SQL로 데이터 다루기 2

01 집합 연산자 & 계층형 질의

/* elice */

목차

01

STANDARD SQL

관계형 대수

관계형 데이터베이스에서

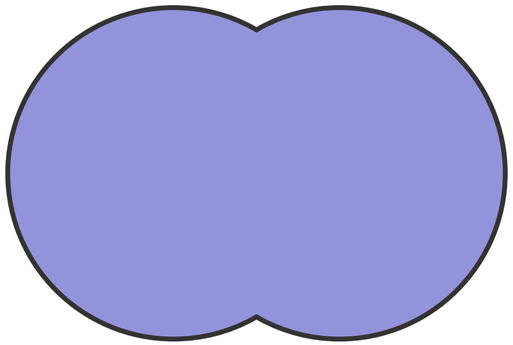
원하는 정보를 유도하기 위한 기본 연산 집합

일반 집합 연산

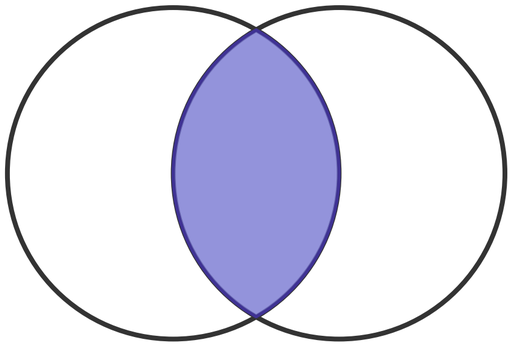
순수 관계 연산

일반 집합 연산

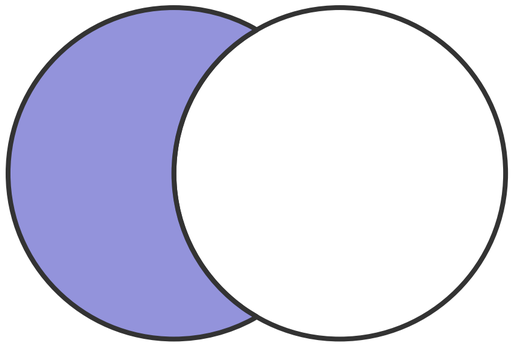
합집합



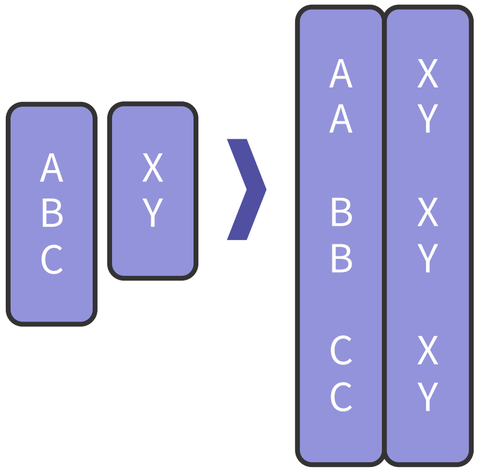
교집합



차집합



카디션 프로덕트



일반 집합 연산 별 SQL

A 릴레이션

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3

B 릴레이션

A	B	C
A1	B1	1
A4	B4	4

C 릴레이션

D
3
4

합집합

A∪B

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3
A4	B4	4

UNION

교집합

A∩B

A	B	C
A1	B1	1

INTERSECT

차집합

A - B

A	B	C
