



[D E E P
N O I D]

데이터 베이스에 대하여 (MySQL편)

2021.09.14

www.deepnoid.com

Presenter: 임효진

목차

▣ 데이터 베이스에 대하여 (MySQL 편)

1. DDL
2. DML - 삽입, 삭제, 갱신문 / SELECT 조회하기
3. DML - JOIN
4. 트리거
5. 프로시저
6. 정규화
7. 트랜잭션 (Transaction)
8. ERD

관계형 DB - 트랜잭션, 정규화

장점

- 데이터의 성능이 일반적으로 좋아 정렬, 탐색, 분류가 빠름
- 신뢰성이 높아 데이터의 무결성을 보장
- 정규화에 따른 갱신 비용을 최소화

단점

- 기존에 작성된 스키마를 수정하기 어려움
- 데이터베이스의 부하를 분석하기 어려움
- 빅데이터를 처리하는데 매우 비효율적임

비관계형 DB

- 거대한 Map으로 key-value 형식 지원
- 관계를 따로 두지 않음
- 스키마 정의가 없음

장점

- 대용량 데이터 처리를 하는데 효율적
- 읽기 작업보다 쓰기 작업이 더 빠르고 관계형 데이터 베이스에 비해 쓰기와 읽기 성능이 빠름
- 데이터 모델링이 유연함
- 뛰어난 확장성으로 검색에 유리함
- 최적화된 키 값 저장 기법을 사용하여 응답 속도나 처리 효율 등에서 성능이 뛰어남
- 복잡한 데이터 구조를 표현할 수 있음

단점

- 쿼리 처리시 데이터를 파싱 후 연산을 해야해서 큰 크기의 document를 다룰 때는 성능이 저하됨

장점

- MySQL은 다중 사용자
- 다른 프로그래밍 언어와 통합 가능
- 큰 RAM이 필요 없음
- 휴대용 소프트웨어 => CMD로 가능
- 테이블 구조가 더 유연 => ALTER나 TABLE로 처리
- 오픈 소스
- 낮은 하드웨어와 함께 사용
- 관리 도구
- 변수 데이터 타입
- 보안 보장
- 좋은 인터페이스
- 우수한 성능

단점

- 가난한 기술 지원
- 큰 데이터베이스로 적용하기 어려움
- 게임 및 모바일 애플리케이션 인기가 없음

기타

- <https://smoh.tistory.com/369>
- <https://altitudetvm.com/ko/komputer/1522-15-kelebihan-dan-kekurangan-mysql-server-yang-perlu-diketahui.html>

- DB 구조, 데이터 형식, 접근 방식 등 DB를 구축하거나 수정할 목적으로 사용하는 언어이다.
 - DDL은 번역한 결과가 데이터 사전이라는 특별한 파일에 여러 개의 테이블로서 저장된다.
 - DDL에는 CREATE SCHEMA, CREATE DOMAIN, CREATE TABLE, CREATE VIEW, CREATE INDEX, ALTER TABLE, DROP CREATG 등이 있다.
- ▣ 데이터 유형 (날짜 유형 첨부: <https://lightblog.tistory.com/155>)

데이터 유형	설명
VARCHAR	가변 길이 문자 데이터 (최소는 1 이며 최대는 4000)
CHAR	고정 길이 문자 데이터 (최소는 1이며 최대는 2000)
NUMBER	자릿수가 p이며 소수점 이하 자릿수가 s인 숫자
DATE	기원전 4712년 1월 1일부터 서기 9999년 12월 31일 상의 가장 가까운 초 단위에 날짜 및 시간 값
LONG	가변 길이 문자 데이터 (최대 2GB)
CLOB	문자 데이터 (최대 4GB)

KEY = 무엇인가를 유일하게 식별한다는 의미 - 1

- 추가자료: <https://wikidocs.net/135080>

- 슈퍼키 => 튜플을 유일하게 식별할 수 있는 하나의 속성 혹은 속상의 집합. 튜플을 식별할 수 있으면 모두 슈퍼키가 될 수 있다.
- 후보키 => 튜플을 유일하게 식별할 수 있는 속성의 최소 집합이다. 즉 슈퍼키 중에서 최소의 속성으로 집합된 것이 후보키가 된다.
- 기본키 => 후보키 중 하나를 선정하여 대표로 삼는 키를 말한다. 후보키가 하나뿐이라면 그 후보키를 기본키로 사용하면 되고 여러 개라면 릴레이션의 특성을 반영하여 하나를 선택한다, 스키마를 표현할 때 따로 표시

- 기본 키 선정시 고려사항
 - 릴레이션 내 튜플을 식별할 수 있는 고유한 값을 가져야 한다.
 - NULL 값은 허용하지 않는다.
 - 키 값의 변동이 일어나지 않아야 한다.
 - 최대한 적은 수의 속성을 가진 것 이어야 한다.
 - 향후 키를 사용하는데 있어서 문제 발생 소지가 없어야 한다.

[각 키의 포함 관계 및 정의 요약]



- 대체키 => 대체키는 기본키로 선정되지 않은 후보키를 말한다.
- 외래키 => 다른 릴레이션의 기본키를 참조하는 속성을 말한다. 외래키는 다른 릴레이션의 기본키를 참조하여 관계 데이터 모델의 특징인 릴레이션 간의 관계를 표현 한다.

외래키 조건 : 외래키가 성립하기 위해 참조하고 참조되는 양쪽 릴레이션의 도메인이 서로 같아야 한다.

- 기본 키 => 후보키 중 하나를 선정하여 대표로 삼는 키를 말한다. 후보키가 하나뿐이라면 그 후보키를 기본키로 사용하면 되고 여러 개라면 릴레이션의 특성을 반영하여 하나를 선택 단, 스키마를 표현 할 때 따로 표시
- 기본 키 선정시 고려사항
 - 릴레이션 내 튜플을 식별 할 수 있는 고유한 값을 가져야 한다.
 - NULL 값은 허용하지 않는다.
 - 키 값의 변동이 일어나지 않아야 한다.
 - 최대한 적은 수의 속성을 가진 것 이어야 한다.
 - 향후 키를 사용하는데 있어서 문제 발생 소지가 없어야 한다.

