



SQL

✓ 원리를 알면 IT가 맛있다

SQL for Beginners



chapter 03.

SELECT

- SELECT 문
- null 값 특징 및 NVL 함수
- WHERE 조건
- 비교 연산자
- 조건 연산자
- 정렬

■ SELECT 기능

: 데이터베이스로부터 **데이터를 검색**하는 기능을 갖는다.

- selection : 질의에 대해 테이블의 행을 선택하기 위해 사용.
- projection : 질의에 대해 테이블의 열을 선택하기 위해 사용.
- join : 여러 테이블이 공통적으로 가진 컬럼을 이용해서 다른 테이블에 저장되어 있는 데이터를 가져오기 위해 사용.

■ 기본적인 SELECT 문법

```
SELECT [DISTINCT] { *, column [alias],... }  
FROM table;
```

Projection

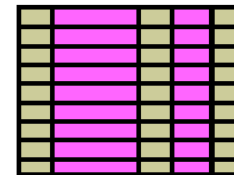


Table 1

Selection

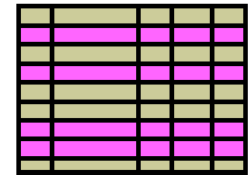


Table 1

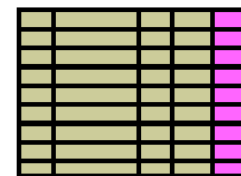


Table 1

Join

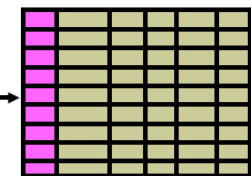


Table 2

- scott 계정이 소유한 테이블 목록 보기

```
SQL> SELECT * FROM TAB;
```

TNAME	TABTYPE	CLUSTERID
BONUS	TABLE	
DEPT	TABLE	
EMP	TABLE	
SALGRADE	TABLE	

```
SQL>
```

- 특정 테이블의 컬럼 구조 보기

```
SQL> DESC DEPT
```

이름	널?	유형
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

■ 테이블내의 모든 데이터 보기

```
SQL> SELECT * FROM DEPT;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

■ 테이블내의 특정 컬럼 데이터 보기

- SELECT 뒤에 해당 컬럼을 차례대로 기술. 쉼표로 구분해서 여러 개 지정 가능.

```
SQL> SELECT EMPNO, ENAME, JOB, HIREDATE FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE
7369	SMITH	CLERK	80/12/17
7499	ALLEN	SALESMAN	81/02/20
7521	WARD	SALESMAN	81/02/22
...			
7900	JAMES	CLERK	81/12/03
7902	FORD	ANALYST	81/12/03
7934	MILLER	CLERK	82/01/23

□ 2) 산술연산을 이용한 SQL 문

SQL

- SQL 문장내의 숫자 및 날짜 타입에는 + , - , * , / , () 사용 가능하다.
- 연산식이 컬럼명으로 표시된다.

```
SQL> SELECT EMPNO, ENAME, SAL * 1.1 FROM EMP;
```

EMPNO	ENAME	SAL*1.1
7369	SMITH	880
7499	ALLEN	1760
7521	WARD	1375
...		
7900	JAMES	1045
7902	FORD	3300
7934	MILLER	1430

- 컬럼에 별칭(alias) 사용.

```
SQL> SELECT EMPNO AS 사번, ENAME AS 성명, SAL AS 급여 FROM EMP;
```

사번	성명	급여
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
...		
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300

- 이용 불가능한(unavailable), 비교 자체가 불가능한
 - 지정되지 않은(unassigned)
 - 알 수 없는 (unknown)
 - 적용할 수 없는(Inapplicable) 값을 의미한다.
-
- 오라클은 컬럼에 기본적으로 null값을 허용하며 제약조건을 이용해서 null 값을 허용하지 않을 수도 있다. 주의할 점은 null 값의 연산결과는 null 값으로 나온다는 것이다.
 - null값의 비교는 IS NULL , IS NOT NULL 이라는 정해진 문구를 사용해야 제대로 된 결과를 얻을 수 있다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM FROM EMP;
```

EMPNO	ENAME	SAL	COMM
7369	SMITH	800	
7499	ALLEN	1600	300
7521	WARD	1250	500
7566	JONES	2975	
7654	MARTIN	1250	1400
...			
7902	FORD	3000	
7934	MILLER	1300	