A2	B2	2
A3	B3	3

EXCEPT

카티션 곱

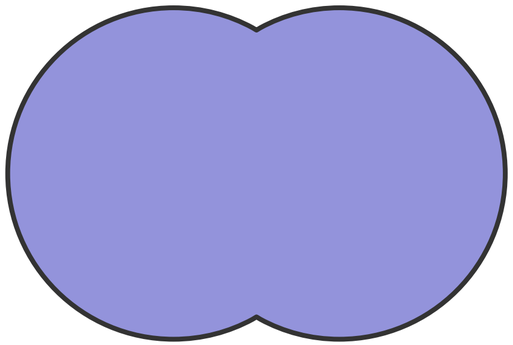
B X C

A	B	C	D
A1	B1	1	3
A1	B1	1	4
A4	B4	4	3
A4	B4	4	4

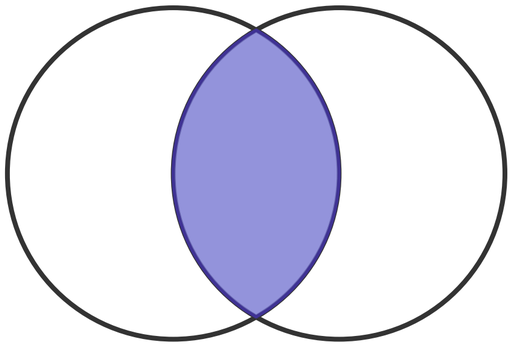
CROSS JOIN

일반 집합 연산

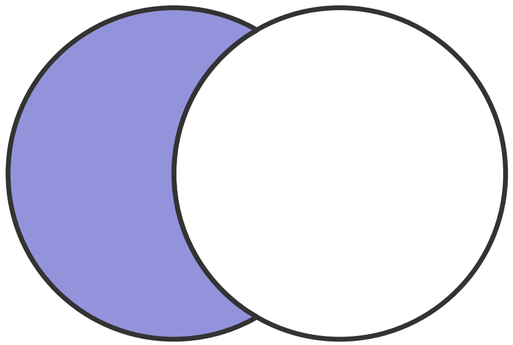
합집합



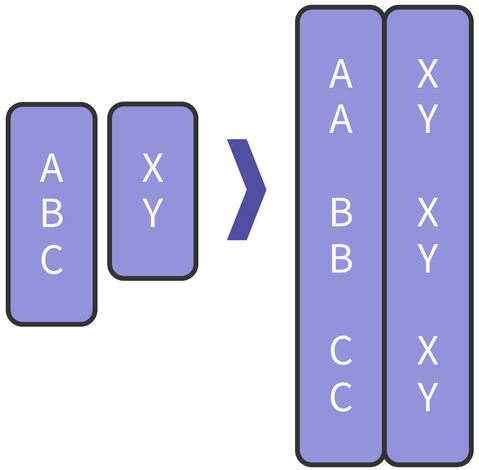
교집합



차집합



카디션 프로덕트



UNION

INTERSECT

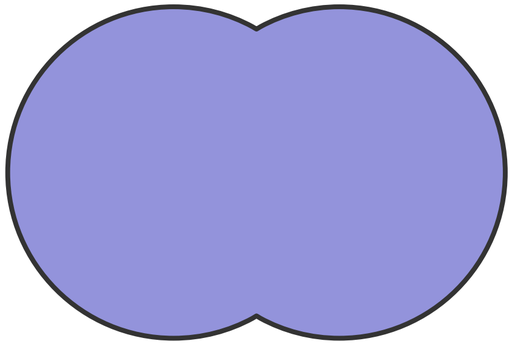
EXCEPT

CROSS JOIN

일반 집합 연산

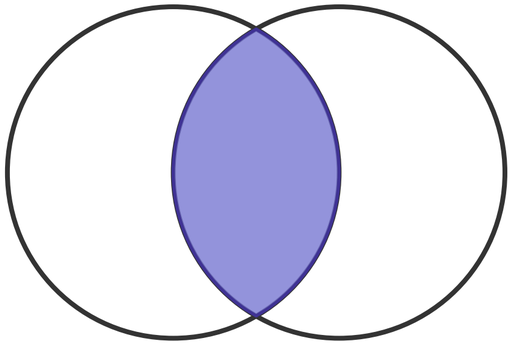
2장 – JOIN에서 다루자

합집합



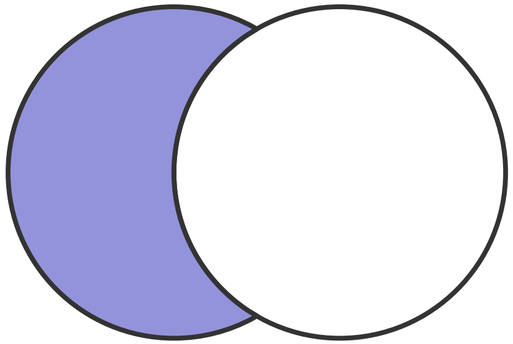
UNION

교집합



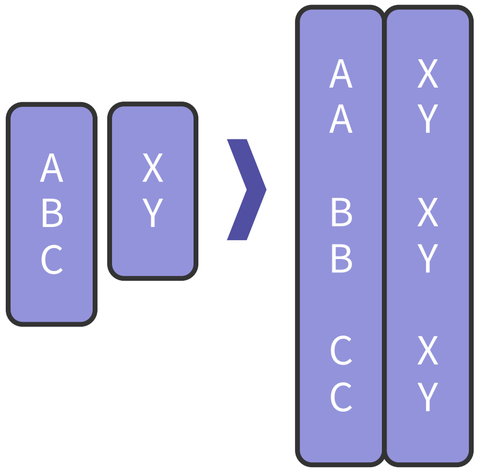
INTERSECT

차집합



EXCEPT

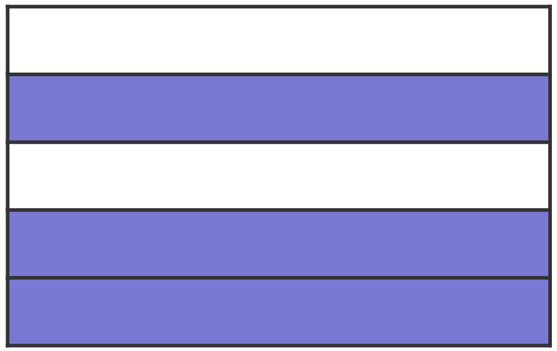
카디션 프로덕트



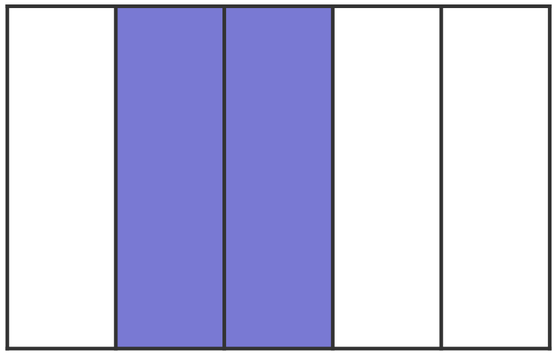
CROSS JOIN

순수 관계 연산

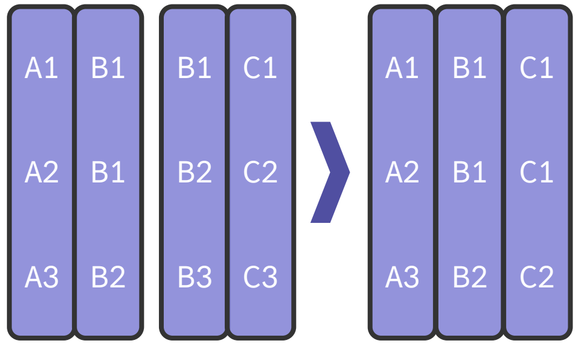
셀렉션



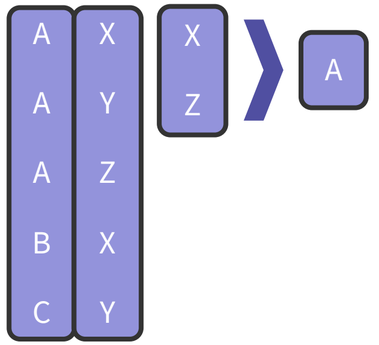
프로젝션



조인



디비전



순수 관계 연산 별 SQL

A 릴레이션

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3

B 릴레이션

A	B	C
A1	B1	1
A4	B4	4

C 릴레이션

A	D
A1	3
A4	4

D 릴레이션

A
A4

셀렉션

$\sigma_{C<3}(A)$

A	B	C
A1	B1	1
A2	B2	2

WHERE절

프로젝션

$\prod_{A,B}(A)$

A	B
A1	B1
A2	B2
A3	B3

SELECT절

조인

(세타조인, 동등조인, **자연조인**, 외부조인)

$B \bowtie_N C$

A	B	C	D
A1	B1	1	3
A4	B4	4	4

다양한 JOIN

디비전

$B \div D$

A	B	C
A4	B4	4

사용 X

02

집합 연산자

집합 연산자란?