CREATE TABLE

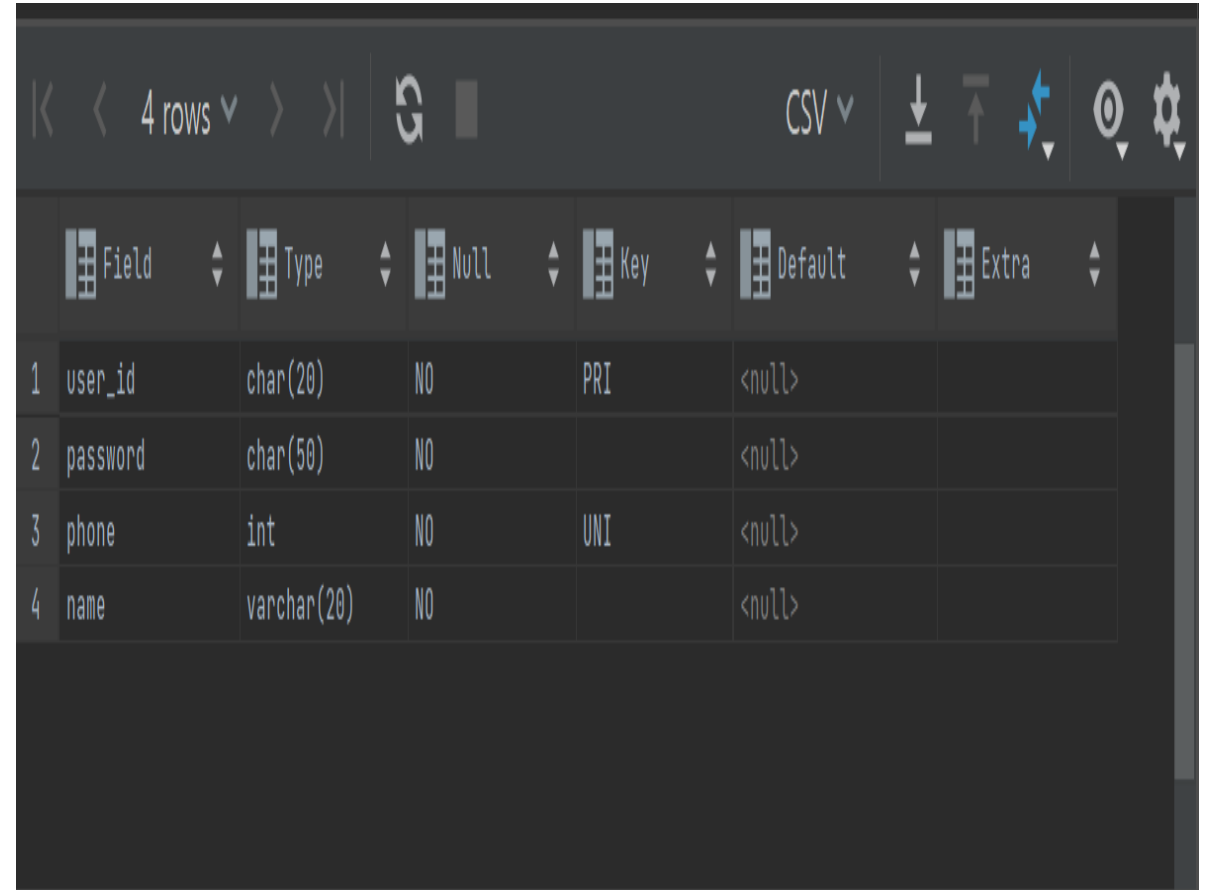
- 참고 실습: http://tcpschool.com/mysql/mysql_basic_create

user table 사용 예정 , 기본 키 값만 예제로 넣음

- 이때 기본 키값, 외래키 값 등에 대해서 잘 적어야 함 우선은 기본 키 값만 예제로 활용

```
CREATE TABLE user (  
    user_id char(20) not null unique,  
    password char(50) not null,  
    phone int(50) not null unique,  
    name varchar(20) not null,  
    primary key (user_id)  
);
```

```
DESC user;
```



	Field	Type	Null	Key	Default	Extra
1	user_id	char(20)	NO	PRI	<null>	
2	password	char(50)	NO		<null>	
3	phone	int	NO	UNI	<null>	
4	name	varchar(20)	NO		<null>	

ALTER TABLE

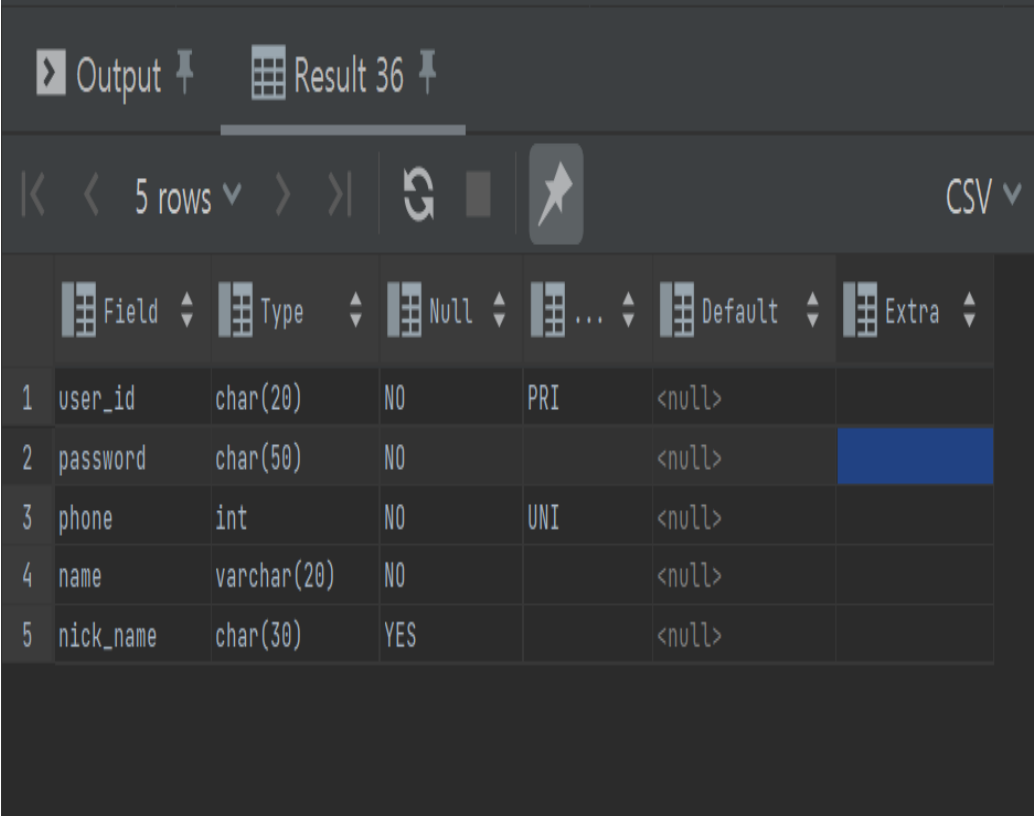
- 참고 실습: http://tcpschool.com/mysql/mysql_basic_alter

