# 뷰

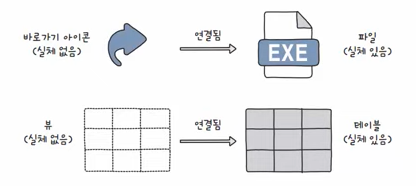
- 데이터베이스 개체 중 하나

- 뷰는 한번 생성해 놓으면 테이블이라고 생각하고 사용해도 될 정도로 사용자들의 입장에서는 테이블과 거의 동일한 개체로 취급

- 뷰는 테이블처럼 데이터를 가지고 있지는 않음

- 비유하자면 바탕 화면의 ‘바로 가기 아이콘’과 비슷함

- 단순 뷰는 하나의 테이블과 연관된 뷰, 복합 뷰는 2개 이상의 테이블과 연관된 뷰



- 뷰를 만드는 형식



- 뷰에 접근하는 방식



Ex)

USE market\_db;

CREATE VIEW v\_member

AS

SELECT mem\_id, mem\_name, addr FROM member;

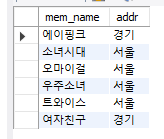
SELECT \* FROM v\_member;



Ex2)

SELECT mem\_name, addr FROM v\_member

WHERE addr IN ('서울', '경기');



- 뷰를 사용하는 이유

1. 보안에 도움이 됩니다

Ex) v\_member 뷰에는 사용자의 아이디, 이름, 주소만 있을 뿐 사용자의 중요한 개인 정보인 연락처, 평균 키, 데뷔 일자 등의 정보는 들어있지 않음

또한, 다른 예시로 인터넷 마켓 회원의 이름과 주소를 확인하는 작업을 진행하려고 합니다. 작업량이 많아서 아르바이트생에게 지시한다고 했을 때 아르바이트생이 회원 테이블에 접근 할 수 있도록 한다면 사용자의 중요한 개인 정보(연락처, 평균 키, 데뷔 일자 등)까지 모두 노출될 것입니다 이런 경우 아이디, 이름, 주소만 보이는 뷰를 생성해서 아르바이트생은 뷰에만 접근할 수 있도록 권한을 준다면 이러한 문제를 쉽게 해결할 수 있습니다.

2. 복잡한 SQL을 단순하게 만들 수 있음

만약 복잡한 쿼리문을 자주 사용해야 한다면, 뷰로 생성해 놓고 사용자들은 해당 뷰에만 접근하도록 하면 복잡한 SQL을 입력할 필요가 없어짐

Ex)

CREATE VIEW v\_memberbuy

AS

SELECT B.mem\_id, M.mem\_name, B.prod\_name, M.addr,

CONCAT(M.phone1, M.phone2) '연락처'

FROM buy B

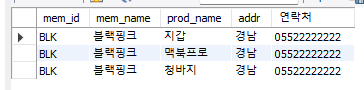
INNER JOIN member M

ON B.mem\_id = M.mem\_id;

SELECT \* FROM v\_memberbuy;



SELECT \* FROM v\_memberbuy WHERE mem\_name = '블랙핑크';



# 뷰의 실제 생성, 수정, 삭제

- 기본적인 뷰를 생성하면서 뷰에서 사용될 열 이름을 테이블과 다르게 지정 할 수 있음

- 기존에 배운 별칭을 사용하면 되는데, 중간에 띄어쓰기 사용 가능

- 별칭은 열 이름 뒤에 작은따옴표 또는 큰따옴표로 묶어주고, 형식상 AS를 붙여 줌

- 단 뷰를 조회할 때는 열 이름에 공백이 있으면 따옴표가 아닌 백틱( ` )으로 묶어줘야 함

**1. 뷰의 생성**

Ex)

USE market\_db;

CREATE VIEW v\_viewtest1

AS

SELECT B.mem\_id 'Member ID', M.mem\_name AS 'Member Name',

B.prod\_name "Product Name",

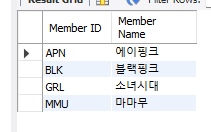
CONCAT(M.phone1, M.phone2) AS "Office Phone"

FROM buy B

INNER JOIN member M

ON B.mem\_id = M.mem\_id;

SELECT DISTINCT `Member ID`, `Member Name` FROM v\_viewtest1; -- 백틱을 사용



Ex2) 한글 이름은 권장하지 않지만 테이블 말고 뷰를 한글로 만들어준다

ALTER VIEW v\_viewtest1

AS

SELECT B.mem\_id '회원 아이디', M.mem\_name AS '회원 이름',

B.prod\_name "제품 이름",

CONCAT(M.phone1, M.phone2) AS "연락처"

FROM buy B

INNER JOIN member M

ON B.mem\_id = M.mem\_id;

SELECT DISTINCT `회원 아이디`, `회원 이름` FROM v\_viewtest1; -- 백틱을 사용

- 뷰를 생성할 때 CREATE VIEW는 기존에 뷰가 있으면 오류가 발생하지만, CREATE OR REPLACE VIEW는 기존에 뷰가 있어도 덮어쓰는 효과를 내기 때문에 오류가 발생하지 않음

즉, DORP VIEW와 CREATE VIEW를 연속으로 작성한 효과를 갖는다

Ex)

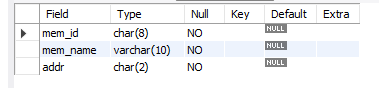
USE market\_db;

CREATE OR REPLACE VIEW v\_viewtest2

AS

SELECT mem\_id, mem\_name, addr FROM member;

DESCRIBE v\_viewtest2;



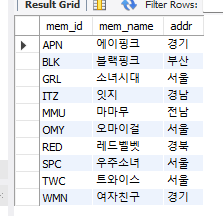
SHOW CREATE VIEW v\_viewtest2; // 뷰를 만든 구문을 살펴보고 싶을 때

**2. 뷰의 수정**

Ex)

UPDATE v\_member SET addr = '부산' WHERE mem\_id='BLK' ;

SELECT \* FROM v\_member;



\* 일반적으로 뷰에서 데이터를 삽입하는 것은 바람직하지 않다 -> 참조한 테이블에 걸려있는 제약 조건이 있을 수 있기 때문

Ex2) 뷰에 입력 조건을 걸어줄 때

ALTER VIEW v\_height167

AS

SELECT \* FROM member WHERE height >= 167

WITH CHECK OPTION ;

INSERT INTO v\_height167 VALUES('TOB','텔레토비', 4, '영국', NULL, NULL, 140, '1995-01-01') ; // 키가 167이상인 데이터만 체크해서 넣겠다는 조건이 걸려있기 때문에 에러

\* 뷰가 참조하고 있는 테이블도 삭제 할 수 있음

DROP TABLE IF EXISTS buy, member;

SELECT \* FROM v\_height167; // 참조하고 있는 테이블이 삭제 되었기 때문에 에러