Built-in function

Oracle 기준.

- SELECT, WHERE, ORDER BY 절에 사용 가능.
- Arguments로 함수 사용 가능.

Data Manipulation Language, SQL, Database

Single-row function

문자형 함수

LOWER, UPPER

Case 변환

ASCII, CHR

아스키코드 변환

- ASCII('문자열')

```
ASCII('A')
-- 65
```

- CHR(ASCII번호)

```
CHR(65)
-- 'A'
```

CONCAT

문자열 붙이기. '문자열1'||'문자열2' 과 동일.

- CONCAT('문자열1', '문자열2')

```
CONCAT('Hello', ' World')
-- 'Hello World'
```

SUBSTR

문자열 추출

- SUBSTR('문자열', INDEX[, 추출할_개수])

[, 추출할_개수]를 생략할 경우 문자열 끝까지 추출

```
SUBSTR('Hello World', 7, 2)
-- 'or'
```

Length

Length('문자열')

LTRIM, RTRIM, TRIM

- LTRIM('문자열'[, '지정문자'])

```
LTRIM('xxxHello Worldxxx', 'x')
-- ''Hello Worldxxx'
```

- RTRIM([, '지정문자'])

```
LTRIM('xxxHello Worldxxx', 'x')
-- 'xxxHello World'
```

- TRIM([leading|trailing|both] '지정문자' FROM '문자열')

Default : both

```
TRIM('x' FROM 'xxxHello Worldxxx')
-- 'Hello World'
```

숫자형 함수

ABS

ABS(숫자)

SIGN

양수/음수/0 구별

```
SIGN(-99)
-- -1

SIGN(0)
-- 0

SIGN(77)
-- 1
```

CEIL, FLOOR

- CEIL(숫자)

올림(숫자보다 크거나 같은 최소 정수 반환)

```
CEIL(38.123)
-- 39

CEIL(-38.123)
-- -38
```

- FLOOR(숫자)

버림(숫자보다 작거나 같은 최소 정수 반환)

```
FLOOR(38.123)
-- 38

FLOOR(-38.123)
-- -39
```

ROUND, TRUNC

- ROUND(숫자[, m])

m자리 올림

```
ROUND(77.77777, 3)
-- 77.778
```

- TRUNC(숫자[, m])

m자리 뒤 버림 (m 생략시 Default : 0)

```
ROUND(77.77777, 3)
-- 77.777
```

SIN, COS, TAN

삼각함수

EXP, POWER, SQRT, LOG, LN

- EXP : 지수
- POWER : 거듭 제곱
- SQRT : 제곱근
- LOG : 로그
- LN : 자연 로그

날짜형 함수

- SYSGATE
  - 현재 날짜와 시각 출력
- EXTRACT(YEAR|MONTH|DAY from d)
  - 년/월/일 / 시/분/초 출력
- TO_NUMBER(TO_CHAR(d, 'YYYY'|'MM'|'DD'))
  - 년/월/일 출력

```
SELECT ENAME, SYSDATE+1, HIREDATE, EXTRACT(YEAR from HIREDATE),
TO_NUMBER(TO_CHAR(HIREDATE, 'MMDD'))
FROM EMP;
```

	ENAME	SYSDATE+1	HIREDATE	EXTRACT(YEARFROMHIREDATE)	TO_NUMBER(TO_CHAR(HIREDATE,'MMDD'))
1	SMITH	15/03/07	80/12/17	1980	1217
2	ALLEN	15/03/07	81/02/20	1981	220
3	WARD	15/03/07	81/02/22	1981	222
4	JONES	15/03/07	81/04/02	1981	402
5	MARTIN	15/03/07	81/09/28	1981	928
6	BLAKE	15/03/07	81/05/01	1981	501
7	CLARK	15/03/07	81/06/09	1981	609
8	SCOTT	15/03/07	87/07/13	1987	713
9	KING	15/03/07	81/11/17	1981	1117
10	TURNER	15/03/07	81/09/08	1981	908
11	ADAMS	15/03/07	87/07/13	1987	713
12	JAMES	15/03/07	81/12/03	1981	1203
13	FORD	15/03/07	81/12/03	1981	1203
14	MILLER	15/03/07	82/01/23	1982	123