#바꾸기 시도 = ALTER에
nick_name을 추가함

ALTER TABLE user

ADD nick_name char(30);

DESC user;



Output Result 36

5 rows

CSV

	Field	Type	Null	...	Default	Extra
1	user_id	char(20)	NO	PRI	<null>	
2	password	char(50)	NO		<null>	
3	phone	int	NO	UNI	<null>	
4	name	varchar(20)	NO		<null>	
5	nick_name	char(30)	YES		<null>	

ALTER TABLE

- 참고 실습: http://tcpschool.com/mysql/mysql_basic_drop

테이블 삭제

- 이 때 삭제를 하면 바로 사라지기 때문에 주의
- 또한 DATAGRIP에서 활용시에는 빨간색 글꼴로 변함을 알 수 있음

```
DROP TABLE user;
```

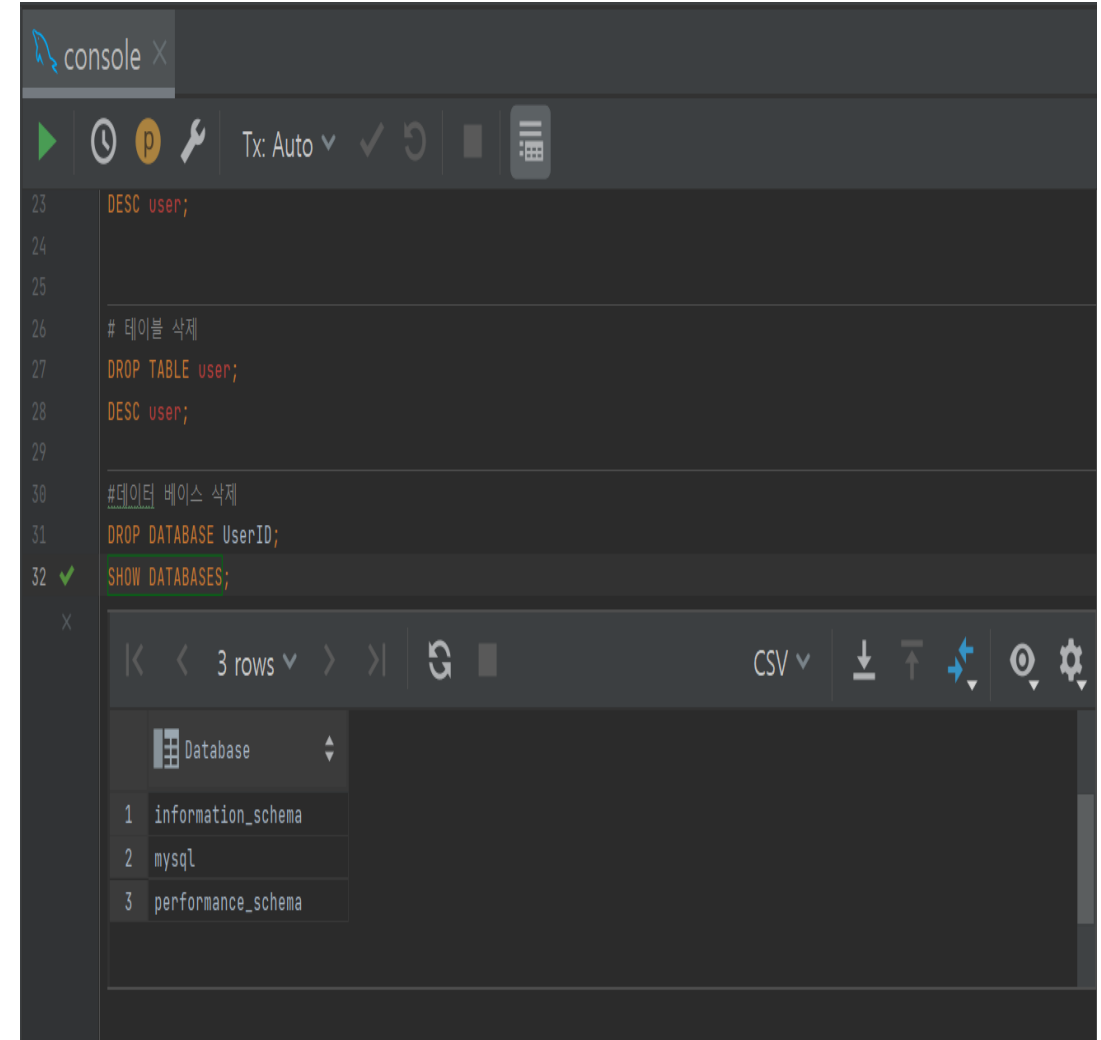
```
DESC user;
```

#데이터 베이스 삭제

- 데이터베이스 전체가 삭제되는 현상을 경험

```
DROP DATABASE UserID;
```

```
SHOW DATABASES;
```



The screenshot shows a MySQL console window with the following content:

```
console X
Tx: Auto
23 DESC user;
24
25
26 # 테이블 삭제
27 DROP TABLE user;
28 DESC user;
29
30 #데이터 베이스 삭제
31 DROP DATABASE UserID;
32 ✓ SHOW DATABASES;
```

Below the console, the results of the `SHOW DATABASES;` command are displayed in a table:

	Database
1	information_schema
2	mysql
3	performance_schema

CREATE VIEW

- 참고 실습:

http://tcpschool.com/mysql/mysql_view_createReplace

- 단순 뷰 ? <https://wikidocs.net/4176>

- 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.
- 저장 장치 내에 물리적으로 존재하지 않는다.
- CREATE문을 사용하여 정의한다.
- 제거할 때는 DROP문을 사용한다.
- 데이터의 논리적 독립성을 제공한다.

