

코멘토 실무PT 클래스

개발자 없이 시작하는 데이터 분석 - 비개발자를 위한 SQL

1주차

2주차

3주차

4주차

5주차

6주차

22.10.23(일) 20:00~23:00

3주차

데이터들의 "조합"으로
뽐아봅시다

01 2주차 세션 회고 및 과제 피드백

02 JOIN

- JOIN의 종류, 결합키 찾기
- ERD 통한 테이블 관계 확인

03 SUB QUERY

- SUB QUERY의 종류(WITH절), 사용시 유의사항

04 현직자 이야기 : SQL을 잘 읽고 쓰는 법 (코드컨벤션 설명)

05 VSCODE 설치 및 소개

06 실시간 Q&A

과제 JOIN, SUB QUERY 활용하기

- 공공데이터(자전거대여BikeShare) 데이터 추출 연습 진행

Chapter.

1. 지난시간

1주차

2주차

3주차

4주차

5주차

6주차

22.10.02(일) 20:00~23:00

2주차

데이터를 "집계"해서 봅시다

01 1주차 세션 회고 및 과제 피드백

02 GROUP BY
- 집계 함수 정의, 그룹 별 집계 방법03 HAVING
- 그룹 별로 조건을 걸어 원하는 데이터 추출하기04 ORDER BY
- 오름차순 또는 내림차순으로 정렬하기

05 현직자 이야기 : 데이터 요청 잘하는 방법

06 실시간 Q&A

과제 GROUP BY, HAVING, ORDER BY 활용하기
- 공공데이터 (자전거대여BikeShare) 데이터 추출 연습 진행

Chapter.

1. 과제피드백

최댓값 구하기

최댓값 구하기

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
<code>ANIMAL_ID</code>	<code>VARCHAR(N)</code>	FALSE
<code>ANIMAL_TYPE</code>	<code>VARCHAR(N)</code>	FALSE
<code>DATETIME</code>	<code>DATETIME</code>	FALSE
<code>INTAKE_CONDITION</code>	<code>VARCHAR(N)</code>	FALSE
<code>NAME</code>	<code>VARCHAR(N)</code>	TRUE
<code>SEX_UPON_INTAKE</code>	<code>VARCHAR(N)</code>	FALSE

가장 최근에 들어온 동물은 언제 들어왔는지 조회하는 SQL 문을 작성해주세요.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT MAX(DATETIME)
3  FROM ANIMAL_INS
```

실행 결과

실행 결과가 여기에 표시됩니다.

최솟값 구하기

최솟값 구하기

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
INTAKE_CONDITION	VARCHAR(N)	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_INTAKE	VARCHAR(N)	FALSE

동물 보호소에 가장 먼저 들어온 동물은 언제 들어왔는지 조회하는 SQL 문을 작성해주세요.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT MIN(DATETIME)
3  FROM ANIMAL_INS
```

실행 결과

실행 결과가 여기에 표시됩니다.

동물 수 구하기

동물 수 구하기

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
<code>ANIMAL_ID</code>	<code>VARCHAR(N)</code>	FALSE
<code>ANIMAL_TYPE</code>	<code>VARCHAR(N)</code>	FALSE
<code>DATETIME</code>	<code>DATETIME</code>	FALSE
<code>INTAKE_CONDITION</code>	<code>VARCHAR(N)</code>	FALSE
<code>NAME</code>	<code>VARCHAR(N)</code>	TRUE
<code>SEX_UPON_INTAKE</code>	<code>VARCHAR(N)</code>	FALSE

동물 보호소에 동물이 몇 마리 들어왔는지 조회하는 SQL 문을 작성해주세요.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT COUNT(ANIMAL_ID)
3  FROM ANIMAL_INS
```

실행 결과

실행 결과가 여기에 표시됩니다.

중복 제거하기

중복 제거하기

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
INTAKE_CONDITION	VARCHAR(N)	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_INTAKE	VARCHAR(N)	FALSE

동물 보호소에 들어온 동물의 이름은 몇 개인지 조회하는 SQL 문을 작성해주세요. 이때 이름이 NULL인 경우는 집계하지 않으며 중복되는 이름은 하나로 칩니다.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT COUNT(DISTINCT NAME)
3  FROM ANIMAL_INS
```

실행 결과

실행 결과가 여기에 표시됩니다.

고양이와 개는 몇 마리 있을까

고양이와 개는 몇 마리 있을까

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
<code>ANIMAL_ID</code>	<code>VARCHAR(N)</code>	FALSE
<code>ANIMAL_TYPE</code>	<code>VARCHAR(N)</code>	FALSE
<code>DATETIME</code>	<code>DATETIME</code>	FALSE
<code>INTAKE_CONDITION</code>	<code>VARCHAR(N)</code>	FALSE
<code>NAME</code>	<code>VARCHAR(N)</code>	TRUE
<code>SEX_UPON_INTAKE</code>	<code>VARCHAR(N)</code>	FALSE

동물 보호소에 들어온 동물 중 고양이와 개가 각각 몇 마리인지 조회하는 SQL문을 작성해주세요. 이때 고양이를 개보다 먼저 조회해주세요.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT ANIMAL_TYPE, COUNT(ANIMAL_ID)
3  FROM ANIMAL_INS
4  GROUP BY 1
```

실행 결과

실행 결과가 여기에 표시됩니다.

동명 동물 수 찾기

동명 동물 수 찾기

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `INTAKE_CONDITION`, `NAME`, `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
<code>ANIMAL_ID</code>	<code>VARCHAR(N)</code>	FALSE
<code>ANIMAL_TYPE</code>	<code>VARCHAR(N)</code>	FALSE
<code>DATETIME</code>	<code>DATETIME</code>	FALSE
<code>INTAKE_CONDITION</code>	<code>VARCHAR(N)</code>	FALSE
<code>NAME</code>	<code>VARCHAR(N)</code>	TRUE
<code>SEX_UPON_INTAKE</code>	<code>VARCHAR(N)</code>	FALSE

동물 보호소에 들어온 동물 이름 중 두 번 이상 쓰인 이름과 해당 이름이 쓰인 횟수를 조회하는 SQL문을 작성해주세요. 이때 결과는 이름이 없는 동물은 집계에서 제외하며, 결과는 이름 순으로 조회해주세요.

sql

```
1  -- 코드를 입력하세요
2  SELECT NAME, COUNT(ANIMAL_ID)
3  FROM ANIMAL_INS
4  WHERE NAME IS NOT NULL
5  GROUP BY NAME
6  HAVING COUNT(ANIMAL_ID) > 1
7  ORDER BY NAME ASC
```

실행 결과

실행 결과가 여기에 표시됩니다.

입양 시각 구하기(1)

입양 시각 구하기(1)

문제 설명

`ANIMAL_OUTS` 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. `ANIMAL_OUTS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `NAME`, `SEX_UPON_OUTCOME` 는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_OUTCOME	VARCHAR(N)	FALSE

보호소에서는 몇 시에 입양이 가장 활발하게 일어나는지 알아보려 합니다. 09:00부터 19:59까지, 각 시간대별로 입양이 몇 건이나 발생했는지 조회하는 SQL문을 작성해주세요. 이때 결과는 시간대 순으로 정렬해야 합니다.

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT
3      EXTRACT(HOUR FROM DATETIME) as HOUR,
4      COUNT(ANIMAL_ID)
5  FROM ANIMAL_OUTS
6  WHERE EXTRACT(HOUR FROM DATETIME) BETWEEN 9 and 19
7  GROUP BY HOUR
8  ORDER BY HOUR
```

실행 결과

입양 시각 구하기(2)

입양 시각 구하기(2)

문제 설명

`ANIMAL_OUTS` 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. `ANIMAL_OUTS` 테이블 구조는 다음과 같으며, `ANIMAL_ID`, `ANIMAL_TYPE`, `DATETIME`, `NAME`, `SEX_UPON_OUTCOME` 는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_OUTCOME	VARCHAR(N)	FALSE

보호소에서는 몇 시에 입양이 가장 활발하게 일어나는지 알아보려 합니다. 0시부터 23시까지, 각 시간대별로 입양이 몇 건이나 발생했는지 조회하는 SQL문을 작성해주세요. 이때 결과는 시간대 순으로 정렬해야 합니다.