변환형 함수

- TO_NUMBER('문자열')
- TO_CHAR(숫자|날짜[, FORMAT])
- TO_DATE('문자열'[, FORMAT])

```
SELECT TO_CHAR(855726.872/1101.32159, 'L999,999,999.999') --L은 Local 화폐
FROM DUAL;
-- ￦777.000
```

CASE 표현식

```
--※ 모든 반환값의 Type은 일치되어야 한다!

--SIMPLE CASE EXPRESSION 조건 : (CASE 뒤 기준 표현이) EQUI조건인 경우
SELECT
    CASE 칼럼|조건식 WHEN 결과값 THEN 반환값(칼럼명|칼럼값)
                     ...
                     [ELSE 반환값(칼럼명|칼럼값)] -- 생략시 NULL
    END[ [AS] 별칭]
FROM 테이블명


--SEARCHED CASE EXPRESSION 조건 : EQUI조건이 아닌 경우 (좀 더 다양한 조건을 가질 경우)
SELECT
    CASE WHEN 조건식 THEN 반환값(칼럼명|칼럼값)
         ...
         [ELSE 반환값(칼럼명|칼럼값)] -- 생략시 NULL
    END[ [AS] 별칭]
```

```
FROM 테이블명
```

- 예시

```
SELECT ENAME, HIREDATE,
       CASE EXTRACT(YEAR FROM HIREDATE) WHEN 1981 THEN 'Since 1980'
                                        ELSE 'ETC'
       END HIREYEAR,
       SAL,
       CASE WHEN SAL>=2000 THEN 1000
            ELSE (CASE WHEN SAL>=1000 THEN 500
                  ELSE 0
                  END)
       END BONUS
FROM EMP;
```

	ENAME	HIREDATE	HIREYEAR	SAL	BONUS
1	SMITH	80/12/17	ETC	800	0
2	ALLEN	81/02/20	Since 1980	1600	500
3	WARD	81/02/22	Since 1980	1250	500
4	JONES	81/04/02	Since 1980	2975	1000
5	MARTIN	81/09/28	Since 1980	1250	500
6	BLAKE	81/05/01	Since 1980	2850	1000
7	CLARK	81/06/09	Since 1980	2450	1000
8	SCOTT	87/07/13	ETC	3000	1000
9	KING	81/11/17	Since 1980	5000	1000
10	TURNER	81/09/08	Since 1980	1500	500
11	ADAMS	87/07/13	ETC	1100	500
12	JAMES	81/12/03	Since 1980	950	0
13	FORD	81/12/03	Since 1980	3000	1000
14	MILLER	82/01/23	ETC	1300	500

DECODE

Oracle 전용. CASE 표현이 EQUI 조건일 때, 짧게 표현할 수 있으므로 유용.

```
DECODE(표현식|칼럼, 기준값1, 반환값1[, 기준값2, 반환값2, ... , Default값])
-- Default값 : CASE표현의 ELSE 뒤 반환값
```

- 예시

부서별로 월별 입사자의 평균 급여 조회

```

```

```
SELECT DEPTNO,
  AVG(CASE HIREMONTH WHEN 1 THEN SAL END) M01,
  AVG(CASE HIREMONTH WHEN 2 THEN SAL END) M02,
  AVG(CASE HIREMONTH WHEN 3 THEN SAL END) M03,
  AVG(CASE HIREMONTH WHEN 4 THEN SAL END) M04,
  AVG(CASE HIREMONTH WHEN 5 THEN SAL END) M05,
  AVG(CASE HIREMONTH WHEN 6 THEN SAL END) M06,
  AVG(DECODE(HIREMONTH, 7, SAL)) M07,
  AVG(DECODE(HIREMONTH, 8, SAL)) M08,
  AVG(DECODE(HIREMONTH, 9, SAL)) M09,
  AVG(DECODE(HIREMONTH, 10, SAL)) M10,
  AVG(DECODE(HIREMONTH, 11, SAL)) M11,
  AVG(DECODE(HIREMONTH, 12, SAL)) M12
FROM (SELECT ENAME, DEPTNO, EXTRACT(MONTH FROM HIREDATE) HIREMONTH, SAL FROM
EMP)
GROUP BY DEPTNO;
```

	DEPTNO	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12
1	30	(null)	1425	(null)	(null)	2850	(null)	(null)	(null)	1375	(null)	(null)	950
2	20	(null)	(null)	(null)	2975	(null)	(null)	2050	(null)	(null)	(null)	(null)	1900
3	10	1300	(null)	(null)	(null)	(null)	2450	(null)	(null)	(null)	(null)	5000	(null)

NULL 관련 함수

NVL

- 반환값을 NULL이 아닌 값으로 대체할 때 사용.
  - 산술연산에서 데이터값이 NULL일 때 유용!
  - 공집합 대신 다른 값 출력할 때!

```
SELECT
    NVL(NULL 판단 대상, NULL일 때 대체값)
FROM 테이블명;
```

NULLIF

- 반환값을 NULL로 대체할 때 사용.
- 두 인자가 같으면 NULL을, 다르면 표현식1 반환

```
SELECT
    NULLIF(EXPR1, EXPR2)
FROM 테이블명;
```

COALESCE

- 인자들값 중 NULL이 아니면 그 인자값을, 모두 NULL이면 NULL 반환

```
SELECT
    COALESCE(EXPR1, EXPR2, ...)
FROM 테이블명;
```

Multi-row function

Aggregate function

여러 행들의 그룹이 모여서 그룹당 단 하나의 결과를 돌려주는 함수

- GROUP BY 절은 행들을 소그룹화 한다.
- SELECT, HAVING, ORDER BY절에 사용할 수 있다.

문법	설명
COUNT(*|표현식)	NULL 제외 계산 문자,날짜 Type에도 사용 가능
MAX([DISTINCT|ALL] 표현식) MIN([DISTINCT|ALL] 표현식)	문자,날짜 Type에도 사용 가능
SUM([DISTINCT|ALL] 표현식)	NULL 제외 계산
AVG([DISTINCT|ALL] 표현식)	
STDDEV([DISTINCT|ALL] 표현식)	표준편차
VARIAN([DISTINCT|ALL] 표현식)	분산

- 활용법
  - “집계함수(CASE()) ~ GROUP BY” (예제 참조)
  - NULL 처리 함수
- 예시

팀별 각 포지션(FW, MF, DF, GK)의 인원수와 팀별 전체 인원수, 평균키 출력

```
SELECT TEAM_ID,
  NVL(SUM(DECODE(POSITION, 'FW', 1)), 0) FW,
  NVL(SUM(DECODE(POSITION, 'MF', 1)), 0) MF,
  NVL(SUM(DECODE(POSITION, 'DF', 1)), 0) DF,
  NVL(SUM(DECODE(POSITION, 'GK', 1)), 0) GK,
  COUNT(*) SUM,
  ROUND(AVG(HEIGHT), 2) AVG_HEIGHT
FROM PLAYER
GROUP BY TEAM_ID;
```

Group function

(집계 함수 제외하고,) 소그룹 간의 소계(전체가 아닌 어느 한 부분만을 셈한 합계) 계산

ROLLUP

- 집계 함수 제외하고, 소그룹 간의 소계(전체가 아닌 어느 한 부분만을 셈한 합계) 계산

```
SELECT DNAME, JOB, COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ROLLUP(DNAME, JOB) --DNAME, JOB에 대한 "소"그룹 집"계" 출력
ORDER BY DNAME, JOB;
```

	DNAME	JOB	TOTAL EMP	TOTAL SAL
1	ACCOUNTING	CLERK	1	1300
2	ACCOUNTING	MANAGER	1	2450
3	ACCOUNTING	PRESIDENT	1	5000
4	RESEARCH	ANALYST	2	6000
5	RESEARCH	CLERK	2	1900
6	RESEARCH	MANAGER	1	2975
7	SALES	CLERK	1	950
8	SALES	MANAGER	1	2850
9	SALES	SALESMAN	4	5600

