

## DS 데이터분석 과정

### <기본 SQL학습 및 DB연동 프로그래밍>

일 자	2020년 8월 17일	과 정	JBFG 데이터분석(DS) 과정
강 사	김진수(bipycraft@gmail.com)	구 분	DB특강

## MariaDB 설치 및 DB 구축과정 실습

### 1. DBMS 개요와 MariaDB 소개

#### 1.1 DBMS 개요

##### 데이터베이스 정의

- 대용량의 데이터 집합을 체계적으로 구성해 놓은 것
- 통합, 저장, 운영, 공유

##### 데이터베이스 특징

- 데이터의 무결성 : 트랜잭션이 동시에 처리되는 동안에도 논리적 일관성을 보장
- 데이터의 독립성 : 응용P/G와 상호 의존적 관계 탈피 cf. 크기/저장소 변경시
- 보안 : 접근권한, 정보유출방지 cf. OOP의 Encapsulation, Scope관리
- 중복의 최소화 : 통합관리, 코드관리
- 응용P/G 제작 및 수정의 용이함
- 데이터의 안전성 : 백업/복원 기능으로 논리적 프로세스 단위의 수행

##### DBMS, 데이터베이스관리시스템 분류

- 계층형(Hierarchical) DBMS: 트리구조, 검색은 빠르나 구조 변경이 어려움
- 망형(Network) DBMS: 빠른 데이터 추출 가능하나, 아주 복잡한 구조
- 관계형(Relational) DBMS: 데이터의 효율적인 저장관리, Parent/Child 구조
- 객체지향형(Object-Oriented) DBMS,
- 객체관계형(Object-Relational) DBMS

SQL, Structured Query Language

- 관계형 데이터베이스에서 사용되는 언어
- 다양한 벤더사, 이식성/상호호환성 우수, 대화형 언어, 분산환경 C/S구조

참고. 데이터베이스를 제어 관리 위한 SQL

구분	종류	명령어
DDL	데이터 정의 언어 Data Definition	Create(정의), Alter(수정), Drop(삭제), Truncate(Drop 후 Create)
DML	데이터 조작 언어 Data Manipulation	Select(조회), Insert(추가), Delete(삭제), Update(변경), Lock, Explain, Call 등 대상 : 사용자, 테이블 및 테이블 스페이스
DCL	데이터 제어 언어 Transaction Control	Commit : 트랜잭션의 작업 결과를 반영 Rollback : 트랜잭션의 작업을 취소 및 원래대로 복구 Grant : 사용자에게 권한 부여 Revoke : 사용자 권한 취소

## 1.2 MariaDB 소개

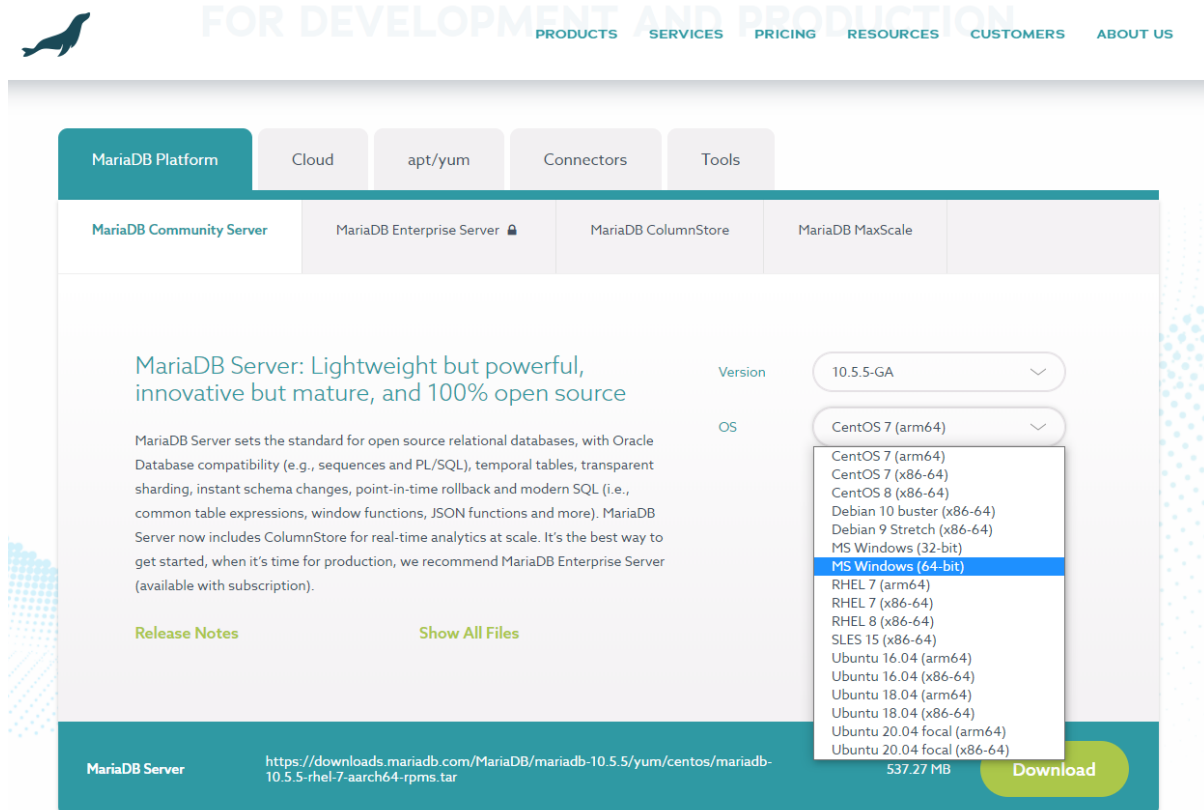
- RDBMS, Open Source로 제공
- 2010년 오라클이 썬마이크로시스템즈 인수  
→ JAVA, MySQL 오라클 소유, 즉 상용화
- MySQL 핵심창업자인 몬티 와이드니어스가 별도회사 설립 후 MariaDB 개발  
→ MariaDB 10.3 과 MySQL 8.0 버전과 기능이 대응 (2017.04)

## 2. MariaDB 설치



### MariaDB 설치

- MariaDB 최신버전 : ver10.x
- Download : <https://mariadb.com/downloads/>



The screenshot shows the MariaDB Platform website. At the top, there's a navigation bar with links: PRODUCTS, SERVICES, PRICING, RESOURCES, CUSTOMERS, and ABOUT US. Below this, a secondary navigation bar highlights 'MariaDB Platform' and includes tabs for Cloud, apt/yum, Connectors, and Tools. The main content area features a hero section for 'MariaDB Server' with the tagline 'Lightweight but powerful, innovative but mature, and 100% open source'. It describes the server's compatibility with Oracle Database and its inclusion of ColumnStore. A 'Version' dropdown is set to '10.5.5-GA', and an 'OS' dropdown is open, showing a list of operating systems including CentOS, Debian, MS Windows, RHEL, SLES, Ubuntu, and Fedora. The 'MS Windows (64-bit)' option is selected. At the bottom, there's a 'Download' button and a link to the download page.

MariaDB Platform

Cloud apt/yum Connectors Tools

MariaDB Community Server MariaDB Enterprise Server MariaDB ColumnStore MariaDB MaxScale

MariaDB Server: Lightweight but powerful, innovative but mature, and 100% open source

MariaDB Server sets the standard for open source relational databases, with Oracle Database compatibility (e.g., sequences and PL/SQL), temporal tables, transparent sharding, instant schema changes, point-in-time rollback and modern SQL (i.e., common table expressions, window functions, JSON functions and more). MariaDB Server now includes ColumnStore for real-time analytics at scale. It's the best way to get started, when it's time for production, we recommend MariaDB Enterprise Server (available with subscription).

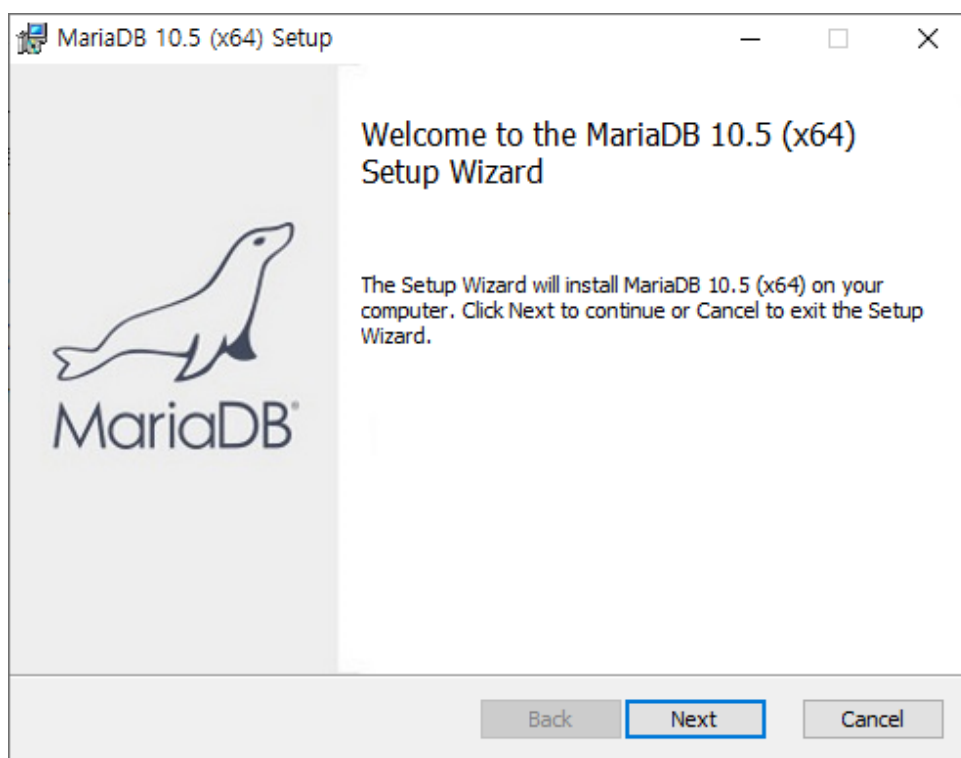
Release Notes Show All Files

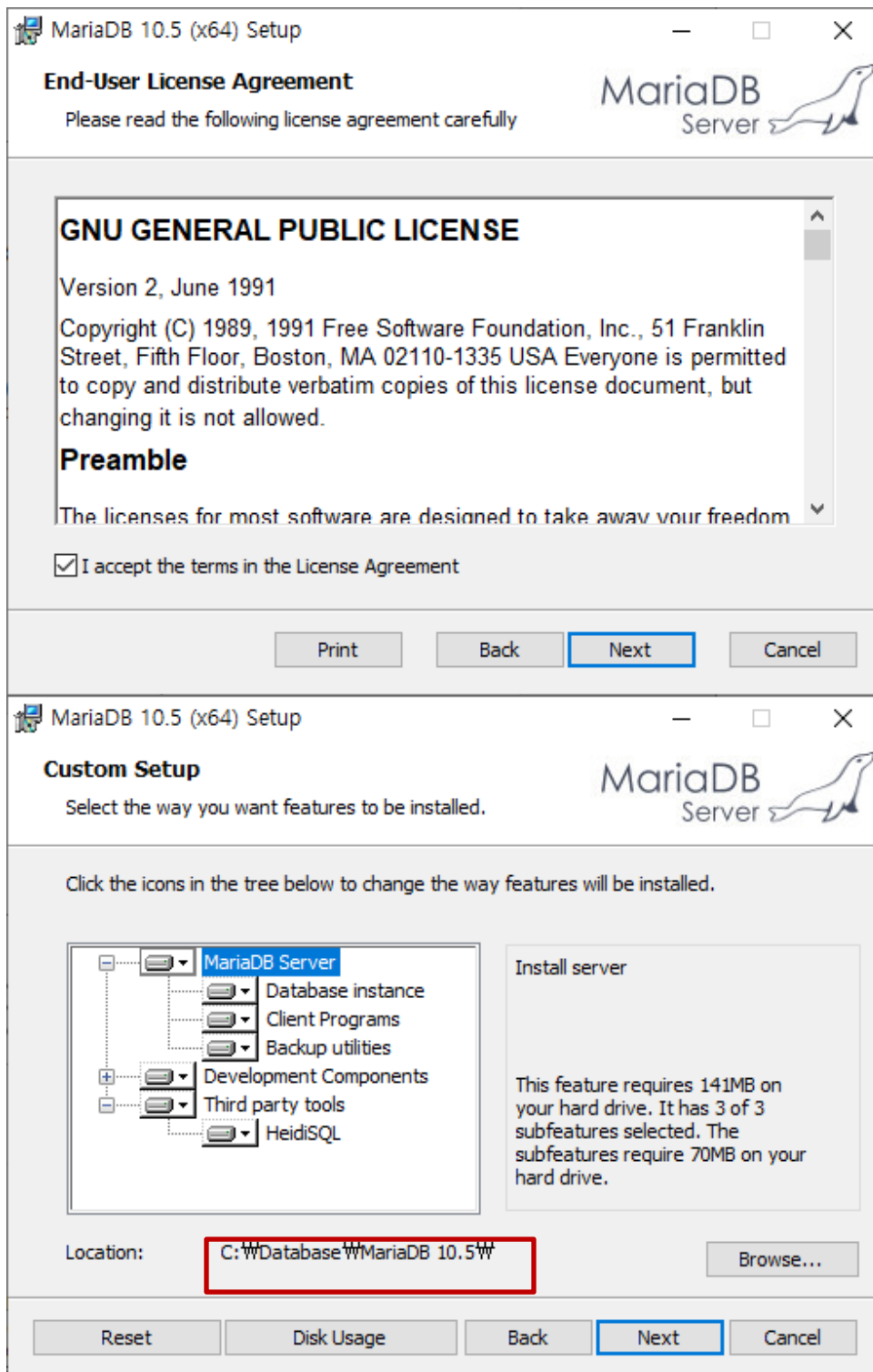
MariaDB Server <https://downloads.mariadb.com/MariaDB/mariadb-10.5.5/yum/centos/mariadb-10.5.5-rhel-7-aarch64-rpms.tar> 537.27 MB Download

Version: 10.5.5-GA

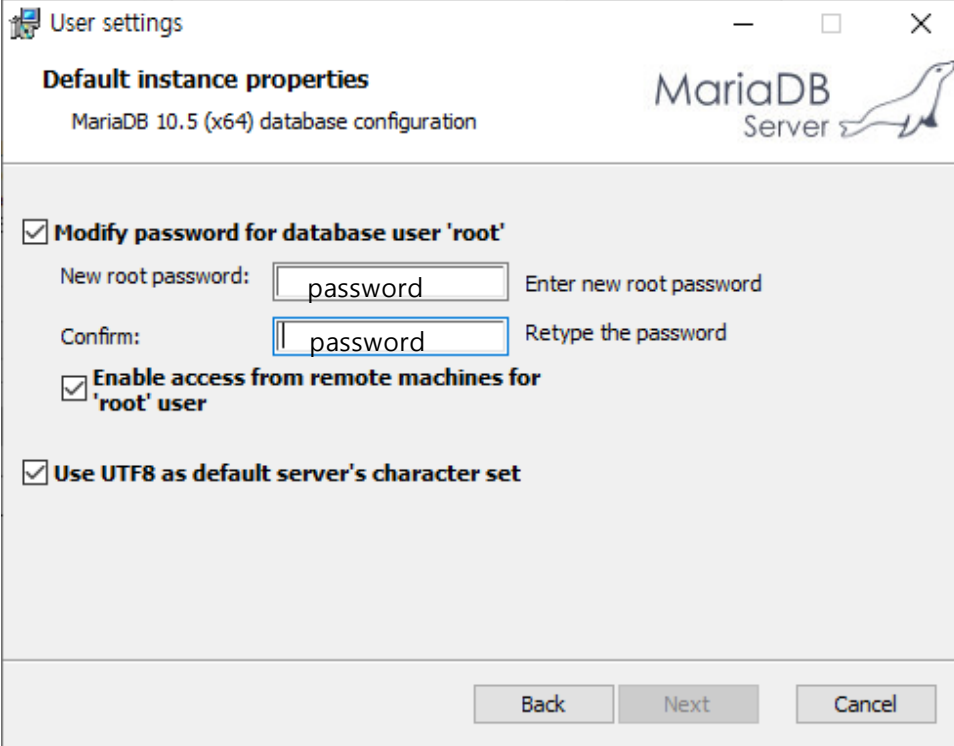
OS: CentOS 7 (arm64)

- CentOS 7 (arm64)
- CentOS 7 (x86-64)
- CentOS 8 (x86-64)
- Debian 10 buster (x86-64)
- Debian 9 Stretch (x86-64)
- MS Windows (32-bit)
- MS Windows (64-bit)
- RHEL 7 (arm64)
- RHEL 7 (x86-64)
- RHEL 8 (x86-64)
- SLES 15 (x86-64)
- Ubuntu 16.04 (arm64)
- Ubuntu 16.04 (x86-64)
- Ubuntu 18.04 (arm64)
- Ubuntu 18.04 (x86-64)
- Ubuntu 20.04 focal (arm64)
- Ubuntu 20.04 focal (x86-64)





cf. 설치위치 : 유지보수관리, 백업관리 용이하기 위한 디렉토리에 설치



The 'User settings' window for MariaDB 10.5 (x64) database configuration. It features the MariaDB Server logo in the top right. The 'Default instance properties' section includes three checked options: 'Modify password for database user 'root'', 'Enable access from remote machines for 'root' user', and 'Use UTF8 as default server's character set'. The password fields are pre-filled with 'password'. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

User settings

Default instance properties

MariaDB 10.5 (x64) database configuration

MariaDB Server

☒ **Modify password for database user 'root'**

New root password:  Enter new root password

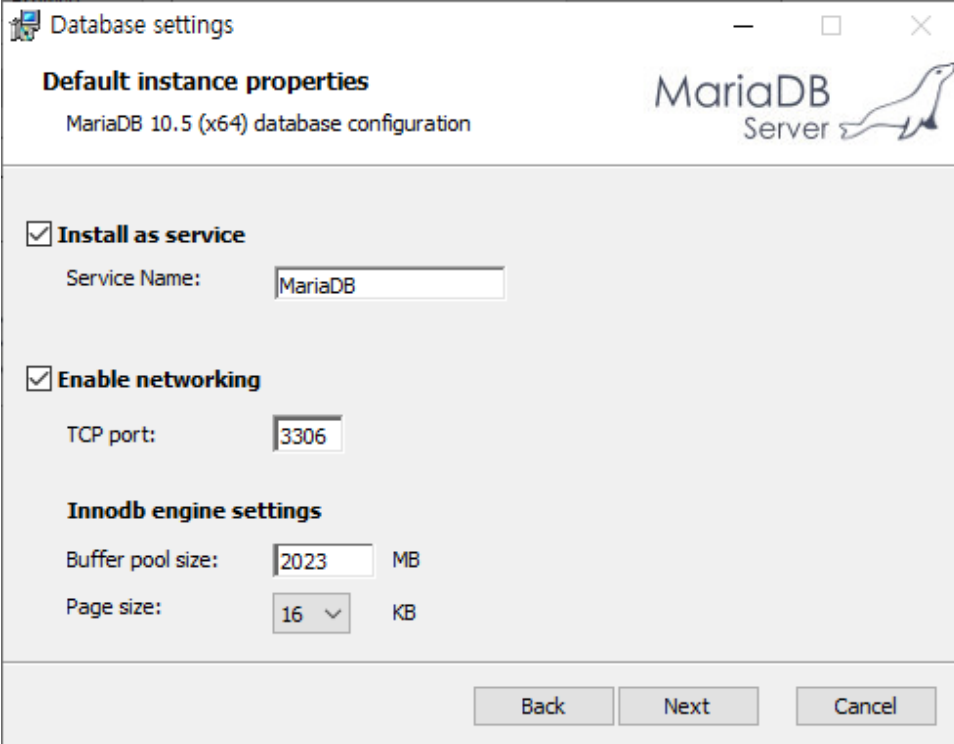
Confirm:  Retype the password

☒ **Enable access from remote machines for 'root' user**

☒ **Use UTF8 as default server's character set**

Back Next Cancel

cf. 패스워드 : password 라고 입력 → 보안철저 & 기억하기 쉽게, 까 먹으면 안됨!!



The 'Database settings' window for MariaDB 10.5 (x64) database configuration. It features the MariaDB Server logo in the top right. The 'Default instance properties' section includes two checked options: 'Install as service' (with 'Service Name' set to 'MariaDB') and 'Enable networking' (with 'TCP port' set to '3306'). The 'InnoDB engine settings' section shows 'Buffer pool size' as '2023 MB' and 'Page size' as '16 KB'. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

Database settings

Default instance properties

MariaDB 10.5 (x64) database configuration

MariaDB Server

☒ **Install as service**

Service Name:

☒ **Enable networking**

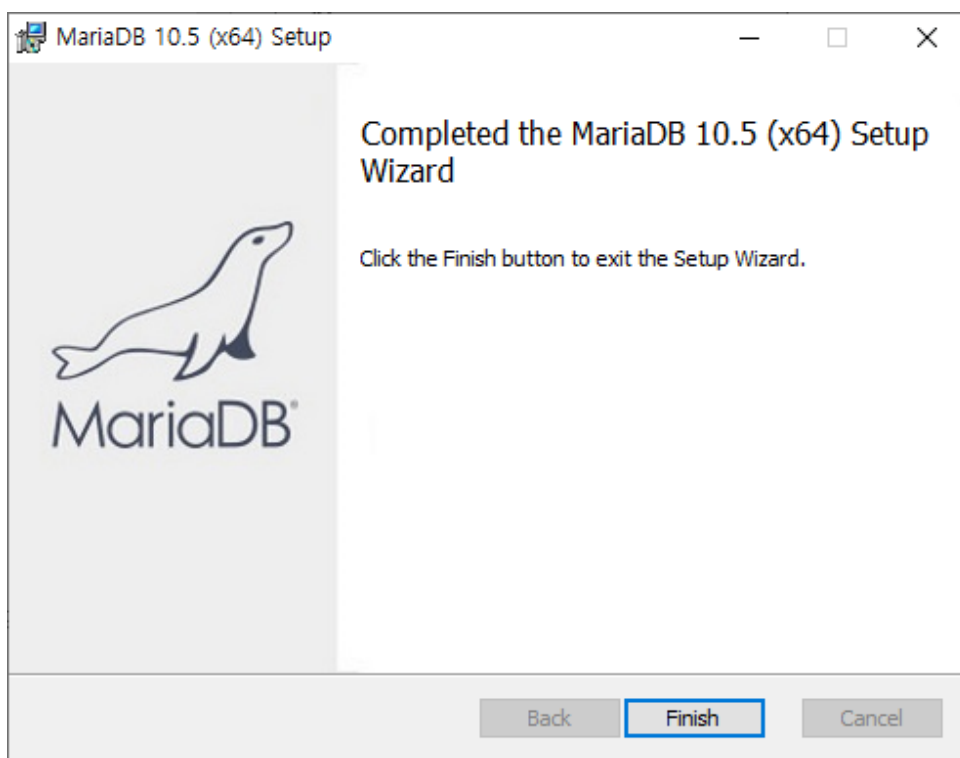
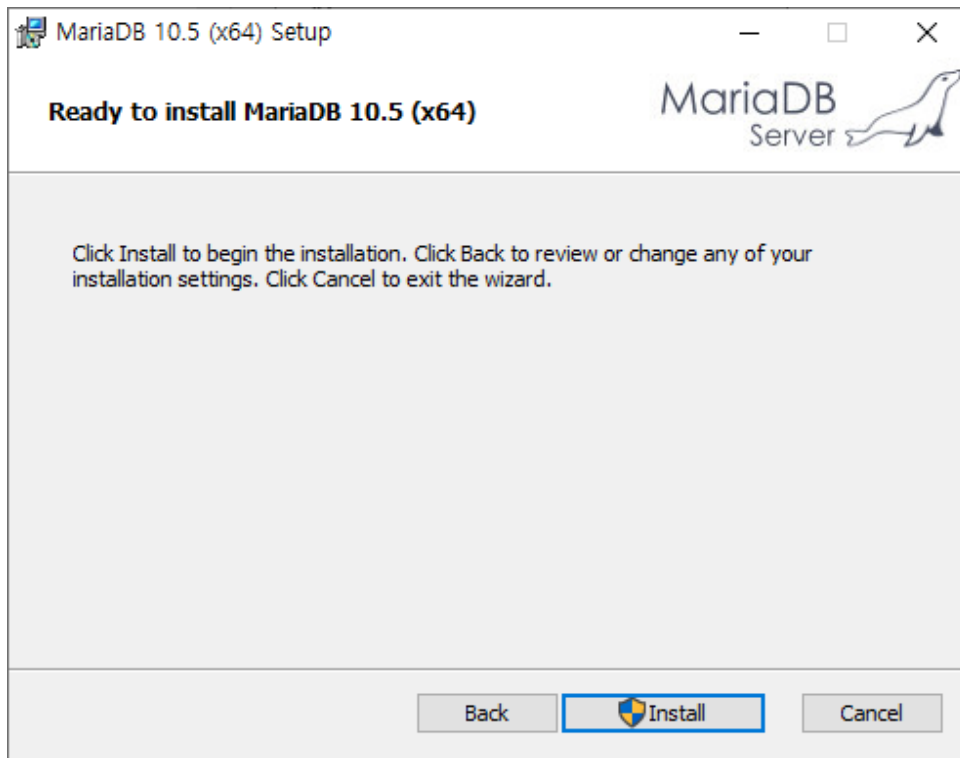
TCP port:

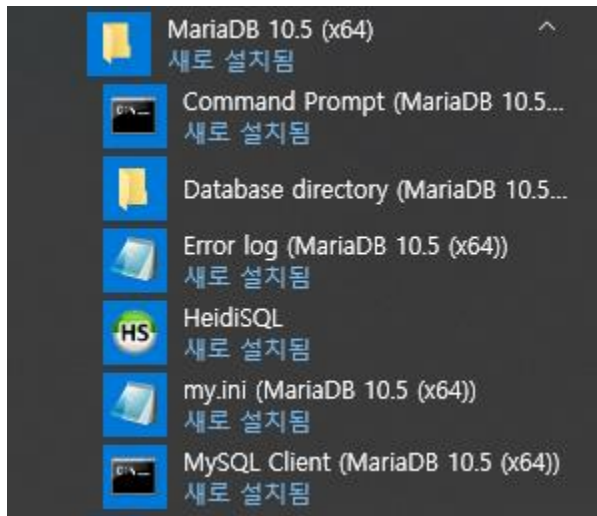
**InnoDB engine settings**

Buffer pool size:  MB

Page size:  KB

Back Next Cancel

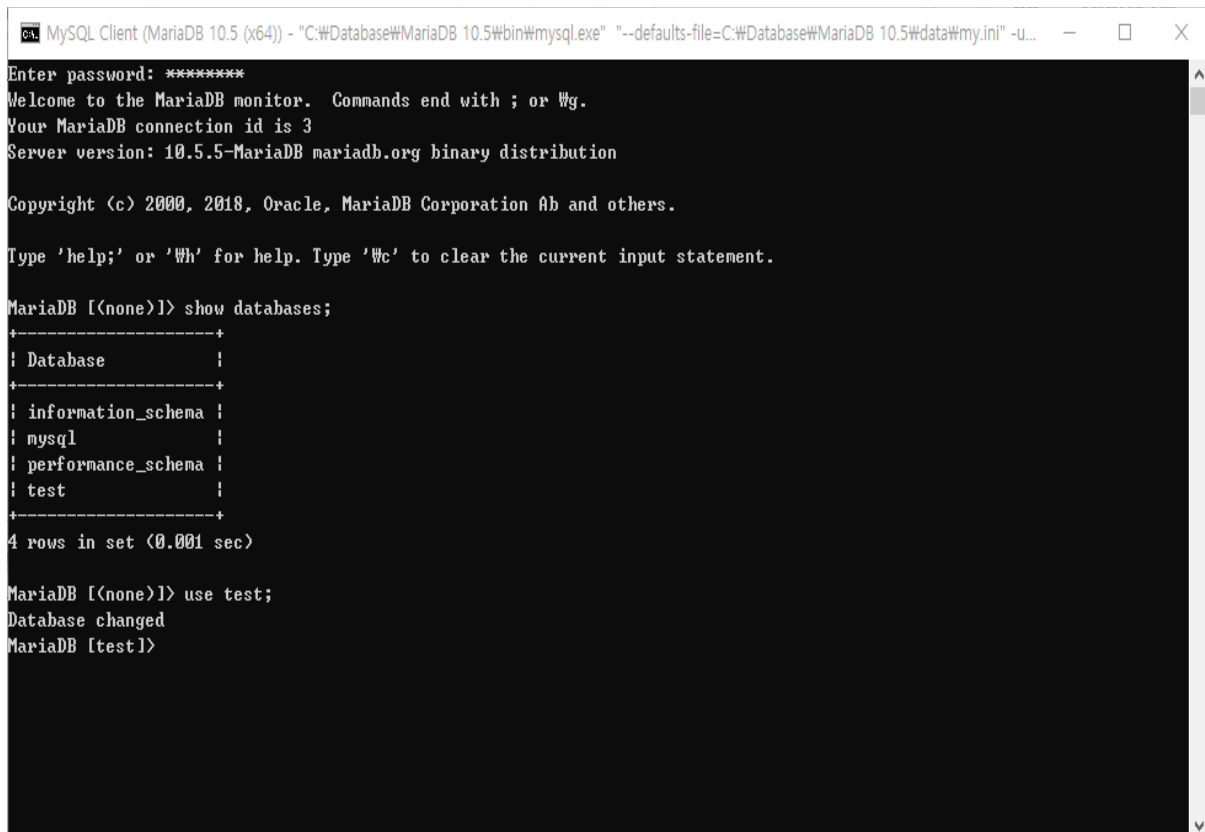






## 실습 : MariaDB 실행

### # MySQL Client Command 실행



```
MySQL Client (MariaDB 10.5 (x64)) - "C:\\Database\\MariaDB 10.5\\bin\\mysql.exe" --defaults-file=C:\\Database\\MariaDB 10.5\\data\\my.ini" -u...
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \\g.
Your MariaDB connection id is 3
Server version: 10.5.5-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\\h' for help. Type '\\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]> use test;
Database changed
MariaDB [test]>
```

```
MariaDB (none)> show databases;
MariaDB (none)> use mysql
MariaDB (mysql)> show tables;
MariaDB (mysql)> desc user;
MariaDB (mysql)> SELECT * FROM user;
MariaDB (mysql)> SELECT count(*) FROM user;
MariaDB (mysql)> SELECT host, user, password
-> FROM user;
```

## # 기본 Commnad 창에서 실행

```
C:\Windows\system32\cmd.exe - mysql -u root -p

C:\Database\MariaDB 10.5\bin>mysql -u root -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.5.5-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

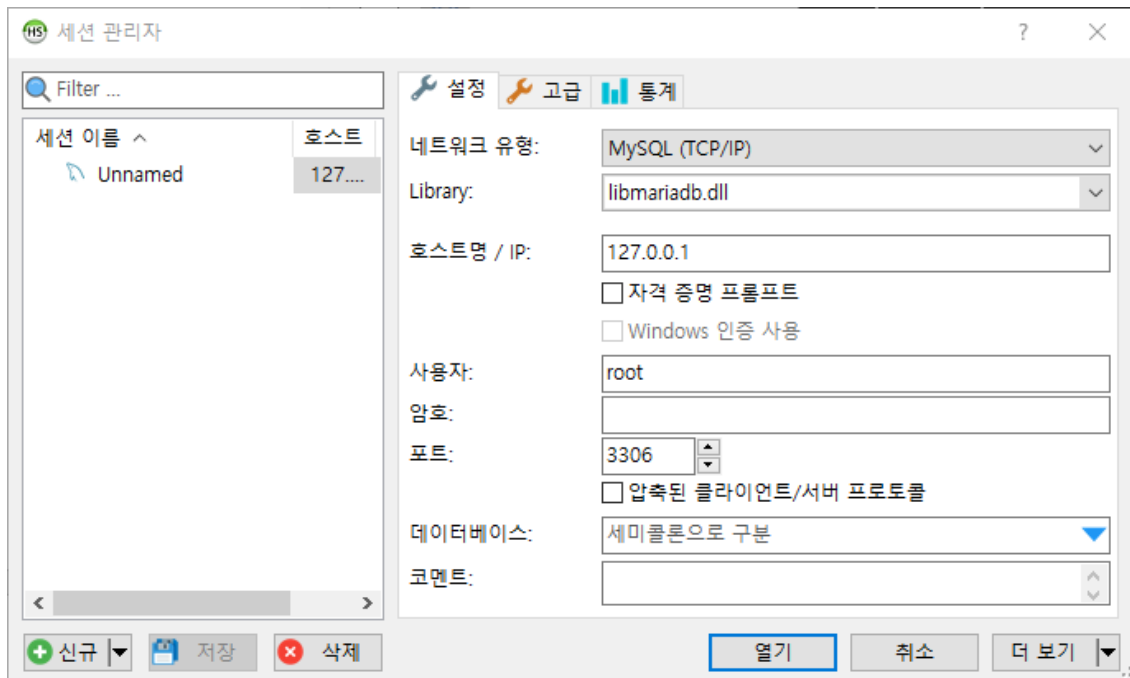
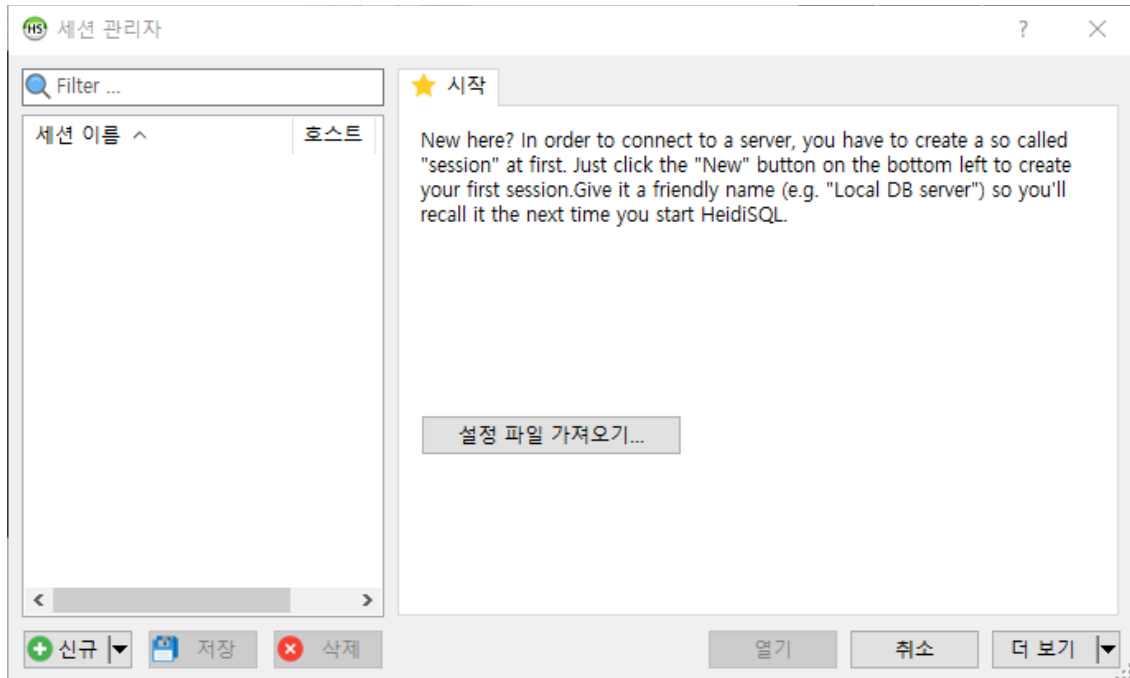
MariaDB [(none)]>
```

C:\MaridB\bin> mysql -u root -p

## # DBMS를 추천

DBMS Tools	Brand	Download
 <b>HeidiSQL</b>	하이드SQL HeidiSQL	<a href="https://www.heidisql.com/download.php">https://www.heidisql.com/download.php</a>
 <b>DBeaver Community</b> <small>Free Universal Database Tool</small>	디비버, DBeaver	<a href="https://dbeaver.io/download/">https://dbeaver.io/download/</a>
 <b>Toad</b> <small>by Quest</small> Downloads	토드, Toad	<a href="https://www.toadworld.com/downloads">https://www.toadworld.com/downloads</a>
	오렌지, Orange	<a href="https://orange.biolab.si/download">https://orange.biolab.si/download</a>

## # HeidiSQL 실행



## 테이블 생성

CREATE TABLE 테이블 생성, 제약조건(CONSTRAINT), 확인(DESC)

### 1. 테이블 생성 구문 형식

```
CREATE TABLE 테이블 이름 (  
    컬럼명1 DATATYPE [DEFAULT 형식],  
    컬럼명2 DATATYPE [DEFAULT 형식],  
    컬럼명3 DATATYPE [DEFAULT 형식]  
);
```

- 테이블명은 객체를 의미할 수 있는 적절한 이름을 사용한다. 가능한 단수형을 권고한다.
- 테이블명은 다른 테이블의 이름과 중복되지 않아야 한다.
- 한 테이블 내에서는 컬럼명이 중복되게 지정될 수 없다.
- 테이블 이름을 지정하고 각 컬럼들은 괄호 "(" 로 묶어 지정한다.
- 각 컬럼들은 콤마 ","로 구분되고, 테이블 생성문의 끝은 항상 세미콜론 ";"로 끝난다.
- 컬럼에 대해서는 다른 테이블까지 고려하여 데이터베이스 내에서는 일관성 있게 사용하는 것이 좋다.
- 컬럼 뒤에 데이터 유형은 꼭 지정되어야 한다.
- 테이블명과 컬럼명은 반드시 문자로 시작해야 하고, 벤더별로 길이에 대한 한계가 있다.
- 벤더에서 사전에 정의한 예약어(Reserved word)는 쓸 수 없다.
- A-Z, a-z, 0-9, \_ \$, # 문자만 허용된다.

## 2. CREATE TABLE 예제

```
CREATE TABLE PLAYER (
    PLAYER_ID          CHAR(7)          NOT NULL,
    PLAYER_NAME        VARCHAR2(20)     NOT NULL,
    TEAM_ID            CHAR(3)          NOT NULL,
    E_PLAYER_NAME      VARCHAR2(40),
    NICKNAME            VARCHAR2(30),
    JOIN_YYYY          CHAR(4),
    POSITION            VARCHAR2(10),
    BACK_NO            NUMBER(2),
    NATION              VARCHAR2(20),
    BIRTH_DATE          DATE,
    SOLAR              CHAR(1),
    HEIGHT              NUMBER(3),
    WEIGHT              NUMBER(3),

    CONSTRAINT PLAYER_PK PRIMARY KEY (PLAYER_ID),
    CONSTRAINT PLAYER_FK FOREIGN KEY (TEAM_ID) REFERENCES TEAM(TEAM_ID)
);
```

- 테이블 생성시 대/소문자 구분은 하지 않는다. (기본적으로 테이블이나 컬럼명은 대문자로 만들어진다.)
- DATE 유형은 별도로 크기를 지정하지 않는다.
- 문자 데이터 유형은 반드시 가질 수 있는 최대 길이를 표시해야 한다.
- 컬럼과 컬럼의 구분은 콤마로 하되, 마지막 컬럼은 콤마를 찍지 않는다.
- 컬럼에 대한 제약조건이 있으면 CONSTRAINT를 이용하여 추가할 수 있다.

제약조건은 PLAYER\_NAME, TEAM\_ID 컬럼의 데이터 유형 뒤에 NOT NULL을 정의한 것과 같이 컬럼 LEVEL 방식과, CONSTRAINT로 테이블 생성 마지막에 모든 제약조건을 기술하는 테이블 LEVEL 방식이 있다.

