Chapter 21. 분석 함수

이번 장에서는 Oracle 8.1.6 버전부터 제공되는 분석함수에 대하여 설명한다. 분석함수를 사용하면 복잡한 쿼리를 간단하게 기술 할 수 있으며, 수행 속도 또한 향상된다.

랭킹(Ranking) 함수

랭킹과 관련된 함수들은 지정된 기준에 따라 해당 행의 순위를 계산하며, 사용 방법은 다음과 같다.

```
analytic_function() OVER (
        [PARTITION BY <value expression1>][,....]
        ORDER BY <value expression2>[collate clause][ASC|DESC]
        [NULLS FIRST|NULLS LAST][,...])
```

여기서,
- OVER : 해당 결과 집합을 이용해 동작하는 함수라는 의미이다.
- PARTITION BY : 결과 집합이 *value expression1*에 지정된 값으로 분할된다.
- ORDER BY : PARTITION BY에 의해 분할된 각 파티션을 *value expression2*에 지정된 컬럼으로 정렬한다.
- NULLS FIRST|NULLS LAST : NULL이 포함된 행이 PARTITION내에서 가장 먼저 위치 할 것인지 가장 마지막에 위치 할 것인지를 지정

■ RANK

RANK 함수는 각 파티션 내에서 ORDER BY 뒤에 지정된 컬럼을 기준으로 정렬한 뒤, 해당 행에 대한 순위를 계산한다. 사원 테이블에서 입사일이 빠른 순서대로 순위를 계산하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, RANK() OVER (ORDER BY HIREDATE) RANK
  2  FROM EMP;

     EMPNO ENAME      HIREDATE       RANK
---------- ---------- -------- ----------
      7369 SMITH      80/12/17          1
      7499 ALLEN      81/02/20          2
      7521 WARD       81/02/22          3
      7566 JONES      81/04/02          4
      7698 BLAKE      81/05/01          5
      7782 CLARK      81/06/09          6
      7844 TURNER     81/09/08          7
      7654 MARTIN     81/09/28          8
      7839 KING       81/11/17          9
      7900 JAMES      81/12/03         10
      7902 FORD       81/12/03         10
      7934 MILLER     82/01/23         12
      7788 SCOTT      87/04/19         13
      7876 ADAMS      87/05/23         14
```

위의 경우에서 OVER에 PARTITION BY가 생략되었으며 테이블의 모든 행이 하나의 파

티션으로 지정된다.

이번에는 각 부서에 대하여 사원들의 급여에 따라 순위를 계산한다. 부서별로 파티션이 되고, 해당 파티션별로 순위가 계산된다.

```
SQL> SELECT EMPNO, ENAME, DEPTNO, SAL,
  2  RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) RANK
  3  FROM EMP;

     EMPNO ENAME          DEPTNO        SAL       RANK
---------- ---------- ---------- ---------- ----------
      7839 KING               10       5000          1
      7782 CLARK              10       2450          2
      7934 MILLER             10       1300          3
      7788 SCOTT              20       3000          1
      7902 FORD               20       3000          1
      7566 JONES              20       2975          3
      7876 ADAMS              20       1100          4
      7369 SMITH              20        800          5
      7698 BLAKE              30       2850          1
      7499 ALLEN              30       1600          2
      7844 TURNER             30       1500          3
      7521 WARD               30       1250          4
      7654 MARTIN             30       1250          4
      7900 JAMES              30        950          6
```

■ DENSE_RANK

RANK는 동일한 순위가 존재하면 그 다음 순위가 존재하지 않는다. 즉, RANK의 경우, 2등이 2건이면 3등은 존재 할 수 없으며, 그 다음 순위는 4등이 되는 반면, DENSE_RANK는 동일한 순위가 있더라도, 순위가 차례대로 계산된다.

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  RANK() OVER (ORDER BY SAL DESC) RANK,
  3  DENSE_RANK() OVER (ORDER BY SAL DESC) DENSE_RANK
  4  FROM EMP;

     EMPNO ENAME          SAL       RANK DENSE_RANK
---------- ---------- ---------- ---------- ----------
      7839 KING           5000          1          1
      7788 SCOTT          3000          2          2
      7902 FORD           3000          2          2
      7566 JONES          2975          4          3
      7698 BLAKE          2850          5          4
      7782 CLARK          2450          6          5
      7499 ALLEN          1600          7          6
      7844 TURNER         1500          8          7
      7934 MILLER         1300          9          8
      7521 WARD           1250         10          9
      7654 MARTIN         1250         10          9
      7876 ADAMS          1100         12         10
      7900 JAMES           950         13         11
      7369 SMITH           800         14         12
```

■ CUME_DIST

CUME_DIST는 RANK 함수와 같이 파티션 내에서 ORDER BY에 지정된 컬럼을 기준으로 순위를 계산하지만, 최대값 1을 기준으로 순위를 0~1사이의 값으로 표시한다.