solution.sql

```
1  set @a := -1;
2  SELECT A.HOUR, IFNULL(B.COUNT, 0)
3  FROM (
4      SELECT @a:=@a+1 as HOUR
5      FROM ANIMAL_OUTS
6  ) A
7  LEFT JOIN
8  (
9      SELECT HOUR(DATETIME) as HOUR,
10         COUNT(*) as COUNT
11      FROM ANIMAL_OUTS
12      GROUP BY HOUR(DATETIME)
13  ) B
14  ON A.HOUR = B.HOUR
15  WHERE A.HOUR <= 23
```

실행 결과

HOUR	IFNULL(B.COUNT, 0)
------	--------------------

<WINDOW FUNCTION>

solution.sql

```
1  WITH dummy_hour_t AS ( -- 1. 0~23시 더미데이터.
2  SELECT
3      ROW_NUMBER() OVER(PARTITION BY '') -1 AS HOUR
4  FROM ANIMAL_OUTS
5  LIMIT 24
6  ), animal_out_t AS ( -- 2. 시간대별 입양된 동물의 수. (7~16시. 새벽시간에, 늦은밤 데이터X)
7  SELECT
8      HOUR(DATETIME) as HOUR,
9      COUNT(*) as ANIMAL_COUNT
10 FROM ANIMAL_OUTS
11 GROUP BY
12     1
13 )
14
15 SELECT
16     dht.HOUR as dummy_hour,
17     IFNULL(ANIMAL_COUNT,0) AS ANIMAL_COUNT
18 FROM dummy_hour_t dht -- 1. 0~23시 더미데이터.(큰)
19     LEFT JOIN animal_out_t ao -- 2. 7~19시. 시간대별 입양된 동물의 수.(새벽시간에, 늦은밤 데이터X)
20         ON dht.HOUR = ao.HOUR -- 시간으로 연결.
21 GROUP BY 1,2
22 ORDER BY 1,2
```

원하는 OUTPUT

HOUR	COUNT
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	3
8	1
9	1
10	2
11	13
12	10
13	14
14	9
15	7
16	10
17	12
18	16
19	2
20	0
21	0
22	0
23	0

우리가 돌려본 OUTPUT

HOUR	COUNT
7	3
8	1
9	1
10	2
11	13
12	10
13	14
14	9
15	7
16	10
17	12
18	16
19	2

데이터가 없는 시간의 날짜값은 어떻게 구해야지?

더미데이터 생성

더미Dummy데이터의 목적.

더미는 일종의 “임시” 데이터. 특히 날짜나 시간과 같은 곳에서 주로 많이 사용함.

현업에서는 특정 일자에 데이터가 없는 경우, 데이터 표기가 되지 않을 수 있음.

그럼에도 “0”이라는 값으로 데이터를 표기해야하는 니즈가 종종 발생함.

그럴 때 해당 방법으로 데이터를 임의로 생성해서 표기할 수 있음.

DB별 생성하는 방법이 다르지만 최대한 가장 보편적으로 쓰는 방법으로 안내.

빅쿼리에서 더미데이터 생성

빅쿼리에서 숫자/날짜 더미데이터를 생성할 때.

```
SELECT num FROM UNNEST(GENERATE_ARRAY(1, 10)) AS num;  
SELECT *FROM UNNEST(GENERATE_DATE_ARRAY('2016-10-05', '2016-10-08'))
```

UNNEST는 이번 과정에서 설명은 별도로 하지 않음.
궁금하신 분들은 참고.
<https://medium.com/firebase-developers/using-the-unnest-function-in-bigquery-to-analy>

실행

저장

공유

일정

더보기

1 SELECT num FROM UNNEST(GENERATE_ARRAY(1, 10)) AS num;

쿼리 결과

작업 정보

결과

JSON

실행 세부정보

행	num
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

(참고) 올바르게 검색하는 방법

{찾고 싶은 내용} in {어디에서}

대체로 답을 찾을 수 있다.

너무 오래된 포스팅은 변경될 수 있으니 참고.

“공식문서”를 가장 먼저 신뢰하고 찾아보기.

Google

generate date in mysql

3 임영웅

전체 동영상 뉴스 이미지 쇼핑 더보기

도구

17,600,000개의 구글 검색결과 중 도움이 되는 결과만 모아보세요.

검색결과 최적화

https://stackoverflow.com › questions › generating-a-se...

Generating a series of dates - mysql - Stack Overflow

2015. 1. 18. — Set @i:=0; SELECT DATE(DATE_ADD(X, INTERVAL @i:=@i+1 DAY)) AS datesSeries FROM yourtable, (SELECT @i:=0) r where @i < DATEDIFF(now(), date Y) ;. Not sure if ...

답변 4개 · 인기 답변: if you're in a situation like me where creating temporary tables is prohibited, an...

generate days from date range - sql - Stack Overflow

답변 30개 2011년 11월 16일

How to get list of dates between two dates in mysql select query

답변 6개 2012년 2월 15일

Create date from day, month, year fields in MySQL

답변 6개 2011년 11월 7일

Create a date range in mysql - Stack Overflow

답변 5개 2010년 1월 27일

stackoverflow.com 검색결과 더보기

https://notestoself.dev › mysql-generate-date-range

Generating a range of dates in MySQL - NotesToSelf.Dev

Generating a range of dates in MySQL ... It's often useful to have a range of contiguous dates when building SQL queries, for example by using a left join to ...

https://dba.stackexchange.com › questions › generate-d...

Generate Dates between Date Ranges in mysql

2018. 12. 5. · 답변 1개

I tried this solution : WITH recursive Date_Ranges AS (select '2018-11-30' as Date union all select Date + interval 1 day from Date_Ranges ...

라이너 추천 검색어

MySQL generate date range

Mysql date list

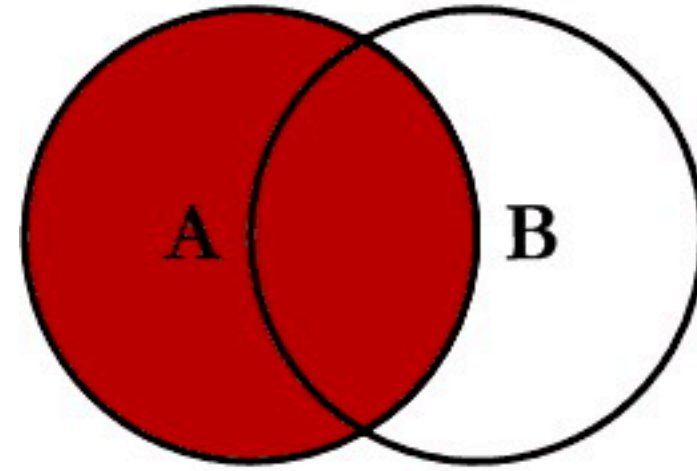
Mssql generate date range

Sql date_range

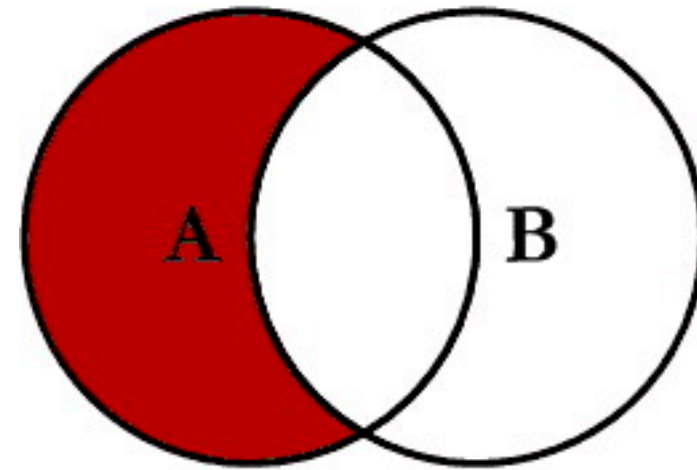
Chapter.

