan_statistics */
       d.country_name, SUM(a.salary) sal_amt
  FROM employees a,
       departments b,
       locations c,
       countries d
 WHERE a.department_id = b.department_id
   AND b.location_id = c.location_id
   AND c.country_id = d.country_id
GROUP BY d.country_name
ORDER BY 1;
```



Optimizer goal: All rows

Tree | HTML | Text | XML

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			8	25	800
SORT GROUP BY			8	25	800
HASH JOIN			7	106	3,392
HASH JOIN			4	29	725
NESTED LOOPS			2	23	414
VIEW	HR	index$_join$_003	2	23	138
HASH JOIN					
INDEX FAST FULL SCAN	HR	LOC_COUNTRY_IX	1	23	138
INDEX FAST FULL SCAN	HR	LOC_ID_PK	1	23	138
INDEX UNIQUE SCAN	HR	COUNTRY_C_ID_PK	0	1	12
VIEW	HR	index$_join$_002	2	29	203
HASH JOIN					
INDEX FAST FULL SCAN	HR	DEPT_ID_PK	1	29	203
INDEX FAST FULL SCAN	HR	DEPT_LOCATION_IX	1	29	203
TABLE ACCESS FULL	HR	EMPLOYEES	3	107	749

4. SQL 처리과정

```
SELECT /*+ gather_plan_statistics */
       d.country_name, SUM(a.salary) sal_amt
  FROM employees a,
       departments b,
       locations c,
       countries d
 WHERE a.department_id = b.department_id
   AND b.location_id = c.location_id
   AND c.country_id = d.country_id
GROUP BY d.country_name
ORDER BY 1;
```

```
SELECT *
FROM V$SQL
WHERE sql_text LIKE '%gather%' ;
```

	SQL_TEXT	SQL_FULLTEXT	SQL_ID
1	SELECT /*+ gather_plan_statistics */ d.country_name,	<CLOB>	gx8yap8azwk86
2	WITH coun_sal AS (/*+ gather_plan_statistics */ SELECT c.	<CLOB>	1qhf1d8jyx8r0
3	SELECT * FROM V$SQL WHERE sql_text LIKE '%gather%'	<CLOB>	85pnhrcjk7tpf

4. SQL 처리과정

```
select *
from table(dbms_xplan.display_cursor ( 'gx8yap8azwk86', null, 'ADVANCED ALLSTATS LAST'));
```

	PLAN_TABLE_OUTPUT
1	SQL_ID gx8yap8azwk86, child number 0
2	------------------------------------
3	SELECT /*+ gather_plan_statistics */ d.country_name,
4	SUM(a.salary) sal_amt FROM employees a, departments b,
5	locations c, countries d WHERE a.department_id =
6	b.department_id AND b.location_id = c.location_id AND
7	c.country_id = d.country_id GROUP BY d.country_name ORDER BY 1
8	
9	Plan hash value: 2024698841
10	
11	----------
12	| Id | Operation | Name | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-T
13	----------

4. SQL 처리과정

select *

from table(dbms_xplan.display_cursor ('gx8yap8azwk86', null, 'ADVANCED ALLSTATS LAST'));

```
SQL_ID  gx8yap8azwk86, child number 0
-------------------------------------
SELECT /*+ gather_plan_statistics */        d.country_name,
SUM(a.salary) sal_amt   FROM employees a,        departments b,
locations c,        countries d  WHERE a.department_id =
b.department_id    AND b.location_id = c.location_id    AND
c.country_id = d.country_id GROUP BY d.country_name ORDER BY 1

