2021-09-15

**정리**

###########################################################################

SQL 정리 사이트 : <https://psun.tistory.com/entry/SQL-%EB%AC%B8%EB%B2%95-%EC%A0%95%EB%A6%AC>

###########################################################################

가장 최근 값을 추출 : MAX(DATETIME)

가장 옛날 값을 추출 : MIN(DATETIME)

###########################################################################

NULL 혹은 빈 문자열 제외 하는 방법들 ( 함수 / LENGTH )

NULL 값만 불러오기.

###########################################################################

중복된 데이터를 제거할 경우에는 DISTINCT를 쓰자.

DISTINCT : 중복을 없애주지만 정렬을 해주지 않는다.

-> <https://jhnyang.tistory.com/422>

ORDER BY : 중복을 없애주고 정렬도 해준다.

###########################################################################

전체 행 숫자 : count(\*)

###########################################################################

WHERE과 AND를 쓴 예시

->

**SELECT** user\_id , user\_mobile , user\_gender , user\_amount  , user\_addr   **FROM** User\_Table

**WHERE** user\_gender  **=** 'male' **AND** user\_amount   **>=** 5500 **;**

###########################################################################

AS ( alias ) 는 SELECT 뒤에

###########################################################################

SQL 구문 순서 - <https://data-make.tistory.com/23>

SELECT 컬럼명

FORM 테이블명

ON

JOIN

WHERE 테이블 조건

GROUP BY 컬럼명

WITH CUBE 또는 WITH ROLLUP

HAVING 그룹 조건

SELECT

DISTINCT

ORDER BY 컬럼명

TOP

==> 이거 뭔가 이상하다 섞인 것 같다. 수정하기.

LIMIT 1 ; 위에서 하나만 가져오는 것

###########################################################################

WHERE 조건

비교 | =,<>,<,<=,>,>= | ex) price < 20000 \* <>은 다름을 의미한다.

범위 | BETWEEN | ex) price BETWEEN 10000 AND 20000

집합 | IN, NOT IN | ex) price IN ( 10000, 20000, 30000 )

패턴 | LIKE | bookname LIKE ‘축구의 역사’

NULL | IS NULL, IS NOT NULL | ex) price is NULL

복합조건 | AND, OR, NOT | ex) (price < 20000) AND ( book name LIKE ‘축구의 역사’)

###########################################################################

카티션 곱

조인이라고 할 수 없는 조인으로 WHERE 절에 조인 조건을 주지 않는 것을 말함.

두 테이블 기준으로 FROM 절에는 두 개의 테이블을 명시하지만 WHERE 절에서 조인 조건을 주지 않음.

(WHERE 절을 추가하지 않거나 WHERE 절을 추가해도 조인 조건을 주지 않는 경우).

그 결과 두 테이블의 데이터를 기준으로 가능한 모든 조합의 데이터가 조회된다.

ex)

SELECT a.emp\_id, a.emp\_name, a.gender, a.age, a.dept\_id,

b.dept\_id, b.dept\_name, b.use\_yn

FROM emp\_master a

,dept\_master b

ORDER BY a.emp\_id;

###########################################################################

HAVING 절

WHERE 절에서 집계함수를 사용할 수 없다.

HAVING 절은 집계함수를 가지고 조건비교를 할 때 사용한다.

HAVING 절은 GROUP BY 절과 함께 사용이 된다. [\*\*\*]

ex)

**SELECT** b.dname, COUNT(a.empno) "사원수"

**FROM** emp a, dept b

**WHERE** a.deptno = b.deptno

**GROUP** **BY** dname

**HAVING** COUNT(a.empno) > 5;

=> 이런 식으로

###########################################################################

HAVING 절은 WHERE 절과 비슷하지만 그룹 전체, 즉 그룹을 나타내는 결과 집합의 행에만 적용된다.

반면 WHERE 절은 개별 행에 적용된다.

쿼리에는 WHERE 절과 HAVING절 모두 포함될 수 있다. 이 때 이런 식으로 수행하게 된다.