2. 서로 다른 데이터를 연결하는 JOIN

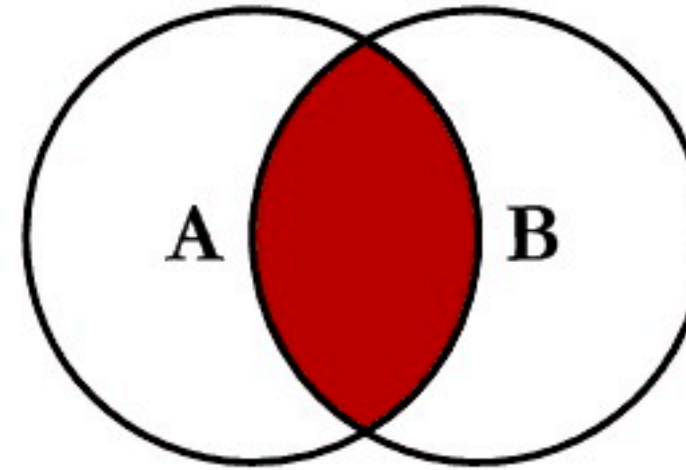
SQL JOINS



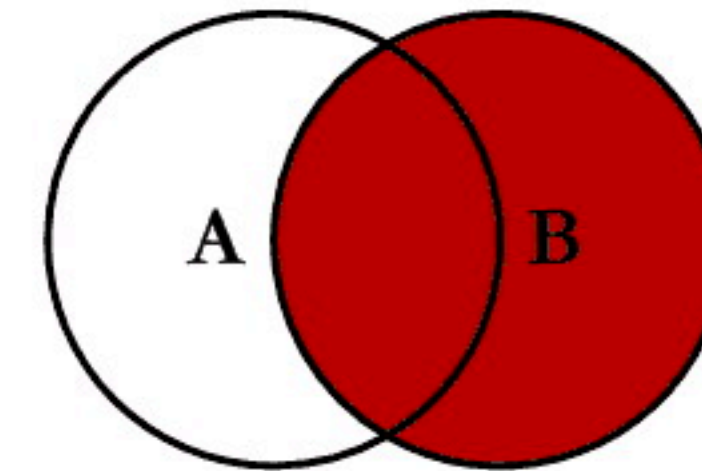
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



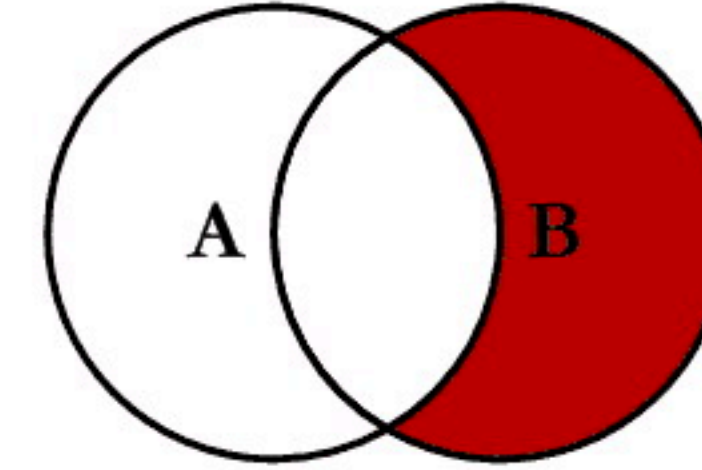
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



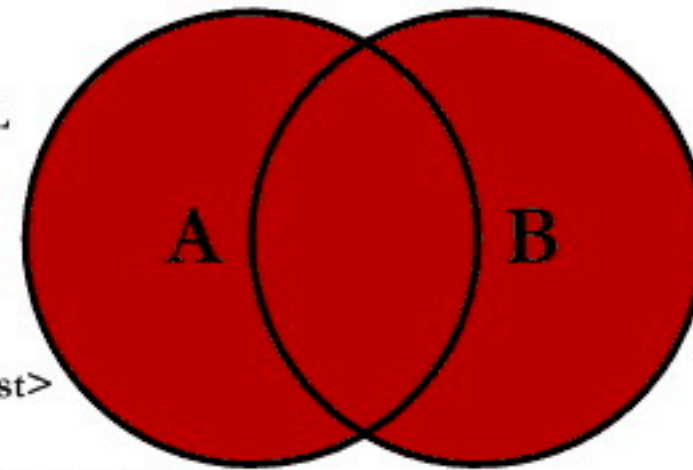
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



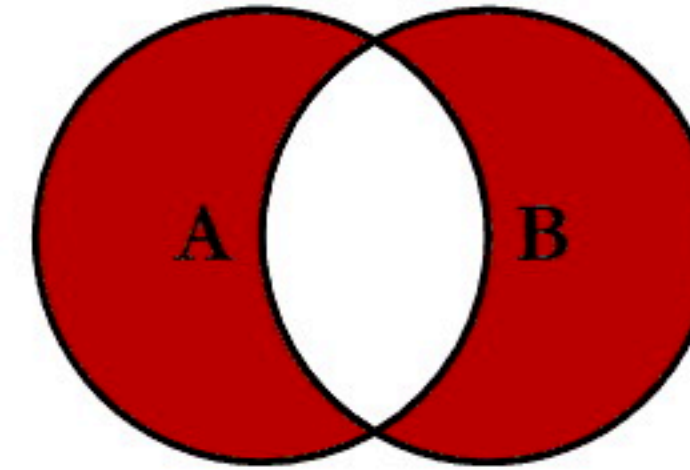
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