CUBE

- GROUP BY 항목들간 다차원적인 소계 계산
- ROLLUP 함수 같은 각 소계 출력 및 전체 총계까지 출력

```
SELECT DNAME, JOB, COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY CUBE(DNAME, JOB)
ORDER BY DNAME, JOB;
```

	DNAME	JOB	TOTAL EMP	TOTAL SAL
1	ACCOUNTING	CLERK	1	1300
2	ACCOUNTING	MANAGER	1	2450
3	ACCOUNTING	PRESIDENT	1	5000
4	ACCOUNTING	(null)	3	8750
5	RESEARCH	ANALYST	2	6000
6	RESEARCH	CLERK	2	1900
7	RESEARCH	MANAGER	1	2975
8	RESEARCH	(null)	5	10875
9	SALES	CLERK	1	950
10	SALES	MANAGER	1	2850
11	SALES	SALESMAN	4	5600
12	SALES	(null)	6	9400
13	(null)	ANALYST	2	6000
14	(null)	CLERK	4	4150
15	(null)	MANAGER	3	8275
16	(null)	PRESIDENT	1	5000
17	(null)	SALESMAN	4	5600
18	(null)	(null)	14	29025

GROUPING

- ROLLUP/CUBE 함수와 함께 사용되는 함수로, 어떤 칼럼이 해당 Grouping 작업에 사용되었는 지 아닌 지 구별해주는 역할.
    - 사용되었으면 0, 아니면 1 (= ROLLUP/CUBE 함수에 의해 계산된 결과는 1, 아니면 0)
- **CASE/DECODE** 함수를 이용해, 소계를 나타내는 필드에 사용자 문자열을 정의해 보고서 작성시 유용하게 활용 가능.

```
SELECT DNAME, GROUPING(DNAME),
       JOB, GROUPING(JOB),
       COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ROLLUP(DNAME, JOB);
```

	DNAME	GROUPING(DNAME)	JOB	GROUPING(JOB)	TOTAL EMP	TOTAL SAL
1	SALES	0	CLERK	0	1	950
2	SALES	0	MANAGER	0	1	2850
3	SALES	0	SALESMAN	0	4	5600
4	SALES	0	(null)	1	6	9400
5	RESEARCH	0	CLERK	0	2	1900
6	RESEARCH	0	ANALYST	0	2	6000
7	RESEARCH	0	MANAGER	0	1	2975
8	RESEARCH	0	(null)	1	5	10875
9	ACCOUNTING	0	CLERK	0	1	1300
10	ACCOUNTING	0	MANAGER	0	1	2450
11	ACCOUNTING	0	PRESIDENT	0	1	5000
12	ACCOUNTING	0	(null)	1	3	8750
13	(null)	1	(null)	1	14	29025

GROUPING SETS

특정 항목에 대한 소계 계산

- 일반 그룹함수 이용

```
SELECT 'ALL DEPTS' DNAME, JOB, COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY JOB
UNION ALL
SELECT DNAME, 'ALL JOBS' JOB, COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DNAME;
```

- GROUPING SETS 함수 이용

```
SELECT DECODE(GROUPING(DNAME), 1, 'ALL DEPTS', DNAME) DNAME,
       DECODE(GROUPING(JOB), 1, 'ALL JOBS', JOB) JOB,
       COUNT(*) "TOTAL EMP", SUM(SAL) "TOTAL SAL"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY GROUPING SETS(JOB, DNAME); --칼럼간의 평등관계로 순서가 바뀌어도 결과는 같다.
```

	DNAME	JOB	TOTAL EMP	TOTAL SAL
1	ALL DEPTS	CLERK	4	4150
2	ALL DEPTS	SALESMAN	4	5600
3	ALL DEPTS	PRESIDENT	1	5000
4	ALL DEPTS	MANAGER	3	8275
5	ALL DEPTS	ANALYST	2	6000
6	ACCOUNTING	ALL JOBS	3	8750
7	RESEARCH	ALL JOBS	5	10875
8	SALES	ALL JOBS	6	9400