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  RANK() OVER (ORDER BY SAL DESC) RANK,
  3  CUME_DIST() OVER (ORDER BY SAL DESC) CUME_DIST
  4  FROM EMP;

     EMPNO ENAME             SAL       RANK  CUME_DIST
---------- ---------- ---------- ---------- ----------
      7839 KING             5000          1 .071428571  <-  1건/14건
      7788 SCOTT            3000          2 .214285714
      7902 FORD             3000          2 .214285714
      7566 JONES            2975          4 .285714286
      7698 BLAKE            2850          5 .357142857
      7782 CLARK            2450          6 .428571429
      7499 ALLEN            1600          7          .5
      7844 TURNER           1500          8 .571428571
      7934 MILLER           1300          9 .642857143
      7521 WARD             1250         10 .785714286
      7654 MARTIN           1250         10 .785714286
      7876 ADAMS            1100         12 .857142857
      7900 JAMES             950         13 .928571429
      7369 SMITH             800         14          1  <- 14건/14건
```

■ PERCENT_RANK

PERCENT_RANK는 CUME_DIST와 유사하지만, 계산 방법은 약간 다르다.

PERCENT_RANK = (파티션 내 자신의 RANK-1)/(파티션 내의 행 개수-1)

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  RANK() OVER (ORDER BY SAL DESC) RANK,
  3  PERCENT_RANK() OVER (ORDER BY SAL DESC) PERCENT_RANK
  4  FROM EMP;

     EMPNO ENAME             SAL       RANK PERCENT_RANK
---------- ---------- ---------- ---------- ------------
      7839 KING             5000          1            0 <-  (1-1)/(14-1)
      7788 SCOTT            3000          2   .076923077
      7902 FORD             3000          2   .076923077
      7566 JONES            2975          4   .230769231
      7698 BLAKE            2850          5   .307692308
      7782 CLARK            2450          6   .384615385
      7499 ALLEN            1600          7   .461538462
      7844 TURNER           1500          8   .538461538
      7934 MILLER           1300          9   .615384615
      7521 WARD             1250         10   .692307692
      7654 MARTIN           1250         10   .692307692
      7876 ADAMS            1100         12   .846153846
      7900 JAMES             950         13   .923076923
      7369 SMITH             800         14            1 <- (14-1)/(14-1)
```

■ NTILE(N)

파티션 내의 행들을 N개로 분류하여 행의 위치를 표시한다.

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  NTILE(7) OVER (ORDER BY SAL DESC) NTILE
  3  FROM EMP;

     EMPNO ENAME             SAL      NTILE
---------- ---------- ---------- ----------
      7839 KING             5000          1
      7788 SCOTT            3000          1
      7902 FORD             3000          2
      7566 JONES            2975          2
      7698 BLAKE            2850          3
      7782 CLARK            2450          3
      7499 ALLEN            1600          4
      7844 TURNER           1500          4
      7934 MILLER           1300          5
      7521 WARD             1250          5
      7654 MARTIN           1250          6
      7876 ADAMS            1100          6
      7900 JAMES             950          7
      7369 SMITH             800          7
```

■ ROW_NUMBER

파티션 내에 행들에 대하여 차례로 순번을 표시한다. 가상 컬럼은 ROWNUM과 유사하다.

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  ROW_NUMBER() OVER (ORDER BY SAL DESC) ROW_NUMBER
  3  FROM EMP;

     EMPNO ENAME             SAL ROW_NUMBER
---------- ---------- ---------- ----------
      7839 KING             5000          1
      7788 SCOTT            3000          2
      7902 FORD             3000          3
      7566 JONES            2975          4
      7698 BLAKE            2850          5
      7782 CLARK            2450          6
      7499 ALLEN            1600          7
      7844 TURNER           1500          8
      7934 MILLER           1300          9
      7521 WARD             1250         10
      7654 MARTIN           1250         11
      7876 ADAMS            1100         12
      7900 JAMES             950         13
      7369 SMITH             800         14
```