-다이어그램 창에서 테이블이나 테이블 반환 개체의 개별 행에 WHERE 절이 먼저 적용된다. WHERE 절의 조건에 맞는 행만 그룹화된다.

-그 다음 집합의 행에 HAVING 절이 적용된다. HAVING 조건에 맞는 그룹만 쿼리 출력에 표시된다. 집계 함수나 GROUP BY 절에도 나타나는 열에만 HAVING 절을 적용할 수 있다.

###########################################################################

INNER JOIN, OUTER JOIN, CROSS JOIN 정리 -> <https://gent.tistory.com/376>

조인 -> INNER JOIN: 기준 테이블과 조인 테이블 모두 데이터가 존재해야 조회됨

아우터 조인 -> OUTER JOIN : 기준 테이블에만 데이터가 존재하면 조회됨

###########################################################################

[문자열 자르기] LEFT(문자열, 길이) / RIGHT(문자열, 길이) / SUBSTRING(문자열, 시작위치, 길이)

ex) SELECT LEFT(‘가나다라마바사’, 3)

=> 가나다

ex) SELECT RIGHT(‘가나다라마바사’,3)

=> 마바사

ex) SELECT SUBSTRING(‘가나다라마바사’, 3, 3)

=> 다라마

###########################################################################

IFNULL

: 해당 Column의 값이 NULL을 반환할 때, 다른 값으로 출력할 수 있도록 하는 함수

기본구조 : SELECT IFNULL(Column명, “NULL일 경우 대체 값”) FORM 테이블명;

ex) Name Column이 NULL인 경우 “No name”을 출력, NULL이 아닌 경우 NAME Column을 출력

Select IFNULL(NAME. “No name”) as NAME

FROM ANIMAL\_INS

###########################################################################

IFNULL / NULLIF / IF

IFNULL : IFNULL(expr1, expr2)

- expr1이 NULL이면 expr2를 리턴하고, NULL이 아니면 expr1을 리턴한다.

NULLIF : NULLIF(expr1, expr2)

- expr1 = expr2가 True이면 NULL을 리턴하고, 그렇지 않으면 expr1을 리턴한다.

IF : IF(expr1, expr2, expr3)

-expr1이 True이면 expr2를 리턴하고, 그렇지 않은 경우 expr3을 리턴한다.

###########################################################################

CASE WHEN ~ THEN ~ ELSE END

CASE 컬럼

WHEN 조건1 THEN 값1

WHEN 조건2 THEN 값2

WHEN 조건3 THEN 값3

ELSE 값4

END

ex)

USE AdventureWorks2008R2;

GO

SELECT ProductNumber, Name, 'Price Range' =

CASE

WHEN ListPrice = 0 THEN 'Mfg item - not for resale'

WHEN ListPrice < 50 THEN 'Under $50'

WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'

WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'

ELSE 'Over $1000'

END

FROM Production.Product ORDER BY ProductNumber ; GO

=> GO 를 언제 쓰는가

 나타나는 위치를 보면 문장의 끝에 위치 하고 있습니다.

맞았어요~~

 GO 는 그 문장의 끝을 말하는 역할을 합니다.

########################################################################

:a1 컬럼의 값이 null인 경우만 가져오기

SELECT

FROM db\_name

WEHRE a1 is null -- a1컴럼의 NULL값만

:a1 컬럼의 값이 빈 값인 경우만 가져오기

SELECT

FROM db name

WHERE a1 = ‘ ‘ - - a1컬럼의 null 값만

:a1 컬럼에 있는 값의 길이가 0인 경우만 가져오기

SELECT

FROM db\_name

WHERE length(a1) = 0 - - a1 컬럼의 null 값만

########################################################################

ANSI SQL - **ANSI SQL**

**DBMS**(Oracle, My-SQL, DB2 등등)**들에서 각기 다른 SQL를 사용**하므로, **미국 표준 협회**(American National Standards Institute)**에서** 이를 표준화하여 **표준 SQL문을 정립 시켜 놓은 것**이다.