Window function

- 부분적으로 행과 행간의 관계를 쉽게 정의하기 위한 함수
  - PARTITION BY : 전체 집합을 기준에 의해 소그룹으로 나눔.
  - ORDER BY : 어떤 항목에 대해 **순위**를 지정할 지 기술.
  - WINDOWING : 함수의 대상이 되는 **행** 기준의 **범위**를 강력하게 지정. (SQL Server 미지원)
    - ROWS : 물리적인 결과 행의 수
    - RANGE : 논리적인 값에 의한 범위

```
SELECT WINDOW_FUNCTION (ARGUMENTS) OVER
       ([PARTITION BY 칼럼] [ORDER BY 절] [WINDOWING 절])
FROM 테이블명;
```

그룹 내 순위(Rank) 관련 함수

RANK

특정 칼럼에 대한 순위를 구하는 함수 (단, 동일 순위가 있으면 그 수만큼 순위가 넘어가버림)

```
SELECT JOB, ENAME, SAL,
       RANK() OVER (ORDER BY SAL DESC) ALL_RANK, --전체 급여 순위
       RANK() OVER (PARTITION BY JOB ORDER BY SAL DESC) JOB_RANK --직업별 급여
순위
FROM EMP;
```

DENSE_RANK

RANK 함수와 흡사하나, 동일한 순위를 하나의 건수로 취급.

```
SELECT JOB, ENAME, SAL,
       RANK()       OVER (ORDER BY SAL DESC) RANK,
```

```
       DENSE_RANK() OVER (ORDER BY SAL DESC) DENSE_RANK
FROM EMP;
```

- 4, 12행 주목!

	JOB	ENAME	SAL	RANK	DENSE_RANK
1	PRESIDENT	KING	5000	1	1
2	ANALYST	FORD	3000	2	2
3	ANALYST	SCOTT	3000	2	2
4	MANAGER	JONES	2975	4	3
5	MANAGER	BLAKE	2850	5	4
6	MANAGER	CLARK	2450	6	5
7	SALESMAN	ALLEN	1600	7	6
8	SALESMAN	TURNER	1500	8	7
9	CLERK	MILLER	1300	9	8
10	SALESMAN	WARD	1250	10	9
11	SALESMAN	MARTIN	1250	10	9
12	CLERK	ADAMS	1100	12	10
13	CLERK	JAMES	950	13	11
14	CLERK	SMITH	800	14	12

ROW_NUMBER

동일값이라도 고유한 순위 부여. 동일 등수를 인정하지 않음.

```
SELECT JOB, ENAME, SAL,
       RANK()       OVER (ORDER BY SAL DESC) RANK,
       ROW_NUMBER() OVER (ORDER BY SAL DESC) DENSE_RANK --SAL 기준에 따른 줄번호
순
FROM EMP;
```

그룹 내 집계(Aggregate) 관련 함수

SUM

```
SELECT MGR, ENAME, SAL,
       SUM(SAL) OVER (PARTITION BY MGR) MGR_SUM --MGR별 급여 합계
FROM EMP;
```

MAX

```
SELECT MGR, ENAME, SAL,
       MAX(SAL) OVER (PARTITION BY MGR) MGR_MAX --MGR별 급여 최대값
FROM EMP;
```

MIN

```
SELECT MGR, ENAME, SAL,
       MIN(SAL) OVER (PARTITION BY MGR ORDER BY HIREDATE) MGR_MIN --MGR별 급여 최소값, 입사일 기준 정렬
FROM EMP;
```

AVG

```
SELECT MGR, ENAME, SAL,
       ROUND(AVG(SAL) OVER (PARTITION BY MGR), 2) MGR_AVG --MGR별 급여 평균
FROM EMP;
```

COUNT

```
SELECT ENAME, SAL,
       COUNT(*) OVER (ORDER BY SAL  RANGE BETWEEN 50 PRECEDING AND 150 FOLLOWING) SIM_CNT
FROM EMP;

--RANGE BETWEEN 50 PRECEDING AND 150 FOLLOWING : 현재 행의 급여값을 기준으로 급여가 -50 ~ +150 범위 내에 포함된 모든 행이 대상
```

그룹 내 행 순서 관련 함수

모두 SQL Server에서 미지원.