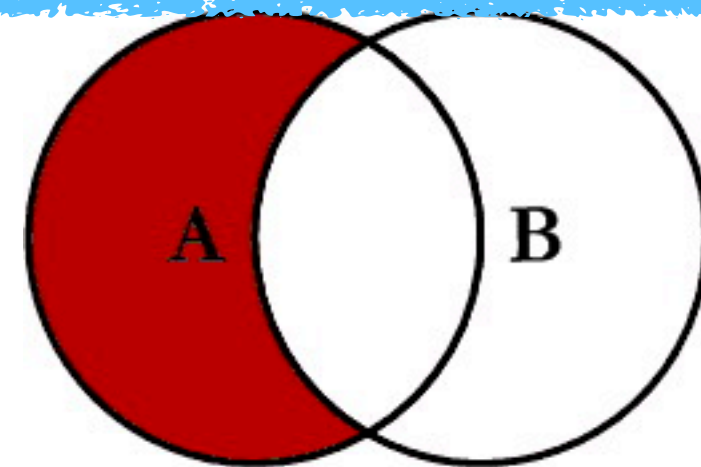
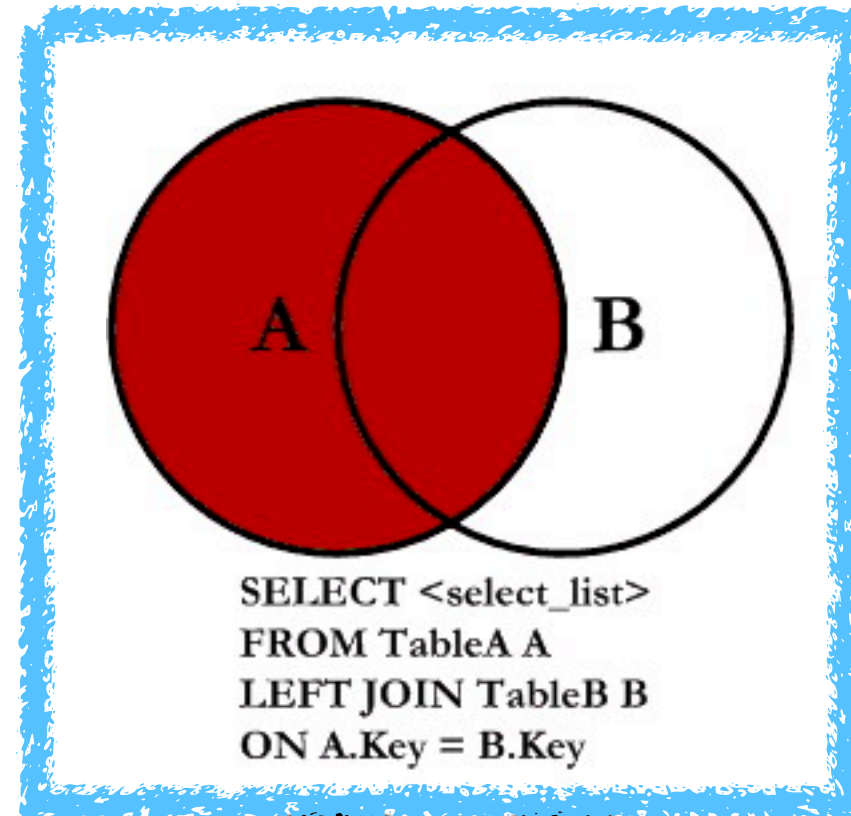
```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

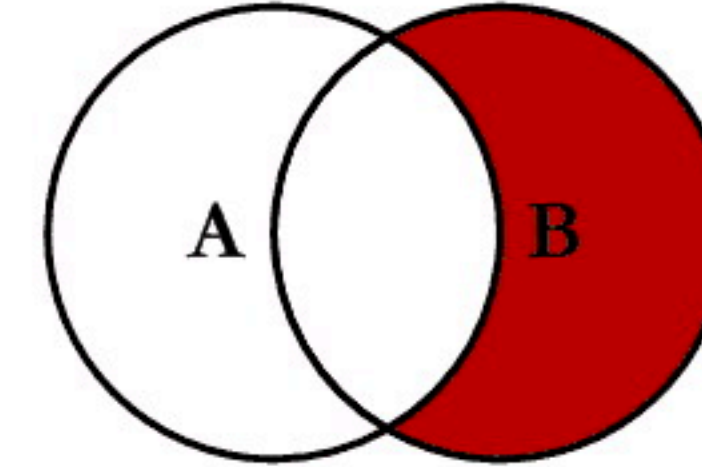
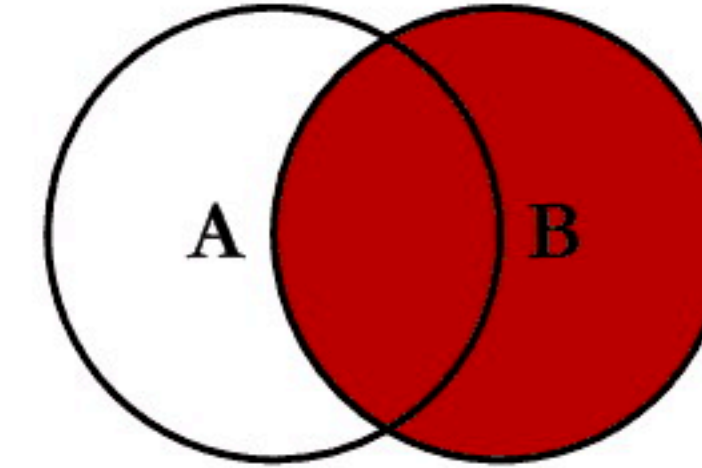
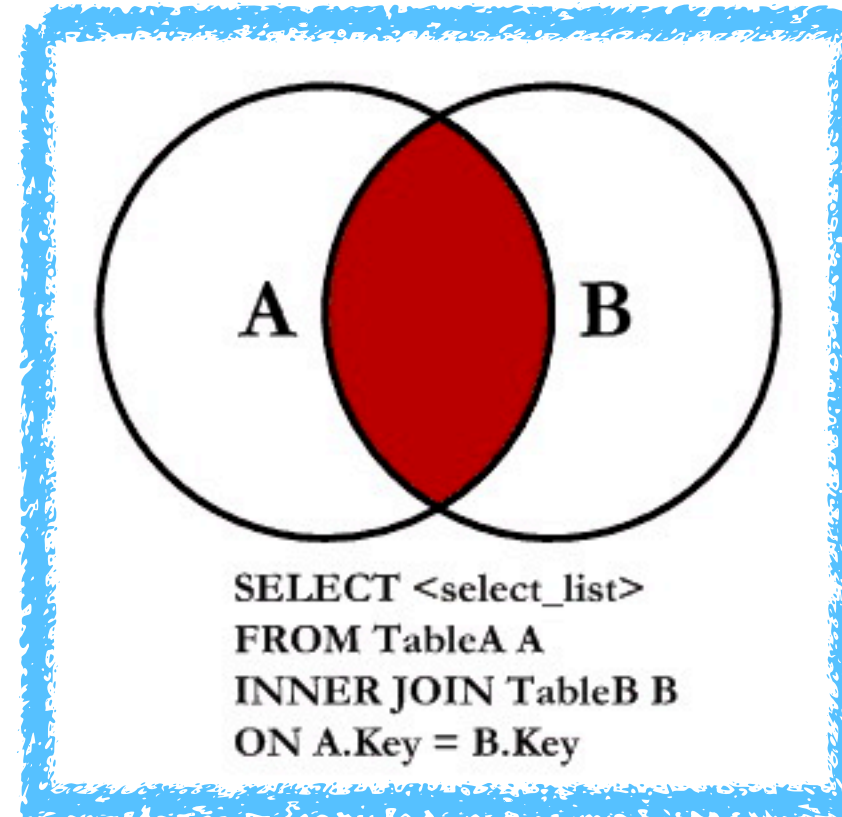
JOIN 개념

SQL JOINS

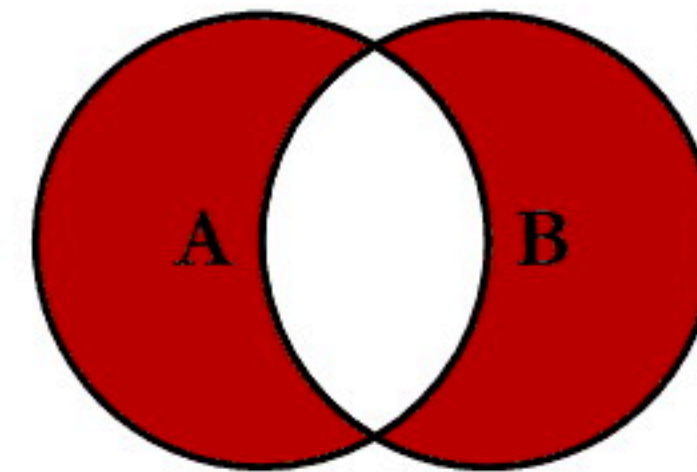
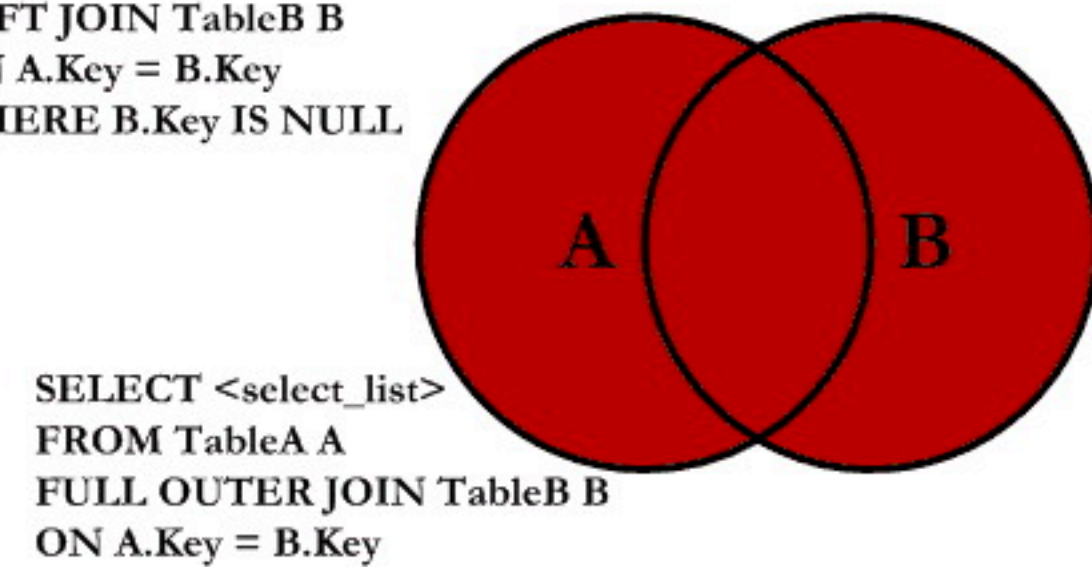
우선 이것만 이해 해봅시다.



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

