SQL 활용 프로그래밍

SQL Application Programming

CHAPTER 05 SQL 고급

데이터 형식과 변수의 사용, 대용량 데이터의 저장 방식, PostgreSQL 프로그래밍을 위한 내용을 살펴본다.

Contents

Contents

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL에서 지원하는 데이터 형식의 종류

- Data Type으로 표현
  - 데이터 형식, 데이터형, 자료형, 데이터 타입 등 다양하게 불림
- 데이터 형식에 대한 이해가 필요한 이유
  - SELECT문 더욱 잘 활용
  - 테이블의 생성 효율적으로 하기 위해 필요
- PostgreSQL에서 데이터 형식의 종류는 30개 정도
  - **중요하고 자주 쓰는 형식에 대해 중점 학습**

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL에서 지원하는 데이터 형식의 종류

- 숫자 데이터 형식

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

PostgreSQL: Documentation: 14: 8.1. Numeric Types

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL에서 지원하는 데이터 형식의 종류

- 문자 데이터 형식 PostgreSQL: Documentation: 14: 8.3. Character Types

Name	Description
character varying(*n*), varchar(*n*)	variable-length with limit
character(*n*), char(*n*)	fixed-length, blank padded
text	variable unlimited length

- 날짜와 시간데이터 형식 PostgreSQL: Documentation: 14: 8.5. Date/Time Types

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [(*p*)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
timestamp [(*p*)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [(*p*)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
time [(*p*)] with time zone	12 bytes	time of day (no date), with time zone	00:00:00+1559	24:00:00-1559	1 microsecond
interval [*fields*] [(*p*)]	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL에서 지원하는 데이터 형식의 종류

- Array 데이터 형식 PostgreSQL: Documentation: 14: 8.15. Arrays
  - 배열의 column은 문자열 배열 혹은 정수 배열 등의 다양한 형태의 배열로 저장될 수 있음
  - 몇 달, 일 년 또는 일주일 등을 저장하는 데이터를 관리할 때 편할 수 있음

```
CREATE TABLE sal_emp (
    name            text,
    pay_by_quarter  integer[],
    schedule        text[][]
);
```

	name text	pay_by_quarter integer[]	schedule text[]
1	Bill	{10000,10000,10000,10000}	{{meeting,lunch},{training,presentatio...
2	Carol	{20000,25000,25000,25000}	{{breakfast,consulting},{meeting,lunc...

- JSON 데이터 형식 PostgreSQL: Documentation: 14: 8.14. JSON Types
  - JSON, JSONB(Binary JSON) 2가지 유형을 지원함
  - JSON 데이터 형식은 쿼리에서 호출할 때마다 구문 분석되는 일반 JSON 데이터를 저장하는데 사용된다.
  - JSONB 데이터 형식은 JSON 데이터를 이진 형식으로 저장하는 데 사용된다

```
SELECT '{"bar": "baz", "balance": 7.77, "active":false}'::json;
```

	json json
1	{"bar": "baz", "balance": 7.77, "active":false}

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL에서 지원하는 데이터 형식의 종류

- Geometric 데이터 형식
  - 2차원의 공간객체를 나타냄

Name	Storage Size	Description	Representation
point	16 bytes	Point on a plane	(x,y)
line	32 bytes	Infinite line	{A,B,C}
lseg	32 bytes	Finite line segment	((x1,y1),(x2,y2))
box	32 bytes	Rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	Open path	[(x1,y1),...]
polygon	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

PostgreSQL: Documentation: 14: 8.8. Geometric Types

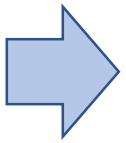
SECTION 01 PostgreSQL의 데이터 형식

변수의 사용

1. with 문법을 사용하여 임시 테이블 생성

```
with price as(select 45 as product_price)

select product_price from price
```

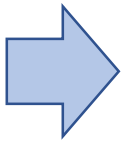


	product_price integer
1	45

2. session 변수 생성

```
set session vars.prices = 45;

select current_setting('vars.prices')
```



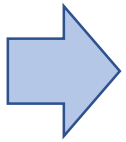
	current_setting text
1	45

SECTION 01 PostgreSQL의 데이터 형식

데이터 형식과 형 변환

1. CAST 함수를 이용한 형 변환

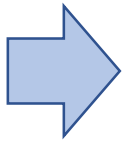
```
set schema 'shop';
select cast(avg(amount) as integer) as "평균 구매 개수" from buytbl;
```



	평균 구매 개수 integer
1	3

2. '::'를 이용한 형 변환

```
select avg(amount)::integer as "평균 구매 개수" from buytbl;
```

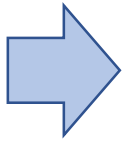


	평균 구매 개수 integer
1	3

3. 함수형 변환

   이 방식은 항상 작용하는 것이 아니라 유효한 유형에만 작동함

```
select DOUBLE(avg(amount)) as "평균 구매 개수" from buytbl; -- ERROR
select FLOAT8(avg(amount)) as "평균 구매 개수" from buytbl; -- 성공적으로 작동함
```



	평균 구매 개수 double precision
1	2.9166666666666665

SECTION 01 PostgreSQL의 데이터 형식

데이터 형식과 형 변환

- 암시적인 형 변환
  - 명시적 형 변환을 사용하지 않고 데이터 형식이 변환되는 것

```
select '300' > '100';
-- true, '300'과 '100'이 integer형으로 변환되어 비교를 진행

select 100 > '60';
-- true, '60'이 integer형으로 변환되어 비교를 진행

select 100 +'60';
-- 160, '60'이 integer형으로 변환되어 덧셈연산을 진행

select '1.1' > '2.2';
-- false, '1.1'과 '2.2'가 numeric type으로 변환되어 비교 연산 수행

select concat('100','200');
-- 100200, text

select concat(100,'200');
-- 100200, 100이 문자열로 변환되어 '200'과 결합
```

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 내장 함수
  - 흐름 함수, 문자열 함수, 수학 함수, 날짜/시간 함수, 전체 텍스트 검색 함수, 형 변환 함수, XML 함수, 비트 함수, 보안/압축 함 수, 정보 함수, 공간 분석 함수, 기타 함수 등
- 제어 흐름 함수
  - 프로그램의 흐름 제어
  - COALESCE(인자1, 인자2)
    - 인자1이 NULL이 아니면 인자1이 반환되고 인자1이 NULL이면 인자2가 반환
  - NULLIF(수식1, 수식2)
    - 수식1과 수식2가 같으면 NULL을 반환, 다르면 수식1을 반환

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 문자열 함수

Function	Return Type	Description	Example	Result
string || string	text	String concatenation	'Post' || 'greSQL'	PostgreSQL
string || non-string or non-string || string	text	String concatenation with one non-string input	'Value: ' || 42	Value: 42
bit_length(string)	int	Number of bits in string	bit_length('jose')	32
char_length(string) or character_length(string)	int	Number of characters in string	char_length('jose')	4
lower(string)	text	Convert string to lower case	lower('TOM')	tom
octet_length(string)	int	Number of bytes in string	octet_length('jose')	4
overlay(string placing string from int [for int])	text	Replace substring	overlay('Txxxxas' placing 'hom' from 2 for 4)	Thomas
position(substring in string)	int	Location of specified substring	position('om' in 'Thomas')	3
substring(string [from int] [for int])	text	Extract substring	substring('Thomas' from 2 for 3)	hom
substring(string from *pattern*)	text	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.	substring('Thomas' from '...$')	mas
substring(string from *pattern* for *escape*)	text	Extract substring matching SQL regular expression. See Section 9.7 for more information on pattern matching.	substring('Thomas' from '%#"o_a#"_' for '#')	oma
trim([leading | trailing | both] [characters] from string)	text	Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string	trim(both 'x' from 'xTomxx')	Tom
upper(string)	text	Convert string to upper case	upper('tom')	TOM

PostgreSQL: Documentation: 9.1: String Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 문자열 함수 예시

```
select 'Post' || 'greSQL';
-- 두 문자열을 합침

select 'Value' || 42;
-- 하나의 non-string input과 문자열을 합침

select lower('TOM');
-- 대문자를 소문자로 변경

select substring('Thomas' from 2 for 3);
-- from 'start_index' for 'number', 시작 인덱스 포함부터 number만큼의 문자열을 추출(1부터 시작)

select trim(both 'x' from 'xTomxx');
-- 문자열의 시작/끝/양쪽 끝에서 문자(기본적으로 공백)만 포함하는 가장 긴 문자열을 제거합니다.
```

PostgreSQL: Documentation: 9.1: String Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 수학 함수

Function	Return Type	Description	Example	Result
abs(*x*)	(same as input)	absolute value	abs(-17.4)	17.4
cbrt(dp)	dp	cube root	cbrt(27.0)	3
ceil(dp or numeric)	(same as input)	nearest integer greater than or equal to argument	ceil(-42.8)	-42
ceiling(dp or numeric)	(same as input)	nearest integer greater than or equal to argument (same as ceil)	ceiling(-95.3)	-95
degrees(dp)	dp	radians to degrees	degrees(0.5)	28.6478897565412
div(y numeric, x numeric)	numeric	integer quotient of y/x	div(9,4)	2
exp(dp or numeric)	(same as input)	exponential	exp(1.0)	2.71828182845905
floor(dp or numeric)	(same as input)	nearest integer less than or equal to argument	floor(-42.8)	-43
ln(dp or numeric)	(same as input)	natural logarithm	ln(2.0)	0.693147180559945
log(dp or numeric)	(same as input)	base 10 logarithm	log(100.0)	2
log(b numeric, x numeric)	numeric	logarithm to base b	log(2.0, 64.0)	6.0000000000
mod(y, x)	(same as argument types)	remainder of y/x	mod(9,4)	1
pi()	dp	"π" constant	pi()	3.14159265358979
power(a dp, b dp)	dp	a raised to the power of b	power(9.0, 3.0)	729
power(a numeric, b numeric)	numeric	a raised to the power of b	power(9.0, 3.0)	729
radians(dp)	dp	degrees to radians	radians(45.0)	0.785398163397448
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(v numeric, s int)	numeric	round to s decimal places	round(42.4382, 2)	42.44
sign(dp or numeric)	(same as input)	sign of the argument (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(v numeric, s int)	numeric	truncate to s decimal places	trunc(42.4382, 2)	42.43
width_bucket(op numeric, b1 numeric, b2 numeric, count int)	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	width_bucket(5.35, 0.024, 10.06, 5)	3
width_bucket(op dp, b1 dp, b2 dp, count int)	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	width_bucket(5.35, 0.024, 10.06, 5)	3

PostgreSQL: Documentation: 9.5: Mathematical Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 수학 함수 예시

```
select ceil(40.7);
-- 주어진 argument보다 더 큰 가장 가까운 정수 혹은 같은 정수를 반환

select div(7,2);
-- div(y,x), y를 x로 나눈 몫을 반환

select mod(10,3);
-- mod(10,3), 10을 3으로 나눈 나머지를 반환

select power(2,4);
-- power(a,b), a의 b승

select round(5.4);
-- 가까운 정수로 반올림 진행

select sqrt(16);
-- 주어진 수의 제곱근을 구함
```

PostgreSQL: Documentation: 9.5: Mathematical Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 날짜 및 시간 함수

Function	Return Type	Description	Example	Result
age(timestamp, timestamp)	interval	Subtract arguments, producing a "symbolic" result that uses years and months	age(timestamp '2001-04-10', timestamp '1957-06-13')	43 years 9 mons 27 days
age(timestamp)	interval	Subtract from current_date (at midnight)	age(timestamp '1957-06-13')	43 years 8 mons 3 days
clock_timestamp()	timestamp with time zone	Current date and time (changes during statement execution); see Section 9.9.4		
current_date	date	Current date; see Section 9.9.4		
current_time	time with time zone	Current time of day; see Section 9.9.4		
current_timestamp	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
date_part(text, timestamp)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('hour', timestamp '2001-02-16 20:38:40')	20
date_part(text, interval)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('month', interval '2 years 3 months')	3
date_trunc(text, timestamp)	timestamp	Truncate to specified precision; see also Section 9.9.2	date_trunc('hour', timestamp '2001-02-16 20:38:40')	2001-02-16 20:00:00
extract(field from timestamp)	double precision	Get subfield; see Section 9.9.1	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(field from interval)	double precision	Get subfield; see Section 9.9.1	extract(month from interval '2 years 3 months')	3
isfinite(date)	boolean	Test for finite date (not +/-infinity)	isfinite(date '2001-02-16')	true
isfinite(timestamp)	boolean	Test for finite time stamp (not +/-infinity)	isfinite(timestamp '2001-02-16 21:28:30')	true
isfinite(interval)	boolean	Test for finite interval	isfinite(interval '4 hours')	true
justify_days(interval)	interval	Adjust interval so 30-day time periods are represented as months	justify_days(interval '35 days')	1 mon 5 days
justify_hours(interval)	interval	Adjust interval so 24-hour time periods are represented as days	justify_hours(interval '27 hours')	1 day 03:00:00
justify_interval(interval)	interval	Adjust interval using justify_days and justify_hours, with additional sign adjustments	justify_interval(interval '1 mon -1 hour')	29 days 23:00:00
localtime	time	Current time of day; see Section 9.9.4		
localtimestamp	timestamp	Current date and time (start of current transaction); see Section 9.9.4		
now()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
statement_timestamp()	timestamp with time zone	Current date and time (start of current statement); see Section 9.9.4		
timeofday()	text	Current date and time (like clock_timestamp, but as a text string); see Section 9.9.4		
transaction_timestamp()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		

PostgreSQL: Documentation: 14: 9.9. Date/Time Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

PostgreSQL 내장 함수

- 날짜 및 시간 함수 예시

```
select date_part('year',timestamp '2001-02-16 20:38:40');
-- date_part(text, timestamp), 주어진 timestamp에서 text(year,month,day,hour 등)에 해당하는 값을 반환한다.

select extract(hour from timestamp '2001-02-16 20:38:40');
-- extract(field from timestamp), field(year,month,day,hour 등)에 해당하는 값을 반환한다.

select now();
-- 현재 날짜 및 시간을 반환

select current_date;
-- 현재 날짜를 반환

select current_time;
-- 현재 시간을 반환

select localtime;
-- 현재 시간을 반환
```

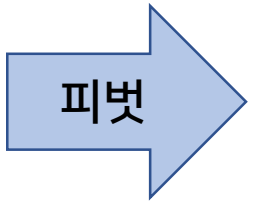
PostgreSQL: Documentation: 14: 9.9. Date/Time Functions and Operators

SECTION 01 PostgreSQL의 데이터 형식

피벗의 구현

- 피벗(Pivot) 이란?
  - 한 열에 포함된 여러 값 출력, 이를 여러 열로 변환하여 테이블 반환식 회전, 필요하면 집계까지 수행

	uname character (3)	season character (2)	amount integer
1	김범수	겨울	10
2	윤종신	여름	15
3	김범수	가을	25
4	김범수	봄	3
5	김범수	봄	37
6	윤종신	겨울	40
7	김범수	여름	14
8	김범수	겨울	22
9	윤종신	여름	64



	uname character (3)	봄 bigint	여름 bigint	가을 bigint	겨울 bigint	합계 bigint
1	김범수	40	14	25	32	111
2	윤종신	0	79	0	40	119

[실습 1] 간단한 피벗 테이블 실습

1. 샘플테이블 작성

```
set schema 'public';
CREATE TABLE pivotTest
(   uName CHAR(3),
    season CHAR(2),
    amount INT );
```

2. 데이터 입력 및 테이블 출력

```
INSERT  INTO  pivotTest VALUES  ('김범수' , '겨울',  10) , ('윤종신' , '여름',  15) , ('김범수' , '가을',  25) , ('김범수' , '봄',    3)
                               ,('김범수' , '봄',    37) , ('윤종신' , '겨울',  40) , ('김범수' , '여름',  14) , ('김범수' , '겨울',  22)
                               ,('윤종신' , '여름',  64);

SELECT * FROM pivotTest;
```

	uname character (3)	season character (2)	amount integer
1	김범수	겨울	10
2	윤종신	여름	15
3	김범수	가을	25
4	김범수	봄	3
5	김범수	봄	37
6	윤종신	겨울	40
7	김범수	여름	14
8	김범수	겨울	22
9	윤종신	여름	64

[실습 1] 간단한 피벗 테이블 실습

3. SUM(), IF(), GROUP BY 활용

```
SELECT uName,    SUM(case when (season='봄') then amount else 0 end) AS "봄",
                 SUM(case when (season='여름') then amount else 0 end) AS "여름",
                 SUM(case when (season='가을') then amount else 0 end) AS "가을",
                 SUM(case when (season='겨울') then amount else 0 end) AS "겨울",
                 SUM(amount) AS "합계" FROM pivotTest GROUP BY uName;

select * from pivotTest;
```

	uname character (3)	봄 bigint	여름 bigint	가을 bigint	겨울 bigint	합계 bigint
1	김범수	40	14	25	32	111
2	윤종신	0	79	0	40	119

SECTION 02 조인

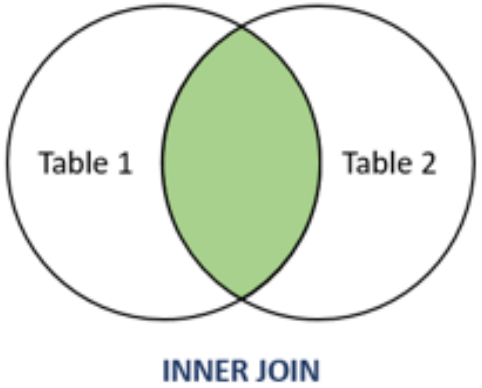
조인(Join)

- 조인
  - 두 개 이상의 테이블을 서로 묶어서 하나의 결과 집합으로 만들어 내는 것
  - 종류 : INNER JOIN, OUTER JOIN, CROSS JOIN, SELF JOIN

- 데이터베이스의 테이블
  - 중복과 공간 낭비를 피하고 데이터의 무결성을 위해서 여러 개의 테이블로 분리하여 저장
  - 분리된 테이블들은 서로 관계(Relation)를 가짐
  - 1대 다 관계 보편적

SECTION 02 조인

INNER JOIN(내부 조인)



- 조인 중에서 가장 많이 사용되는 조인
  - 대개의 업무에서 조인은 INNER JOIN 사용
  - 일반적으로 JOIN이라고 얘기하는 것이 이 INNER JOIN 지칭
  - 사용 형식

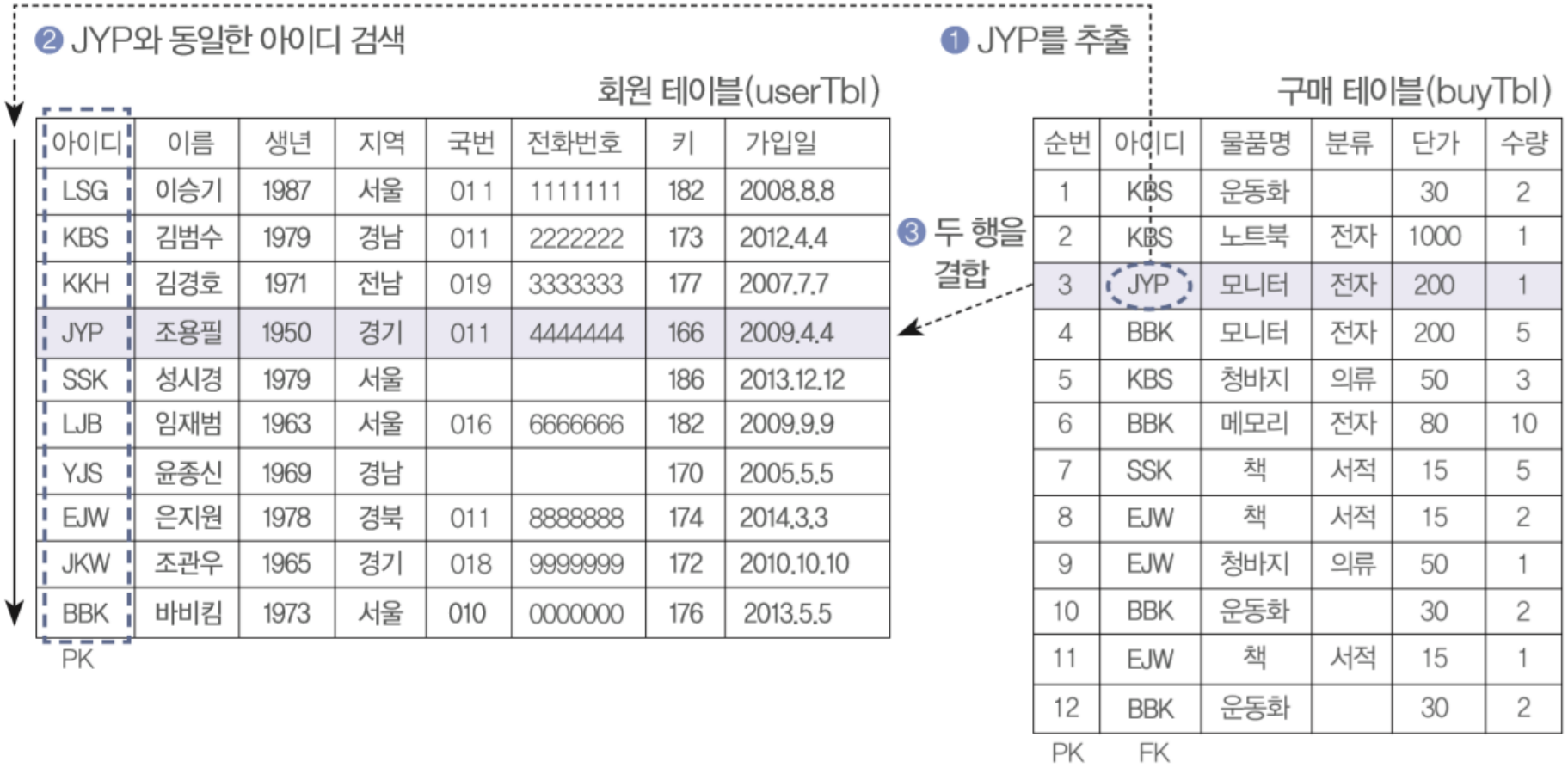
```
SELECT <열 목록>
FROM <첫 번째 테이블>
        INNER JOIN <두 번째 테이블>
        ON <조인될 조건>
[WHERE 검색조건]
```

  - JOIN만 써도 INNER JOIN으로 인식함

SECTION 02 조인

INNER JOIN(내부 조인)

- 조인 중에서 가장 많이 사용되는 조인



아이디	이름	생년	지역	국번	전화번호	키	가입일
LSG	이승기	1987	서울	011	1111111	182	2008.8.8
KBS	김범수	1979	경남	011	2222222	173	2012.4.4
KKH	김경호	1971	전남	019	3333333	177	2007.7.7
JYP	조용필	1950	경기	011	4444444	166	2009.4.4
SSK	성시경	1979	서울			186	2013.12.12
LJB	임재범	1963	서울	016	6666666	182	2009.9.9
YJS	윤종신	1969	경남			170	2005.5.5
EJW	은지원	1978	경북	011	8888888	174	2014.3.3
JKW	조관우	1965	경기	018	9999999	172	2010.10.10
BBK	바비킴	1973	서울	010	0000000	176	2013.5.5

순번	아이디	물품명	분류	단가	수량
1	KBS	운동화		30	2
2	KBS	노트북	전자	1000	1
3	JYP	모니터	전자	200	1
4	BBK	모니터	전자	200	5
5	KBS	청바지	의류	50	3
6	BBK	메모리	전자	80	10
7	SSK	책	서적	15	5
8	EJW	책	서적	15	2
9	EJW	청바지	의류	50	1
10	BBK	운동화		30	2
11	EJW	책	서적	15	1
12	BBK	운동화		30	2

[그림 7-28] INNER JOIN의 작동

SECTION 02 조인

INNER JOIN(내부 조인)

- 조인 중에서 가장 많이 사용되는 조인

```
select *
from buytbl inner join usertbl
on buytbl.userID = usertbl.userID
where buytbl.userID = 'JYP';
```

	num integer	userid character (8)	prodname character (6)	groupname character (4)	price integer	amount smallint	userid character (8)	name character varying (10)	birthyear integer	addr character (2)	mobile1 character (3)	mobile2 character (8)	height smallint	mdate date
1	3	JYP	모니터	전자	200	1	JYP	조용필	1950	경기	011	4444444	166	2009-04-04

[실습 2] INNER JOIN 예시

1. INNER JOIN

```
set schema 'shop';
select * from buytbl;
```

	num [PK] integer	userid character (8)	prodname character (6)	groupname character (4)	price integer	amount smallint
1	1	KBS	운동화	[null]	45	2
2	2	KBS	노트북	전자	1500	1
3	3	JYP	모니터	전자	300	1
4	4	BBK	모니터	전자	300	5
5	5	KBS	청바지	의류	75	3
6	6	BBK	메모리	전자	120	10
7	7	SSK	책	서적	23	5

```
select * from usertbl;
```

	userid [PK] character (8)	name character varying (10)	birthyear integer	addr character (2)	mobile1 character (3)	mobile2 character (8)	height smallint	mdate date
1	LSG	이승기	1987	서울	011	1111111	182	2008-08-08
2	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
3	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
4	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
5	SSK	성시경	1979	서울	[null]	[null]	186	2013-12-12
6	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
7	YJS	윤종신	1969	경남	[null]	[null]	170	2005-05-05

```
select *
from buytbl inner join usertbl
on buytbl.userId = usertbl.userID
where buytbl.userID = 'JYP';
```

buytbl, usertbl 두 개의 테이블에 동일한 열 이름이 모두 존재하기 때문에
'테이블 이름.열 이름' 형식으로 작성해야 함

	num	userID	prodName	groupName	price	amount	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
▶	3	JYP	모니터	전자	675	1	JYP	조용필	1950	경기	011	4444444	166	2009-04-04

buytbl

usertbl

[실습 2] INNER JOIN 예시

2. INNER JOIN 실행 시 WHERE절을 생략할 경우

```
select *
from buytbl inner join usertbl
on buytbl.userID = usertbl.userID
order by num;
```

	num integer	userid character (8)	prodname character (6)	groupname character (4)	price integer	amount smallint	userid character (8)	name character varying (10)	birthyear integer	addr character (2)	mobile1 character (3)	mobile2 character (8)	height smallint	mdate date
1	1	KBS	운동화	[null]	45	2	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
2	2	KBS	노트북	전자	1500	1	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
3	3	JYP	모니터	전자	300	1	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
4	4	BBK	모니터	전자	300	5	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
5	5	KBS	청바지	의류	75	3	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
6	6	BBK	메모리	전자	120	10	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
7	7	SSK	책	서적	23	5	SSK	성시경	1979	서울	[null]	[null]	186	2013-12-12
8	8	EJW	책	서적	23	2	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
9	9	EJW	청바지	의류	75	1	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
10	10	BBK	운동화	[null]	45	2	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
11	11	EJW	책	서적	23	1	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
12	12	BBK	운동화	[null]	45	2	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05

이전 WHERE절 있을 경우

	num integer	userid character (8)	prodname character (6)	groupname character (4)	price integer	amount smallint	userid character (8)	name character varying (10)	birthyear integer	addr character (2)	mobile1 character (3)	mobile2 character (8)	height smallint	mdate date
1	3	JYP	모니터	전자	300	1	JYP	조용필	1950	경기	011	4444444	166	2009-04-04

[실습 2] INNER JOIN 예시

3. 필요한 열 추출 및 테이블.열 지정

```
SELECT userID, name, prodName, addr, CONCAT(mobile1, mobile2)AS "연락처"
FROM buytbl INNER JOIN usertbl
ON buytbl.userID = usertbl.userID
ORDER BY num;
```

```
ERROR: 오류:  칼럼 참조 "userid" 가 모호합니다.
LINE 300: select userID, name, prodName, addr, concat(mobile1,mobile2)...
```

userID가 불확실하다는 오류 메세지

테이블, 열 이름 지정

```
SELECT buytbl.userID, name, prodName, addr, CONCAT(mobile1, mobile2)AS "연락처"
FROM buytbl INNER JOIN usertbl
ON buytbl.userID = usertbl.userID
ORDER BY num;
```

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	KBS	김범수	운동화	경남	0112222222
2	KBS	김범수	노트북	경남	0112222222
3	JYP	조용필	모니터	경기	0114444444
4	BBK	바비킴	모니터	서울	0100000000
5	KBS	김범수	청바지	경남	0112222222
6	BBK	바비킴	메모리	서울	0100000000
7	SSK	성시경	책	서울	
8	EJW	은지원	책	경북	0118888888
9	EJW	은지원	청바지	경북	0118888888
10	BBK	바비킴	운동화	서울	0100000000
11	EJW	은지원	책	경북	0118888888
12	BBK	바비킴	운동화	서울	0100000000

[실습 2] INNER JOIN 예시

4. 테이블 별칭 지정

```
SELECT B.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)  AS "연락처"
FROM buytbl B
INNER JOIN usertbl U
ON B.userID = U.userID
ORDER BY B.num;
```

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	KBS	김범수	운동화	경남	0112222...
2	KBS	김범수	노트북	경남	0112222...
3	JYP	조용필	모니터	경기	0114444...
4	BBK	바비킴	모니터	서울	0100000...
5	KBS	김범수	청바지	경남	0112222...
6	BBK	바비킴	메모리	서울	0100000...
7	SSK	성시경	책	서울	
8	EJW	은지원	책	경북	0118888...
9	EJW	은지원	청바지	경북	0118888...
10	BBK	바비킴	운동화	서울	0100000...
11	EJW	은지원	책	경북	0118888...
12	BBK	바비킴	운동화	서울	0100000...

테이블에 별칭을 지정하여 실행했을 때에도
그렇지 않은 것과 같은 결과값이 나타나는 것을 확인할 수 있음

- 테이블에 별칭을 주기 위해서는 FROM절에 나오는 테이블의 이름 뒤에 별칭을 붙여주면 됨
- 여러 개의 테이블이 관련되는 조인에서는 별칭을 붙이는 방식을 사용할 것을 적극 권장함

[실습 2] INNER JOIN 예시

5. 테이블 별칭 지정 및 특정 열 출력

```
SELECT B.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)  AS "연락처"
FROM buytbl B
INNER JOIN usertbl U
ON B.userID = U.userID
WHERE B.userID = 'JYP';
```

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	JYP	조용필	모니터	경기	0114444444

[실습 2] INNER JOIN 예시

6. 테이블 별칭 지정, 특정 열 출력, 정렬

```
SELECT B.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)  AS "연락처"
FROM usertbl U
INNER JOIN buytbl B
ON U.userID = B.userID
ORDER BY U.userID;
```

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	BBK	바비킴	운동화	서울	0100000...
2	BBK	바비킴	메모리	서울	0100000...
3	BBK	바비킴	운동화	서울	0100000...
4	BBK	바비킴	모니터	서울	0100000...
5	EJW	은지원	청바지	경북	0118888...
6	EJW	은지원	책	경북	0118888...
7	EJW	은지원	책	경북	0118888...
8	JYP	조용필	모니터	경기	0114444...
9	KBS	김범수	운동화	경남	0112222...
10	KBS	김범수	노트북	경남	0112222...
11	KBS	김범수	청바지	경남	0112222...
12	SSK	성시경	책	서울	

ORDER BY U.userID로 사용자 이름으로 정렬했을 경우

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	KBS	김범수	운동화	경남	0112222222
2	KBS	김범수	노트북	경남	0112222222
3	JYP	조용필	모니터	경기	0114444444
4	BBK	바비킴	모니터	서울	0100000000
5	KBS	김범수	청바지	경남	0112222222
6	BBK	바비킴	메모리	서울	0100000000
7	SSK	성시경	책	서울	
8	EJW	은지원	책	경북	0118888888
9	EJW	은지원	청바지	경북	0118888888
10	BBK	바비킴	운동화	서울	0100000000
11	EJW	은지원	책	경북	0118888888
12	BBK	바비킴	운동화	서울	0100000000

정렬하지 않은 경우

[실습 2] INNER JOIN 예시

7. DISTINCT를 이용한 중복 제거

```
SELECT DISTINCT U.userID, U.name,  U.addr
FROM usertbl U
INNER JOIN buytbl B
ON U.userID = B.userID
ORDER BY U.userID ;
```

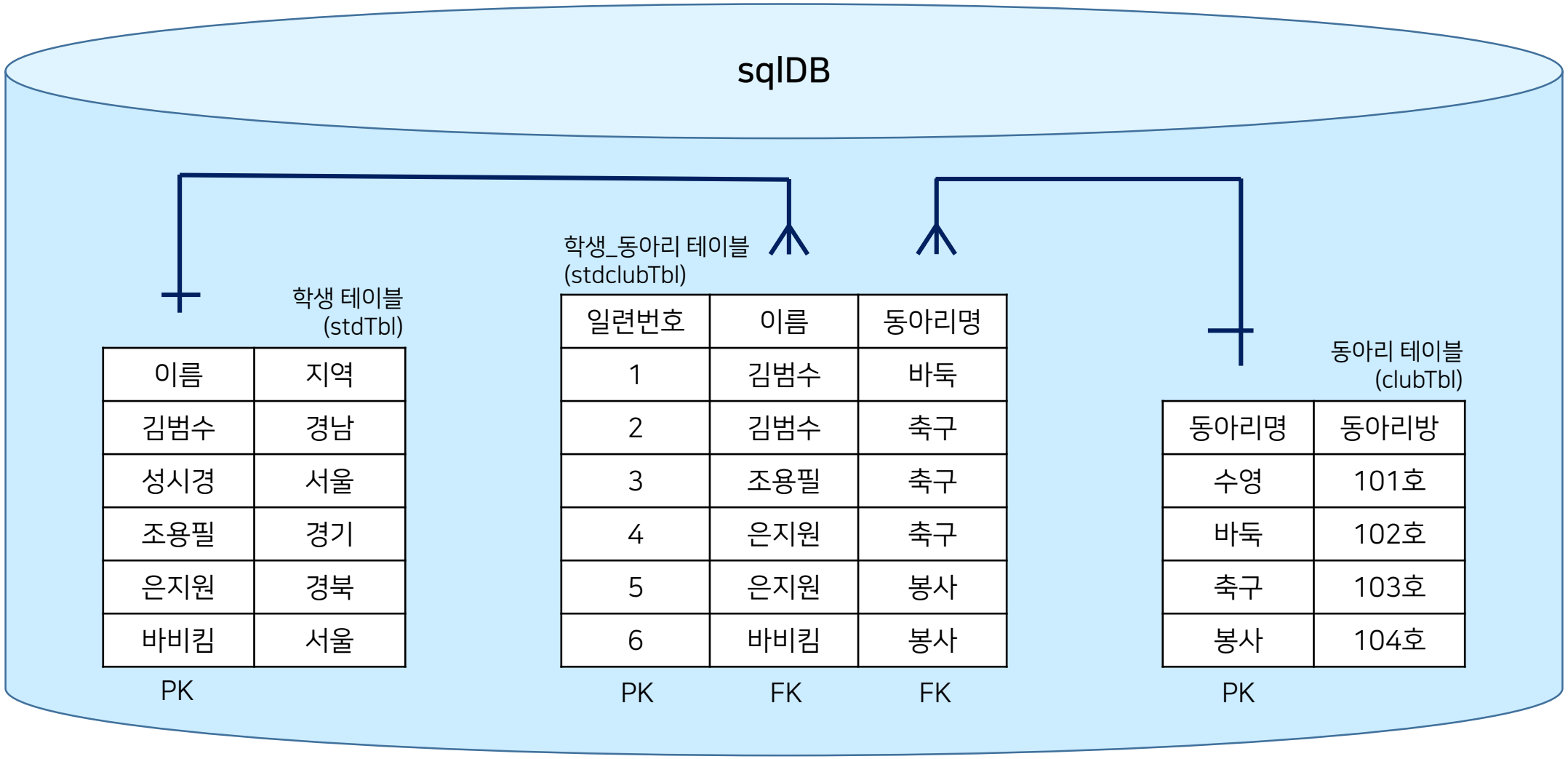
	userid [PK] character (8)	name character varying (10)	addr character (2)
1	BBK	바비킴	서울
2	EJW	은지원	경북
3	JYP	조용필	경기
4	KBS	김범수	경남
5	SSK	성시경	서울

한 번이라도 구매한 이력이 있는 회원 조회

- 여태 실습하면서 단 한 번도 구매하지 않은 회원인 이승기, 김경호, 임재범, 윤종신, 조관우는 나타나지 않았음(usertbl 조회해볼 것)
- 구매한 회원의 기록도 나오면서, 구매하지 않았어도 그 회원의 이름/주소 등은 나오도록 조인할 필요가 있을 수 있음
- 그럴 경우 사용하는 방식이 OUTER JOIN임

[실습 3] 3개 테이블 JOIN 실습

1. 세 개의 테이블 샘플



[실습 3] 3개 테이블 JOIN 실습

2. 테이블 생성 및 데이터 입력

```
set schema 'public';
CREATE TABLE stdtbl
( stdName    VARCHAR(10) NOT NULL PRIMARY KEY,
  addr    CHAR(4) NOT NULL
);
CREATE TABLE clubtbl
( clubName    VARCHAR(10) NOT NULL PRIMARY KEY,
  roomNo    CHAR(4) NOT NULL
);
CREATE TABLE stdclubtbl
(  num serial NOT NULL PRIMARY KEY,
   stdName    VARCHAR(10) NOT NULL,
   clubName    VARCHAR(10) NOT NULL,
FOREIGN KEY(stdName) REFERENCES stdtbl(stdName),
FOREIGN KEY(clubName) REFERENCES clubtbl(clubName)
);
INSERT INTO stdtbl VALUES ('김범수','경남'), ('성시경','서울'), ('조용필','경기'), ('은지원','경북'),('바비킴','서울');
INSERT INTO clubtbl VALUES ('수영','101호'), ('바둑','102호'), ('축구','103호'), ('봉사','104호');
INSERT INTO stdclubtbl VALUES (DEFAULT, '김범수','바둑'), (DEFAULT,'김범수','축구'), (DEFAULT,'조용필','축구'), (DEFAULT,'은지원','축구')
                            , (DEFAULT,'은지원','봉사'), (DEFAULT,'바비킴','봉사');
```

[실습 3] 3개 테이블 JOIN 실습

3. 학생 기준 학생 이름/지역/가입한 동아리/동아리방 출력

```
select*from stdtbl;
```

	stdname [PK] character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울

```
select*from clubtbl;
```

	clubname [PK] character varying (10)	roomno character (4)
1	수영	101호
2	바둑	102호
3	축구	103호
4	봉사	104호

```
select*from stdclubtbl;
```

	num [PK] integer	stdname character varying (10)	clubname character varying (10)
1	1	김범수	바둑
2	2	김범수	축구
3	3	조용필	축구
4	4	은지원	축구
5	5	은지원	봉사
6	6	바비킴	봉사

```
SELECT S.stdName, S.addr, SC.clubName, C.roomNo
   FROM stdtbl S
      INNER JOIN stdclubtbl SC
            ON S.stdName = SC.stdName
      INNER JOIN clubtbl C
            ON SC.clubName = C.clubName
   ORDER BY S.stdName;
```

JOIN

	stdname character varying (10)	addr character (4)	clubname character varying (10)	roomno character (4)
1	김범수	경남	바둑	102호
2	김범수	경남	축구	103호
3	바비킴	서울	봉사	104호
4	은지원	경북	축구	103호
5	은지원	경북	봉사	104호
6	조용필	경기	축구	103호

3개 테이블 조인 결과

- 학생동아리 테이블과 학생 테이블 (일대다 관계) INNER JOIN
- 학생동아리 테이블과 동아리 테이블(일대다 관계) INNER JOIN

[실습 3] 3개 테이블 JOIN 실습

4. 동아리 기준 가입한 학생 목록 출력

```
select*from stdtbl;
```

	stdname [PK] character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울

```
select*from clubtbl;
```

	clubname [PK] character varying (10)	roomno character (4)
1	수영	101호
2	바둑	102호
3	축구	103호
4	봉사	104호

```
select*from stdclubtbl;
```

	num [PK] integer	stdname character varying (10)	clubname character varying (10)
1	1	김범수	바둑
2	2	김범수	축구
3	3	조용필	축구
4	4	은지원	축구
5	5	은지원	봉사
6	6	바비킴	봉사

```
SELECT C.clubName, C.roomNo, S.stdName, S.addr
   FROM  stdtbl S
      INNER JOIN stdclubtbl SC
         ON SC.stdName = S.stdName
      INNER JOIN clubtbl C
          ON SC.clubName = C.clubName
    ORDER BY C.clubName;
```

JOIN

	clubname character varying (10)	roomno character (4)	stdname character varying (10)	addr character (4)
1	바둑	102호	김범수	경남
2	봉사	104호	은지원	경북
3	봉사	104호	바비킴	서울
4	축구	103호	김범수	경남
5	축구	103호	조용필	경기
6	축구	103호	은지원	경북

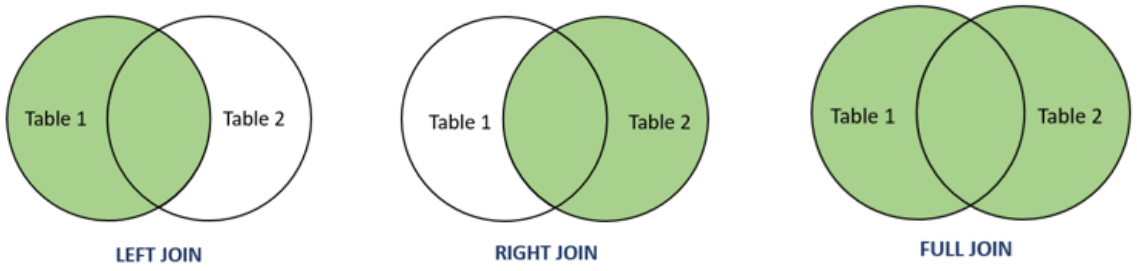
SECTION 02 조인

OUTER JOIN(외부 조인)

- 조인의 조건에 만족되지 않는 행까지도 포함시키는 것

```
SELECT <열 목록>
FROM <첫 번째 테이블(LEFT 테이블)>
 <LEFT ¦ RIGHT ¦ FULL> OUTER JOIN <두 번째 테이블(RIGHT 테이블)>
      ON <조인될 조건>
[WHERE 검색조건] ;
```

- LEFT OUTER JOIN
  - 왼쪽 테이블의 것은 모두 출력되어야 한다로 이해
  - 줄여서 LEFT JOIN으로 쓸 수있음
- RIGHT OUTER JOIN
  - 오른쪽 테이블의 것은 모두 출력되어야 한다로 이해



SECTION 02 조인

OUTER JOIN(외부 조인)

- LEFT OUTER JOIN

```
set schema 'shop';
SELECT U.userID, U.name, B.prodName, U.addr, CONCAT(U.mobile1, U.mobile2)  AS "연락처"
   FROM usertbl U
      LEFT OUTER JOIN buytbl B
         ON U.userID = B.userID
   ORDER BY U.userID;
```

	userid character (8)	name character varying (10)	prodname character (6)	addr character (2)	연락처 text
1	BBK	바비킴	운동화	서울	0100000...
2	BBK	바비킴	메모리	서울	0100000...
3	BBK	바비킴	운동화	서울	0100000...
4	BBK	바비킴	모니터	서울	0100000...
5	EJW	은지원	청바지	경북	0118888...
6	EJW	은지원	책	경북	0118888...
7	EJW	은지원	책	경북	0118888...
8	JKW	조관우	[null]	경기	0189999...
9	JYP	조용필	모니터	경기	0114444...
10	KBS	김범수	운동화	경남	0112222...
11	KBS	김범수	노트북	경남	0112222...
12	KBS	김범수	청바지	경남	0112222...
13	KKH	김경호	[null]	전남	0193333...
14	LJB	임재범	[null]	서울	0166666...
15	LSG	이승기	[null]	서울	0111111...
16	SSK	성시경	책	서울	
17	YJS	윤종신	[null]	경남	

[실습 4] LEFT/RIGHT/FULL OUTER JOIN 실습(앞의 실습 3 INNER JOIN 결과를 OUTER JOIN으로 고려, 두 개 JOIN 고려한 FULL JOIN 테스트)

1. 동아리에 가입하지 않은 학생도 출력되도록 수정

[실습 3] 에서는 동아리에 가입되지 않은 성시경은 출력되지 않았음
성시경도 출력해보도록 하자

간단히 INNER JOIN을 LEFT OUTER JOIN으로 변경

```
select*from stdtbl;
```

	stdname [PK] character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울

```
select*from clubtbl;
```

	clubname [PK] character varying (10)	roomno character (4)
1	수영	101호
2	바둑	102호
3	축구	103호
4	봉사	104호

```
select*from stdclubtbl;
```

	num [PK] integer	stdname character varying (10)	clubname character varying (10)
1	1	김범수	바둑
2	2	김범수	축구
3	3	조용필	축구
4	4	은지원	축구
5	5	은지원	봉사
6	6	바비킴	봉사

```
set schema 'public';
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdtbl S
      LEFT OUTER JOIN stdclubtbl SC
          ON S.stdName = SC.stdName
      LEFT OUTER JOIN clubtbl C
          ON SC.clubName = C.clubName
   ORDER BY S.stdName;
```

JOIN

	stdname character varying (10)	addr character (4)	clubname character varying (10)	roomno character (4)
1	김범수	경남	축구	103호
2	김범수	경남	바둑	102호
3	바비킴	서울	봉사	104호
4	성시경	서울	[null]	[null]
5	은지원	경북	축구	103호
6	은지원	경북	봉사	104호
7	조용필	경기	축구	103호

[실습 4] LEFT/RIGHT/FULL OUTER JOIN 실습(앞의 실습 3 INNER JOIN 결과를 OUTER JOIN으로 고려, 두 개 JOIN 고려한 FULL JOIN 테스트)

2. 동아리 기준 가입된 학생을 출력하되, 가입 학생이 하나도 없는 동아리도 출력

```
select*from stdtbl;
```

	stdname [PK] character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울

```
select*from clubtbl;
```

	clubname [PK] character varying (10)	roomno character (4)
1	수영	101호
2	바둑	102호
3	축구	103호
4	봉사	104호

```
select*from stdclubtbl;
```

	num [PK] integer	stdname character varying (10)	clubname character varying (10)
1	1	김범수	바둑
2	2	김범수	축구
3	3	조용필	축구
4	4	은지원	축구
5	5	은지원	봉사
6	6	바비킴	봉사

```
SELECT C.clubName, C.roomNo, S.stdName, S.addr
   FROM  stdtbl S
      LEFT OUTER JOIN stdclubtbl SC
          ON SC.stdName = S.stdName
      RIGHT OUTER JOIN clubtbl C
          ON SC.clubName = C.clubName
   ORDER BY C.clubName ;
```

JOIN

	clubname character varying (10)	roomno character (4)	stdname character varying (10)	addr character (4)
1	바둑	102호	김범수	경남
2	봉사	104호	은지원	경북
3	봉사	104호	바비킴	서울
4	수영	101호	[null]	[null]
5	축구	103호	은지원	경북
6	축구	103호	김범수	경남
7	축구	103호	조용필	경기

클럽을 기준으로 조인해야 하므로 두 번째 조인은 RIGHT OUTER JOIN으로 처리해서 clubtbl이 조인의 기준이 되도록 설정

[실습 4] LEFT/RIGHT/FULL OUTER JOIN 실습(앞의 실습 3 INNER JOIN 결과를 OUTER JOIN으로 고려, 두 개 JOIN 고려한 FULL JOIN 테스트)

3. 1, 2의 결과를 하나로 통합

```
select*from stdtbl;
```

	stdname [PK] character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울

```
select*from clubtbl;
```

	clubname [PK] character varying (10)	roomno character (4)
1	수영	101호
2	바둑	102호
3	축구	103호
4	봉사	104호

```
select*from stdclubtbl;
```

	num [PK] integer	stdname character varying (10)	clubname character varying (10)
1	1	김범수	바둑
2	2	김범수	축구
3	3	조용필	축구
4	4	은지원	축구
5	5	은지원	봉사
6	6	바비킴	봉사

```
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM stdtbl S
      LEFT OUTER JOIN stdclubtbl SC
          ON S.stdName = SC.stdName
      LEFT OUTER JOIN clubtbl C
          ON SC.clubName = C.clubName
UNION
SELECT S.stdName, S.addr, C.clubName, C.roomNo
   FROM  stdtbl S
      LEFT OUTER JOIN stdclubtbl SC
          ON SC.stdName = S.stdName
      RIGHT OUTER JOIN clubtbl C
          ON SC.clubName = C.clubName;
```

JOIN

	stdname character varying (10)	addr character (4)	clubname character varying (10)	roomno character (4)
1	김범수	경남	축구	103호
2	성시경	서울	[null]	[null]
3	김범수	경남	바둑	102호
4	[null]	[null]	수영	101호
5	은지원	경북	축구	103호
6	바비킴	서울	봉사	104호
7	은지원	경북	봉사	104호
8	조용필	경기	축구	103호

동아리에 가입하지 않은 성시경과
가입한 학생이 없는 수영 동아리가 모두 출력됨

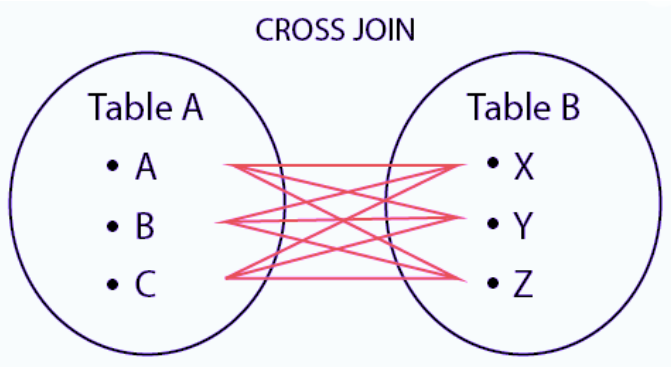
동아리에 가입하지 않은 학생도 출력되고 학생이 한 명도 없는 동아리도 출력되게 해야 함

앞의 두 쿼리를 UNION으로 합쳐줌

SECTION 02 조인

CROSS JOIN(상호 조인)

- 한쪽 테이블의 모든 행들과 다른 쪽 테이블의 모든 행을 조인(연결)시키는 기능
- CROSS JOIN의 결과 개수 = 두 테이블 개수를 곱한 개수




```
SELECT *
    FROM buytbl
      CROSS JOIN usertbl ;
```

회원 테이블(userTbl)

아이디	이름	생년	지역	국번	전화번호	키	가입일
LSG	이승기	1987	서울	011	1111111	182	2008.8.8
KBS	김범수	1979	경남	011	2222222	173	2012.4.4
KKH	김경호	1971	전남	019	3333333	177	2007.7.7
JYP	조용필	1950	경기	011	4444444	166	2009.4.4
SSK	성시경	1979	서울			186	2013.12.12
LJB	임재범	1963	서울	016	6666666	182	2009.9.9
YJS	윤종신	1969	경남			170	2005.5.5
EJW	은지원	1978	경북	011	8888888	174	2014.3.3
JKW	조관우	1965	경기	018	9999999	172	2010.10.10
BBK	바비킴	1973	서울	010	0000000	176	2013.5.5

PK

구매 테이블(buyTbl)

순번	아이디	물품명	분류	단가	수량
1	KBS	운동화		30	2
2	KBS	노트북	전자	1000	1
3	JYP	모니터	전자	200	1
4	BBK	모니터	전자	200	5
5	KBS	청바지	의류	50	3
6	BBK	메모리	전자	80	10
7	SSK	책	서적	15	5
8	EJW	책	서적	15	2
9	EJW	청바지	의류	50	1
10	BBK	운동화		30	2
11	EJW	책	서적	15	1
12	BBK	운동화		30	2

PK FK

[그림 7-43] CROSS JOIN(상호 조인) 방식

	num integer	userid character (8)	prodname character (6)	groupname character (4)	price integer	amount smallint	userid character (8)	name character varying (10)	birthyear integer	addr character (2)	mobile1 character (3)	mobile2 character (8)	height smallint	mdate date
1	1	KBS	운동화	[null]	30	2	LSG	이승기	1987	서울	011	1111111	182	2008-08-08
2	1	KBS	운동화	[null]	30	2	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
3	1	KBS	운동화	[null]	30	2	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
4	1	KBS	운동화	[null]	30	2	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
5	1	KBS	운동화	[null]	30	2	SSK	성시경	1979	서울	[null]	[null]	186	2013-12-12
6	1	KBS	운동화	[null]	30	2	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
7	1	KBS	운동화	[null]	30	2	YJS	윤종신	1969	경남	[null]	[null]	170	2005-05-05
8	1	KBS	운동화	[null]	30	2	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
9	1	KBS	운동화	[null]	30	2	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
10	1	KBS	운동화	[null]	30	2	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
11	2	KBS	노트북	전자	1000	1	LSG	이승기	1987	서울	011	1111111	182	2008-08-08
12	2	KBS	노트북	전자	1000	1	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
13	2	KBS	노트북	전자	1000	1	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
14	2	KBS	노트북	전자	1000	1	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
15	2	KBS	노트북	전자	1000	1	SSK	성시경	1979	서울	[null]	[null]	186	2013-12-12
16	2	KBS	노트북	전자	1000	1	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
17	2	KBS	노트북	전자	1000	1	YJS	윤종신	1969	경남	[null]	[null]	170	2005-05-05
18	2	KBS	노트북	전자	1000	1	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
19	2	KBS	노트북	전자	1000	1	JKW	조관우	1965	경기	018	9999999	172	2010-10-10

SECTION 02 조인

CROSS JOIN(상호 조인)

- 테스트로 사용할 많은 용량의 데이터를 생성할 때 주로 사용
- ON 구문을 사용할 수 없음
- 대량의 데이터를 생성하면 시스템이 다운되거나 디스크 용량이 모두 찰 수 있어 COUNT(*) 함수로 개수만 카운트

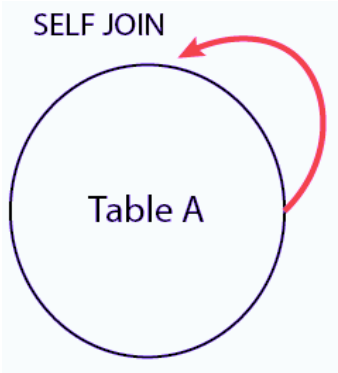
```
set schema 'shop';
select count(*) as "데이터 개수"
from buytbl cross join usertbl;
```

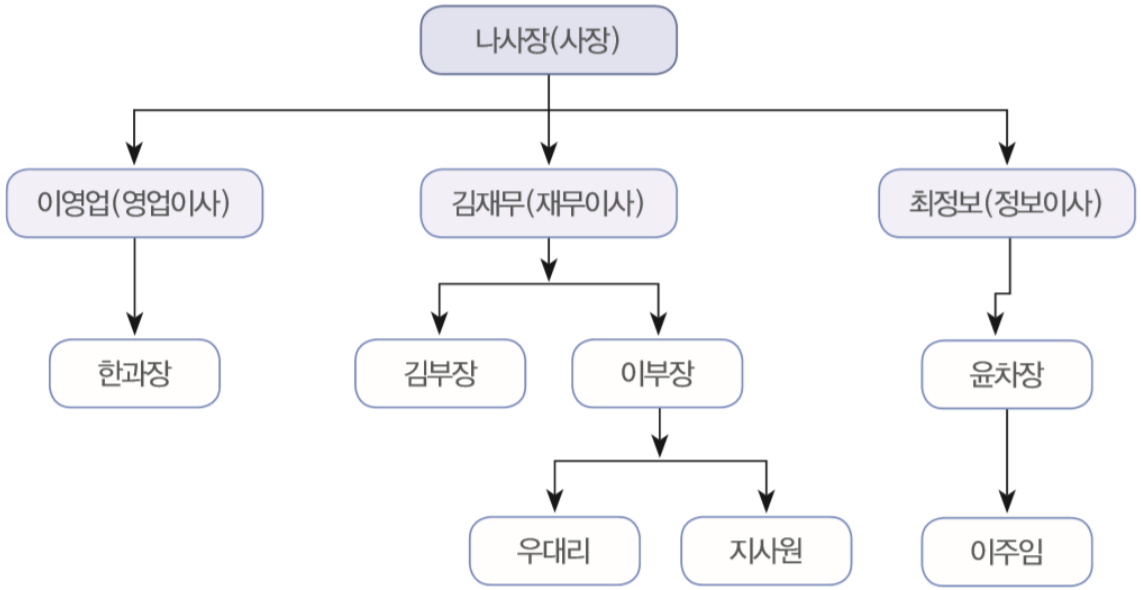
	데이터 개수 bigint
1	120

SECTION 02 조인

SELF JOIN(자체 조인)

- 자기 자신과 자기 자신이 조인한다는 의미
  - 대표적인 예
    - 조직도와 관련된 테이블





[그림 7-45] 간단한 조직도 예

직원 이름(EMP) - 기본 키	상관 이름(MANAGER)	구내 번호
나사장	없음 (NULL)	0000
김재무	나사장	2222
김부장	김재무	2222-1
이부장	김재무	2222-2
우대리	이부장	2222-2-1
지사원	이부장	2222-2-2
이영업	나사장	1111
한과장	이영업	1111-1
최정보	나사장	3333
윤차장	최정보	3333-1
이주임	윤차장	3333-1-1

[표 7-5] 조직도 테이블

[실습 5] 하나의 테이블에서 SELF JOIN 실습

1. 테이블 정의 및 데이터 입력

```
set schema 'public';
CREATE TABLE empTbl (emp CHAR(3), manager CHAR(3), empTel VARCHAR(8));

INSERT INTO empTbl VALUES('나사장',NULL,'0000');
INSERT INTO empTbl VALUES('김재무','나사장','2222');
INSERT INTO empTbl VALUES('김부장','김재무','2222-1');
INSERT INTO empTbl VALUES('이부장','김재무','2222-2');
INSERT INTO empTbl VALUES('우대리','이부장','2222-2-1');
INSERT INTO empTbl VALUES('지사원','이부장','2222-2-2');
INSERT INTO empTbl VALUES('이영업','나사장','1111');
INSERT INTO empTbl VALUES('한과장','이영업','1111-1');
INSERT INTO empTbl VALUES('최정보','나사장','3333');
INSERT INTO empTbl VALUES('윤차장','최정보','3333-1');
INSERT INTO empTbl VALUES('이주임','윤차장','3333-1-1');
```

	emp character (3)	manager character (3)	emptel character varying (8)
1	나사장	[null]	0000
2	김재무	나사장	2222
3	김부장	김재무	2222-1
4	이부장	김재무	2222-2
5	우대리	이부장	2222-2-1
6	지사원	이부장	2222-2-2
7	이영업	나사장	1111
8	한과장	이영업	1111-1
9	최정보	나사장	3333
10	윤차장	최정보	3333-1
11	이주임	윤차장	3333-1-1

2. SELF JOIN 활용

```
SELECT A.emp AS "부하직원" , B.emp AS "직속상관", B.empTel AS "직속상관연락처"
   FROM empTbl A
      INNER JOIN empTbl B
         ON A.manager = B.emp
   WHERE A.emp = '우대리';
```

JOIN

	부하직원 character (3)	직속상관 character (3)	직속상관연락처 character varying (8)
1	우대리	이부장	2222-2

SECTION 02 조인

UNION / UNION ALL / NOT IN / IN

- UNION : 두 쿼리의 결과를 행으로 합치는 것

```
SELECT 문장1
   UNION [ALL]
SELECT 문장2
```

- SELECT 문장1과 SELECT 문장2의 결과 열의 개수가 같아야 함
- 데이터 형식도 각 열 단위로 같거나 서로 호환되는 데이터 형식이어야 함
- 문장 1의 열 이름을 따름
- 문장 1의 결과가 INT인데 문장 2의 결과가 CHAR라면 오류 발생!!

SELECT stdName, addr FROM stdTbl

stdName	addr
김범수	경남
성시성	서울
조용필	경기
은지원	경북
바비킴	서울

SELECT clubName, roomNo FROM clubTbl

clubName	roomNo
수영	101호
바둑	102호
축구	103호
봉사	104호

UNION ALL

stdName	addr
김범수	경남
성시성	서울
조용필	경기
은지원	경북
바비킴	서울
수영	101호
바둑	102호
축구	103호
봉사	104호

[그림 7-47] UNION의 결합과정

- UNION만 사용하면 중복된 열은 제거되고 데이터가 정렬됨
- UNION ALL을 사용하면 중복된 열까지 모두 출력됨

[실습 6] UNION ALL / NOT IN / IN 실습

1. UNION ALL

```
set schema 'shop';
SELECT stdName, addr FROM stdtbl
   UNION ALL
SELECT clubName, roomNo FROM clubtbl;
```

	stdname character varying (10)	addr character (4)
1	김범수	경남
2	성시경	서울
3	조용필	경기
4	은지원	경북
5	바비킴	서울
6	수영	101호
7	바둑	102호
8	축구	103호
9	봉사	104호

2. NOT IN

```
SELECT name, CONCAT(mobile1, mobile2) AS "전화번호" FROM usertbl
   WHERE name NOT IN ( SELECT name FROM usertbl WHERE mobile1 IS NULL) ;
```

첫 번째 쿼리의 결과 중에서
두 번째 쿼리에 해당하는 것을 제외하기 위한 구문

usertbl의 사용자를 모두 조회하되 전화가 없는 사람을 제외함

	name character varying (10)	전화번호 text
1	이승기	0111111...
2	김범수	0112222...
3	김경호	0193333...
4	조용필	0114444...
5	임재범	0166666...
6	은지원	0118888...
7	조관우	0189999...
8	바비킴	0100000...

3. IN

```
SELECT name, CONCAT(mobile1, mobile2) AS "전화번호" FROM usertbl
   WHERE name IN ( SELECT name FROM usertbl WHERE mobile1 IS NULL);
```

	name character varying (10)	전화번호 text
1	성시경	
2	윤종신	

NOT IN과 반대로 첫 번째 쿼리의 결과 중에서
두 번째 쿼리에 해당되는 것만 조회하기 위한 구문

전화가 없는 사람만 조회함

SECTION 03 SQL 프로그래밍

스토어드 프로시저 생성 및 사용

- SQL도 분기, 흐름제어, 반복의 기능이 있음

```
CREATE [or replace] PROCEDURE 스토어드 프로시저이름
                        ( 파라미터명1 데이터타입,
                          파라미터명2 데이터타입, …)
LANGUAGE plpgsql;
AS $$
DECLARE
        변수 선언
BEGIN


        이 부분에 SQL 프로그래밍 코딩…



END; $$

CALL 스토어드 프로시저이름();
```

CREATE PROCEDURE로 생성한 스토어드 프로시저를 호출(=실행)함

$$ ~
END; $$
스토어드 프로시저의
코딩할 부분을 묶어줌

END; $$가 나올 때까지를
스토어드 프로시저로 인식
하게 함

IF…ELSE

- 조건에 따라 분기
  - 참 / 거짓 두 가지만 있기에 2중 분기
- 한 문장 이상 처리되어야 할 때 BEGIN.. END로 묶어주기
- 형식

```
형식:
IF 〈부울 표현식〉 THEN
        SQL문장들1..
ELSE
        SQL문장들2..
END IF;
```

  - 부울 표현식 부분이 참이면 SQL문장들1 수행 / 거짓이면 SQL문장들2 수행
  - 거짓일 경우이면서 아무 것도 할 것이 없다면 ELSE 이하 생략 가능

[실습 8] IF…ELSE 실습

1. var1이 100인가, 100이 아닌가?

```
create procedure ifProc()
as $$
declare
    var1 integer := 100; -- 변수 선언 및 값 대입
begin
    if var1 = 100 then
        raise notice '100입니다.';
    else
        raise notice '100이 아닙니다.';
    end if;
end;
$$
language plpgsql;

call ifProc();
```

```
알림:  100입니다.
CALL
```

[실습 8] IF…ELSE 실습

2. 직원번호 10001에 해당하는 직원의 입사일이 5년이 넘었는지 확인

```
set schema 'employees';
CREATE PROCEDURE ifProc2()
as $$
DECLARE
    hireDATE DATE; -- 입사일
    curDATE DATE; -- 오늘
    days INT; -- 근무한 일수
BEGIN
    SELECT hire_date INTO hireDate -- hire_date 열의 결과를 hireDATE에 대입
       FROM employees
       WHERE emp_no = 10001;

    curDATE := CURRENT_DATE; -- 현재 날짜
    days := extract(day from curDATE::timestamp-hireDATE::timestamp); -- 날짜의 차이, 일 단위

    IF (days/365) >= 5 THEN -- 5년이 지났다면
        raise notice '입사한지 %일이나 지났습니다. 축하합니다!',days;
    ELSE
        raise notice '입사한지 %일밖에 안되었네요. 열심히 일하세요.',days;
    END IF;
END;
$$
language plpgsql;

CALL ifProc2();
```

```
알림:  입사한지 13178일이나 지났습니다. 축하합니다!
CALL
```

[실습 8] IF…ELSE 실습

3. Inserting data using a procedure

```
set schema 'public';

drop procedure if exists emp_insert_data;

create procedure emp_insert_data("emp" character,"manager" character
                                 ,"emptel" character varying)
as $$
begin
    insert into empTbl values ("emp","manager","emptel");
end;
$$
language plpgsql;

call emp_insert_data('황주임','서대리','4444');
```

	emp character (3)	manager character (3)	emptel character varying (8)
1	나사장	[null]	0000
2	김재무	나사장	2222
3	김부장	김재무	2222-1
4	이부장	김재무	2222-2
5	우대리	이부장	2222-2-1
6	지사원	이부장	2222-2-2
7	이영업	나사장	1111
8	한과장	이영업	1111-1
9	최정보	나사장	3333
10	윤차장	최정보	3333-1
11	이주임	윤차장	3333-1-1

	emp character (3)	manager character (3)	emptel character varying (8)
1	나사장	[null]	0000
2	김재무	나사장	2222
3	김부장	김재무	2222-1
4	이부장	김재무	2222-2
5	우대리	이부장	2222-2-1
6	지사원	이부장	2222-2-2
7	이영업	나사장	1111
8	한과장	이영업	1111-1
9	최정보	나사장	3333
10	윤차장	최정보	3333-1
11	이주임	윤차장	3333-1-1
12	황주임	서대리	4444

CASE

- 조건에 따라 분기
  - 다중 분기
  - 조건에 맞는 WHEN이 여러 개더라도 먼저 조건이 만족하는 WHEN 처리됨
  - SELECT문에서 많이 사용됨
  - 점수로 성적을 판단하는 경우처럼 여러 단계로 분기 될 때 사용

[실습 9] CASE 실습

1. 학점 계산 프로그램

```
CREATE PROCEDURE ifProc3()
AS $$
DECLARE
    point INT ;
    credit CHAR(1);
BEGIN
    point := 77 ;

    IF point >= 90 THEN
        credit := 'A';
    ELSEIF point >= 80 THEN
        credit := 'B';
    ELSEIF point >= 70 THEN
        credit := 'C';
    ELSEIF point >= 60 THEN
        credit := 'D';
    ELSE
        credit := 'F';
    END IF;

    RAISE NOTICE '취득점수==>%', point;
    RAISE NOTICE '학점==>%',credit;
END $$
LANGUAGE plpgsql;

CALL ifProc3();
```

```
알림:  취득점수==>77
알림:  학점==>C
CALL
```

[실습 9] CASE 실습

2. 1번의 IF문을 CASE문으로 변경

```
CREATE PROCEDURE caseProc()
AS $$
DECLARE
    point INT ;
    credit CHAR(1);
BEGIN
    point := 77 ;

    case
      WHEN point >= 90 THEN
        credit := 'A';
      WHEN point >= 80 THEN
        credit := 'B';
      WHEN point >= 70 THEN
        credit := 'C';
      WHEN point >= 60 THEN
        credit := 'D';
      ELSE
        credit := 'F';
    END case;

    RAISE NOTICE '취득점수==>%', point;
    RAISE NOTICE '학점==>%',credit;
END $$
LANGUAGE plpgsql;
```

```
알림:  취득점수==>77
알림:  학점==>C
CALL
```