Name	ReserveDate	Dday
홍길동	2016-01-05	-10103
임꺽정	2016-02-12	-9996
장길산	2016-01-16	-10092
홍길동	2016-03-17	-9891

Name, ReserveDate,
ReserveDate -
Curdate() AS Dday를
기반으로 만들어진 뷰
이때 as 는 alias를 의
미

```
CREATE VIEW MyView AS  
SELECT Name, ReserveDate,  
ReserveDate - Curdate() AS Dday  
FROM Reservation;
```

▣ DCL - GRANT / REVOKE

- 데이터 베이스 관리자가 데이터 베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.
- GRANT: 권한 부여를 위한 명령어
\$ 예제 :: 김영웅에게 <사원>테이블에 대한 접근 및 조작에 관한 모든 권한을 부여하는 SQL 문을 작성하시오.
 - > GRANT ALL ON 사원 TO 김영웅
- REVOKE: 권한 취소를 위한 명령어
\$ 예제 :: 이민지에게 부여된 <고객> 테이블에 대한 SELECT, INSERT,DELETE 권한을 취소하는 SQL 문을 작성하시오.
 - > REVOKE SELECT , INSERT,DELETE ON 고객 FROM 이민지

▣ DCL - COMMIT/ROLLBACK/SAVEPOINT

- COMMIT : 트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영해야함
- ROLLBACK: 아직 COMMIT 되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어
- SAVE POINT: 트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어로, 저장점을 지정할때는 이름을 부여하며, ROLLBACK 시 지정된 저장점까지의 트랜잭션 처리 내용이 취소

삽입, 삭제, 갱신문

```
INSERT INTO user VALUES ("lim",12345678,01312345678,"임효진");
INSERT INTO user VALUES ("front_sin",9101112,0109101112,"신소진");
INSERT INTO user VALUES ("jin",131415161718,0113141516,"진태정");
INSERT INTO user VALUES ("back_sin",17181829,01171819,"신준영");

select * from user;
```

	user_id	password	phone	name
1	back_sin	17181829	1171819	신준영
2	front_sin	9101112	109101112	신소진
3	jin	131415161718	113141516	진태정
4	lim	12345678	1312345678	임효진

```
UPDATE user SET name="deepnoid" WHERE name="임효진";

select * from user;
```

	user_id	password	phone	name
1	back_sin	17181829	1171819	신준영
2	front_sin	9101112	109101112	신소진
3	jin	131415161718	113141516	진태정
4	lim	12345678	1312345678	deepnoid

- INSERT INTO 테이블명 VALUES : 테이블 안 데이터 추가 이때 모두가 들어가면 생략이 가능
 - 예 : insert into user (user_id, password, phone, name) values ("아이디",패스워드,전화번호,"이름");
- UPDATE 테이블명 SET 고칠 데이터 WHERE 고치기 전 데이터
 - 예 : UPDATE user SET name="deepnoid" WHERE name="임효진";

삽입, 삭제, 갱신문

```
delete from user where name="deepnoid";
select * from user;
```

	user_id	password	phone	name
1	back_sin	17181829	1171819	신준영
2	front_sin	9101112	109101112	신소진
3	jin	131415161718	113141516	진태정

- DELETE FROM 테이블 WHERE 삭제할 데이터
- 예 : delete from user where name="deepnoid";

My_Table로 부터 모든 칼럼 조회

```
SELECT * FROM My_Table;
```

My_Table의 No_Emp, Nm_Kor, Age 칼럼 조회

```
SELECT No_Emp, Nm_Kor, Age FROM My_Table;
```

이름이 '홍길동'인 사람 검색

```
SELECT * FROM My_Table WHERE Nm_Kor = '홍길동';
```

나이가 25살인 사원의 한국 이름과 나이 조회

```
SELECT Nm_Kor, Age FROM My_Table WHERE Age=25;
```

나이가 25살이 아닌 사원 조회

```
SELECT * FROM My_Table WHERE Age <> 25;
```

사원번호가 '0315' 이고 나이가 25살보다 작거나 이름이 '홍길동'인 사원 조회

```
SELECT * FROM My_Table WHERE No_Emp = '0315' AND (Age < 25 OR Nm_Kor = '홍길동');
```

사원번호가 '0315' 이거나 나이가 25살 이상이면서 이름이 '홍길동'인 사원 조회

```
SELECT * FROM My_Table WHERE No_Emp = '0315' OR (Age >= 25 AND Nm_Kor = '홍길동');
```

'김'으로 시작하는 사원 조회

```
SELECT * FROM My_Table WHERE Nm_Kor LIKE '김%';
```

'김'이 들어가는 시작하는 사원 조회

```
SELECT * FROM My_Table WHERE Nm_Kor LIKE '%김%';
```

'김'으로 끝나는 사원의 사원번호 조회

```
SELECT No_Emp FROM My_Table WHERE Nm_Kor LIKE '%김';
```

▣ JOIN의 개념

- 2개의 테이블에 대해 연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환
- JOIN은 일반적으로 FROM절에서 기술하지만, 릴레이션이 사용되는 어느 곳에서나 사용

▣ INNER_JOIN의 개념

- 가장 일반적인 JOIN의 형태로, 관계가 설정된 두 테이블에서 조인된 필드가 일치하는 행만 표시

▣ OUTER_JOIN의 개념

- 릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과로 출력하기 위한 JOIN 방법

JOIN 예제1 - 실습 예제 : <https://programmers.co.kr/learn/courses/30/lessons/59042>

solution.sql

```
1  -- 코드를 입력하세요
2  SELECT OUTS.ANIMAL_ID, OUTS.NAME
3  FROM ANIMAL_OUTS OUTS
4  #LEFT OUTER JOIN 공부하기
5  LEFT OUTER JOIN ANIMAL_INS INS
6  ON OUTS.ANIMAL_ID = INS.ANIMAL_ID
7  WHERE INS.ANIMAL_ID is NULL
8  ORDER BY OUTS.ANIMAL_ID;
```

실행 결과

ANIMAL_ID	NAME
A349480	Daisy
A349733	Allie
A349990	Spice
A362137	*Darcy

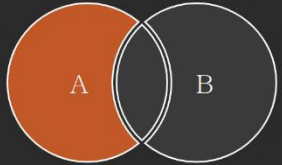
예를 들어, ANIMAL_INS 테이블과 ANIMAL_OUTS 테이블을 참고하게 된다면

ANIMAL_ID와 NAME이 조회되어야하는 상태이니 먼저 SELECT로 조회를 시도 합니다.