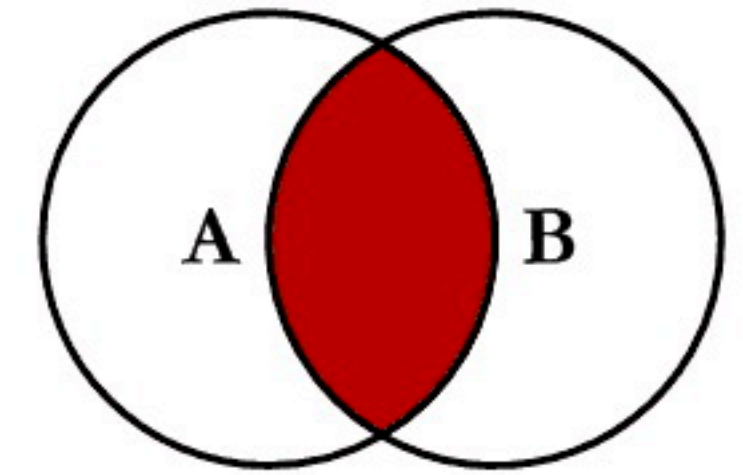
JOIN 개념

너와 나의 연결고리~



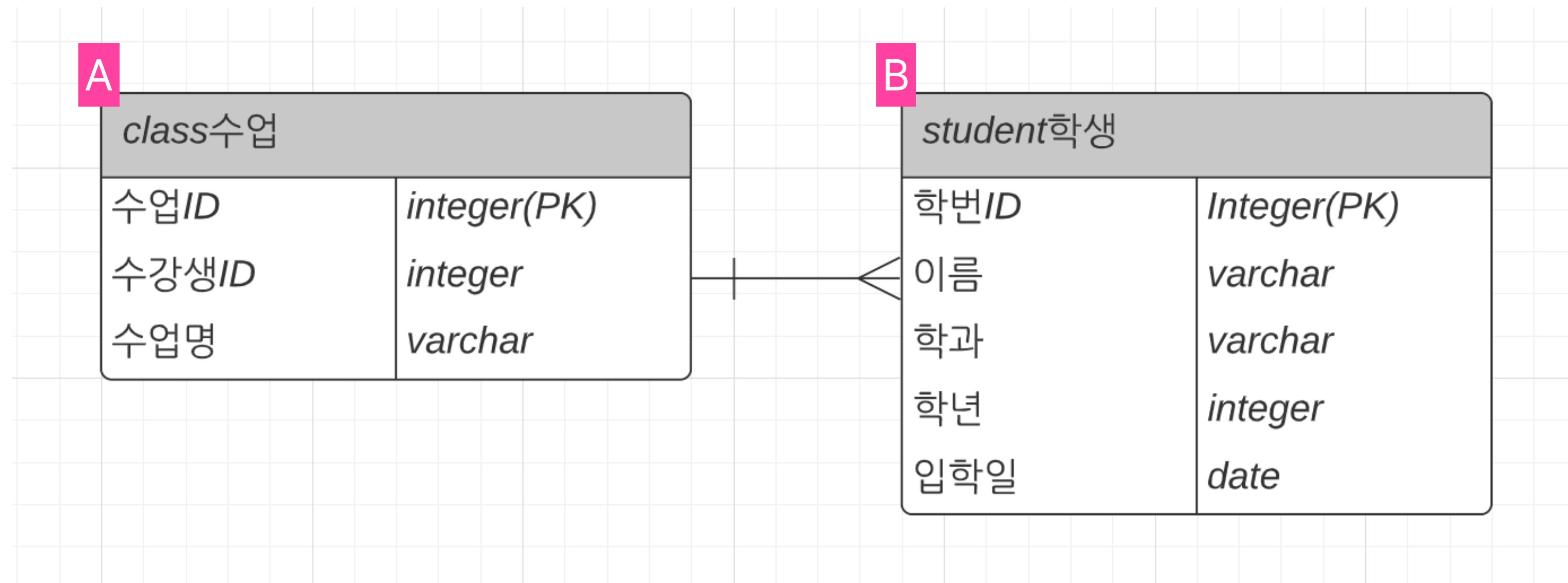
데이터는 서로 독립적으로 존재하지만
이것을 서로 하나로 묶어주는 매개체가 JOIN

JOIN 개념



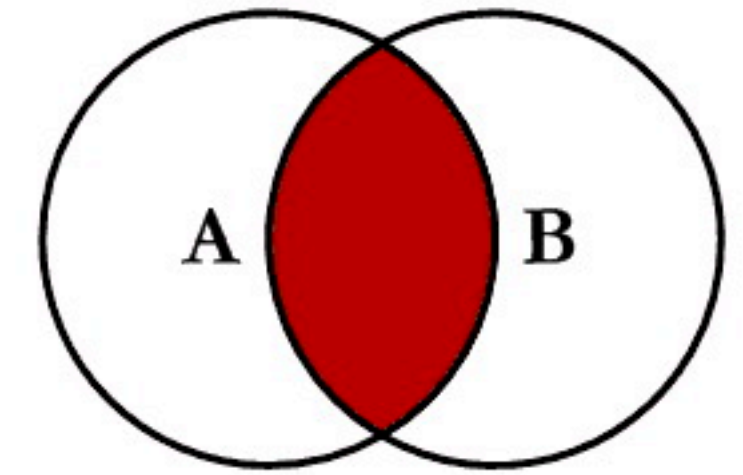
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

우리는 묶는다는 것을 하기 위해서 데이터 테이블의 ROW의 단위를 이해하고 있어야 함.



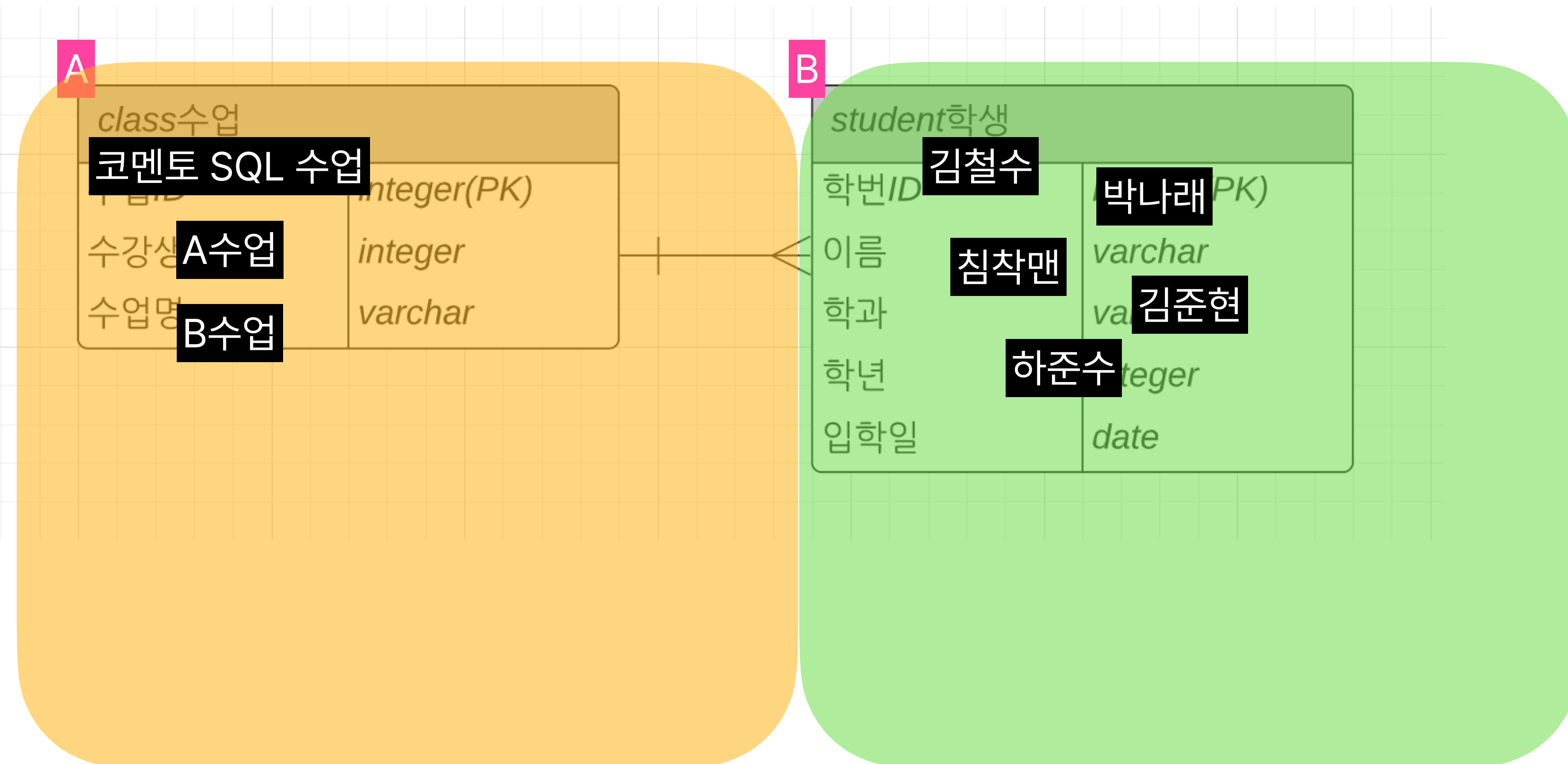
"수업" 하나에 여러명의 "학생"이 신청할 수 있다.
이때 우리는 수업에 참여한 학생들이 누구인지 알 수 있어야 한다.

JOIN 개념

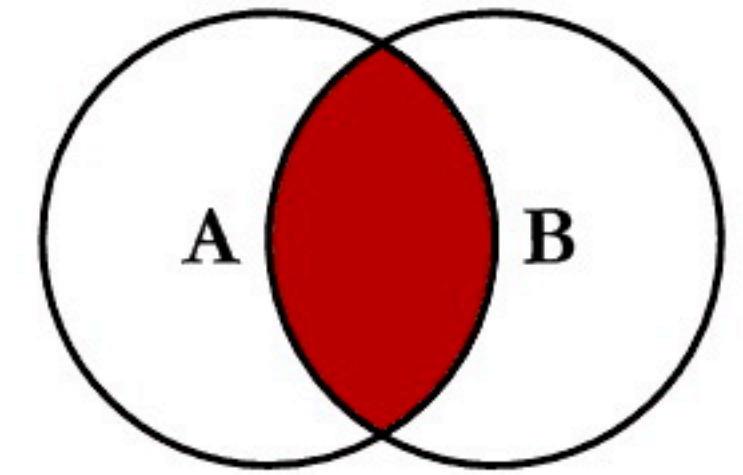