윈도우(Windowing) 함수

윈도우 함수란 기존의 집계 함수에 대하여 다음과 같은 기능을 지원하는 함수이다.

```
{SUM|AVG|MAX|MIN|COUNT|STDDEV|VARIANCE|FIRST_VALUE|LAST_VALUE}
        ({<value expression1>|*}) OVER
                ([PARTITION BY <value expression2>[,...]]
                ORDER BY <value expression3>[collate clause]
                [ASC|DESC][NULLS FIRST|NULLS LAST][,...]
                ROWS|RANGE
                {{UNBOUNDED PRECEDING|<value expression4> PRECEDING}
                |BETWEEN
                {UNBOUNDED PRECEDING|<value expression5> PRECEDING}
                AND {CURRENT ROW|<value expression6> FOLLOWING}}
```

여기서,

- OVER : 결과 집합을 이용해 동작하는 함수라는 의미이다.
- PARTITION BY : 결과 집합이 *value expression1*에 지정된 값으로 분할된다.
- ORDER BY : 각 파티션 내의 결과가 *value expression2*에 지정된 값으로 정렬된다.
- NULLS FIRST|NULLS LAST : NULL이 포함된 행이 파티션 내에서 가장 먼저 위치 할 것인지 가장 마지막에 위치 할 것인지를 지정
- ROWS|RANGE : 연산 할 행들을 파티션 내에서 물리적(ROWS) 순서를 이용하여 선택 할 것인지 논리적(RANGE) 순서를 이용하여 선택 할 것인지를 결정
- BETWEEN ... AND ... : 파티션 내에서 연산 할 행들의 범위를 결정
- UNBOUNDED PRECEDING : 파티션 내에서 지정된 행 이전의 모든 행을 포함
- UNBOUNDED FOLLOWING : 파티션 내에서 지정된 행 이후의 모든 행을 포함
- CURRENT ROW : 파티션 내에서 현재 행을 시작 행 또는 마지막 행으로 이용 할 때 사용

다음과 같은 다양한 예제를 통해 윈도우 함수의 사용법에 대해서 살펴본다.

먼저, 사원을 입사일로 정렬하고, 각 사원의 급여와 누적 급여를 출력하는 방법은 다음과 같다. UNBOUNDED PRECEDING은 현재 행과 이전 행을 대상으로 집계 함수를 적용한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, SAL,
  2  SUM(SAL) OVER (ORDER BY HIREDATE ROWS UNBOUNDED PRECEDING) CUM_SAL
  3  FROM EMP;

     EMPNO ENAME      HIREDATE        SAL    CUM_SAL
---------- ---------- -------- ---------- ----------
      7369 SMITH      80/12/17        800        800
      7499 ALLEN      81/02/20       1600       2400
      7521 WARD       81/02/22       1250       3650
      7566 JONES      81/04/02       2975       6625
      7698 BLAKE      81/05/01       2850       9475
      7782 CLARK      81/06/09       2450      11925
      7844 TURNER     81/09/08       1500      13425
      7654 MARTIN     81/09/28       1250      14675
      7839 KING       81/11/17       5000      19675
      7900 JAMES      81/12/03        950      20625
      7902 FORD       81/12/03       3000      23625
      7934 MILLER     82/01/23       1300      24925
      7788 SCOTT      87/04/19       3000      27925
      7876 ADAMS      87/05/23       1100      29025
```

PARTITION BY를 사용하면 부서별 누적 급여를 출력 할 수도 있다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, SAL, DEPTNO,
  2  SUM(SAL) OVER (PARTITION BY DEPTNO ORDER BY HIREDATE
  3                 ROWS UNBOUNDED PRECEDING) CUM_SAL
  4  FROM EMP;

     EMPNO ENAME      HIREDATE        SAL     DEPTNO    CUM_SAL
---------- ---------- -------- ---------- ---------- ----------
      7782 CLARK      81/06/09       2450         10       2450
      7839 KING       81/11/17       5000         10       7450
      7934 MILLER     82/01/23       1300         10       8750
      7369 SMITH      80/12/17        800         20        800
      7566 JONES      81/04/02       2975         20       3775
      7902 FORD       81/12/03       3000         20       6775
      7788 SCOTT      87/04/19       3000         20       9775
      7876 ADAMS      87/05/23       1100         20      10875
      7499 ALLEN      81/02/20       1600         30       1600
      7521 WARD       81/02/22       1250         30       2850
      7698 BLAKE      81/05/01       2850         30       5700
      7844 TURNER     81/09/08       1500         30       7200
      7654 MARTIN     81/09/28       1250         30       8450
      7900 JAMES      81/12/03        950         30       9400
```