이때 LEFT OUTER JOIN을 활용하여 교집합을 만들 수 있도록 하는데 ANIMAL_ID를 기반으로 OUT과 INS를 판단하여 구분을 하고 WHERE절과 ORDER_BY를 통해 확인 할 수 있습니다.

JOIN의 예제 2, 종류

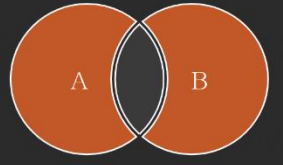
- 참고 사항 예제, 출처 : <https://opentutorials.org/course/3884>
- https://docs.google.com/spreadsheets/d/1OUHANTPdx0ga8P1_HBm6WUuWs02tvV-31mgi_XmRbc/edit#gid=1849152573



```
SELECT * FROM TableA A
LEFT JOIN TableB B ON
A.key = B.key WHERE B.key IS NULL
```

Copy SQL

To produce the set of records only in Table A, but not in Table B, we perform the left (outer) join, then exclude the records we don't want from the right side via a where clause



```
SELECT * FROM TableA A
FULL OUTER JOIN TableB B ON
A.key = B.key WHERE A.key IS NULL
OR B.key IS NULL
```

Copy SQL

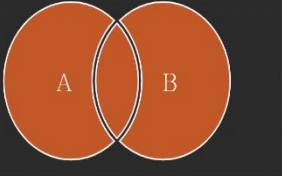
To produce the set of records unique to Table A and Table B, we perform the full outer join, then exclude the records we don't want from both sides via a where clause



```
SELECT * FROM TableA A
INNER JOIN TableB B ON
A.key = B.key
```

Copy SQL

Inner join produces only the set of records that match in both Table A and Table B

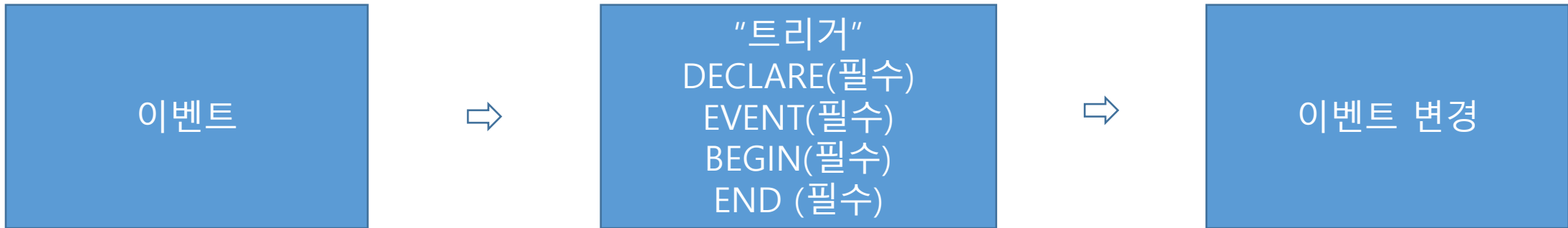


```
SELECT * FROM TableA A
FULL OUTER JOIN TableB B ON
A.key = B.key
```

Copy SQL

Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null

- 데이터베이스 시스템에서 데이터의 삽입(INSERT), 갱신(UPDATE), 삭제(DELETE) 등의 이벤트가 발생할 때 마다 관련 작업이 자동으로 수행되는 절차형 SQL
- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력이 목적



▶ **트리거 예제** - EMP 테이블의 SAL에 UPDATE가 발생하면 로그를 저장하는 EMP_SAL_LOG 테이블과 트리거를 작성하시오. SAL에 UPDATE가 발생하면 트리거에 의해 EMP_SAL_LOG 테이블에 UPDATE가 발생한 날짜, SAL가 변경 된 사원의 번호, 변경 전 SAL의 값, 변경 후 SAL의 값, 증감율((변경 후 SAL -변경 전 SAL) / 변경 전 SAL) 값을 저장한다. 2개 이상의 UPDATE 문을 수행하여 EMP_SAL_LOG의 내용을 보이시오.

```
mysql> CREATE TABLE EMP_SAL_LOG(  
  -> updateAt DATE,  
  -> empno INT,  
  -> old_sal DECIMAL(7,2),  
  -> new_sal DECIMAL(7,2),  
  -> variation DECIMAL(5,2));  
Query OK, 0 rows affected (0.15 sec)
```

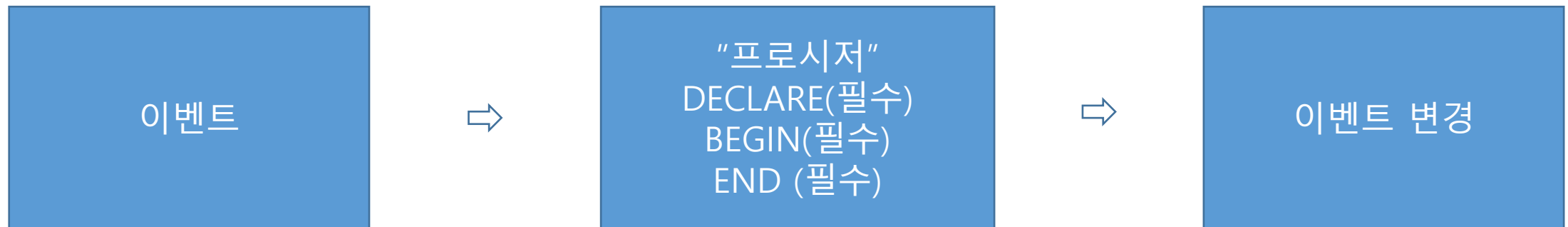
```
mysql> DELIMITER $$  
mysql> CREATE TRIGGER sal_up_trg  
  -> AFTER UPDATE  
  -> ON EMP  
  -> FOR EACH ROW  
  -> BEGIN  
  ->   IF NEW.SAL <> OLD.SAL THEN  
  ->     INSERT INTO EMP_SAL_LOG VALUES (CURDATE(), OLD.EMPNO, OLD.SAL,  
  -> NEW.SAL, ((NEW.SAL - OLD.SAL)/OLD.SAL));  
  ->   END IF;  
  -> END $$  
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> DELIMITER ;  
mysql> UPDATE EMP  
  -> SET SAL = 3500  
  -> WHERE EMPNO = 7499;  
Query OK, 1 row affected (0.03 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE EMP  
  -> SET SAL = 4800  
  -> WHERE EMPNO = 7654;  
Query OK, 1 row affected (0.01 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM EMP_SAL_LOG;  
+-----+-----+-----+-----+-----+  
| updateAt | empno | old_sal | new_sal | variation |  
+-----+-----+-----+-----+-----+  
| 2020-12-01 | 7499 | 1600.00 | 3500.00 | 1.19 |  
| 2020-12-01 | 7654 | 1250.00 | 4800.00 | 2.84 |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.01 sec)
```