```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

수업테이블(A)에는 수업 리스트가 있고
학생테이블(B)에는 학생 리스트가 있습니다.

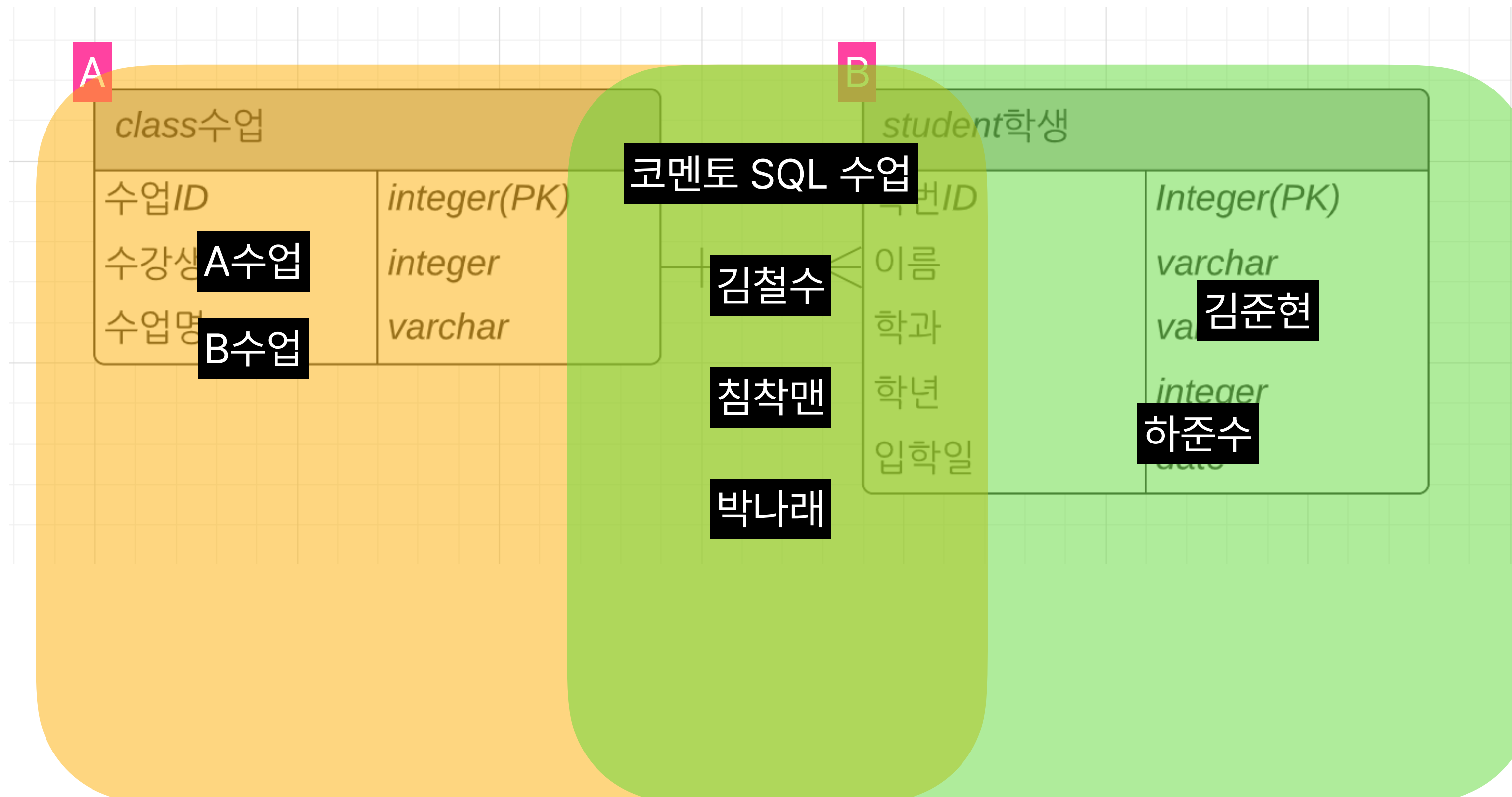


JOIN 개념

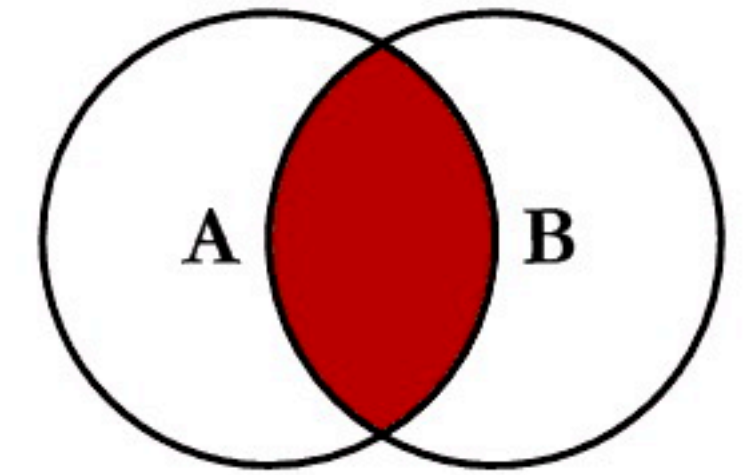


```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

이때 “코멘토 SQL수업”에서 수강신청을 받았고
학생들은 수업을 신청하였습니다.
이때 수업을 신청한 학생들의 학과,학년,이름을 알고 싶어합니다.



JOIN 개념



```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

데이터의 예시는 다음과 같습니다.
현재 우리는 수강생들의 ID만 가지고 있고
수강생들의 정보는 알지 못하는 상황입니다.