```
SQL> SELECT EMPNO, ENAME, COMM, COMM + 100 FROM EMP;
```

EMPNO	ENAME	COMM	COMM+100
7369	SMITH		
7499	ALLEN	300	400
7521	WARD	500	600
7566	JONES		
...			
7902	FORD		
7934	MILLER		

□ 3) null 값

null 값을 가진 컬럼을 연산하기 위해서는 NVL 또는 NVL2 함수를 사용할 수 있다.

NVL 함수의 사용법은 다음과 같으며, 컬럼값이 null인 경우에 기본값으로 설정한다.

NVL (컬럼명, 기본값) or NVL2(컬럼명, A, B)

```
SQL> SELECT empno, ename, comm, NVL(comm,0) + 100 , NVL2( comm, 1, 2 )  
2 FROM emp;
```

EMPNO	ENAME	COMM	NVL(COMM,0)+100	NVL2(COMM,1,2)
7369	SMITH		100	2
7499	ALLEN	300	400	1
7521	WARD	500	600	1
7566	JONES		100	2
7654	MARTIN	1400	1500	1

- 여러 개의 문자열을 연결하여 하나의 문자열로 생성.

```
SQL> SELECT ENAME || JOB AS "이름 직업" FROM EMP;
```

이름 직업

SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
JONESMANAGER
MARTINSALESMAN
BLAKEMANAGER

- 리터럴(Literal)

- SELECT 문장에 포함된 컬럼명 또는 별칭 이외의 문자값, 숫자값, 날짜값이다.
- 반드시 문자값, 날짜값에는 ‘ ’ 을 붙인다.

```
SQL> SELECT ENAME || '의 직급은 ' || JOB || '이다' AS "사원별 직급" FROM EMP;
```

사원별 직급

SMITH의 직급은 CLERK이다
ALLEN의 직급은 SALESMAN이다
WARD의 직급은 SALESMAN이다
...
JAMES의 직급은 CLERK이다
FORD의 직급은 ANALYST이다
MILLER의 직급은 CLERK이다

- DISTINCT 키워드 이용
중복된 값 제거하여 한번만 출력.

```
SQL> SELECT JOB FROM EMP;
```

```
JOB  
-----  
CLERK  
SALESMAN  
SALESMAN  
MANAGER  
SALESMAN  
MANAGER  
MANAGER  
ANALYST  
PRESIDENT  
SALESMAN  
CLERK  
CLERK  
ANALYST  
CLERK
```



```
SQL> SELECT DISTINCT JOB FROM EMP;
```

```
JOB  
-----  
ANALYST  
CLERK  
MANAGER  
PRESIDENT  
SALESMAN
```

■ WHERE 기능

- 테이블내의 모든 행을 검색하는 대신 검색 조건을 지정하여 사용자가 원하는 행들만 검색하는 기능.

■ 기본적인 SELECT ~ WHERE 문법

```
SELECT [DISTINCT] { *, column [alias],... }
FROM table
[WHERE 조건식];
```

```
SQL> SELECT EMPNO, ENAME, JOB, DEPTNO
2 FROM EMP
3 WHERE DEPTNO = 30;
```

EMPNO	ENAME	JOB	DEPTNO
7499	ALLEN	SALESMAN	30
7521	WARD	SALESMAN	30
7654	MARTIN	SALESMAN	30
7698	BLAKE	MANAGER	30
7844	TURNER	SALESMAN	30
7900	JAMES	CLERK	30

6 개의 행이 선택되었습니다.

```
SQL> SELECT EMPNO, ENAME, JOB, DEPTNO
2 FROM EMP
3 WHERE JOB = 'SALESMAN';
```

EMPNO	ENAME	JOB	DEPTNO
7499	ALLEN	SALESMAN	30
7521	WARD	SALESMAN	30
7654	MARTIN	SALESMAN	30
7844	TURNER	SALESMAN	30

```
SQL> SELECT EMPNO , ENAME , JOB, DEPTNO
2 FROM EMP
3 WHERE HIREDATE = '81/11/17';
```

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10

연산자	의미
=	같다
>	보다 크다
>=	보다 크거나 같다
<	보다 작다
<=	보다 작거나 같다
<>	다르다

여기서, != 와 ^= 는 <> 와 동일한 의미를 갖는다.