- 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어
- 프로시저는 데이터베이스에 저장되어 수행되기 때문에 스토어드 프로시저라고도 불린다.
- 프로시저는 시스템의 일일 마감 작업, 일괄 작업 등에 주로 사용된다.
- 요약: DBMS에서 시스템의 주간 마감, 일일 마감 작업 등 주로 일괄 작업에 상요되며, 데이터베이스에 저장되어 수행한다. DECLARE, CONTROL, SQL, EXCEPTION등의 구성요소로 이루어져 있으며 EXECUTE또는 CALL 명령어로 실행되는 절차형 SQL



프로시저 예제 - 커서를 사용하여 EMP 테이블에서 늦게 입사한 순서대로 4명의 사원 이름(ENAME)을 출력하는 emp_dateRank 프로시저를 작성하고 호출한 결과를 작성하시오.

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE emp_dateRank()
-> BEGIN
-> DECLARE cnt INT DEFAULT 0;
-> DECLARE empName VARCHAR(10);
-> DECLARE userCursor CURSOR FOR
-> SELECT ename
-> FROM emp
-> ORDER BY hiredate DESC;
->
-> OPEN userCursor;
-> cloop: WHILE TRUE DO
-> FETCH userCursor INTO empName;
->
-> SET cnt = cnt + 1;
-> SELECT empName as ENAME;
->
-> IF cnt = 4 THEN
-> LEAVE cloop;
-> END IF;
->
-> END WHILE;
->
-> CLOSE userCursor;
-> END $$
Query OK, 0 rows affected (0.03 sec)
```

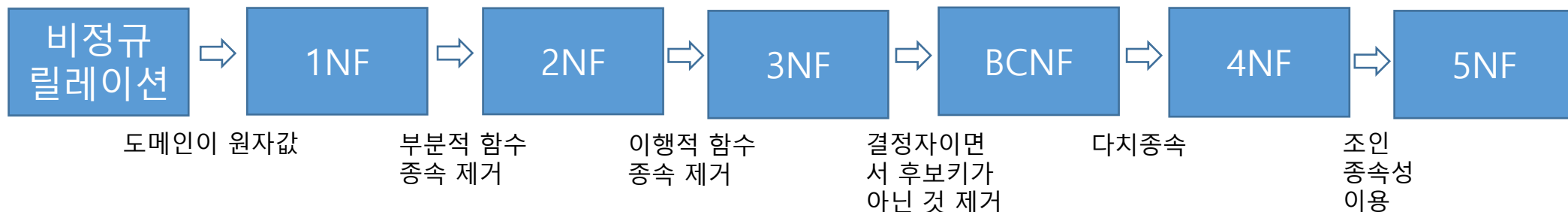
```
mysql> CALL emp_dateRank;
+-----+
| ENAME |
+-----+
| ADAMS |
+-----+
1 row in set (0.01 sec)

+-----+
| ENAME |
+-----+
| SCOTT |
+-----+
1 row in set (0.03 sec)

+-----+
| ENAME |
+-----+
| MILLER |
+-----+
1 row in set (0.05 sec)

+-----+
| ENAME |
+-----+
| JAMES |
+-----+
1 row in set (0.07 sec)
Query OK, 0 rows affected (0.08 sec)
```

- 테이블의 속성들이 상호 종속적인 관계를 갖는 특성을 이용하여 테이블을 무손실 분해하는 과정
- 순서도



- 반정규화란? 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화 데이터 모델을 통합, 중복, 분리하는 과정

- 테이블 통합

- 중복 속성 추가

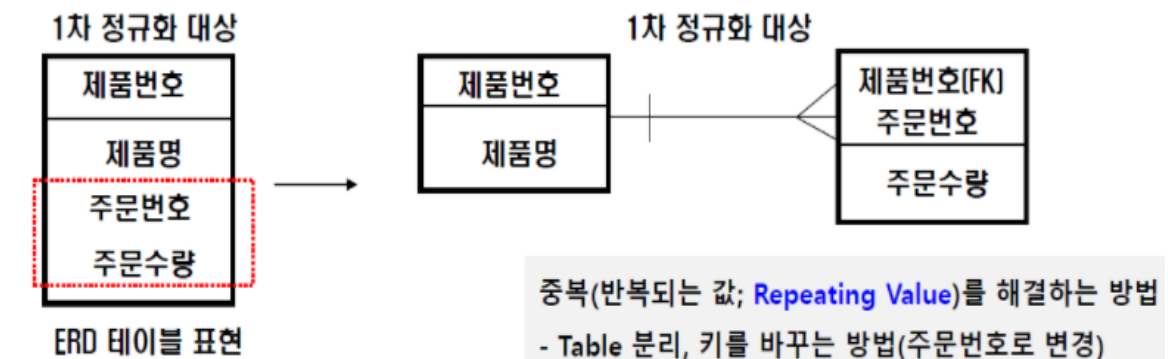
- 테이블 분리

- 중복 테이블 추가

정규화 예제 - 참고 자료 : <https://wikidocs.net/133798>

1NF

- 모든 도메인이 원자값을 가지도록 분해: 의존자의 중복을 제거



ERD 테이블 표현

업무규칙: 여러 제품을 주문

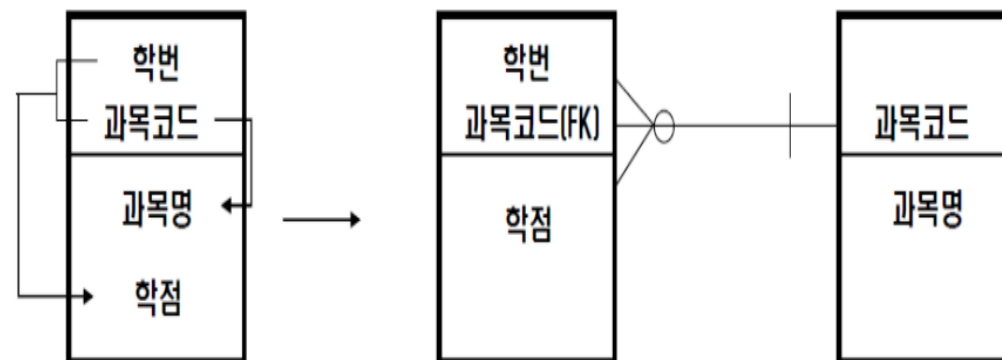
-동일 제품번호에 대해 주문이 중복되어 발생하게 됨 -> 이러한 경우 이상현상(삽입, 갱신, 삭제) 발생

제품번호	제품명	주문번호	주문수량
1001	컴퓨터	20040101001	1
1001	컴퓨터	20040101002	1
1001	컴퓨터	20040101003	2

2NF

- 부분함수 종속성 제거: 식별자를 제외한 모든 속성은 식별자에 종속해야 함

◦ 부분함수적 종속성: $X \rightarrow Y$ 에서 Y 가 X 의 부분집합에 대해서도 함수적으로 종속되는 경우



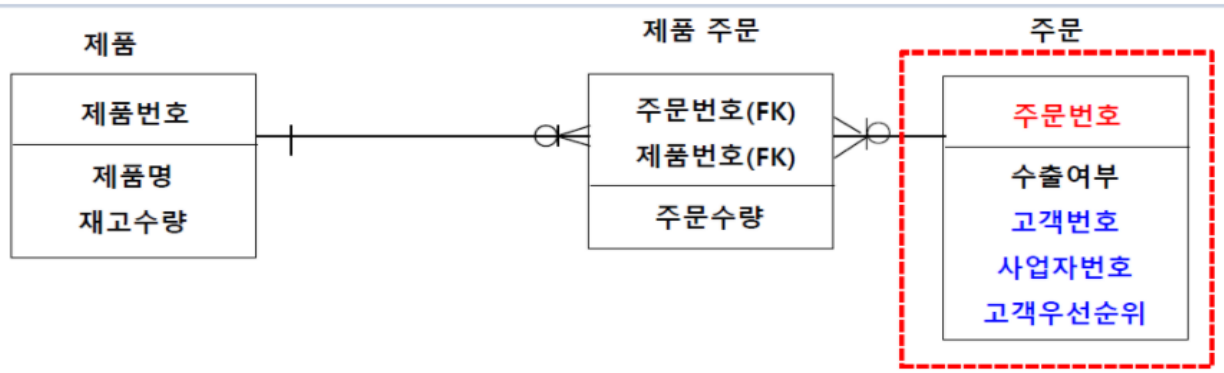
-과목코드->과목명의 관계가 부분함수 종속: 이를 제거해야함

- $X \rightarrow Y$ 에서 Y 가 X 의 부분집합에 대해서도 함수적으로 종속되는 경우: 위에서 과목명(Y)이 X (학번+과목코드)의 부분집합인 '과목코드'에 대해서 함수적으로 종속관계

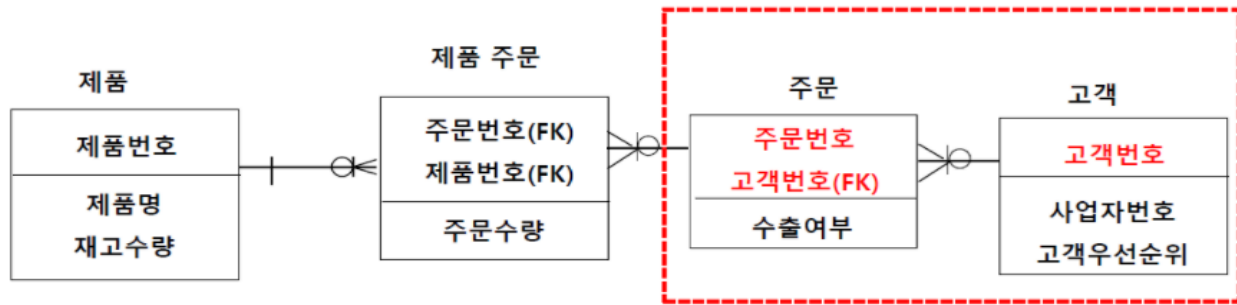
정규화 예제 - 참고 자료 : <https://wikidocs.net/133798>

3NF

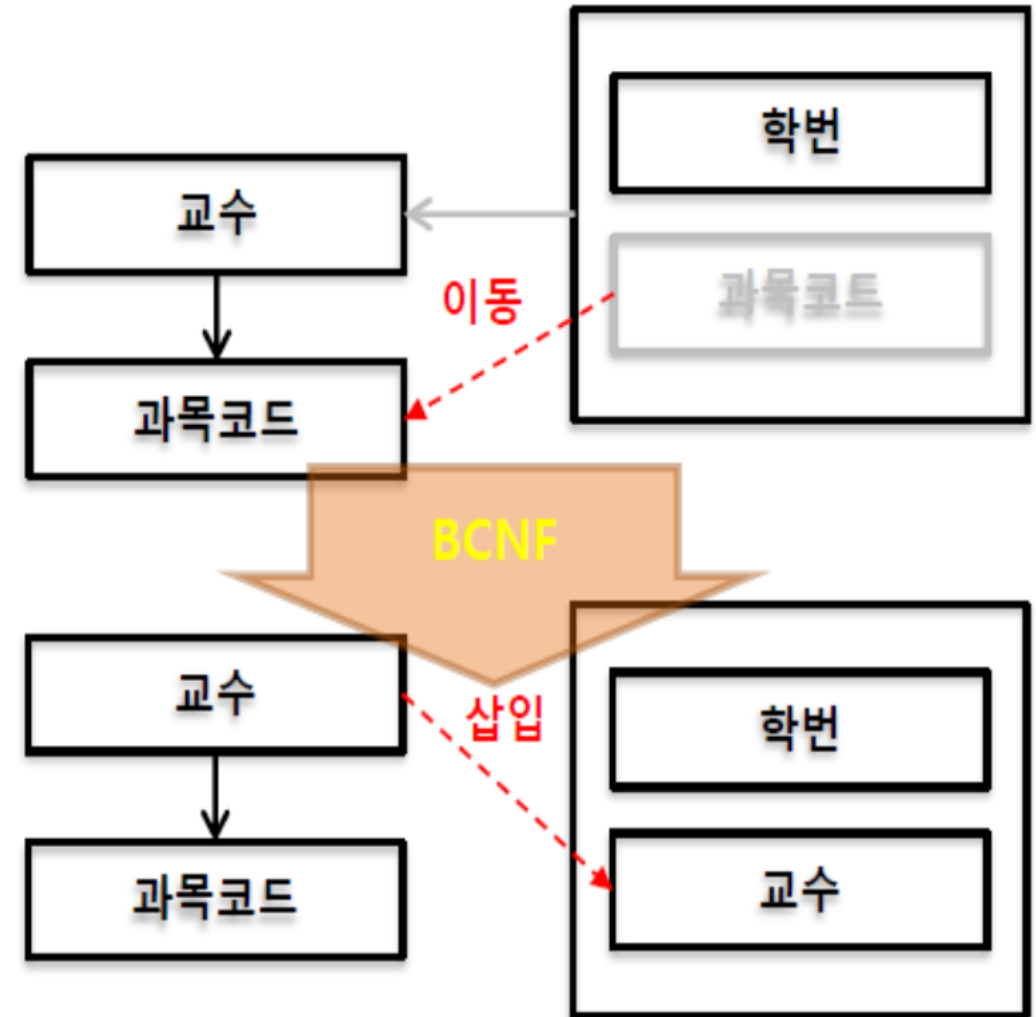
- 어떤 속성이 기본키에 대해 이행함수 종속성인 경우 이를 제거
 - 이행함수 종속성: 관계 R에서 속성 $A \rightarrow X$ 이고 $X \rightarrow Y$ 이면 $A \rightarrow Y$ 임



- 이 경우, 주문 테이블을 보면 '주문번호'가 결정자인데, 종속하는 속성들에서 '고객번호'가 '사업자번호'와 '고객우선순위'를 결정하고 있다. 따라서 (주문번호, 고객번호)를 키로 하는 테이블과 (고객번호)를 키로 하는 테이블로 분리해도 join에 의해서 주문번호->고객번호->사업자번호 등을 찾을 수 있기에, 두 테이블로 분리하라는 것



-해소 방법(BCNF 정규화)



- ▶ 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미
 - 일관성 : 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다.
 - 원자성 : 트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(RollBack) 되어야 한다. (All or Nothing)
 - 지속성 : 성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다.
 - 독립성 : 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없다.

ERD 란? 데이터 모델 표기법

개체 관계도

- 현실 세계에 존재하는 데이터와 그들 간의 관계를 사람이 이해할 수 있는 형태로 명확하게 표현하기 위해서 가장 널리 사용되고 있는 모델 // 개념적 모델링의 대표적인 도구

<새발표기법의 예>

	- 개체 A와 개체 B가 반드시 하나씩 존재 - 1 : 1
	- 개체 A는 하나가 존재하지만 개체 B는 없거나 한개만 존재 - 1 : 0 or 1 : 1
	- 개체 A가 하나가 존재하면 개체 B는 반드시 여러개 존재 - 1 : N
	- 개체 A가 하나 존재하는데 개체 B는 한개 이상 존재 - 1 : 1 or 1 : N
	- 개체 A가 하나 존재하는데 개체 B는 없거나 한개 존재하거나 여러개 존재 - 1 : 0 or 1 : 1 or 1 : N

관련된 TOOL.

ERD CLOUD

MY SQL WORKBENCH

EXERD

STAR UML

ERD MASTER







AQUERY TOOL

ERD 작업 순서

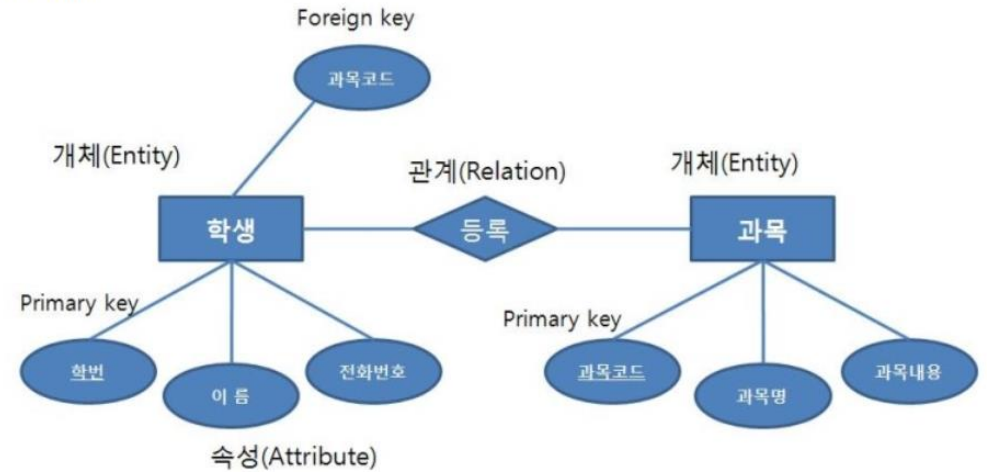
1. 엔터티를 그린다.
2. 엔터티를 적절하게 배치한다.
3. 엔터티간 관계를 설정한다.
4. 관계명을 기술한다.
5. 관계의 참여도를 기술한다.
6. 관계의 필수여부를 기술한다.

ERD

ERD 표준 기호

기 호	의 미
	개체
	속성
	기본키
	관계
	개체 타입과 속성을 연결
	개체간의 관계 타입

ERD예시



#관계타입의유형

- 1대1 관계: 개체 집합 A의 각 원소가 개체 집합 B의 원소 1개와 대응
- 예: 한명의 교수는 한과목만 강의하고, 한개의 과목은 한명의 교수에 의해 강의됨
- 1대다 관계: 개체 집합 A의 각 원소는 개체 집합 B의 원소 여러개와 대응할 수 있고, 개체 집합 B의 각 원소는 개체 집합 A의 원소 1개와 대응
- 예: 한학과에는 여러명의 학생이 있을 수 있고 한학생은 한개의 학과에 소속된다.
- 다대다 관계: 개체 집합 A의 각 원소는 개체 집합 B의 원소 여러개와 대응할 수 있고, 개체 집합 B의 각 원소는 개체 집합 A의 원소 여러개와 대응할 수 있음
- 예: 한명의 학생은 여러과목을 수강할 수 있고, 한과목은 여러명의 학생에 의해 수강되어질 수 있다.

감사합니다