각 사원들의 급여와 해당 사원 및 먼저 입사한 사원 3명의 평균 급여를 출력하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE, SAL,
  2  AVG(SAL) OVER (ORDER BY HIREDATE ROWS 3 PRECEDING) AVG_SAL
  3  FROM EMP;

     EMPNO ENAME      HIREDATE        SAL    AVG_SAL
---------- ---------- -------- ---------- ----------
      7369 SMITH      80/12/17        800        800
      7499 ALLEN      81/02/20       1600       1200
      7521 WARD       81/02/22       1250 1216.66667
      7566 JONES      81/04/02       2975    1656.25 <- (800+1600+1250+2975)/4
      7698 BLAKE      81/05/01       2850    2168.75
      7782 CLARK      81/06/09       2450    2381.25
      7844 TURNER     81/09/08       1500    2443.75
      7654 MARTIN     81/09/28       1250     2012.5
      7839 KING       81/11/17       5000       2550
      7900 JAMES      81/12/03        950       2175
      7902 FORD       81/12/03       3000       2550
      7934 MILLER     82/01/23       1300     2562.5
      7788 SCOTT      87/04/19       3000     2062.5
      7876 ADAMS      87/05/23       1100       2100 <- (3000+1300+3000+1100)/4
```

사원의 급여와 해당 부서의 최대 급여 및 최소 급여를 출력하는 방법은 다음과 같다. FIRST_VALUE는 해당 PARTITION의 첫 번째 행들을 선택하며, LAST_VALUE는 해당 파티션의 마지막 행들을 선택한다. 아래 예제에서는 각각 MIN, MAX를 사용해도 결과는 동일하다.

```
SQL> SELECT EMPNO, ENAME, SAL, DEPTNO,
  2  FIRST_VALUE(SAL) OVER (PARTITION BY DEPTNO ORDER BY SAL
  3     ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) MIN_SAL,
  4  LAST_VALUE(SAL) OVER (PARTITION BY DEPTNO ORDER BY SAL
  5     ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) MAX_SAL
  6  FROM EMP;

     EMPNO ENAME             SAL     DEPTNO    MIN_SAL    MAX_SAL
---------- ---------- ---------- ---------- ---------- ----------
      7934 MILLER           1300         10       1300       5000
      7782 CLARK            2450         10       1300       5000
      7839 KING             5000         10       1300       5000
      7369 SMITH             800         20        800       3000
      7876 ADAMS            1100         20        800       3000
      7566 JONES            2975         20        800       3000
      7788 SCOTT            3000         20        800       3000
      7902 FORD             3000         20        800       3000
      7900 JAMES             950         30        950       2850
      7521 WARD             1250         30        950       2850
      7654 MARTIN           1250         30        950       2850
      7844 TURNER           1500         30        950       2850
      7499 ALLEN            1600         30        950       2850
      7698 BLAKE            2850         30        950       2850
```

각 사원의 급여와 전체 급여에서 각 사원의 급여가 차지하는 비율을 출력하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, SAL,
  2  RATIO_TO_REPORT(SAL) OVER() RATIO
  3  FROM EMP;

     EMPNO ENAME             SAL      RATIO
---------- ---------- ---------- ----------
      7369 SMITH             800 .027562446
      7499 ALLEN            1600 .055124892
      7521 WARD             1250 .043066322
      7566 JONES            2975 .102497847
      7654 MARTIN           1250 .043066322
      7698 BLAKE            2850 .098191214
      7782 CLARK            2450 .084409991
      7788 SCOTT            3000 .103359173
      7839 KING             5000 .172265289
      7844 TURNER           1500 .051679587
      7876 ADAMS            1100 .037898363
      7900 JAMES             950 .032730405
      7902 FORD             3000 .103359173
      7934 MILLER           1300 .044788975
```

각 사원의 급여와 해당 사원보다 직전에 입사한 사원의 급여 및 직후에 입사한 사원의 급여를 출력하는 방법은 다음과 같다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE,
  2  LAG(HIREDATE, 1) OVER (ORDER BY HIREDATE) LAG_HIREDATE,
  3  LEAD(HIREDATE, 1) OVER (ORDER BY HIREDATE) LEAD_HIREDATE
  4  FROM EMP;

