

SQL 검색문 - SELECT

- **SELECT 문**: 관계형 데이터베이스에서 데이터를 검색하는 유일한 문장 형식
- 기본 형식: SELECT-FROM-WHERE

SELECT 애트리뷰트

: 출력할 애트리뷰트 지정, 관계대수의
프로젝트 연산자와 유사

FROM 테이블

: 검색대상 테이블

WHERE 조건

: 검색 조건 지정, 관계대수의 선택
연산자와 유사

SELECT-FROM-WHERE 예

학생
(STUDENT)

- ③ **SELECT** sno, sname
- ① **FROM** student
- ② **WHERE** dept = '컴퓨터'

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

(해석)

FROM절: student 테이블에 존재하는 각 튜플에 대하여,

WHERE절: 학과가 '컴퓨터'인 조건을 만족하는 튜플들을 필터링하고,

SELECT절: 애트리뷰트 sno, sname을 출력하라.

Sno	Sname
100	L.S.Y
300	J.G.T
400	S.B.G

(결과)

sno,sname의 두 애트리뷰트를 가지는 새로운 테이블

SELECT문은 결과를 테이블로 리턴함.

SQL 과 관계 데이터베이스 이론

- SQL 기반 DBMS와 관계 데이터 모델 이론과의 차이점
 - 이론적 관계데이터 모델에서는 한 릴레이션 안에서 중복된 튜플을 허용하지 않음
 - 실제 SQL기반 관계 데이터베이스 시스템에서의 테이블은 기본 키를 반드시 가져야 하는 것은 아님
 - 기본키나 unique constraint가 명시되지 않은 테이블은 같은 원소의 중복을 허용하는 multiset으로 정의됨
- SQL과 관계대수의 공통점
 - 관계 대수와 마찬가지로 입력과 출력이 테이블이라는 점
 - 폐쇄속성(**closure property**): 검색 결과가 또 다시 테이블이 됨
 - 중첩 질의문(nested query)을 구성할 수 있음
- SQL과 관계대수의 차이점
 - 관계대수는 결과에서 중복을 허용하지 않음
 - SQL은 결과에서 중복된 튜플을 허용함

Boolean expression 및 *

학생
(STUDENT)

```
SELECT *  
FROM student  
WHERE not (dept = '컴퓨터' or syear = 3 )
```

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

- *는 모든 애트리뷰트를 출력하고자 할 때 빈번하게 사용함.
- WHERE절에는 and, or, not, () 등을 활용하여 Boolean expression의 조합을 표현할 수 있음.
- 질의문 해석1: 학생 테이블에서 (학과가 컴퓨터 또는 학년이 3학년) 이 아닌 학생들의 모든 애트리뷰트를 검색하라.
- 질의문 해석2: 학생 테이블에서 학과가 컴퓨터가 아니고, 학년도 3학년이 아닌 학생들의 모든 애트리뷰트를 검색하라.

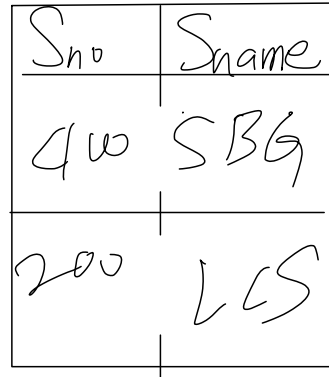
ORDER BY

학생
(STUDENT)

```
SELECT sno, sname  
FROM student
```

```
WHERE syear = 3
```

```
order by sname desc
```



Sno	Sname
400	SBG
200	LLS

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

- SQL 검색결과는 순서가 정해지지 않은 형태로 출력됨. 왜 ? 출력 테이블도 집합이기 때문. 출력 순서를 정해주기 위해서는 order by 절을 사용함.
- 기본 순서는 오름차순(ascending order). desc 는 내림차순(descending order)을 지정함.

문자열 검색

학생
(STUDENT)

```
SELECT sno, sname  
FROM student  
WHERE sname = '정기태'
```

```
SELECT sno, sname  
FROM student  
WHERE sname like '정%'
```

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

- 완전일치 매치(Exact match): WHERE sname = '정기태'
- SQL 에서 부분문자열 매칭(substring matching) 검색이 가능함.
- Regular expression 표현
 - %: 0개 이상의 임의의 문자열을 나타냄
 - _: 임의의 1개 문자를 나타냄

Handwritten diagram explaining SQL wildcards:

% 수 .
= (string) 수 (char)
n개 1개

The diagram shows a box containing the text "% 수 ." and an equals sign followed by "(string) 수 (char)". Below "(string)" is a red arrow pointing to it with the text "n개" (n times). Below "(char)" is a red arrow pointing to it with the text "1개" (1 time).