두 개 이상의 테이블에서 **조인을 사용하지 않고**

연관된 데이터를 조회하는 방법 중 하나

테이블에서 SELECT한 **컬럼의 수와 각 컬럼의 데이터타입**이

테이블 간 **상호 호환** 가능해야 한다

UNION

UNION ALL

INTERSECT

EXCEPT

UNION

두 개의 테이블을 하나로 만드는 연산

UNION에 사용할 컬럼의 수와 데이터 형식이 일치해야 하며

합친 후에 테이블에서 **중복된 데이터는 제거**

이를 위해 UNION은 테이블을 합칠 때 **정렬 과정**을 발생시킴

(하지만 최종 결과에 대해 올바른 정렬을 위해서는 ORDER BY 구문을 사용해야함)

관계형 대수의 일반 집합 연산에서 **합집합**의 역할

UNION

ALPHA 테이블

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3

BETA 테이블

A	B	C
A1	B1	1
A4	B4	4

예시

```
SELECT * FROM ALPHA
UNION
SELECT * FROM BETA;
```

데이터 연결 및 정렬

A	B	C
A1	B1	1
A1	B1	1
A2	B2	2
A3	B3	3
A4	B4	4

중복제거

최종 결과

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3
A4	B4	4

UNION ALL

UNION과 거의 같은 기능을 수행

다만, UNION과 달리 **중복 제거와 정렬을 하지 않음**

중복 제거를 하지 않는다는 것이

제일 중요한 차이점!

UNION ALL

- 관계형 대수의 일반 집합 연산에서 합집합의 역할 +) 정렬X, 중복제거X

예시

```
SELECT * FROM ALPHA
UNION
SELECT * FROM BETA;
```

ALPHA 테이블

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3

BETA 테이블

A	B	C
A1	B1	1
A4	B4	4

데이터 연결

결과

A	B	C
A1	B1	1
A2	B2	2
A3	B3	3
A1	B1	1
A4	B4	4

INTERSECT

두 개의 테이블에 대해 겹치는 부분을 추출하는 연산

추출 후에는 **중복된 결과를 제거**

관계형 대수의 일반 집합 연산에서 **교집합**의 역할

INTERSECT

ALPHA 테이블

A	B	C
A1	B1	1
A2	B3	2
A3	B4	4

BETA 테이블

A	B	C
A1	B1	1
A1	B1	3

예시

```
SELECT A, B FROM ALPHA
INTERSECT
SELECT A, B FROM BETA;
```

겹치는 부분 추출

A	B
A1	B1
A1	B1
A1	B1

중복제거

최종 결과

A	B
A1	B1

INTERSECT

Oracle/Maria(실습환경) Database에서는 지원되지만,

MySQL에서는 지원되지 않기 때문에

추후 학습하는 JOIN 등을 활용해야 함.

EXCEPT(MINUS)

두 개의 테이블에서 겹치는 부분을

앞의 테이블에서 제외하여 추출하는 연산

추출 후에는 **중복된 결과를 제거**

관계형 대수의 일반 집합 연산에서 **차집합**의 역할

EXCEPT(MINUS)

ALPHA 테이블

A	B	C
A1	B1	1
A2	B3	2
A2	B3	4

BETA 테이블

A	B	C
A1	B1	1
A1	B3	4

예시

```
SELECT A, B FROM ALPHA
EXCEPT
SELECT A, B FROM BETA;
```

겹치는 부분 제외

A	B
A2	B3
A2	B3

중복제거

최종 결과

A	B
A2	B3

EXCEPT(MINUS)

OracleDatabase에서는 지원되지만,

실습 환경인 Maria DataBase에서는 10.3 version부터

EXCEPT 키워드로 지원함.

MySQL에서는 지원되지 않기 때문에

추후 학습하는 JOIN 등을 활용해야 함.

03

계층형 질의

계층형 질의란?