Plan hash value: 2024698841

-----------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |      1 |        |       |     8 (100)|          |      4 |00:00:00.01 |      67 |       |       |          |
|   1 |  SORT GROUP BY               |                  |      1 |     25 |   800 |     8  (13)| 00:00:01 |      4 |00:00:00.01 |      67 |  2048 |  2048 | 2048  (0)|
|*  2 |   HASH JOIN                  |                  |      1 |    106 |  3392 |     7   (0)| 00:00:01 |    106 |00:00:00.01 |      67 |  1298K|  1298K| 1597K (0)|
|*  3 |    HASH JOIN                 |                  |      1 |     29 |   725 |     4   (0)| 00:00:01 |     29 |00:00:00.01 |      20 |  1610K|  1610K| 1635K (0)|
|   4 |     NESTED LOOPS             |                  |      1 |     23 |   414 |     2   (0)| 00:00:01 |     23 |00:00:00.01 |      12 |       |       |          |
|   5 |      VIEW                    | index$_join$_003 |      1 |     23 |   138 |     2   (0)| 00:00:01 |     23 |00:00:00.01 |       8 |       |       |          |
|*  6 |       HASH JOIN              |                  |      1 |        |       |            |          |     23 |00:00:00.01 |       8 |  1610K|  1610K| 1616K (0)|
|   7 |        INDEX FAST FULL SCAN  | LOC_COUNTRY_IX   |      1 |     23 |   138 |     1   (0)| 00:00:01 |     23 |00:00:00.01 |       4 |       |       |          |
|   8 |        INDEX FAST FULL SCAN  | LOC_ID_PK        |      1 |     23 |   138 |     1   (0)| 00:00:01 |     23 |00:00:00.01 |       4 |       |       |          |
|*  9 |      INDEX UNIQUE SCAN       | COUNTRY_C_ID_PK  |     23 |      1 |    12 |     0   (0)|          |     23 |00:00:00.01 |       4 |       |       |          |
|  10 |     VIEW                     | index$_join$_002 |      1 |     29 |   203 |     2   (0)| 00:00:01 |     29 |00:00:00.01 |       8 |       |       |          |
|* 11 |      HASH JOIN               |                  |      1 |        |       |            |          |     29 |00:00:00.01 |       8 |  1610K|  1610K| 1491K (0)|
|  12 |       INDEX FAST FULL SCAN   | DEPT_ID_PK       |      1 |     29 |   203 |     1   (0)| 00:00:01 |     29 |00:00:00.01 |       4 |       |       |          |
|  13 |       INDEX FAST FULL SCAN   | DEPT_LOCATION_IX |      1 |     29 |   203 |     1   (0)| 00:00:01 |     29 |00:00:00.01 |       4 |       |       |          |
|  14 |    TABLE ACCESS FULL         | EMPLOYEES        |      1 |    107 |   749 |     3   (0)| 00:00:01 |    107 |00:00:00.01 |       6 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------
```


5. Top n Query

· 특정 컬럼 값을 기준으로 상위 n개, 혹은 하위 n개 로우를 조회하는 쿼리

· MSSQL, MySQL 등은 기본 문법에서 제공

· 오라클 11g 까지는 제공하지 않았음

➔ 서브쿼리, ROWNUM 을 사용해 구현

· 오라클 12c 부터 기본 문법으로 제공

5. Top 5 Query – ROWNUM 사용

```
SELECT *
FROM (
  SELECT a.employee_id,
                a.first_name || ' ' || a.last_name emp_name,
                a.salary
     FROM employees a
    ORDER BY a.salary DESC
    ) b
WHERE ROWNUM <= 5;
```

	EMPLOYEE_ID	EMP_NAME	SALARY
1	100	Steven King	24000
2	101	Neena Kochhar	17000
3	102	Lex De Haan	17000
4	145	John Russell	14000
5	146	Karen Partners	13500

1. 서브쿼리에서 salary 값을 기준으로 내림차순 정렬

2. rownum을 사용해 5건 이하만 조회

5. Top 5 Query – ROW_NUMBER() 사용

```
SELECT *
FROM ( SELECT a.employee_id,
              a.first_name || ' ' || a.last_name emp_name,
              a.salary,
              ROW_NUMBER() OVER (ORDER BY a.salary DESC) ROW_SEQ
         FROM employees a
       ) b
WHERE ROW_SEQ <= 5;
```

	EMPLOY...	EMP_NAME	SALARY	ROW_SEQ
1	100	Steven King	24000	1
2	101	Neena Kochhar	17000	2
3	102	Lex De Haan	17000	3
4	145	John Russell	14000	4
5	146	Karen Partners	13500	5

1. 서브쿼리에서 분석 함수를 사용해 salary 값을 기준으로 내림차순 순번 계산

2. 계산한 순번을 사용해 5건 이하만 조회

5. Top 5 Query – FETCH FIRST ROWS 구문 (12c 부터)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.salary
  FROM employees a
 ORDER BY a.salary DESC
 FETCH FIRST 5 ROWS ONLY;
```