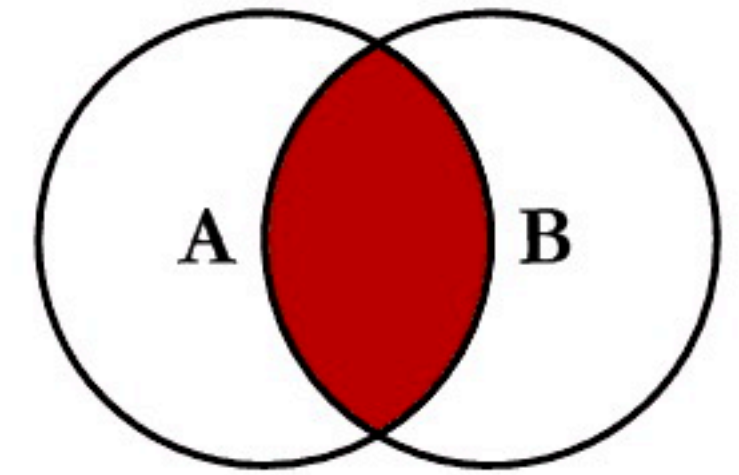
CLASS수업

수업ID	수업명	수강생ID
10001	코멘토 SQL수업	101
10001	코멘토 SQL수업	102
10001	코멘토 SQL수업	103

STUDENT학생

학번ID	이름	학과	학년	입학일
101	김철수	무역학과	1	2020.03.01
102	침착맨	통계학과	4	2017.03.01
103	박나래	통계학과	2	2019.03.01
104	김준현	바둑학과	4	2017.03.01
105	하준수	바둑학과	3	2018.03.01

JOIN 개념



수업을 신청한 3명의 리스트를 뽑기 위해
다음과 같이 SQL문을 짤 수 있습니다.

```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

```
-- # 코멘토 SQL수업을 신청한 3명의 이름을 뽑기  
-- (1) 수업에 신청한 수강생  
✓ SELECT  
    수업ID,  
    수강생ID  
FROM {수업class}  
-- (2) 학생 정보  
✓ SELECT  
    학생ID,  
    이름,  
    학과,  
    학년,  
    입학일  
FROM {학생student}
```

-- (3) 수업을 신청한 수강생들의 정보 확인

SELECT

A.수업ID,
A.수강생ID,
B.학번ID,
B.이름,
B.학과,
B.학년,
B.입학일

FROM {수업class} A -- 첫번째 테이블

JOIN {학생student} B -- 두번째 테이블

ON A.수강생ID = B.학번ID -- 두개 테이블의 "연결고리"는? (학번ID)

OUTPUT

수업ID	수업명	수강생ID	학번ID	이름	학과	학년	입학일
10001	코멘토 SQL수업	101	101	김철수	무역학과	1	2020.03.01
10001	코멘토 SQL수업	102	102	침착맨	통계학과	4	2017.03.01
10001	코멘토 SQL수업	103	103	박나래	통계학과	2	2019.03.01

INNER JOIN (교집합)

모든 JOIN의 핵심은 연결고리(LINK) 입니다.

쉽게 말해 두개 테이블에 있는 공통의 컬럼값을 찾으면 끝입니다.

앞서 수업테이블에는 수강생ID와 학생테이블의 학생ID 값이 서로의 연결고리 였습니다.

LEFT JOIN

INNER (교집합)
7~19 만 취한게된다.

—

LEFT
0~23시 전체.
해당시간에 데이터가 없으면
0 으로 표기한다.

기준

HOUR
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

우리가 돌려본 OUTPUT

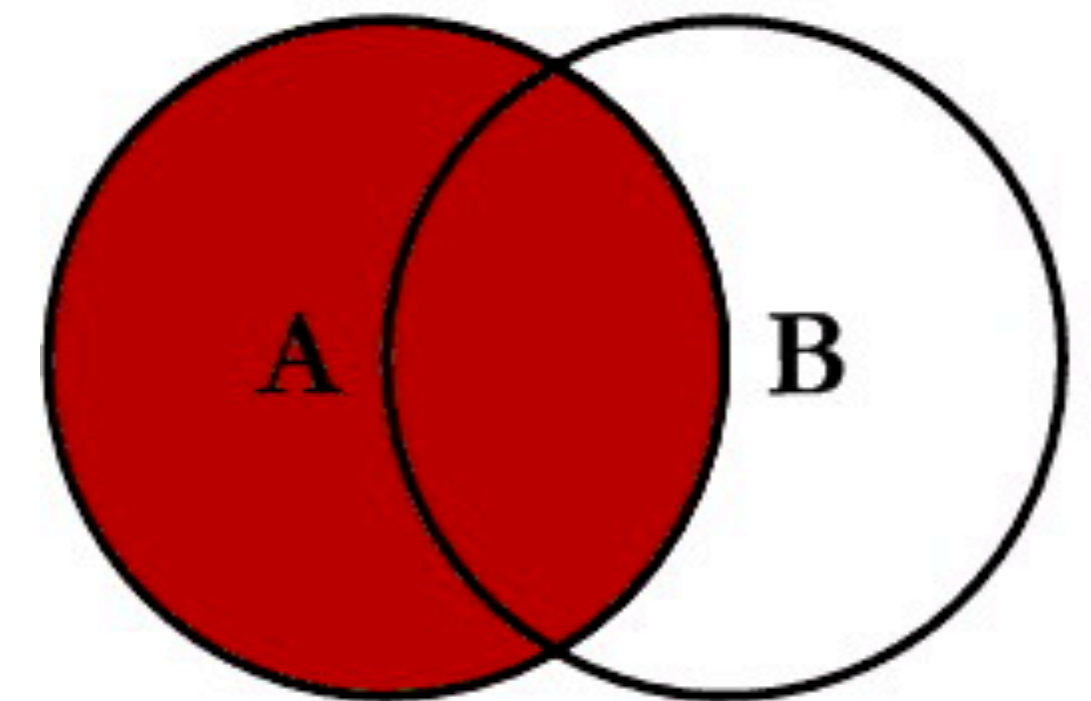
LEFT	HOUR	COUNT
	7	3
	8	1
	9	1
	10	2
	11	13
	12	10
	13	14
	14	9
	15	7
	16	10
	17	12
	18	16
	19	2

다시 한번 가져와봤습니다.

LEFT JOIN

- 두 개의 데이터 덩어리가 있을 때, 어느 한쪽의 덩어리가 좀 더 큰 경우 **큰 덩어리**를 기준으로 데이터를 구합니다.
 - 어떻게 큰 덩어리인지 어떻게 알 수 있냐구요? 그건 상황에 따라 판단할 수 있습니다. :-)
- 문제를 풀어가면서 설명해드릴게요.