테이블에 **계층형 데이터가 존재**하는 경우

데이터를 조회하기 위해 사용하는 것

대표적인 데이터베이스

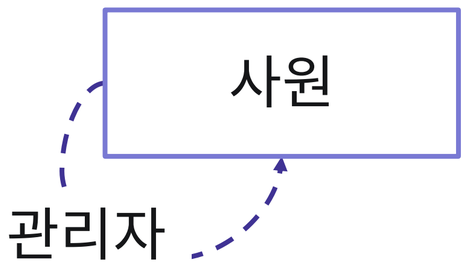
ORACLE

SQL Server

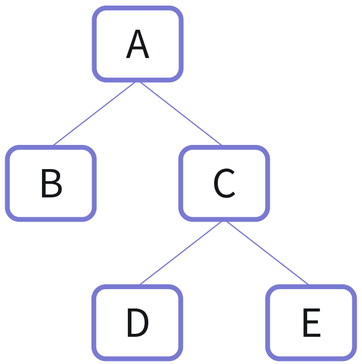
계층형 데이터

동일 테이블에 계층적으로 상위와 하위 데이터가
포함되어 있는 데이터

순환관계 데이터 모델



계층형 구조



샘플 데이터

사원	관리자
A	
B	A
C	A
D	C
E	C

계층형 질의 예시(Oracle)

Expression

```
SELECT LEVEL, 자식 컬럼, 부모 컬럼, 원하는 컬럼
FROM 테이블명
START WITH 부모 컬럼 IS NULL
CONNECT BY PRIOR 자식 컬럼 = 부모 컬럼;
```

부모 컬럼이 NULL인 행이 Root(가장 상위)가 됨

상위 데이터와 하위 데이터의 연결 방식

계층형 질의 예시(Oracle)

예시

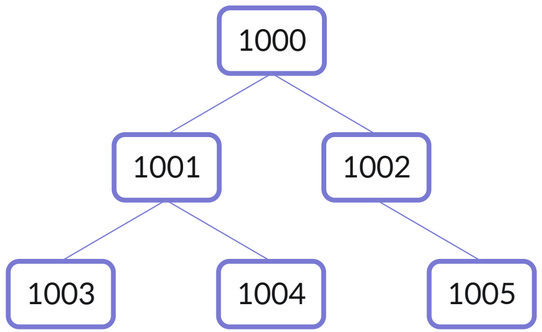
```
SELECT LEVEL, 사원번호, 관리자 FROM 직원

                                  상위 데이터          하위 데이터
START WITH 관리자 IS NULL CONNECT BY PRIOR 사원번호 = 관리자;
```

사원번호 1000이 Root가 됨

직원 테이블

사원번호	직급	관리자	연봉
1000	사장		8000
1001	대리	1000	2800
1002	대리	1000	2800
1003	사원	1001	2700
1004	사원	1001	2600
1005	사원	1002	2600

계층형 구조



결과값

LEVEL	사원번호	관리자
1	1000	
2	1001	1000
2	1002	1000
3	1003	1001
3	1004	1001
3	1005	1002

계층형 질의 예시(Oracle)

LPAD(‘ ’, n)은 왼쪽에 n자리의 공백 추가를 의미
Root는 LEVEL값이 1이기 때문에 4*(LEVEL-1) = 0 이 된다

예시

```
SELECT LEVEL, LPAD(‘ ’, 4*(LEVEL-1))||사원번호, 관리자

FROM 직원 START WITH 관리자 IS NULL CONNECT BY PRIOR 사원번호 = 관리자;
```

LEVEL 확인

LEVEL	사원번호	관리자
1	1000	
2	1001	1000
2	1002	1000
3	1003	1001
3	1004	1001
3	1005	1002

결과값

LEVEL	LPAD(‘ ’, 4*(LEVEL-1))||사원번호	관리자
1	1000	
2	1001	1000
2	1002	1000
3	1003	1001
3	1004	1001
3	1005	1002

CONNECT BY 키워드(Oracle)

키워드	설명
LEVEL	검색 항목의 깊이를 의미하며, 계층구조에서 루트(최상위)의 레벨이 1
CONNECT_BY_ROOT	현재 전개할 데이터의 루트(최상위) 데이터 값 표시
CONNECT_BY_ISLEAF	현재 전개할 데이터가 리프(최하위) 데이터 인지에 대한 값 표시(0 or 1)
SYS_CONNECT_BY_PATH(A, B)	루트 데이터부터 현재까지 전개한 경로 표시(A: 컬럼명, B: 구분자)

계층형 질의 예시(SQL Server)