FIRST_VALUE, LAST_VALUE

- FIRST_VALUE : 파티션별 윈도우에서 가장 먼저 나온 값 반환.
- LAST_VALUE : 파티션별 윈도우에서 가장 나중에 나온 값 반환.

```
SELECT DEPTNO, ENAME, SAL,
```

```
       FIRST_VALUE(ENAME) OVER (PARTITION BY DEPTNO ORDER BY SAL DESC
                                ROWS UNBOUNDED PRECEDING) AS DEPT_RICH
FROM EMP;

--ROWS UNBOUNDED PRECEDING : 현재 행 지준으로 "파티션 내 첫 번째 행까지" 범위 지정
```

	DEPTNO	ENAME	SAL	DEPT_RICH
1	10	KING	5000	KING
2	10	CLARK	2450	KING
3	10	MILLER	1300	KING
4	20	SCOTT	3000	SCOTT
5	20	FORD	3000	SCOTT
6	20	JONES	2975	SCOTT
7	20	ADAMS	1100	SCOTT
8	20	SMITH	800	SCOTT
9	30	BLAKE	2850	BLAKE
10	30	ALLEN	1600	BLAKE
11	30	TURNER	1500	BLAKE
12	30	MARTIN	1250	BLAKE
13	30	WARD	1250	BLAKE
14	30	JAMES	950	BLAKE

LAG, LEAD

- LAG : 파티션별 윈도우에서 이전 몇 번째 행의 값 반환. (지연, 미루기)
- LEAD : 파티션별 윈도우에서 이후 몇 번째 행의 값 반환. (당기기, 이끌기)

	ENAME	HIREDATE	SAL	PREV_SAL
1	ALLEN	81/02/20	1600	0
2	WARD	81/02/22	1250	0
3	TURNER	81/09/08	1500	1600
4	MARTIN	81/09/28	1250	1250

그룹 내 비율 관련 함수

NTILE 이외 함수들은 SQL Server에서 미지원.

RATIO_TO_REPORT

파티션 내 전체 칼럼 합계값에 대한 백분율 반환.

PERCENT_RANK

파티션별 윈도우에서 제일 먼저 나오는 행값을 0, 가장 늦게 나오는 행값을 1로 하여 행 순서별 백분율 반환.

(상위 몇퍼센트에 있는가?)

- 예시

같은 부서 소속 사원들의 집합에서 본인의 급여가 순서상 몇 번째 위치쯤에 있는 지 0과 1사이의 값으로 조회.

```
SELECT DEPTNO, ENAME, SAL,
       PERCENT_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) PER_RANK
FROM EMP;
```

	DEPTNO	ENAME	SAL	PER_RANK
1	10	KING	5000	0
2	10	CLARK	2450	0.5
3	10	MILLER	1300	1
4	20	SCOTT	3000	0
5	20	FORD	3000	0
6	20	JONES	2975	0.5
7	20	ADAMS	1100	0.75
8	20	SMITH	800	1
9	30	BLAKE	2850	0
10	30	ALLEN	1600	0.2
11	30	TURNER	1500	0.4
12	30	MARTIN	1250	0.6
13	30	WARD	1250	0.6
14	30	JAMES	950	1

CUME_DIST

파티션별 윈도우의 전체건수에서 현재 행보다 작거나 같은 건수에 대한 누적백분율 반환.

NTILE(n)

파티션별 전체 건수를 n등분한 Argument값 반환 (조 짜기)

ROWNUM

테이블이나 집합에서 **원하는 만큼의 행만 가져오고 싶을 때** 사용하는 가상 칼럼(Pseudo Column)

SQL Server에서는

```
-- (SELECT 절에서)
TOP(n) WITH TIES
-- WITH TIES : 동일 수치 데이터 추가
```

```
SELECT문
WHERE ROWNUM 비교연산자 수치

-- 예시 : 3행까지 가져올 때
SELECT *
FROM MYTABLE
WHERE ROWNUM <= 3;
```

- 테이블 내의 고유한 키나 인덱스 값 생성. (☞ 오름차순으로 번호가 가상 생성되는 특성 이용!)

```
UPDATE 테이블명
SET 칼럼명 = ROWNUM;
```

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=relational_database:built-in_function

Last update: **2021/02/07 04:15**