**ANSI SQL 특징**

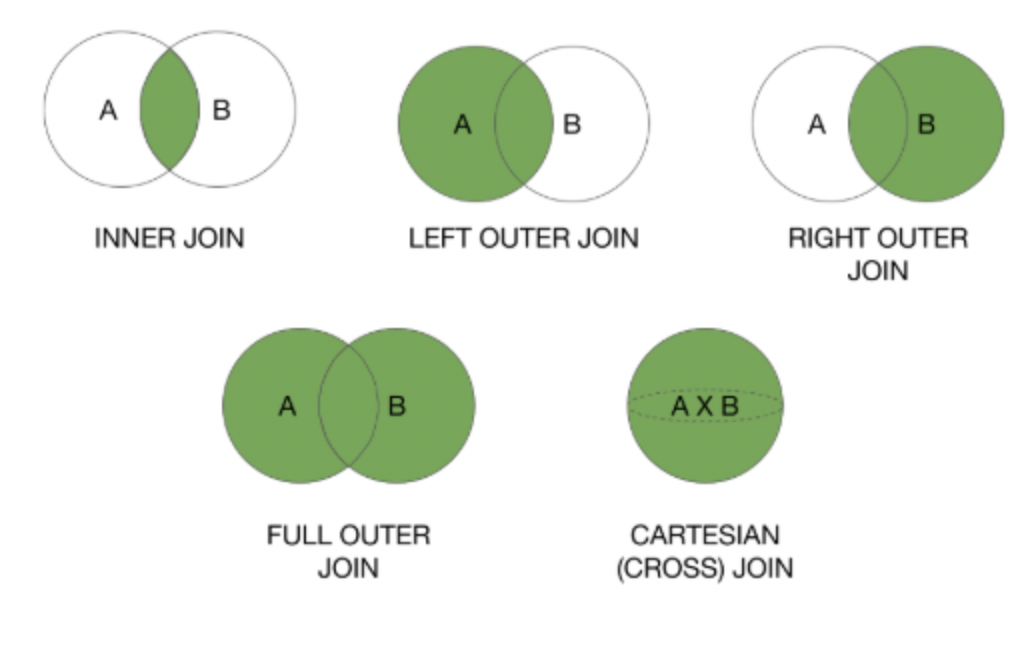
표준 SQL문이기 때문에 DBMS의 종류에 제약을 받지 않는다. (MySQL, Oracle..) 즉*,* 특정벤더에종속적이지않아다른벤더의 *DBMS*로교체하더라도빠르게다른벤더사를이동할수있다*.* 특정 *DBMS*의이탈이가속되는것도 *ANSI SQL*의영향이크다고할수있다*.*

테이블간의 Join 관계가 FROM 에서 명시되기 때문에 WHERE 문에서 조건만 확인하면 된다. 즉*,* 가독성이일반 *Query*문보다좋다*.*

<https://velog.io/@gillog/ANSI-SQL%EC%9D%B4%EB%9E%80>

<https://it-jin-developer.tistory.com/40>

이 2가지로 INNER JOIN 과 ANSI JOIN 개념 이해하기



########################################################################

SELECT

outs.animal\_id, outs.name

FROM animal\_outs AS outs

LEFT JOIN animal\_ins AS ins

ON outs.animal\_id = ins.animal\_id

WHERE

ins.animal\_id IS NULL;

=> 이런식으로 animal\_outs를 outs로 이름 바꾸고 animanl\_outs와 animal\_ins 이렇게 2가지가 있을 때 animal\_outs의 값을 받고 싶을 때

animal\_outs.animal\_id, animal\_outs.name 이런식으로 받는다.

추가로

LEFT JOIN animal\_ins AS ins

ON outs.animal\_id = ins.animal\_id

=> 이 부분을 이해해야 할 것 같다.