SQL Server version.2000 이전

→ 저장 프로시저를 재귀 호출 / While 루프 문에서 임시테이블 사용

SQL Server version.2005 이후

→ CTE(Common Table Expression)을 이용하여 재귀 호출

계층형 질의 예시(SQL Server)

코드

```
WITH EMPLOYEES_ANCHOR AS ( SELECT EMPLOYEEID), LASTNAME, FIRSTNAME, REPORTSTO, 0 AS LEVEL
                           FROM EMPLOYEES
                           WHERE REPORTSTO IS NULL
                           UNION ALL
                           SELECT R.EMPLOYEEID, R.LASTNAME, R.FIRSTNAME, R.REPORTSTO, A.LEVEL + 1
                           FROM EMPLOYEES_ANCHOR A, EMPLOYEES R
                           WHERE A.EMPLOYEEID = R.REPORTSTO )

SELECT LEVEL, EMPLOYEEID, LASTNAME, FIRSTNAME, REPORTSTO
  FROM EMPLOYEES_ANCHOR GO;
```

계층형 질의 예시 (MySQL / MariaDB)

특정 버전 이후 (MariaDB의 경우 10.2 이후)

CTE(Common Table Expression)을 이용하여 재귀 호출

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드

```
WITH RECURSIVE CTE(member_id, manager_id, lvl)
AS (
    SELECT member_id, manager_id, 0 AS lvl
    FROM MEMBER
    WHERE manager_id IS NULL
    UNION ALL
    SELECT a.member_id, a.manager_id, b.lvl + 1
    FROM MEMBER a
    JOIN CTE AS b
    ON a.manager_id = b.member_id
)

SELECT member_id, manager_id, lvl
FROM CTE
ORDER BY member_id, lvl;
```

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1002	1001	2
1003	1002	3
1004	1002	3
1005	1000	1
1006	1005	2
1007	1006	3

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 1번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL;
```

member_id	manager_id	lvl
1000	NULL	0

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 1번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 1번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 2번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl
1001	1000	1
1005	1000	1

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 2번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1005	1000	1

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl
1001	1000	1
1005	1000	1

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 3번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1005	1000	1

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl
1001	1000	1
1005	1000	1
1002	1001	2
1006	1005	2

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 3번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1005	1000	1
1002	1001	2
1006	1005	2

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl
1001	1000	1
1005	1000	1
1002	1001	2
1006	1005	2

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 4번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1005	1000	1
1002	1001	2
1006	1005	2

member_id	manager_id	lvl
1000	NULL	0

UNION ALL

member_id	manager_id	lvl
1001	1000	1
1005	1000	1
1002	1001	1
1006	1005	2
1003	1002	2
1004	1002	3
1007	1006	3

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드 4번째 순환

```
SELECT member_id, manager_id, 0 AS lvl
FROM MEMBER
WHERE manager_id IS NULL

UNION ALL

SELECT a.member_id, a.manager_id, b.lvl + 1
FROM MEMBER a
JOIN CTE AS b
ON a.manager_id = b.member_id;
```

CTE table →

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1005	1000	1
1002	1001	1
1006	1005	2
1003	1002	2
1004	1002	3
1007	1006	3

계층형 질의 예시 (MySQL / MariaDB)

member_id	manager_id
1000	NULL
1001	1000
1002	1001
1003	1002
1004	1002
1005	1000
1006	1005
1007	1006

코드

```
WITH RECURSIVE CTE(member_id, manager_id, lvl)
AS (
    SELECT member_id, manager_id, 0 AS lvl
    FROM MEMBER
    WHERE manager_id IS NULL
    UNION ALL
    SELECT a.member_id, a.manager_id, b.lvl + 1
    FROM MEMBER a
    JOIN CTE AS b
    ON a.manager_id = b.member_id
)

SELECT member_id, manager_id, lvl
FROM CTE
ORDER BY member_id, lvl;
```

member_id	manager_id	lvl
1000	NULL	0
1001	1000	1
1002	1001	2
1003	1002	3
1004	1002	3
1005	1000	1
1006	1005	2
1007	1006	3

크레딧

/* elice */

코스 매니저
하주희

콘텐츠 제작자
하주희, 문범우

강사
문범우

감수자
장석준

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io