```
-- 1-2 LEFT OUTER JOIN
SELECT <열 목록>
FROM <첫번째 테이블> A -- **기준이 되는 큰 덩어리 테이블**
LEFT OUTER JOIN <두번째 테이블> B
ON A.<연결고리> = B.<연결고리>
```



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

문제를 풀어보면서 이해해볼까요?

서브쿼리 SUB-QUERY

15 X 12

어떻게 계산하시나요?

$$\begin{aligned} &15 * (10+2) \\ &15*10 + 15*2 \\ &150 + 30 \\ &=180 \end{aligned}$$

인지하지 못할 뿐 이렇게 계산합니다.

서브쿼리 SUB-QUERY

“쿼리 안에 쿼리”를 의미합니다. 즉, 이중쿼리문.

우리가 쿼리를 쓸 때 완전한 OUTPUT이 나올때 까지는 모든 과정이 임시적인 테이블로 저장됩니다.

15 x 12 를 한번 계산을 해볼까요? 결과는 180이 나옵니다. 어떻게 계산하셨나요?

우리가 곱셈을 할때 잘 인지하지는 않지만 십의자리, 일의자리를 각각 계산하고 더할 것입니다.

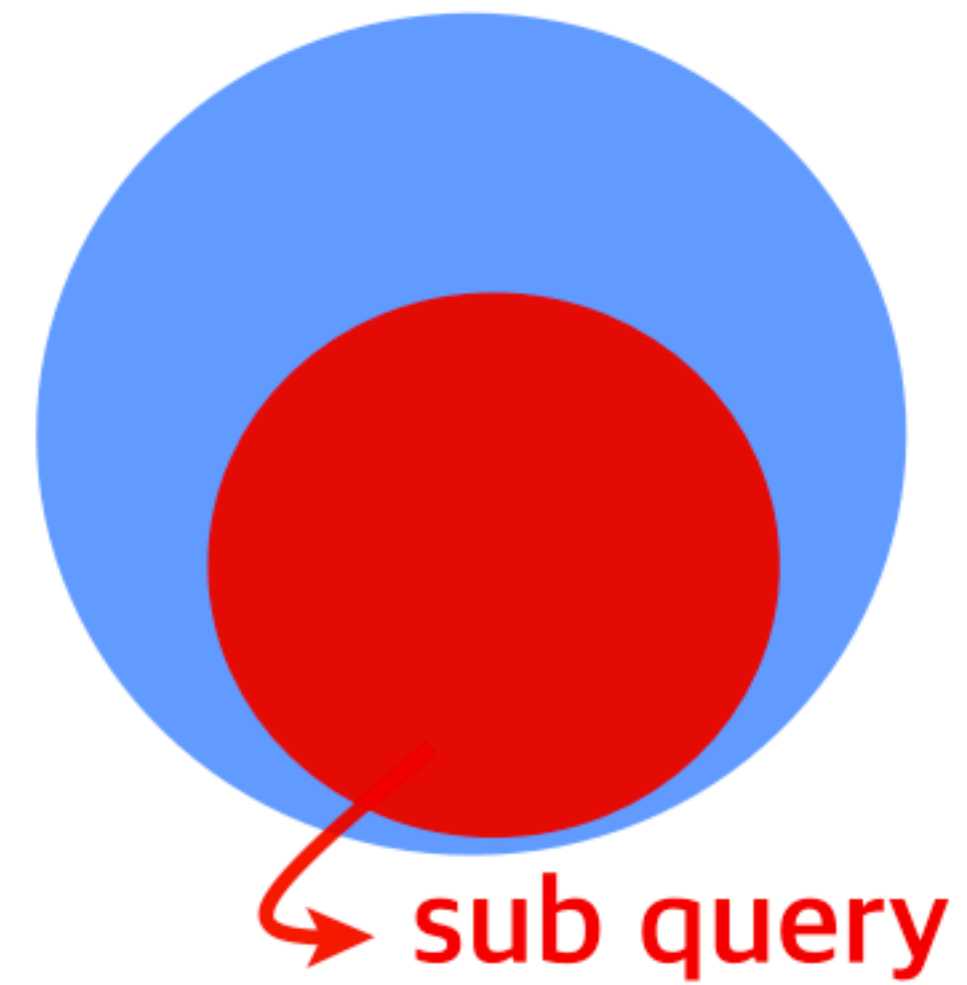
눈에 보이지 않지만 분명 우리는 각각 계산한 값을 어딘가에 저장을 해두었습니다.

서브쿼리도 마찬가지입니다.

결과로는 나오지는 않지만 중간에 저장되는 테이블로 사용되는 구조로 이해하면 됩니다.

Q. 학생 이름이 "침착맨"의 학과명을 알고 싶다.

```
SELECT 학과명  
FROM 학과 테이블  
WHERE 학과번호 = (  
  SELECT 학과번호  
  FROM 학생 테이블  
  WHERE 학생 이름 = '침착맨'  
);
```



#1. 서브쿼리

```
1  SELECT 학과명
2  FROM 학과테이블
3  ✓ WHERE 학과번호 = (
4      SELECT 학과번호
5      FROM 학생테이블
6      WHERE 학생이름 = '침착맨'
7  )
```

#2. 조인을 사용

```
9  SELECT 학과명
10 ✓ FROM 학과테이블 t1
11 ✓ JOIN 학생테이블 t2
12   ON t1.학과번호 = t2.학과번호
13 WHERE t2.학생이름 = '침착맨'
```

WITH절 (서브쿼리 대체제)

서브쿼리랑 동일하지만 주로 모듈module처럼 목적에 맞게 분리해서 사용가능하고, 재사용이 가능하다는 이유로 사용합니다.

-- 3. WITH절 사용

```
WITH B AS (  
  SELECT  
    학생ID,  
    이름  
  FROM {학생student}  
  WHERE 학과 = '통계학과'  
)
```

```
SELECT  
  수업ID,  
  수강생ID  
FROM {수업class} A  
JOIN B  
ON A.수강생ID = B.학번ID -- 두개 테이블의 "연결고리"는? (학번ID)
```



Q. 선생님 쿼리문이 너무 길어서 읽기가 어려운데, 쉽게 읽는 법이 있을까여?

정보의 “덩어리”로 읽으면 됩니다. 덩어리를 묶는 기준은 테이블이 머릿속에 그려지는지 입니다. 머릿속에 덩어리만 기억하시면 로직에 맞추어서 쿼리를 짜면 쉽습니다.

- 주황색 덩어리 = 학생정보(통계학과만)
- 녹색 덩어리 = 수업정보

-- 3. WITH절 사용


```
WITH B AS (  
  SELECT  
    학생ID,  
    이름  
  FROM {학생student}  
  WHERE 학과 = '통계학과'  
)
```

```
SELECT  
  수업ID,  
  수강생ID  
FROM {수업class} A  
JOIN B  
ON A.수강생ID = B.학번ID -- 두개 테이블의 "연결고리"는? (학번ID)
```


Chapter.^{별도}

VSCODE 설치

<https://code.visualstudio.com/>

 Visual Studio Code

[Docs](#) [Updates](#) [Blog](#) [API](#) [Extensions](#) [FAQ](#) [Learn](#)

 [Download](#)

Version 1.65 is now available! Read about the new features and fixes from February.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

[Download Mac Universal](#)
Stable Build

[Other platforms](#) and [Insiders Edition](#)

By using VS Code, you agree to its [license and privacy statement](#).



The screenshot displays the Visual Studio Code interface. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar is open, showing a list of installed and available extensions such as Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, and Vetur. The main editor area shows a JavaScript file named 'blog-post.js' with code for a Gatsby blog post. The code includes imports for GraphQL, React, and Gatsby Image, and a GraphQL query to fetch blog posts. A hover tooltip is visible over the 'data' variable, showing its type as 'any'. At the bottom, a terminal window shows the output of a command, indicating that the code was compiled successfully in 70ms.

반드시 알아야 할 VSCode 단축키

- Cmd/Ctrl + N : 새로운 편집기(파일) 생성하기
- Cmd/Ctrl + B : 사이드바 숨기기
- Cmd/Ctrl + P : 파일 검색
- (중요) **Cmd/Ctrl + Shift + F** : 코드 검색
- (중요)***Cmd/Ctrl + Shift + P** : 명령팔레트 실행
- (중요)***Cmd/Ctrl+D** : 일괄적으로 바꾸야 할 경우 사용 ***
- Opt/Alt+Click : 클릭하는 곳마다 커서를 만들어줌
- Opt/Alt + 방향키 위/아래 : 커서에 있는 줄을 옮겨줌
- Cmd/Ctrl + / : 주석 Comment 만들어주기 **
- Opt/Alt + Shift + i : 선택한 영역의 줄 모두에 커서를 생성
- Opt/Alt + Shift + Drag : 마우스 드래그 영역에 커서를 생성
- formatter로 정리시키기 : **Shift + Opt/Alt + F** **

Chapter.

**이번 6주 과정으로 알아갈 것.
(매 시간마다 Wrap-up)**

3주차 강의를 모두 들으시느라 고생하셨습니다!

궁금한 내용은 언제든지 문의주세요. :-)

