SQL, Database

- DML 명령어의 경우, 조작하려는 테이블을 메모리 버퍼에 올려놓고 작업을 하기 때문에 **실시간으로 테이블에 영향을 미치는 것은 아니다.** 따라서 **COMMIT** 명령어를 입력하여 Transaction을 종료시켜 실제 테이블에 반영시켜야 한다!

INSERT

```
--유형1 : 칼럼 순서대로 기술 필요X. 빠진 칼럼은 Default로 "NULL"(PK, NOT NULL 제외)
INSERT INTO 테이블명(속성명1, 속성명2, ...)
VALUES (데이터1, 데이터2, ...);

--유형2 : 칼럼 순서대로 빠짐없이!
INSERT INTO 테이블명
VALUES (데이터1, 데이터2, ...);
```

UPDATE

```
UPDATE 테이블명
SET 속성명1=데이터1[, 속성명2=데이터2, ...]
[WHERE 조건];
```

DELETE

```
DELETE [FROM] 테이블명
[WHERE 조건];
```

- Table 전체 데이터 삭제 경우, Log를 저장하지 않아(= ROLLBACK 불가) 시스템 부하가 적은 **TRUNCATE TABLE** 권장.

SELECT

```
SELECT [Predicate] [테이블명|테이블별칭.]속성명[ [AS] 별칭] [, [테이블명|테이블별칭.]속성명[ [AS] 별칭], ...]
 --전체 칼럼 지정은 "*"(Asterisk)
FROM 테이블명1[ [AS] 별칭][, 테이블명2[ [AS] 별칭], ...]
```

```
   --□ TABLE에 별칭(Alias)를 지정했을 경우, SELECT와 WHERE절에 별칭을 반드시 사용해야 한다!
[WHERE 조건]
[GROUP BY 표현식|속성명[, 속성명, ...]
[HAVING 조건]] --GROUP에 대한 조건 기술
[ORDER BY 표현식|속성명 [ASC|DESC][, 속성명 [ASC|DESC], ...]]


-- 별칭 지정 방법2 : 공백, 특수문자 포함 or 대소문자 구분이 필요한 경우
SELECT [Predicate] [테이블명.]속성명 [AS] "별칭" [, [테이블명.]속성명 [AS] "별칭", ...]
...
```

- Predicate : 검색할 Tuple을 제한할 목적으로 사용되는 조건
  - ALL : Default 옵션으로 생략 가능. 중복된 데이터가 있어도 모두 출력. (기본합)
  - DISTINCT : 중복 데이터 제거, 1건으로 처리 후 출력. (중복 제거)
  - DISTINCTROW : Tupe 전체에 대한 중복 제거. (DISTINCT와 동의어. ORACLE 지원X. MySQL, CUBRID에서 지원.)
- 임의의 문자열/숫자 칼럼을 추가할 수 있다. (칼럼명[[AS] 별칭])

SELECT문의 실행(읽는) 순서

```
□ SELECT
□ FROM
□ WHERE
□ GROUP BY
□ HAVING
□ ORDER BY

추출 대상 테이블 참조                (FROM)
→ 추출 대상 데이터만 거르기          (WHERE)
 → 행들을 소그룹화                    (GROUP BY)
  → 소그룹의 대상 데이터만 거르기  (HAVING)
   → 데이터 값 출력/계산             (SELECT)
    → 데이터 정렬                     (ORDER BY)
```

증명1) SELECT 절에 없는 칼럼을 ORDER BY 절에 사용. (⑤ SELECT → ⑥ ORDER BY)

```
SELECT EMPNO
FROM EMP
ORDER BY MGR;
```

증명2) Sub Query(인라인 뷰에서 정의된 SELECT) 칼럼을 Main Query에서 사용 (① FROM → ~ → ⑤ SELECT)

```
SELECT EMPNO -- 만약 FROM절에서 정의되지 않은 테이블|표현식(칼럼) 사용하면 Error.
```

```
FROM (SELECT EMPNO, ENAME
      FROM EMP
      ORDER BY MGR);
```

연산자 (주로 SELECT)

연산자를 사용할 경우, 칼럼 Label이 길어지고 칼럼에 새 의미를 부여하기 때문에 **별칭(Alias)**를 부여하는 것이 좋다!