     EMPNO ENAME      HIREDATE LAG_HIRE LEAD_HIR
---------- ---------- -------- -------- --------
      7369 SMITH      80/12/17          81/02/20
      7499 ALLEN      81/02/20 80/12/17 81/02/22
      7521 WARD       81/02/22 81/02/20 81/04/02
      7566 JONES      81/04/02 81/02/22 81/05/01
      7698 BLAKE      81/05/01 81/04/02 81/06/09
      7782 CLARK      81/06/09 81/05/01 81/09/08
      7844 TURNER     81/09/08 81/06/09 81/09/28
      7654 MARTIN     81/09/28 81/09/08 81/11/17
      7839 KING       81/11/17 81/09/28 81/12/03
      7900 JAMES      81/12/03 81/11/17 81/12/03
      7902 FORD       81/12/03 81/12/03 82/01/23
      7934 MILLER     82/01/23 81/12/03 87/04/19
      7788 SCOTT      87/04/19 82/01/23 87/05/23
      7876 ADAMS      87/05/23 87/04/19
```

복습

1. 다음과 같이 급여가 높은 순서대로 순위를 출력하는 쿼리를 작성하시오.

EMPNO	ENAME	SAL	순위
7839	KING	5000	1
7788	SCOTT	3000	2
7902	FORD	3000	2
7566	JONES	2975	4
7698	BLAKE	2850	5
7782	CLARK	2450	6
7499	ALLEN	1600	7
7844	TURNER	1500	8
7934	MILLER	1300	9
7521	WARD	1250	10
7654	MARTIN	1250	10
7876	ADAMS	1100	12
7900	JAMES	950	13
7369	SMITH	800	14

2. 다음과 같이 업무별 급여가 높은 순으로 순위를 출력하는 쿼리를 작성하시오.

EMPNO	ENAME	SAL	JOB	업무별순위
7788	SCOTT	3000	ANALYST	1
7902	FORD	3000	ANALYST	1
7934	MILLER	1300	CLERK	1
7876	ADAMS	1100	CLERK	2
7900	JAMES	950	CLERK	3
7369	SMITH	800	CLERK	4
7566	JONES	2975	MANAGER	1
7698	BLAKE	2850	MANAGER	2
7782	CLARK	2450	MANAGER	3
7839	KING	5000	PRESIDENT	1
7499	ALLEN	1600	SALESMAN	1
7844	TURNER	1500	SALESMAN	2
7521	WARD	1250	SALESMAN	3
7654	MARTIN	1250	SALESMAN	3

3. 다음과 같이 부서별 최대 급여가 높은 순서대로 순위를 출력하시오.

DEPTNO	MAX(SAL)	순위
10	5000	1
20	3000	2
30	2850	3

4. 다음과 같이 동일한 연도에 입사한 사원들에 대하여 입사일을 기준으로 순위를 출력하시오.

EMPNO	ENAME	입사	HIREDATE	년도별순위
7369	SMITH	1980	80/12/17	1
7499	ALLEN	1981	81/02/20	1
7521	WARD	1981	81/02/22	2
7566	JONES	1981	81/04/02	3
7698	BLAKE	1981	81/05/01	4
7782	CLARK	1981	81/06/09	5
7844	TURNER	1981	81/09/08	6
7654	MARTIN	1981	81/09/28	7
7839	KING	1981	81/11/17	8
7900	JAMES	1981	81/12/03	9
7902	FORD	1981	81/12/03	9
7934	MILLER	1982	82/01/23	1
7788	SCOTT	1987	87/04/19	1
7876	ADAMS	1987	87/05/23	2

5. 다음과 같이 각 사원의 급여와 업무별 누적 급여를 출력하시오.

EMPNO	ENAME	SAL	JOB	누적급여
7788	SCOTT	3000	ANALYST	3000
7902	FORD	3000	ANALYST	6000
7369	SMITH	800	CLERK	800
7900	JAMES	950	CLERK	1750
7876	ADAMS	1100	CLERK	2850
7934	MILLER	1300	CLERK	4150
7782	CLARK	2450	MANAGER	2450
7698	BLAKE	2850	MANAGER	5300
7566	JONES	2975	MANAGER	8275
7839	KING	5000	PRESIDENT	5000
7521	WARD	1250	SALESMAN	1250
7654	MARTIN	1250	SALESMAN	2500
7844	TURNER	1500	SALESMAN	4000
7499	ALLEN	1600	SALESMAN	5600