DISTINCT

학생
(STUDENT)

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

```
SELECT distinct syear  
FROM student  
WHERE dept = '컴퓨터'
```



Year
4
1

$\pi_{\text{Year}} (\sigma_{\text{dept} = \text{'컴퓨터'}} (\text{Student})) \rightarrow$

Year
4
1

- SQL 검색결과는 관계대수와 달리, 중복된 튜플을 허용함.
- 중복된 튜플을 제거한 검색결과를 원할 때에는 SELECT절에 distinct 사용

NULL

학생
(STUDENT)

```
SELECT sno, sname  
FROM student  
WHERE dept is NULL
```

학번 (Sno)	이름 (Sname)	학년 (Year)	학과 (Dept)
100	나 수 영	4	컴퓨터
200	이 찬 수	3	NULL
300	정 기 태	1	컴퓨터
400	송 병 길	4	컴퓨터
500	박 종 화	2	산공

- NULL 은 애트리뷰트 값이 지정되어 없을 때 사용되는 특수한 값
- 길이가 0인 문자열은 NULL 이 아님
- 애트리뷰트 값이 NULL 인지, 아닌지 체크하는 구문: is NULL, is not NULL

dept = NULL (x)

dept <> NULL (x)

```
SELECT sno, sname  
FROM student  
WHERE dept <> '컴퓨터'
```

- <> 는 not equal을 표시하는 구문임
- 어떤 튜플의 dept 값이 NULL 로 지정되어 있을 때, 이 튜플은 이 검색문의 결과로 반환될까 ? No.

→ 무조건 false를 반환

→ select 절에만 쓸 수 있음

집계함수(Aggregation Function)

→ tuple 개수 반환.
SELECT COUNT(*) as e_count
FROM ENROL;

e_count

SELECT COUNT(DISTINCT cno)
FROM ENROL; cno 중복 X 개수.

SELECT AVG(Midterm), MAX(Midterm)
FROM ENROL
WHERE cno = 'C413';

등록
(ENROL)

학번 (Sno)	과목번호 (Cno)	성적 (Grade)	중간성적 (Midterm)	기말성적 (Final)
100	C413	A	90	95
100	E412	A	95	95
200	C123	B	85	80
300	C312	A	90	95
300	C324	C	75	75
300	C413	A	95	90
400	C312	A	90	95
400	C324	A	95	90
400	C413	B	80	85
400	E412	C	65	75
500	C312	B	85	80

- SELECT 절에 집계함수를 ^{int 만} 사용할 수 있음 ^{int 만}
- 집계함수: COUNT, AVG, MIN, MAX, SUM
- 집계함수가 SELECT 절에 나오면, 일반 애트리뷰트는 나올 수 없음. 동시에 여러개의 집계함수를 사용할 수는 있음.
- 집계함수가 SELECT 절에 나오면, 검색결과 테이블의 카디날리티는 1임.

GROUP BY *과 집계/F는 Set.*

등록
(ENROL)

```
SELECT  CNO, AVG(midterm) AS m_avg
FROM    ENROL
WHERE.   Final >= 80
GROUP BY CNO
HAVING count(*) >= 3
ORDER BY AVG(midterm) desc
```

이 조건을 가지는
그룹만 카운다.

C413	90
E412	95
C123	80
C312	...

학번 (Sno)	과목번호 (Cno)	성적 (Grade)	중간성적 (Midterm)	기말성적 (Final)
100	C413	A	90	95
100	E412	A	95	95
200	C123	B	85	80
300	C312	A	90	95
300	C324	C	75	75
300	C413	A	95	90
400	C312	A	90	95
400	C324	A	95	90
400	C413	B	80	85
400	E412	C	65	75
500	C312	B	85	80

- 그룹 애트리뷰트로 입력 테이블을 그룹을 짓고, 각 그룹에 대하여 집계함수를 계산할 수 있음.
- GROUP BY 절을 쓸 경우, SELECT절에는 집계함수 이외에, 그룹 애트리뷰트만 나올 수 있음.
- HAVING은 각 그룹에 대하여 필터링 조건을 명세하는 구문. 집계함수를 사용한 boolean expression을 표현하는 게 일반적임.

GROUP BY

```
SELECT  CNO, AVG(midterm) AS m_avg  
FROM    ENROL  
WHERE.   Final >= 80  
GROUP BY CNO  
HAVING count(*) >= 3  
ORDER BY AVG(midterm) desc
```

- 해석순서: FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY