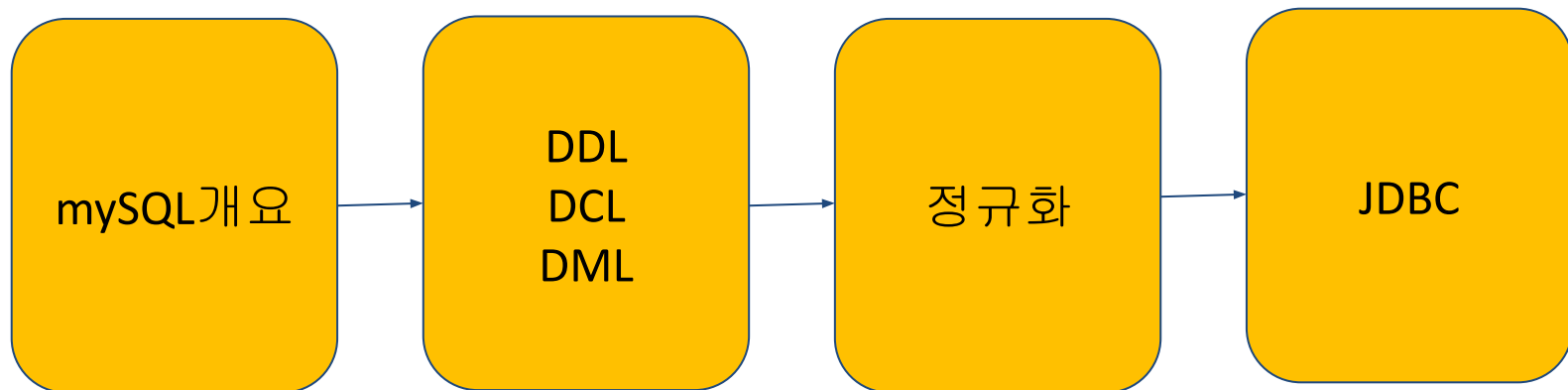


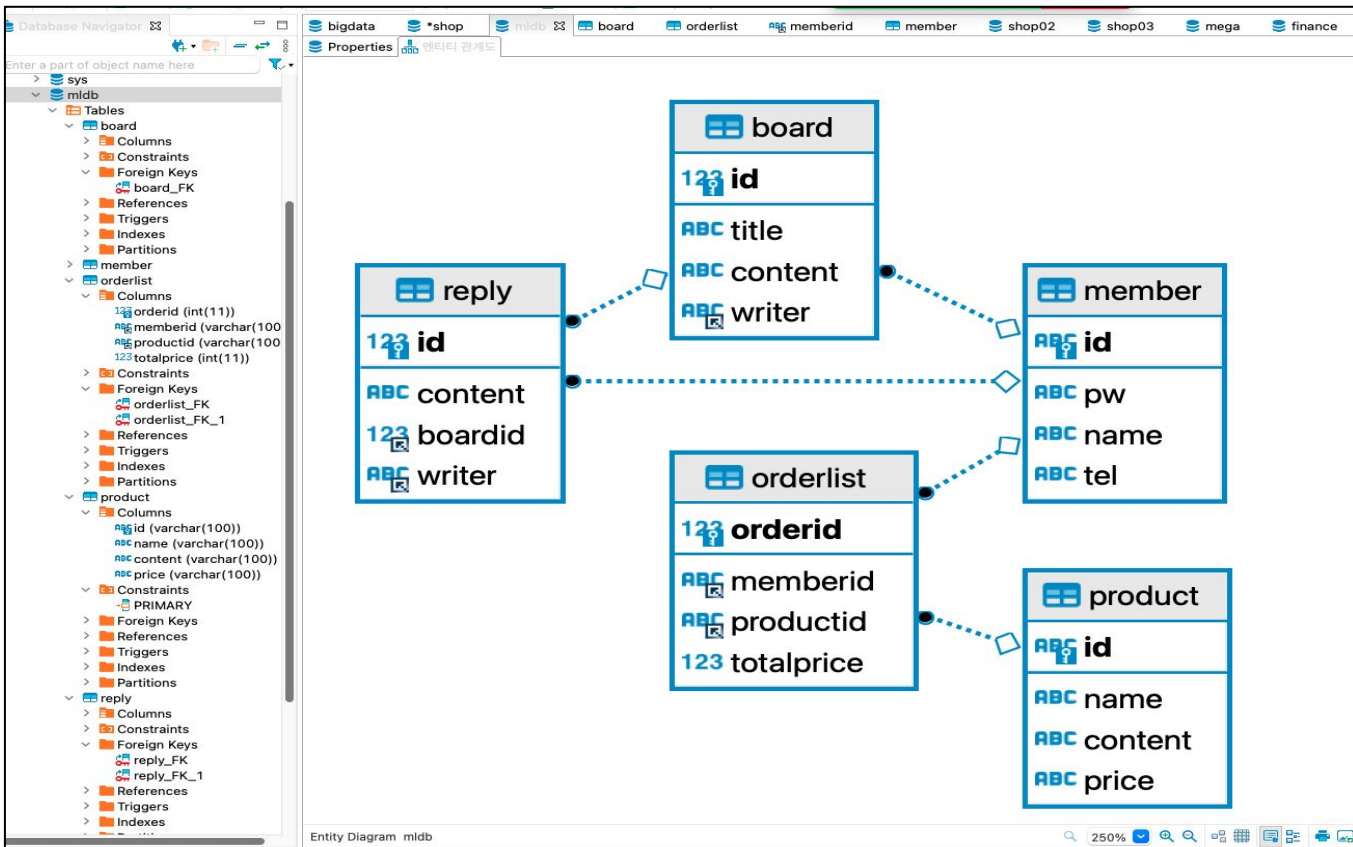
2024년 상반기 K-디지털 트레이닝

# 정규화 (Normalization)

---

[KB] IT's Your Life





## 지금까지 학습내용

- DBMS, RDBMS
- DB schema, relation, entity, attribute, instance
- DDL(create, alter, drop)
- DML(CRUD), DCL
- 기본함수, 집계함수, 그룹함수
- 집합, 조인

## 정규화(스키마가 문제가 없는 상태로 만드는 것)

**정규화 형태** 웹수집 +

**normal** form

**normal**

미국식 ['no:rməl] 🔊 영국식 ['no:ml] 🔊

1. 보통의, 평범한, 정상적인
2. (정신 상태가) 정상인
3. 보통, 평균, 정상

출처: 옥스퍼드 영한사전

BIGDATA - Daum 카페

en.dict.naver.com/#/search?range=word&query=정규화

ai-시원지 출석 수업 aws main this this ELK data-blog 파이썬 kagglebook 토익

**NAVER 사전** 파파고 참여번역 지식백과

사전용 영어 국어 한자 일본어 중국어 프랑스어 스페인어 독일어 U 베트남어

**영어사전**

odd

odd 이상한, 특이한, 이 이상한, 가끔의

Odd 오드

ODD 회귀의약품지정

oddly 이상하게, 이상하게도

odds 공산, 역경, 곤란, 배당률

odds and ends 잡동사니, 자질구레한 것들

정규화 +

NAVER 사전 파파고 참여번역 지식백과

사전종 영어 국어 한자 일본어 중국어 프랑스어 스페인어 독일어 U 베트남어 더보기 N 언어설정 한국어 -

국어사전 **[IT용어]무결성(integrity)** 상세 검색

전체 단어 속담·관용구 뜻풀이 예문 맞춤법 표기법 T T T

오른사전

## [IT용어]무결성(integrity)

검색어 트렌드

1. 데이터 및 네트워크 보안에 있어서 정보가 인가된 사람에 의해서 만이 접근 또는 변경 가능하다는 확실성. 무결성 대책은 네트워크 단말기와 서버의 물리적 환경 통제, 데이터 접근 억제 등의 엄격한 인가 관행을 유지하는 것이다. 한편, 데이터 무결성은 열, 먼지, 전기적 서지(surge)와 같은 환경적 해이에 의해 위협받을 수 있는데, 데이터 무결성의 물리적 환경 대책은 네트워크 관리자만의 서버 접근, 케이블 혹은 콘넥터와 같은 전송선의 외부 접촉 방지 대책, 전력선 서지, 정전기 방전, 자력으로로부터 하드웨어와 저장 장치들을 보호하는 것을 말하며, 데이터 무결성의 네트워크 관리 대책은 현재의 인가 수준을 모든 사람에게 유지시키고, 시스템 관리 절차, 관리 항목, 유지보수 사항을 문서화하며, 정전과 서버 장애, 바이러스 공격과 같은 불시의 재난에 대한 대비책을 세우는 것을 말한다.

cupl\*\*\*\* 2003-11-10 70 0

새로운 뜻을 추가해보세요! 뜻 추가하기 오른사전 더보기 >

**[IT용어]무결성(in...'** 의 전체 검색 결과 보기 >

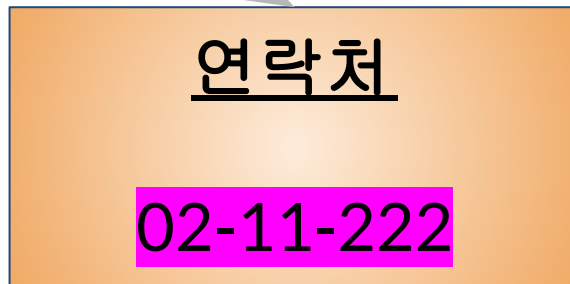
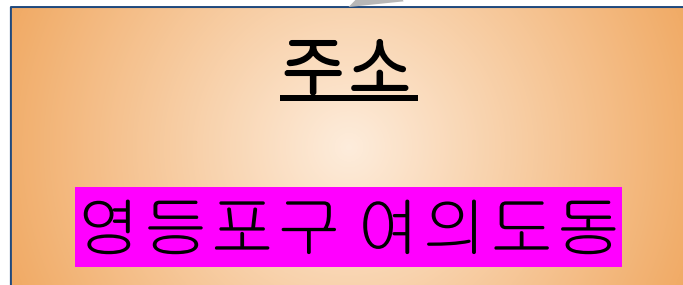
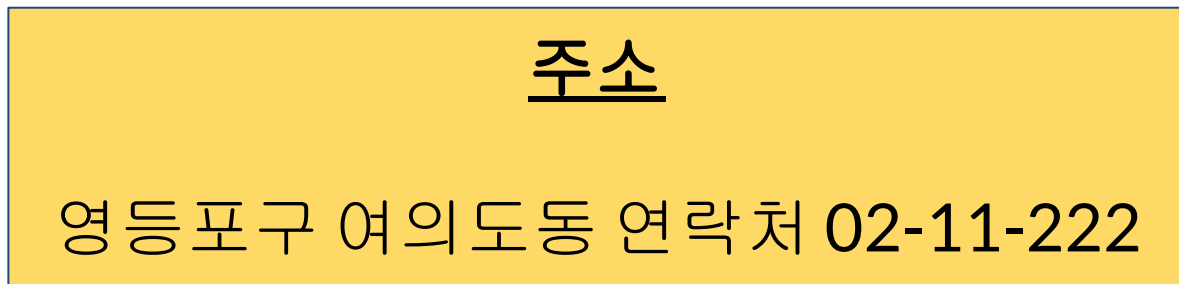
단어장 작은할 사전 오른사전

공지사항 N  
2019년 인기 신조어 · 화제의 단어 공개!  
전라북도 방언사전이 추가되었습니다!

너도 하드캐리 할 수 있어!  
**게임 용어 사전**  
**만들기 도전!**  
지금 바로가기 >

내가 찾은 단어  
[IT용어]무결성(integrity) X

자동저장 모두저장 모두삭제



대부분의 릴레이션은 BCNF까지 정규화하면 실제적인 **이상현상**이 없어지기 때문에 보통 3NF 또는 BCNF까지 **정규화** 진행함.

정규화 되거나 되지 않은 모든 릴레이션

1NF 릴레이션

2NF 릴레이션

3NF 릴레이션

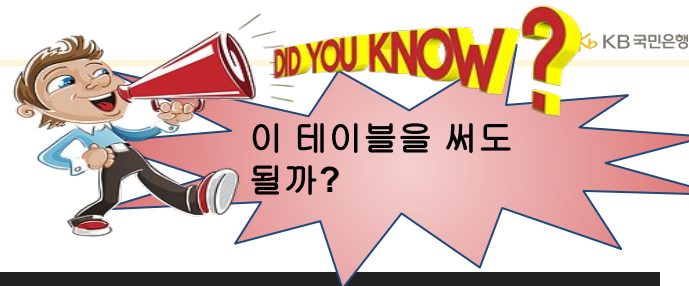
BCNF 릴레이션

4NF 릴레이션

5NF (PJ/NF) 릴레이션



## 테이블 생성(학생수강성적 테이블)



학생번호	학생이름	주소	학과	학과사무실	강좌이름	강의실	성적
501	박지성	맨체스터	컴퓨터과	공학관101	데이터베이스	공학관110	3.5
401	김연아	서울	체육학과	체육관101	데이터베이스	공학관110	4.0
402	장미란	강원도	체육학과	체육관101	스포츠경영학	체육관103	3.5
502	주신수	플리블랜드	컴퓨터과	공학관101	자료구조	공학관111	4.0
501	박지성	맨체스터	컴퓨터과	공학관101	자료구조	공학관111	3.5

## 삽입이상 (Insertion Anomaly)

- 새로운 데이터를 삽입할 때 불필요하게 다른 관련 데이터도 함께 삽입해야 하는 경우

### - 예제 1

- 새로운 학생 503번 (홍길동)이 수강하려고 할 때, 강좌 정보가 없으면 학생 정보를 삽입할 수 없음.

```
INSERT INTO 학생 (학생번호, 학생이름, 주소, 학과) VALUES (503, '홍길동', '부산', '컴퓨터과');
```

→ 강좌 정보 없이 학생 정보를 삽입할 수 없음.

### - 예제 2

- 새로운 강좌 알고리즘을 개설하려고 할 때, 이 강좌를 수강하는 학생 정보가 없으면 강좌 정보를 삽입할 수 없음.

```
INSERT INTO 강좌 (강좌이름, 강의실) VALUES ('알고리즘', '공학관112');
```

→ 학생 정보 없이 강좌 정보를 삽입할 수 없음.

## 삭제이상 (Deletion Anomaly)

- 데이터베이스에서 특정 데이터를 삭제할 때 불필요하게 다른 관련 데이터까지 함께 삭제되는 현상

### - 예제 1

- 만약 강좌이름 = '데이터베이스' 강좌가 폐지되어 해당 강좌 정보를 삭제하려고 할 때, 이 강좌를 수강하는 학생 정보도 함께 삭제됨.

`DELETE FROM 학생_강좌 WHERE 강좌이름 = '데이터베이스';`

→ 학생 김연아와 박지성의 강좌 정보가 함께 삭제되며, 이들의 성적 기록도 사라짐.

### - 예제 2

- 만약 학생 401번 (김연아)이 졸업하여 해당 학생 정보를 삭제하려고 할 때, 이 학생이 수강한 데이터베이스 강좌 정보도 삭제됨.

`DELETE FROM 학생 WHERE 학생번호 = 401;`

→ 강좌 데이터베이스의 성적 정보가 삭제

## 수정이상 (Update Anomaly)

- 특정 데이터를 수정할 때 동일한 정보가 여러 곳에 중복되어 있어 일관성 문제가 발생하는 경우

### - 예제 1

- 학생 501번 (박지성)의 주소를 맨체스터에서 서울로 변경하려고 할 때, 여러 테이블에 중복된 정보가 있으면 모든 정보를 일관되게 수정해야 함.

```
UPDATE 학생 SET 주소 = '서울' WHERE 학생번호 = 501;
```

→ 학생 박지성의 주소 정보가 일관되지 않게 됨.

### - 예제 2

- 컴퓨터과 학과 사무실을 공학관101에서 공학관102로 변경하려고 할 때, 여러 테이블에 중복된 정보가 있으면 모든 정보를 일관되게 수정해야 함.

```
UPDATE 학과 SET 학과사무실 = '공학관102' WHERE 학과 = '컴퓨터과';
```

→ 학과 사무실 정보가 일관되지 않게 됨.



- 왜? 문제가 있음.
- 정규화되지 않음.
- 이상현상 (종류: 삽입이상, 삭제이상, 갱신이상)
- 결점이있기 때문에. 어떤 결점?
- "중복"이 되었기 때문에. => 해결책: 중복을 안하게. => "테이블/컬럼 분리"! => 정규화하는 과정(1, 2, 3단계) => 제1정규화, 제2정규화, 제3정규화
- 검색시 분리된 테이블을 합해야 함.  
→ join/서브쿼리/집합이 필요



삭제이상  
⇒ 연쇄삭제

삽입이상  
⇒ null값 문제

수정이상  
⇒ 정보의  
불일치 문제

## 데이터베이스 정규화

- 데이터베이스 설계 시 데이터를 구조화하여 데이터 중복을 최소화  
데이터의 무결성을 유지하며, 효율적인 데이터 처리를 가능하게 하는 방법
- 정규화의 주요 목표
  - **데이터 중복 최소화**: 동일한 데이터가 여러 장소에 저장되는 것을 방지
  - **데이터 무결성 유지**: 데이터의 일관성과 정확성을 보장
  - **효율적인 데이터 관리**: 데이터 삽입, 삭제, 갱신 시 이상 현상을 방지

## 제1정규형 → 제2정규형 → 제3정규형



정규형	설명	예시
제1정규형 (1NF)	모든 필드가 원자값이어야 함	각 열에 단일 값만 존재
제2정규형 (2NF)	불완전 종속 속성이 없어야 함	기본 키의 모든 부분에 대해 종속성 존재
제3정규형 (3NF)	이행 종속 속성이 없어야 함	비기본 키 속성 간의 종속성 없음



- 데이터베이스에서 키(Key)는 데이터를 식별하고 테이블의 무결성을 유지하는 중요한 요소

## 1. 기본키 (Primary Key)

- 정의: 테이블 내의 각 행(Row)을 고유하게 식별하는 하나의 또는 여러 개의 속성(Attribute)의 집합.
- 특징:
  - 유일한 값을 가짐 (중복 값 없음).
  - NULL 값을 가질 수 없음.
  - 각 테이블에는 하나의 기본키만 존재할 수 있음.

-

## 2. 후보키 (Candidate Key)

- **정의:** 테이블 내의 행을 고유하게 식별할 수 있는 하나 이상의 속성 또는 속성의 집합. 기본키가 될 수 있는 후보들.
- **특징:**
  - 유일한 값만을 가짐 (중복 값 없음).
  - NULL 값을 가질 수 없음.
  - 테이블에는 여러 개의 후보키가 존재할 수 있음.
  - 기본키는 후보키 중에서 선택됨

## 3. 대체키 (Alternate Key)

- **정의:** 기본키로 선택되지 않은 후보키.
- **특징:**
  - 기본키와 같은 특성을 가지지만, 기본키로 사용되지 않는 키.

## 4. 복합키 (Composite Key)

- **정의:** 두 개 이상의 속성으로 구성된 키. 단일 속성이 아닌 여러 속성을 결합하여 고유성을 보장.
- **특징:**
  - 결합된 속성의 조합이 유일성을 보장함.
  - 하나의 속성만으로는 유일성을 보장할 수 없는 경우 사용.

## 5. 외래키 (Foreign Key)

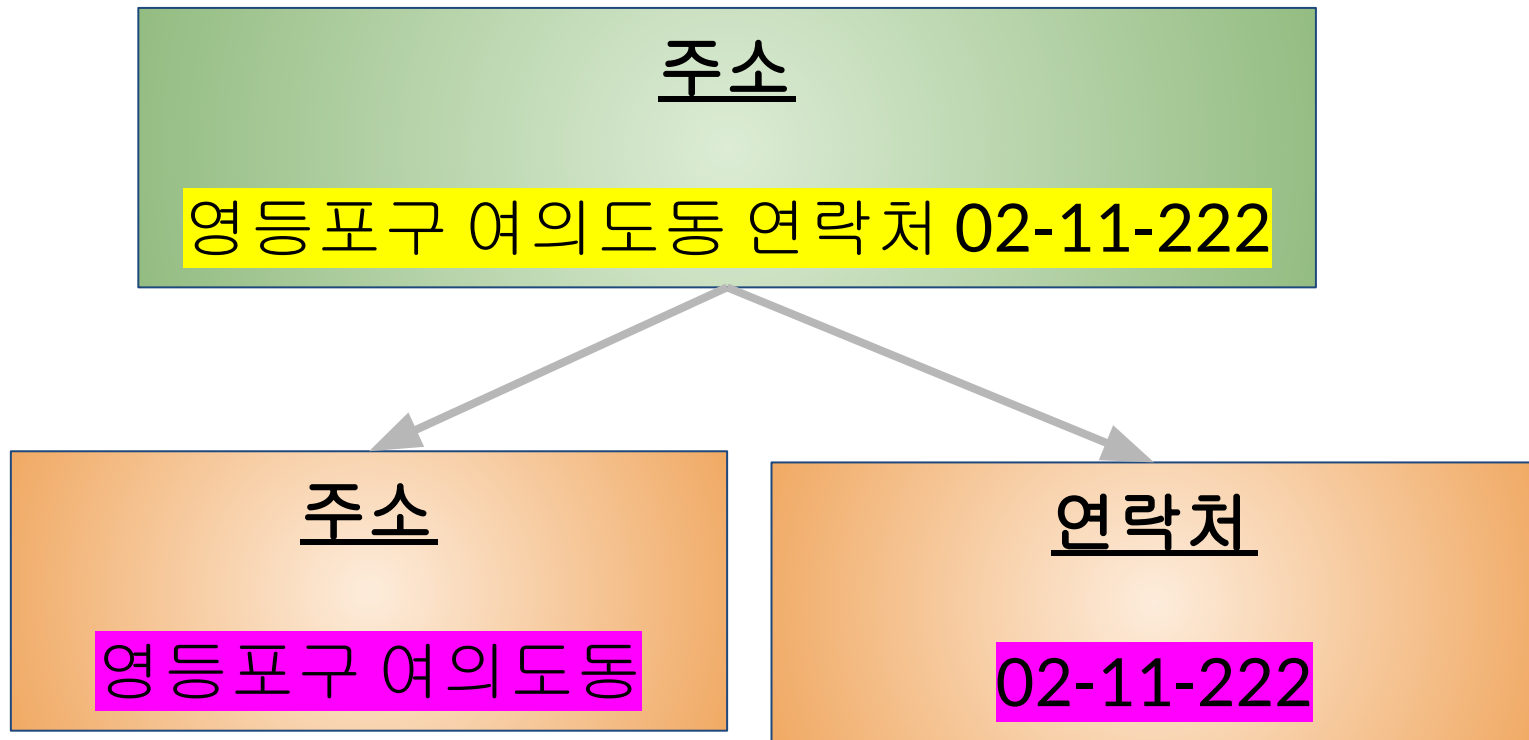
- **정의:** 다른 테이블의 기본키를 참조하는 속성. 두 테이블 간의 관계를 나타냄.
- **특징:**
  - 참조 무결성을 유지함.
  - 참조하는 값은 **NULL**이거나 참조된 테이블의 기본키 값 중 하나여야 함.
  - 외래키 제약 조건을 통해 데이터의 일관성을 유지.

### 6. 유일키 (Unique Key)

- **정의:** 테이블 내의 각 행을 고유하게 식별할 수 있는 키. 기본키와 유사하지만, **NULL** 값을 가질 수 있음.
- **특징:**
  - 유일한 값을 가짐 (중복 값 없음).
  - **NULL** 값을 하나 이상 가질 수 있음.
  - 하나의 테이블에 여러 개의 유일키가 존재할 수 있음.



- **pk는 반드시 있어야 한다?(x)**
  - pk를 만들어주는 것을 권장
- **pk는 컬럼을 반드시 하나만 써야한다.(x)**
  - pk는 컬럼이 하나 이상될 수 있다.
  - 복합키가 pk가 될 수 있다.
  - idx, 학생번호, 강좌이름, 성적
  - idx(pk)



## 제 1 정규형

- 테이블의 모든 필드가 원자값(atomic value)을 가져야 함.
- 즉, 각 열에는 단일 값만 포함되어야 하며, 반복되는 그룹이나 다중 값이 존재해서는 안된다.
  - 원자성: 각 열은 더 이상 쪼갤 수 없는 값이어야 함.
  - 중복 제거: 중복된 데이터가 없어야 함.

수학만  
취소하고자 하는  
경우 문제가  
생김

원본 학생 테이블

학생ID	이름	과목
1	홍길동	수학, 과학
2	이순신	영어

제1정규형 적용 후 학생 테이블

학생ID	이름	과목
1	홍길동	수학
1	홍길동	과학
2	이순신	영어

## 데이터 분석 테이블과 RDB테이블 비교 (df)

- 데이터 분석시 **df(dataframe, 데이터프레임, -I 모양)**
- 앞의 변수들이 뒤에 나오는 타겟을 결정하게 된다.
  - 타겟이 앞에 있는 변수들에 종속된다.
  - 앞의 변수들은 타겟을 설명하는 역할의 변수가 된다.
  - 앞의 변수들: 설명변수
  - 뒤에 나오는 타겟 변수 : 종속변수
  - df의 튜닝의 대상이 되는 변수들은 일반적으로 설명변수가 된다.



## 데이터 분석과 RDB 종속성 차이

데이터분석  
dataframe

	state	color	food	age	height	score
Jane	NY	blue	Steak	30	165	4.6
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0
Penelope	AL	white	Apple	4	80	3.3
Dean	AK	gray	Cheese	32	180	1.8
Christina	TX	black	Melon	33	172	9.5
Cornelia	TX	red	Beans	69	150	2.2

RDB  
table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DE
7,369	SMITH	CLERK	7,902	1980-12-17	800	[NULL]	
7,499	ALLEN	SALESMAN	7,698	1981-02-20	1,600	300	
7,521	WARD	SALESMAN	7,698	1981-02-22	1,250	500	
7,566	JONES	MANAGER	7,839	1981-04-02	2,975	[NULL]	

## 데이터 분석 테이블과 RDB테이블 비교

- database에서 table을 릴레이션이라고 부르기도 한다.
- table에는 일반적으 pk를 만들어주는 편.
- pk에 따라 각 행들은 다른 값을 가지게 된다.
- pk가 학생 id 100이라고 한다면 다른 컬럼들은 학생id 100에 해당하는 컬럼들의 값을 가지게 됨.
- 학생 id 200번의 값들이 100번의 행에 올 수 없다.
- pk가 다른 컬럼들을 종속시킨다.

- 종속성 (dependency)은 한 속성의 값이 다른 속성의 값에 의해 결정되는 관계를 의미
- 종속성은 데이터베이스 정규화 과정에서 매우 중요한 개념으로, 다양한 형태의 종속성이 존재
- 종속성 종류

종류	설명	예시
함수 종속성	한 속성의 값이 다른 속성의 값에 의해 결정됨	학생ID -> 이름
부분 함수 종속성	복합 키의 일부에 의해서만 결정되는 종속성	학생ID -> 학생이름 (복합 키: 학생ID, 과목)
이행 함수 종속성	한 속성이 기본 키가 아닌 다른 속성을 통해 결정됨	학생ID -> 주소, 주소 -> 지역 (간접 종속)

## - 함수 종속성 (Functional Dependency)

- 기본적으로 한 속성의 값이 다른 속성의 값에 의해 고유하게 결정되는 관계를 나타냄.
- 함수 종속성은  $A \rightarrow B$ 로 표현되며, 이는 "A가 B를 함수적으로 결정한다"는 의미
- 예시:

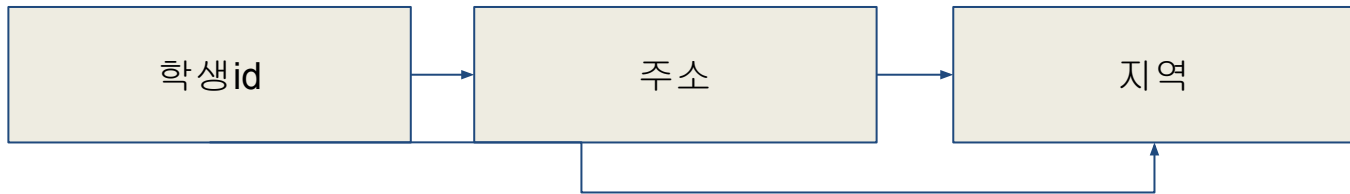
학생 테이블에서  $\text{학생ID} \rightarrow \text{이름}$ 이라는 함수 종속성이 있다면,  
학생ID가 결정되면 그 학생의 이름도 유일하게 결정됨.

## - 부분 함수 종속성 (Partial Dependency)

- 부분 함수 종속성은 복합 키(Composite Key)의 일부에 의해서만 결정되는 종속성을 의미
- 제2정규형(2NF)을 위반하게 됨.
- 부분 함수 종속성을 제거하기 위해서는 테이블을 분리하여 각 부분이 전체 키에 완전히 종속되도록 해야함.
- 예시:
  - 성적 테이블에서 (학생ID, 과목) -> 점수라는 복합 키가 있을 때, 만약 학생ID -> 학생이름이 존재하면 이는 부분 함수 종속성임. 왜냐하면 학생이름은 학생ID에만 종속되기 때문. 이를 제거하려면 학생 정보를 별도의 테이블로 분리해야 함.

## - 이행 함수 종속성 (Transitive Dependency)

- 한 속성이 기본 키가 아닌 다른 속성을 통해서 간접적으로 결정되는 종속성을 의미
- 제3정규형(3NF)을 위반하게 됨.
- 이행 함수 종속성을 제거하기 위해서는 중간 속성을 별도의 테이블로 분리
- 예시:
  - 학생 정보 테이블에서 **학생ID** -> 주소와 주소 -> 지역이라는 두 종속성이 있을 때, **학생ID** -> 지역은 이행 함수 종속성임. 이를 제거하려면 주소와 지역 정보를 별도의 테이블로 분리해야 함.



## 제2정규형 기본키에 의해 완전종속

- 기본키가 단일 속성인 경우와 복합키인 경우 모두 제2정규형의 적용을 이해하는 것이 중요
- 기본키가 단일 속성인 경우, 모든 비기본 속성은 기본키에 대해 완전 종속될 수밖에 없으므로 제2정규형이 자동으로 만족
- 기본키가 복합키인 경우에는 비기본 속성이 기본키의 일부에만 종속되지 않고 전체에 종속되어야 제2정규형을 만족하게 됨.
- 복합키인 경우, 다른 컬럼들은 복합키에 의해 완전히 결정이 되어야 한다.(완전 종속)  
=> 부분종속이 존재하는 경우, 중복현상이 나타남.
- 1NF를 만족하면서, 부분 함수 종속성(partial dependency)이 없어야 함.
- 즉, 기본 키의 일부만으로 종속되는 속성이 없어야 함.
- 기본 키의 모든 부분이 속성에 영향을 미쳐야 함.
- 부분 종속성 제거: 기본 키의 일부에만 종속되는 속성을 제거

**create table test(**

**user\_id varchar(256) not null,**

**user\_pw varchar(256) not null,**

**primary key(user\_id, user\_pw)**

**)**



원본 성적 테이블

학생ID	이름	과목	점수
1	홍길동	수학	90
1	홍길동	과학	85
2	이순신	영어	95

- 복합키인 학생ID, 과목이 점수를 종속시킴.

제2정규형 적용 후

학생 테이블

학생ID	이름
1	홍길동
2	이순신

성적 테이블

학생ID	과목	점수
1	수학	90
1	과학	85
2	영어	95

- 복합키중 학생ID가 이름을 종속시킴.

### 제3정규형 이행종속이 있으면 안된다.

- 데이터베이스 정규화의 세 번째 단계로, 제2정규형(2NF)을 만족하면서, 모든 비기본 키 속성이 기본 키에 비이행적(비트랜지티브)으로 종속되도록 하는 것을 의미
- 제2정규형 (2NF)을 만족해야 한다.
  - 즉, 테이블이 제1정규형(1NF)을 만족하면서 모든 비기본 키 속성이 기본 키 전체에 종속되어야 함. 부분 함수 종속성(기본 키의 일부에만 종속되는 속성)이 없어야 함.
- 이행 함수 종속성 (Transitive Dependency)이 없어야 한다.
  - 이는 비기본 키 속성이 다른 비기본 키 속성에 종속되지 않아야 한다는 의미.  
비기본 키 속성은 기본 키에 직접적으로만 종속되어야 함. 기본 키를 통해 간접적으로 종속되는 경우는 허용되지 않음.

- 교수실은 교수이름에 종속되어 있고, 교수이름은 학생ID와 과목에 종속되어 있음.  
교수실은 학생ID와 과목에 이행적으로 종속됩니다. 이를 제거하기 위해 테이블을 분리해야 함.

학생ID	이름	과목	점수	교수이름	교수실
1	홍길동	수학	90	김명수	101호
1	홍길동	과학	85	이정민	202호
2	이순신	영어	95	박지훈	303호
2	이순신	수학	80	김명수	101호

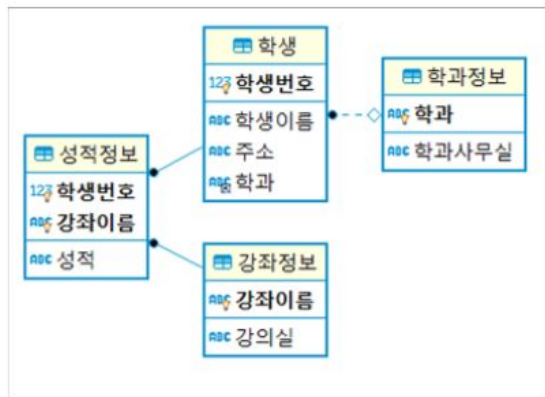
```
CREATE TABLE Student ( 학생ID
INT PRIMARY KEY, 이름
VARCHAR(50) );
```

```
CREATE TABLE Grades (
학생ID INT,
과목 VARCHAR(50),
점수 INT,
PRIMARY KEY (학생ID, 과목),
FOREIGN KEY (학생ID) REFERENCES Student(학생ID)
);
```

```
CREATE TABLE Professor (
교수이름 VARCHAR(50) PRIMARY KEY,
교수실 VARCHAR(50)
);
```

학생수강성적

학생번호	학생이름	주소	학과	학과사무실	강좌이름	강의실	성적
501	박지성	영국 맨체스터	컴퓨터과	공학관101	데이터베이스	공학관 110	3.5
401	김연아	대한민국 서울	체육학과	체육관101	데이터베이스	공학관 110	4.0
402	장미란	대한민국 강원도	체육학과	체육관101	스포츠경영학	체육관 103	3.5
502	추신수	미국 클리블랜드	컴퓨터과	공학관101	자료구조	공학관 111	4.0
501	박지성	영국 맨체스터	컴퓨터과	공학관101	자료구조	공학관 111	3.5



123 학생번호	ABC 학생이름	ABC 주소	ABC 학과
502	추신수	미국 클리블랜드	컴퓨터과
501	박지성	영국 맨체스터	컴퓨터과
402	장미란	대한민국 강원도	체육학과
401	김연아	대한민국 서울	체육학과

ABC 강좌이름	ABC 강의실
데이터베이스	공학관110
스포츠경영학	체육관103
자료구조	공학관111

123 학생번호	ABC 강좌이름	ABC 성적
501	데이터베이스	3.5
401	데이터베이스	4.0
402	스포츠경영학	3.5
502	자료구조	4.0
501	자료구조	3.5

ABC 학과	ABC 학과사무실
컴퓨터과	공학관101
체육학과	체육관101

```
use 학생수강성적;
```

```
select 학생.학생번호, 학생이름, 주소, 학생.학과, 학과사무실, 강좌정보.강좌이름, 강좌정보.강의실, 성적정보.성적 from 학생
inner join 학과정보 on 학과정보.학과 = 학생.학과
inner join 성적정보 on 성적정보.학생번호 = 학생.학생번호
inner join 강좌정보 on 강좌정보.강좌이름 = 성적정보.강좌이름;
```

	123 학생번호	abc 학생이름	abc 주소	abc 학과	abc 학과사무실	abc 강좌이름	abc 강의실	abc 성적
1	401	김연아	대한민국 서울	체육학과	체육관101	데이터베이스	공학관110	4.0
2	501	박지성	영국 맨체스타	컴퓨터과	공학관101	데이터베이스	공학관110	3.5
3	402	장미란	대한민국 강원도	체육학과	체육관101	스포츠경영학	체육관103	3.5
4	501	박지성	영국 맨체스타	컴퓨터과	공학관101	자료구조	공학관111	3.5
5	502	추신수	미국 클리블랜드	컴퓨터과	공학관101	자료구조	공학관111	4.0

123 학생번호	abc 학생이름	abc 주소	abc 학과
502	추신수	미국 클리블랜드	컴퓨터과
501	박지성	영국 맨체스터	컴퓨터과
402	장미란	대한민국 강원도	체육학과
401	김연아	대한민국 서울	체육학과

학생
학생번호(PK)
학생이름
주소
학과

abc 강좌이름	abc 강의실
데이터베이스	공학관110
스포츠경영학	체육관103
자료구조	공학관111

강좌정보
강좌이름(PK)
강의실

123 학생번호	abc 강좌이름	abc 성적
501	데이터베이스	3.5
401	데이터베이스	4.0
402	스포츠경영학	3.5
502	자료구조	4.0
501	자료구조	3.5

성적정보
학생번호(PK)
강좌이름(PK)
성적

abc 학과	abc 학과사무실
컴퓨터과	공학관101
체육학과	체육관101

학과정보
학과(PK)
학과사무실

- 정규화가 왜 필요한가 ? 이상현상이 발생
- 정규화하지 않으면 나타나는 현상 ? 이상현상
- 언제 이상현상이 발생하는가 ? 삽입이상 , 갱신이상 , 삭제이상
- 이상현상이 왜 나타나는가 ? 중복
- 이상현상을 해결하는 방법은 ? 분할
- 분할해 나가는 과정은 ? 정규화 과정
- 정규화순서
  - 1) 제1정규화 -> 2) 제2정규화 -> 3) 제3정규화
    1. 원자값
    2. 완전종속 (부분종속 x)
    3. 이행종속 X
- 데이터를 추출할 때는 분할해놓은 테이블들을 join해서 가져고 와야 한다.
  - on join시 기준이 되는 컬럼을 지정
  - 기준에 해당하는 값들이 양쪽 테이블에 있는 경우 : inner join

- 릴레이션의 모든 속성 값이 원자 값을 가져야 함.
- 아래 테이블에서는 모든 속성 값이 원자 값으로 되어 있으므로 제 1 정규화 과정이 필요 없음

학생번호	학생이름	주소	학과	학과사무실	강좌이름	강의실	성적
501	박지성	맨체스타	컴퓨터과	공학관101	데이터베이스	공학관 110	3.5
401	김연아	서울	체육학과	체육관101	데이터베이스	공학관 110	4.0
402	장미란	강원도	체육학과	체육관101	스포츠경영학	체육관 103	3.5
502	추신수	클리블랜드	컴퓨터과	공학관101	자료구조	공학관 111	4.0
501	박지성	맨체스타	컴퓨터과	공학관101	자료구조	공학관 111	3.5



제 1 정규형을 만족하면서, 부분집합 종속성을 제거해 이상현상을 방지하고 완전 종속을 만족해야 함

학생번호	학생이름	주소	학과	학과 사무실
401	김연아	서울	체육	체육관101
402	장미란	강원도	체육	체육관101
501	박지성	맨체스터	컴퓨터과	공학관101
502	추신수	클리블랜드	컴퓨터과	공학관101

완전 종속이므로  
제2정규화 과정  
필요없음.

학생이름과 주소, 학과는 각각 학생번호에 의해 완전 종속

학생번호	강좌이름	성적
401	데이터베이스	4
402	스포츠경영학	3.5
501	데이터베이스	3.5
501	자료구조	3.5
502	자료구조	4

학생번호,  
강좌이름 → 성적

성적은 (학생 번호, 강좌이름) 의 어떤 부분집합에도 종속되어 있지 않음

강좌이름	강의실
데이터베이스	공학관110
스포츠 경영학	체육관103
자료구조	공학관111

강의실은 강좌이름에 의해 완전 종속

기본키가 아닌 속성이 기본키에 비이행적으로 종속되어야 함.  
 이행적 종속이란  $A \rightarrow B, B \rightarrow C$ 가 성립할 때  $A \rightarrow C$ 가 성립

학생번호	학생이름	주소	학과	학과 사무실
401	김연아	서울	체육	체육관101
402	장미란	강원도	체육	체육관101
501	박지성	맨체스터	컴퓨터과	공학관101
502	추신수	클리블랜드	컴퓨터과	공학관101

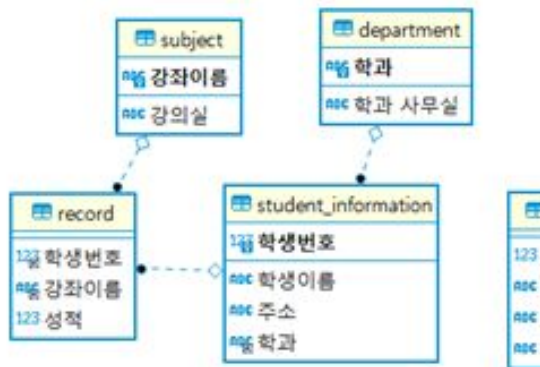
학과	학과 사무실
체육	체육관101
컴퓨터과	공학관101



위 테이블에서 학생 번호  $\rightarrow$  학과  $\rightarrow$  학과 사무실이므로 분해 하여야 함

학생번호	학생이름	주소	학과
401	김연아	서울	체육
402	장미란	강원도	체육
501	박지성	맨체스터	컴퓨터과
502	추신수	클리블랜드	컴퓨터과

## ERD



개체 무결성과 참조 무결성을 지키기 위해 '학생 번호', '강좌이름', '학과'는 PK로 지정해야 함

## TABLE

테이블명	student_information		
Num	Column Id	Type	Null
1	학생번호	Int	NN
2	학생이름	Varchar	NN
3	주소	Varchar	NN
4	학과	Varchar	NN

테이블명	subject		
Num	Column Id	Type	Null
1	강좌이름	Varchar	NN
2	강의실	Varchar	NN

테이블명	record		
Num	Column Id	Type	Null
1	학생번호	Int	NN
2	강좌이름	Varchar	NN
3	성적	Float	NN

테이블명	department		
Num	Column Id	Type	Null
1	학과	Varchar	NN
2	학과사무실	Varchar	NN

```
CREATE TABLE student_information(  
    학생번호 INT PRIMARY KEY,  
    학생이름 VARCHAR(50),  
    주소 VARCHAR(100),  
    학과 VARCHAR(50)  
);
```

```
INSERT INTO student_information(학생번호,  
    학생이름, 주소, 학과) VALUES  
(401, '김연아', '서울', '체육'),  
(402, '장미란', '강원도', '체육'),  
(501, '박지성', '맨체스터', '컴퓨터과'),  
(502, '추신수', '클리블랜드', '컴퓨터과');
```

```
CREATE TABLE department(  
    학과 VARCHAR(50) PRIMARY KEY,  
    학과사무실 VARCHAR(50)  
);
```

```
INSERT INTO department(학과, 학과사무실)  
VALUES  
( '체육', '체육관 101'),  
( '컴퓨터과', '공학관 101');
```

```
CREATE TABLE subject(  
    강좌이름 VARCHAR(50) PRIMARY KEY,  
    강의실 VARCHAR(50)  
);
```

```
INSERT INTO subject(강좌이름, 강의실) VALUES  
( '데이터베이스', '공학관 110'),  
( '스포츠경영학', '체육관 103'),  
( '자료구조', '공학관 111');
```

```
CREATE TABLE record(  
    학생번호 INT,  
    강좌이름 VARCHAR(50),  
    성적 FLOAT,  
    PRIMARY KEY (학생번호, 강좌이름),  
    FOREIGN KEY (학생번호) REFERENCES student_information(학생번호),  
    FOREIGN KEY (강좌이름) REFERENCES subject(강좌이름)  
);
```

```
INSERT INTO record(학생번호, 강좌이름, 성적) VALUES  
(401, '데이터베이스', 4.0),  
(402, '스포츠경영학', 3.5),  
(501, '데이터베이스', 3.5),  
(501, '자료구조', 3.5),  
(502, '자료구조', 4.0);
```

## 1) 성적이 3.5인 학생의 개인 정보

검색 결과 여러개 => in

```
select * from shop.student_information where 학생번호 in
(select 학생번호 from shop.record where 성적 = 3.5);
```

그리드	1	2
학생번호	402	501
학생이름	장미란	박지성
주소	강원도	맨체스터
학과	체육	컴퓨터과

```
SELECT * FROM shop.student_information WHERE 학생번호
IN
(SELECT 학생번호 FROM shop.record WHERE 성적 = 3.5);
```

## 2) 강의실이 '공학관110'인 학생의 성적 정보

검색 결과 한 개 => =

```
select * from shop.record where 강좌이름 =
(select 강좌이름 from shop.subject where 강의실 = '공학관110');
```

그리드	1	2
학생번호	401	501
강좌이름	데이터베이스	데이터베이스
성적	4	3.5

```
SELECT * FROM shop.record WHERE 강좌이름 =
(SELECT 강좌이름 FROM shop.subject WHERE 강의실 =
'공학관110');
```

SELECT \* FROM MEMBER  
WHERE id IN ('apple', 'ice')

SELECT \* FROM "MEMBER"  
WHERE id IN  
(SELECT writer FROM BBS) --apple

Results 1

SELECT \* FROM "MEMBER" WHERE id IN (SELECT writer FROM BBS) Enter a SQL expression

	ABC ID	ABC PW	ABC NAME	ABC TEL	
1	apple	apple	apple	apple	



## 1) Inner Join

'record 테이블의 강좌이름과 subject 테이블의 강좌이름이 일치하는 행 중, 강의실이 '공학관110' 행을 Inner Join을 통해 가져옴

```
SELECT 학생번호 from shop.record inner join shop.subject
where record.강좌이름 = subject.강좌이름 and 강의실 = '공학관110';
```

record 1

SELECT 학생번호 from

학생번호	강좌이름	강의실
401	데이터베이스	공학관110
501	데이터베이스	공학관110

```
SELECT 학생번호 FROM shop.record INNER JOIN
shop.subject
ON record.강좌이름 = subject.강좌이름 AND 강의실 =
'공학관110';
```

## 2) Left Join

1. student\_information과 record 테이블의 학생번호가 일치하는 행을 Left Join
2. student\_information과 department 테이블의 학과가 일치하는 행을 Left Join
3. record와 subject 테이블의 강좌이름이 일치하는 행을 Left Join

```
select * from shop.student_information left join shop.record on
student_information.학생번호 = record.학생번호 left join shop.department on
student_information.학과 = department.학과 left join shop.subject on
record.강좌이름 = subject.강좌이름;
```

student\_information(\*)

select \* from shop.student\_information left join shop.record

학생번호	이름	성명	주요	학과	학과	강좌이름	강의실	강좌이름	강의실
401	김민아	서울	데이터베이스	공학관110	데이터베이스	공학관110			
402	정민아	강원도	데이터베이스	공학관110	데이터베이스	공학관110			
501	박지성	영제스다	컴퓨터과	공학관101	데이터베이스	공학관110			
501	박지성	영제스다	컴퓨터과	공학관101	데이터베이스	공학관110			
502	주신수	울리블랜드	컴퓨터과	공학관101	데이터베이스	공학관110			

```
SELECT * FROM shop.student_information
LEFT JOIN shop.record ON student_information.학생번호 = record.학생번호
LEFT JOIN shop.department ON student_information.학과 = department.학과
LEFT JOIN shop.subject ON record.강좌이름 = subject.강좌이름;
```

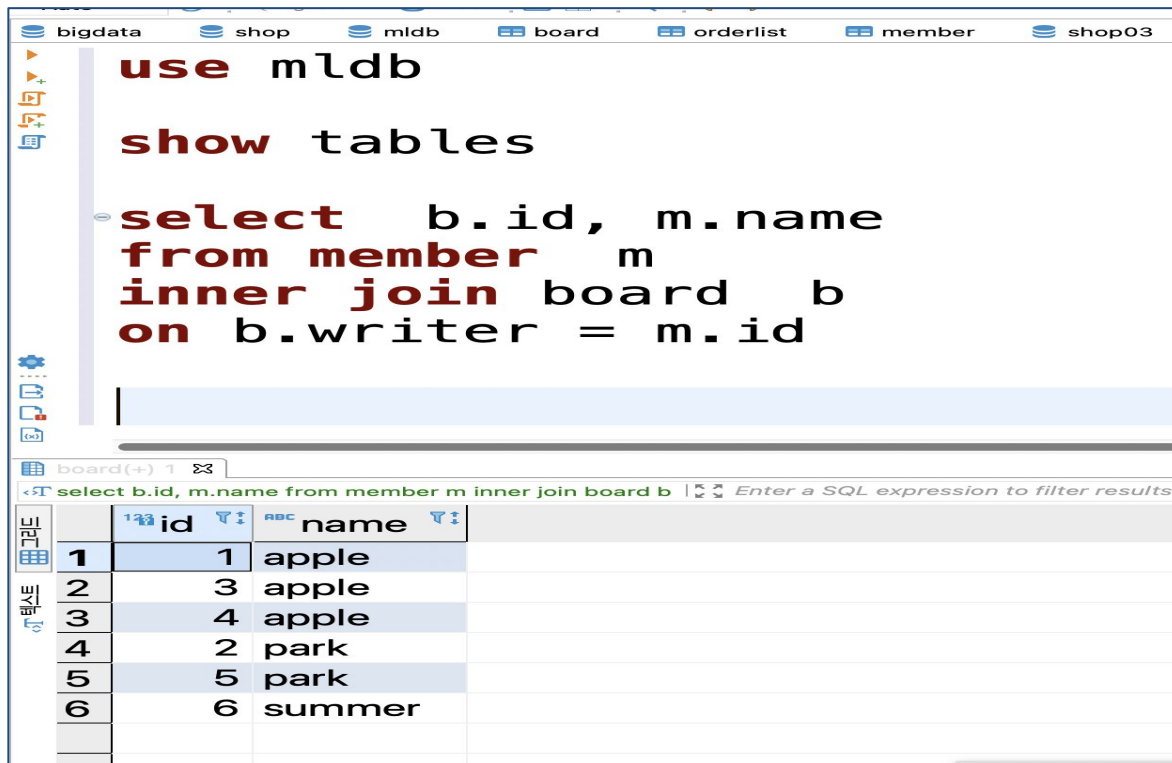


```
SELECT MEMBER.id, BBS.TITLE, ORDERLIST.TOTAL_COUNT  
FROM MEMBER, BBS, ORDERLIST  
WHERE MEMBER.id = bbs.WRITER AND MEMBER.id = ORDERLIST.MEMBER_ID  
AND bbs.TITLE = 'soso'
```

Results 1

SELECT MEMBER.id, BBS.TITLE, ORDERLIST.TOTAL\_COUNT  
Enter a SQL expression to filter results (use Ctrl+Space)

	ID	TITLE	TOTAL_COUNT
1	apple	soso	1000



The screenshot shows a database management interface with a top navigation bar containing tabs for 'bigdata', 'shop', 'mldb', 'board', 'orderlist', 'member', and 'shop03'. The main area displays SQL code for an inner join query. Below the code, a query execution bar shows the same SQL statement. The results are displayed in a table with columns 'id' and 'name'.

```
use mldb  
  
show tables  
  
select b.id, m.name  
from member m  
inner join board b  
on b.writer = m.id
```

board(+) 1

select b.id, m.name from member m inner join board b | Enter a SQL expression to filter results

	id	name
1	1	apple
2	3	apple
3	4	apple
4	2	park
5	5	park
6	6	summer

```

use mldb

show tables

select b.id, m.name, r.content
from member m
inner join board b on b.writer = m.id
inner join reply r on r.boardid = b.id
    
```

board(+) 1

select b.id, m.name, r.content from member m inner join board b on b.writer = m.id inner join reply r on r.boardid = b.id Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	content
1	1	apple	happy
2	1	apple	good
3	2	park	nono
4	1	apple	likw
5	2	park	soso

content: varchar(100)

- 정규화, 정규화과정
- 이상현상, 발생원인(중복), 해결책(분리)
- 제1정규형 - 원자값
- 제2정규형 - 부분종속 X
- 제3정규형 - 이행종속 X