#######

**SELECT column\_name(s)**

**FROM table1**

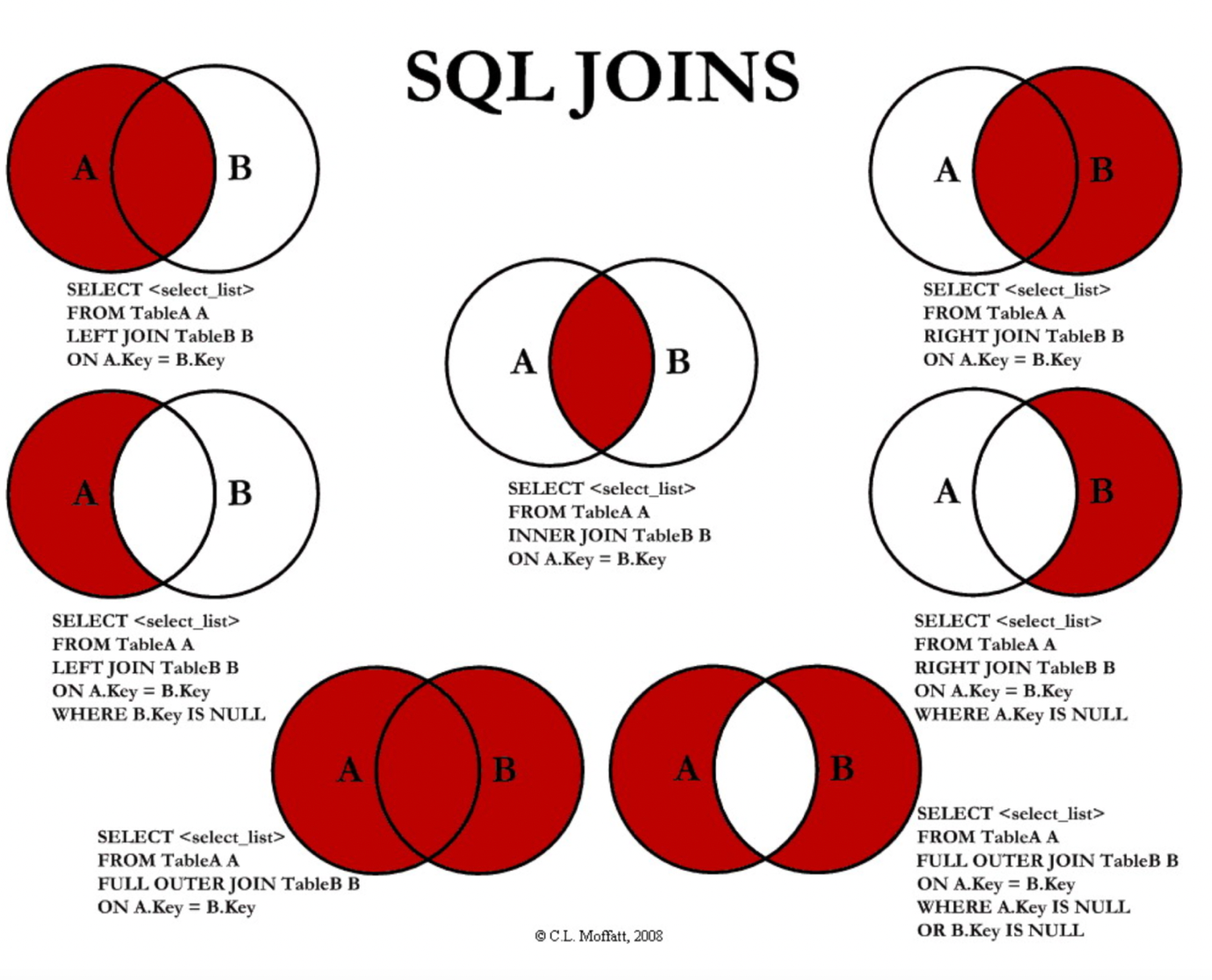
**LEFT JOIN table2**

**ON table1.column\_name=table2.column\_name;**

# inner join과 형태는 같고 inner대신 left로 바뀐 것이다.

테이블이(가) 표시된 사진

자동 생성된 설명



########################################################################