```
SQL> SELECT EMPNO, ENAME, SAL
2  FROM EMP
3  WHERE SAL <= 1000;
```

EMPNO	ENAME	SAL
7369	SMITH	800
7900	JAMES	950

□ 6) WHERE – 비교 연산자 2

연산자	의미
BETWEEN ... AND ...	두 값의 범위에 포함되는
IN (set)	괄호 안의 값과 일치하는
LIKE	문자의 조합이 같은
IS NULL	널 값

```
SQL> SELECT EMPNO, ENAME, SAL
2 FROM EMP
3 WHERE SAL BETWEEN 1000 AND 2000;
```

EMPNO	ENAME	SAL
7499	ALLEN	1600
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7934	MILLER	1300

```
SQL> SELECT EMPNO, ENAME, JOB
2 FROM EMP
3 WHERE EMPNO IN (7839, 7844, 7876);
```

EMPNO	ENAME	JOB
7876	ADAMS	CLERK
7844	TURNER	SALESMAN
7839	KING	PRESIDENT

```
SQL> SELECT EMPNO, ENAME, COMM
2 FROM EMP
3 WHERE COMM IS NULL;
```

EMPNO	ENAME	COMM
7369	SMITH	
7566	JONES	
7698	BLAKE	
7782	CLARK	
7788	SCOTT	
7839	KING	
7876	ADAMS	
7900	JAMES	
7902	FORD	
7934	MILLER	

- 다중 리스트

예> 사원테이블에서 JOB이 MANAGER이면서 20번 부서에 속하거나, JOB이 CLERK이면서 30번 부서에 속하는 사원의 정보를 출력?

```
SQL> SELECT ENAME, JOB, DEPTNO  
2 FROM EMP  
3 WHERE (JOB,DEPTNO) IN ( ('MANAGER',20),('CLERK',30) );
```

ENAME	JOB	DEPTNO
JONES	MANAGER	20
JAMES	CLERK	30

□ 6) WHERE – 비교 연산자 2

- LIKE 연산자

- 검색하고자 하는 문자열을 정확히 알 수 없는 경우에 사용.
- 패턴 매칭 연산자 이용 (와일드 카드)

기호	설명
%	0 글자 이상의 임의 문자를 대표한다.
_	1 글자의 임의 문자를 대표한다

```
SQL> SELECT EMPNO, ENAME, JOB  
2 FROM EMP  
3 WHERE ENAME LIKE 'A%';
```

EMPNO	ENAME	JOB
7499	ALLEN	SALESMAN
7876	ADAMS	CLERK

```
SQL> SELECT EMPNO, ENAME, JOB  
2 FROM EMP  
3 WHERE ENAME LIKE '%T%';
```

EMPNO	ENAME	JOB
7369	SMITH	CLERK
7654	MARTIN	SALESMAN
7788	SCOTT	ANALYST
7844	TURNER	SALESMAN

```
SQL> SELECT EMPNO, ENAME, JOB  
2 FROM EMP  
3 WHERE ENAME LIKE '_L%';
```

EMPNO	ENAME	JOB
7499	ALLEN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER

- 검색하고자 하는 문자열에 패턴 매칭 연산자가 포함되어 있을 때는 ESCAPE 옵션을 사용한다.

```
SQL> SELECT EMPNO, ENAME, JOB  
2 FROM EMP  
3 WHERE ENAME LIKE '%W%' ESCAPE 'W';
```


□ 6) WHERE – 논리 연산자

- WHERE 절에 부여할 조건이 여러 개인 경우에 사용한다.

연산자	의미
AND	두개의 조건이 TRUE이면 TRUE를 리턴
OR	두개의 조건중 하나의 조건이 TRUE이면 TRUE를 리턴
NOT	조건이 FALSE이면 TRUE를 리턴

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
2 FROM EMP
3 WHERE JOB = 'SALESMAN'
4 AND SAL >= 1500;
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7844	TURNER	SALESMAN	1500

```
SQL> SELECT EMPNO, ENAME, COMM
2 FROM EMP
3 WHERE COMM IS NOT NULL;
```

EMPNO	ENAME	COMM
7499	ALLEN	300
7521	WARD	500
7654	MARTIN	1400
7844	TURNER	0

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
2 FROM EMP
3 WHERE JOB = 'SALESMAN'
4 OR SAL >= 1500
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7844	TURNER	SALESMAN	1500
7902	FORD	ANALYST	3000

□ 6) WHERE – 논리 연산자

SQL

- 예> emp 테이블에서 JOB이 'CLERK' 이거나 'ANALYST'이고, COMM이 NULL 이고 sal가 1000보다 크고 3000보다 작은 사원 정보 ?

```
SQL> SELECT ename, job, sal, comm
2 FROM emp
3 WHERE job = 'CLERK' OR job = 'ANALYST'
4 AND comm IS NULL
5 AND sal >= 1000
6 AND sal <= 3000;
```

ENAME	JOB	SAL	COMM
SMITH	CLERK	800	
SCOTT	ANALYST	3000	
ADAMS	CLERK	1100	
JAMES	CLERK	950	
FORD	ANALYST	3000	
MILLER	CLERK	1300	

```
SQL> SELECT ename, job, sal, comm
2 FROM emp
3 WHERE ( job = 'CLERK' OR job = 'ANALYST')
4 AND comm IS NULL
5 AND sal >= 1000
6 AND sal <= 3000;
```

ENAME	JOB	SAL	COMM
SCOTT	ANALYST	3000	
ADAMS	CLERK	1100	
FORD	ANALYST	3000	
MILLER	CLERK	1300	

↓
괄호가 누락되어 OR 연산자보다 AND 연산자를 먼저 실행되어 잘못된 결과가 출력됨.

논리 연산자들이 여러 개가 같이 사용되었을 때의 처리 순서는 (),NOT,AND,OR 순이다.

- WHERE 절에 사용 가능한 부정 연산자.

	연산자	의미
부정 비교 연산자	!=	같지 않다.
	^=	같지 않다.
	<>	같지 않다.
	NOT 컬럼명 =	~ 와 같지 않다.
	NOT 컬럼명 >	~ 보다 크지 않다.
부정 SQL 연산자	NOT BETWEEN a AND b	a와 b값 사이에 있지 않다.
	NOT IN (리스트)	리스트값과 일치하지 않는다.
	IS NOT NULL	NULL 값을 갖지 않는다.

- 연산자 우선 순위

연산자 우선순위	설명
1	괄호()
2	NOT 연산자
3	비교 연산자
4	AND
5	OR

- SELECT 문장에 의해 검색된 결과를 정렬 (기본은 ASC)

```
SELECT [DISTINCT] { *, column [alias],... }
FROM table
[WHERE 조건식]
[ORDER BY {column , exp } [ASC|DESC] ];
```

- null
-오라클은 null값을 가장 큰 값으로 간주한다.

```
SQL> SELECT EMPNO, ENAME, HIREDATE
2 FROM EMP
3 ORDER BY HIREDATE;
```

EMPNO	ENAME	HIREDATE
7369	SMITH	80/12/17
7499	ALLEN	81/02/20
7521	WARD	81/02/22
7566	JONES	81/04/02
...		

```
SQL> SELECT EMPNO, ENAME, HIREDATE
2 FROM EMP
3 ORDER BY HIREDATE DESC;
```

EMPNO	ENAME	HIREDATE
7876	ADAMS	87/05/23
7788	SCOTT	87/04/19
7934	MILLER	82/01/23
7900	JAMES	81/12/03
...		

```
SQL> SELECT EMPNO, ENAME, SAL * 12 ANNUAL
2 FROM EMP
3 ORDER BY ANNUAL;
```

EMPNO	ENAME	ANNUAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7521	WARD	15000
...		

```
SQL> SELECT EMPNO, ENAME, SAL * 12 ANNUAL
2 FROM EMP
3 ORDER BY 3;
```

EMPNO	ENAME	ANNUAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7521	WARD	15000
...		

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

EMPNO	ENAME	SAL
7839	KING	5000
7788	SCOTT	3000
7902	FORD	3000
7566	JONES	2975
7698	BLAKE	2850
7782	CLARK	2450
7499	ALLEN	1600



Thank you