산술 연산자

- (데이터가 수치 연산이 가능한 Type(NUMBER, DATE)값일 때) **SELECT**, **WHERE**, **HAVING**, **ORDER BY** 절에서 사용 가능.
- 우선 순위 : **()** > * > **/** > **+** > **-**

```
SELECT NAME 이름, ROUND(WEIGHT/POWER(HEIGHT,2), 2) "BMI 비만지수" -- 비만지수(Body Mass Index)
FROM SCHOOL;
```

```
표
```

합성(CONCATENATION) 연산자

- **SELECT** 절에서만 사용 가능.
- 칼럼/문자 연결 (속성명 + 문자 or 속성명 + 속성명)
- Oracle : || / SQL Server : **+**

```
SELECT NAME || '의 키는 ' || HEIGHT || 'cm' 몸무게
FROM SCHOOL;
```

```
표
```

WHERE 절

- Column명 (보통 조건의 좌측 위치)
- 비교 연산자
- 문자, 숫자, 표현식 (보통 조건식이 우측 위치)
- 비교 Column명 (JOIN 사용시)

연산자 (WHERE, HAVING 사용)

- 우선 순위

:() > NOT > 비교 연산자, SQL 연산자 > AND > OR

비교 연산자

표

문자 유형 비교 방법

- 비교 연산자 양쪽 모두 CHAR인 경우
  - 길이가 서로 다르면, 짧은 쪽에 **Space** 추가 하여 같게 만들고 비교
    - 서로 다른 문자 나올 때까지 비교
    - 달라진 첫 번째 문자의 값에 따라 크기 결정
    - Blank 수만 다르면, 서로 같은 값으로 결정.
- 비교 연산자 한쪽 VARCHAR인 경우 (**※ VARCHAR은 NOT NULL까지 길이를 가진다!**)
  - 서로 다른 문자 나올 때까지 비교
    - 길이가 다르면, 짧은 것이 끝날 때까지만 비교한 후에 길이가 긴 것이 크다고 판단.
    - 길이가 같고 다른 것이 없다면, 같다고 판단
- 상수값과 비교할 경우
  - 상수 쪽을 상대 변수 타입과 동일하게 바꾸고 비교

SQL 연산자

표

논리 연산자

표

GROUP BY, HAVING 절

- GROUP BY 절 : 특정 속성 같은 값이 같은 소그룹에 대한 **통계** 정보
  - GROUP BY 절을 사용하게 되면, ①**그룹핑 기준에 사용된 Column**과 ②**집계 함수에 사용될 수 있는 숫자형 데이터 Column들의 집합**을 새로 만들게 된다. 그리고 **개별 데이터는 저장하지 않는다.**

:: ☞ GROUP BY **이후**에 수행 절인(SELECT문 실행(읽는) 순서) SELECT절 or ORDER BY절에서 **개별 데이터**를 사용하는 경우 Error!

- HAVING 절 : 소그룹의 조건

예) 최대키가 190cm 이상인 포지션의 평균키 출력

```
SELECT POSITION, MIN(HEIGHT), MAX(HEIGHT), ROUND(AVG(HEIGHT), 2)
FROM PLAYER
GROUP BY POSITION
HAVING MAX(HEIGHT) >= 190;
```

	POSITION	MIN(HEIGHT)	MAX(HEIGHT)	ROUND(AVG(HEIGHT),2)
1	GK	174	196	186.26
2	DF	170	190	180.21
3	FW	168	194	179.91
4	MF	165	189	176.31

- 주 활용법
  - 집계 함수(Aggregate function)
  - "집계함수(CASE()) ~ GROUP BY" (예제 참조) : 모델링의 제1정규화로 인해 반복되는 칼럼이 많을 때, 구분 칼럼을 두고 여러 개의 레코드로 만들어진 집합을 / 정해진 칼럼 수만큼 확장해서 집계 보고서를 만드는 유용한 기법.

ORDER BY절

- 특정 칼럼을 기준으로 **정렬**하여 출력하는 데 사용
- 특징
  - Default : ASC
  - NULL? Oracle에서는 가장 큰 값으로 간주 ☞ ASC일 경우 가장 마지막 (SQL Server에서는 반대)
  - **Alias(별칭)**이나 **SELECT 절의 칼럼 순서(정수)**로 매핑할 수 있다.
    - 예) DEPT 테이블 정보를 부서명, 지역, 부서번호 내림차순으로 정렬

```
SELECT DNAME, LOC CITY, DEPTNO
FROM DEPT
ORDER BY 1, CITY, 3 DESC; -- 가장 마지막에 붙인 옵션이 전체 적용.
```

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

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=data_manipulation_language&rev=1426602694

Last update: **2026/10/24 07:03**