CASE문은 조건에 맞는 WHEN이 여러 개더라도
먼저 조건을 만족하는 WHEN이 처리되고 CASE를 종료함

[실습 10] CASE문을 활용하는 SQL 프로그래밍 작성

shop schema 의 구매 테이블(buytbl)에 구매액(price*amount)이 1500원 이상인 고객은 '최우수 고객', 1000원 이상 '우수 고객', 1원 이상 '일반 고객', 전혀 구매실적 x '유령 고객'

1. buytbl 구매액(price*amount)을 사용자 아이디(userID) 별로 그룹화 및 구매액 높은 순으로 정렬

```
set schema 'shop';
select userID, sum(price*amount) as "총구매액"
    from buytbl
    group by userID
    order by sum(price*amount) desc;
```

	userid character (8)	총구매액 bigint
1	BBK	1920
2	KBS	1210
3	JYP	200
4	EJW	95
5	SSK	75

2. usertbl과 JOIN해서 사용자 이름도 출력

```
select B.userID, U.name, sum(price*amount) as "총구매액"
    from buytbl B
        inner join usertbl U
            on B.userID = U.userID
    group by B.userID, U.name
    order by sum(price*amount) desc;
```

	userid character (8)	name character varying (10)	총구매액 bigint
1	BBK	바비킴	1920
2	KBS	김범수	1210
3	JYP	조용필	200
4	EJW	은지원	95
5	SSK	성시경	75

[실습 10] CASE문을 활용하는 SQL 프로그래밍 작성

shop schema 의 구매 테이블(buytbl)에 구매액(price*amount)이 1500원 이상인 고객은 '최우수 고객', 1000원 이상 '우수 고객', 1원 이상 '일반 고객', 전혀 구매실적 x '유령 고객'

3. 구매하지 않은 고객의 명단도 출력(RIGHT OUTER JOIN)

```
select B.userID, U.name, sum(price*amount) as "총구매액"
    from buytbl B
        right outer join usertbl U
            on B.userID = U.userID
    group by B.userID, U.name
    order by sum(price*amount) desc;
```

	userid character (8)	name character varying (10)	총구매액 bigint
1	[null]	김경호	[null]
2	[null]	조관우	[null]
3	[null]	임재범	[null]
4	[null]	이승기	[null]
5	[null]	윤종신	[null]
6	BBK	바비킴	1920
7	KBS	김범수	1210
8	JYP	조용필	200
9	EJW	은지원	95
10	SSK	성시경	75

4. 구매 기록이 없는 고객 userID가 NULL 이므로 userID 채우기

```
select U.userID, U.name, sum(price*amount) as "총구매액"
    from buytbl B
        right outer join usertbl U
            on B.userID = U.userID
    group by U.userID, U.name
    order by sum(price*amount) desc;
```

	userid [PK] character (8)	name character varying (10)	총구매액 bigint
1	YJS	윤종신	[null]
2	JKW	조관우	[null]
3	LJB	임재범	[null]
4	KKH	김경호	[null]
5	LSG	이승기	[null]
6	BBK	바비킴	1920
7	KBS	김범수	1210
8	JYP	조용필	200
9	EJW	은지원	95
10	SSK	성시경	75

[실습 10] CASE문을 활용하는 SQL 프로그래밍 작성

shop schema 의 구매 테이블(buytbl)에 구매액(price*amount)이 1500원 이상인 고객은 '최우수 고객', 1000원 이상 '우수 고객', 1원 이상 '일반 고객', 전혀 구매실적 x '유령 고객'

5. CASE문만 따로 고려

```
/*
CASE
     WHEN (총구매액  >= 1500) THEN  '최우수고객'
     WHEN (총구매액  >= 1000) THEN  '우수고객'
     WHEN (총구매액 >= 1 ) THEN '일반고객'
     ELSE '유령고객'
END
*/
```

6. 작성한 CASE 구문 SELECT에 추가 (최종쿼리)

```
SELECT U.userID, U.name, SUM(price*amount) AS "총구매액",
       CASE
           WHEN (SUM(price*amount)  >= 1500) THEN '최우수고객'
           WHEN (SUM(price*amount)  >= 1000) THEN '우수고객'
           WHEN (SUM(price*amount) >= 1 ) THEN '일반고객'
           ELSE '유령고객'
       END AS "고객등급"
  FROM buytbl B
     RIGHT OUTER JOIN usertbl U
       ON B.userID = U.userID
  GROUP BY U.userID, U.name
  ORDER BY sum(price*amount) DESC ;
```

	userid [PK] character (8)	name character varying (10)	총구매액 bigint	고객등급 text
1	YJS	윤종신	[null]	유령고객
2	JKW	조관우	[null]	유령고객
3	LJB	임재범	[null]	유령고객
4	KKH	김경호	[null]	유령고객
5	LSG	이승기	[null]	유령고객
6	BBK	바비킴	1920	최우수고객
7	KBS	김범수	1210	우수고객
8	JYP	조용필	200	일반고객
9	EJW	은지원	95	일반고객
10	SSK	성시경	75	일반고객

WHILE과 CONTINUE/EXIT

- WHILE문
  - 다른 프로그래밍 언어의 WHILE과 동일한 개념
  - 해당 부울식이 참인 동안에 계속 반복되는 반복문

```
[ <<label>> ]
WHILE boolean-expression LOOP
    statements
END LOOP [ label ];
```

  - CONTINUE문을 만나면 WHILE문으로 이동해서 비교를 다시함
    - 다른 프로그래밍 언어의 Continue와 동일한 개념
  - EXIT문을 만나면 WHILE문을 빠져 나옴
    - 다른 프로그래밍 언어의 Break와 동일한 개념

[실습 11] WHILE문 실습

1. 1에서 100까지의 값을 모두 더하기

```
DROP PROCEDURE IF EXISTS whileProc;

CREATE PROCEDURE whileProc()
AS $$
DECLARE
    i INT; -- 1에서 100까지 증가할 변수
    hap INT; -- 더한 값을 누적할 변수
BEGIN
    i := 1;
    hap := 0;

    WHILE (i <= 100) LOOP
        hap := hap + i;  -- hap의 원래의 값에 i를 더해서 다시 hap에 넣으라는 의미
        i := i + 1;      -- i의 원래의 값에 1을 더해서 다시 i에 넣으라는 의미
    END LOOP;

    RAISE NOTICE '%',hap;
END $$
LANGUAGE plpgsql;

CALL whileProc();
```

```
알림:  5050
CALL
```

[실습 11] WHILE문 실습

2. CONTINUE문과 EXIT문 사용한 WHILE문 종료

```
CREATE PROCEDURE whileProc2()
as $$
DECLARE
    i INT; -- 1에서 100까지 증가할 변수
    hap INT; -- 더한 값을 누적할 변수
BEGIN
    i := 1;
    hap := 0;

    <<myWhile>>
    WHILE (i <= 100) LOOP  -- While문에 label을 지정
    IF (i%7 = 0) THEN
        i := i + 1;
        continue myWhile; -- 지정한 label문으로 가서 계속 진행
    END IF;

        hap := hap + i;
        IF (hap > 1000) THEN
        EXIT myWhile; -- 지정한 label문을 떠남. 즉, While 종료.
    END IF;
        i := i + 1;
    END LOOP;

    raise notice '%', hap;
END $$
LANGUAGE plpgsql;

CALL whileProc2();
```

```
알림:  1029
CALL
```

오류 처리

- 형식
  - Handler_statements
    - 오류 발생 시에 행동 정의
  - Condition : 어떤 오류를 처리할 것인지를 지정
    - PostgreSQL의 오류 코드 숫자가 오거나 SQLSTATE '상태코드', SQLEXCEPTION, SQLWARNING, NOT FOUND등이 올 수 있음

```
[ <<label>> ]
[ DECLARE
    declarations ]
BEGIN
    statements
EXCEPTION
    WHEN condition [ OR condition ... ] THEN
        handler_statements
    [ WHEN condition [ OR condition ... ] THEN
        handler_statements
      ... ]
END;
```

[실습 12] 오류처리 실습

1. 테이블이 없을 경우 오류 직접 처리

```
DROP PROCEDURE IF EXISTS errorProc;

CREATE PROCEDURE errorProc()
as $$
BEGIN
    SELECT * FROM noTable;  -- noTable은 없음.
    exception
        when others then
            raise notice '테이블이 없어요 ㅠㅠ';
END; $$
LANGUAGE plpgsql;

CALL errorProc();
```

```
알림:  테이블이 없어요 ㅠㅠ
CALL
```

동적 SQL

- PREPARE문
  - SQL문을 실행하지는 않고 미리 준비만 해놓음

- EXECUTE문
  - 준비한 쿼리문 실행
  - 실행 후에는 DEALLOCATE PREFARE로 문장 해제

[실습 13] 동적SQL 실습

1. 동적 SQL 예시(1)

```
PREPARE myQuery as SELECT * FROM usertbl WHERE userID = 'EJW';
EXECUTE myQuery;
DEALLOCATE PREPARE myQuery;
```

- SELECT * FROM usertbl WHERE userID = 'EJW' 문장을 바로 실행하지 않고 myQuery에 입력시켜 놓은 뒤 EXECUTE문으로 실행
- PREPARE문에서 ?으로 향후에 입력될 값을 비워놓고, EXECUTE에서는 USING을 이용해서 값을 전달하여 사용할 수 있음 (아래 예제)

2. 동적 SQL 예시(2)

```
DROP TABLE IF EXISTS myTable;

CREATE TABLE myTable (id serial PRIMARY KEY, mDate timestamp);
PREPARE myQuery(timestamp) as INSERT INTO myTable VALUES(default, $1);
EXECUTE myQuery(now());
DEALLOCATE PREPARE myQuery;
SELECT * FROM myTable;
```

쿼리를 실행하는 순간의 날짜와 시간이 입력되는 기능

	id [PK] integer	mdate timestamp without time zone
1	1	2022-07-25 16:04:36.307992