	EMPLOY...	EMP_NAME	SALARY
1	100	Steven King	24000
2	101	Neena Kochhar	17000
3	102	Lex De Haan	17000
4	145	John Russell	14000
5	146	Karen Partners	13500

1. salary 값을 기준으로 내림차순 정렬

2. FETCH FIRST ROWS 구문을 사용해 5개의 로우만 조회

5. Top 5 Query – FETCH FIRST ROWS 구문 (12c 부터)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.salary
  FROM employees a
FETCH FIRST 5 ROWS ONLY;
```

	EMPLOYEE_ID	EMP_NAME	SALARY
1	100	Steven King	24000
2	101	Neena Kochhar	17000
3	102	Lex De Haan	17000
4	103	Alexander Hunold	9000
5	104	Bruce Ernst	6000

* 정렬을 하지 않았으므로 임으로 5개 로우만 조회됨

5. Top 5 Query – FETCH FIRST ROWS 구문 (12c 부터)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.salary
  FROM employees a
 ORDER BY a.salary DESC
FETCH FIRST 5 PERCENT ROWS ONLY;
```

	EMPLOY...	EMP_NAME	SALARY
1	100	Steven King	24000
2	101	Neena Kochhar	17000
3	102	Lex De Haan	17000
4	145	John Russell	14000
5	146	Karen Partners	13500
6	201	Michael Hartstein	13000

1. salary 값을 기준으로 내림차순 정렬

2. PERCENT를 사용해 5%에 해당하는 로우를 조회

   ➔ EMPLOYEES의 총 건수는 107건, 5%는 5.35건, 6개 로우 조회됨

5. Top 5 Query – FETCH FIRST ROWS 구문 (12c 부터)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.salary
  FROM employees a
 ORDER BY a.salary
 FETCH FIRST 5 PERCENT ROWS ONLY;
```

	EMPLOYEE_ID	EMP_NAME	SALARY
1	132	TJ Olson	2100
2	128	Steven Markle	2200
3	136	Hazel Philtanker	2200
4	127	James Landry	2400
5	135	Ki Gee	2400
6	119	Karen Colmenares	2500

1. salary 값을 기준으로 오름차순 정렬

2. PERCENT를 사용해 5%에 해당하는 로우를 조회

5. Top 5 Query – FETCH FIRST ROWS 구문 (12c 부터)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.salary
  FROM employees a
ORDER BY a.salary
FETCH FIRST 5  PERCENT ROWS WITH TIES;
```

	EMPLOYEE_ID	EMP_NAME	SALARY
1	132	TJ Olson	2100
2	128	Steven Markle	2200
3	136	Hazel Philtanker	2200
4	127	James Landry	2400
5	135	Ki Gee	2400
6	119	Karen Colmenares	2500
7	131	James Marlow	2500
8	140	Joshua Patel	2500
9	144	Peter Vargas	2500
10	182	Martha Sullivan	2500
11	191	Randall Perkins	2500

1. salary 값을 기준으로 오름차순 정렬

2. WITH TIES는 급여가 같은 값을 모두 조회

Quiz

1. **Covid19 테이블에서 2020년 전체 가장 많은 확진자가 나온 상위 5개 국가를 구하는 쿼리를 작성하시오.**

Quiz

2. Covid19 테이블에서 2020년 인구대비 사망률이 가장 많은 상위 20개 국가를 구하는 쿼리를 작성하시오.

	COUNTRY	POPU	DEATH	RATES
1	San Marino	33938	42	0.12376
2	Peru	32971846	34315	0.10407
3	Belgium	11589616	11050	0.09534
4	Andorra	77265	72	0.09319
5	Spain	46754783	35466	0.07586
6	Brazil	212559409	158456	0.07455
7	Bolivia	11673029	8694	0.07448
8	Chile	19116209	14032	0.0734
9	Ecuador	17643060	12608	0.07146
10	Mexico	128932753	90309	0.07004
11	United States	331002647	227700	0.06879
12	United Kingdom	67886004	45675	0.06728
13	Argentina	45195777	30071	0.06653
14	Italy	60461828	37905	0.06269
15	Panama	4314768	2663	0.06172
16	Colombia	50882884	30753	0.06044
17	Sweden	10099270	5927	0.05869
18	France	65273512	35785	0.05482
19	Sint Maarten (Dutch part)	42882	22	0.0513
20	Macedonia	2083380	963	0.04622