하나의 SQL 문장내에 두 가지 방식을 혼용해서 사용할 수 있으며, 같은 기능을 가진다.

### 3. 제약조건 (CONSTRAINT)

- 제약조건은 사용자가 원하는 조건의 데이터만 유지하기 위한 특정 컬럼에 설정하는 제약이다.
- 테이블을 생성할 때 제약조건을 반드시 기술할 필요는 없다.

구분	설명
PRIMARY KEY (기본키)	<ul style="list-style-type: none"> <li>- 테이블에 저장된 행 데이터를 고유하게 식별하기 위한 기본키 정의.</li> <li>- 하나의 테이블에 하나의 기본키 제약만 정의할 수 있다.</li> <li>- 기본키 제약을 정의하면 DBMS는 자동으로 UNIQUE 인덱스를 생성하며, 기본키를 구성하는 컬럼에는 NULL을 입력할 수 없다.</li> </ul> <p>* PRIMARY KEY = UNIQUE KEY &amp; NOT NULL</p>
UNIQUE KEY (고유키)	<ul style="list-style-type: none"> <li>- 테이블에 저장된 행 데이터를 고유하게 식별하기 위한 고유키를 정의한다.</li> </ul> <p>단, NULL은 고유키 제약의 대상이 아니므로, NULL 값을 가진 행이 여러 개가 있더라도 고유키 제약 위반이 되지 않는다.</p>
NOT NULL	<ul style="list-style-type: none"> <li>- NULL 값의 입력을 금지한다. 디폴트 상태에서는 모든 컬럼에서 NULL을 허가하고 있지만, 이 제약을 지정함으로써 해당 컬럼은 입력 필수가 된다.</li> </ul>
CHECK	<ul style="list-style-type: none"> <li>- 입력할 수 있는 값의 범위 등을 제한한다. CHECK 제약으로는 TRUE or FALSE로 평가할 수 있는 논리식을 지정한다.</li> </ul>
FOREIGN KEY	<ul style="list-style-type: none"> <li>- 관계형 데이터베이스에서 테이블 간의 관계를 정의하기 위해 기본키를 다른 테이블의 외래키로 복사하는 경우 외래키가 생성된다.</li> <li>- 외래키 지정시 참조 무결성 제약 옵션을 선택할 수 있다.</li> </ul>

### 4. 생성된 테이블 구조 확인 (DESC)

- 테이블을 생성한 후 테이블의 구조가 제대로 만들어졌는지 확인할 필요가 있다.
- DESCRIBE 또는 DESC 로 테이블 정보를 확인할 수 있다.

```
DESC PLAYER;
```

## 5. SELECT 문장을 통한 테이블 생성

- SELECT 문장을 통해 테이블을 생성할 수도 있다.
- 기존의 테이블을 이용하면 컬럼별로 데이터 유형을 다시 정의하지 않아도 된다.
- 하지만, 기존 테이블의 제약조건 중 NOT NULL만 새로운 복제 테이블에 적용이 되고, PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, CHECK 등의 다른 제약 조건은 없어진다.
- 제약 조건을 추가하기 위해서는 ALTER TABLE 을 사용해야 한다.

```
CREATE TABLE TEMP_PLAYER  
AS SELECT * FROM PLAYER ;
```

## 실습 : Table Creation in MariaDB

### Syntax

[CONSTRAINT [symbol]] constraint\_expression

constraint\_expression:

- | PRIMARY KEY [index\_type] (index\_col\_name, ...) [index\_option] ...
- | FOREIGN KEY [index\_name] (index\_col\_name, ...)  
REFERENCES tbl\_name (index\_col\_name, ...)  
[ON DELETE reference\_option]  
[ON UPDATE reference\_option]
- | UNIQUE [INDEX|KEY] [index\_name]  
[index\_type] (index\_col\_name, ...) [index\_option] ...
- | CHECK (check\_constraints)

index\_type:

USING {BTREE | HASH | RTREE}

index\_col\_name:

col\_name [(length)] [ASC | DESC]

index\_option:

- | KEY\_BLOCK\_SIZE [=] value
- | index\_type
- | WITH PARSER parser\_name
- | COMMENT 'string'
- | CLUSTERING={YES|NO}

reference\_option:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT



MySQL에는 그 특성에 따라 여러 종류의 **데이터베이스 엔진**(스토리지 엔진 이라고도 함)이 존재합니다. 가장 많이 사용하고 있는 엔진은 MyISAM과 InnoDB 이다.

기존 MyISAM이 MySQL의 기본 엔진이었다면 **MySQL 5.5부터는 InnoDB가 기본 엔진**으로 바뀌었습니다.

#### # 데이터베이스 엔진 특징 비교

InnoDB	MyISAM
<ul style="list-style-type: none"> <li>➤ <b>트랜잭션 지원</b></li> <li>➤ 빈번한 쓰기, 수정, 삭제시 처리 능력</li> <li>➤ 디스크, 전원 등의 장애 발생시 복구 성능</li> <li>➤ <b>동시처리</b>가 많은 환경에 적합</li> <li>➤ Row 단위 락킹</li> </ul>	<ul style="list-style-type: none"> <li>➤ <b>상대적으로 높은 성능</b></li> <li>➤ 읽기 위주의 요청에 유리</li> <li>➤ 테이블 단위 락킹</li> </ul>

사실 여러면에서 InnoDB가 좋지만 **성능면에서 MyISAM이 더 좋고**, MySQL이 프리웨어 데이터베이스로 많이 활용되면서 웹 게시판과 같은 단일 트랜잭션 환경 또는 대규모 동시 처리에 대한 요구가 없는 환경에서 사용되다 보니 기본 엔진이 MyISAM이 제격이다.

오라클(Oracle)에 인수 된 이후 엔터프라이즈 용으로도 많이 사용되면서 InnoDB가 더 각광을 받게 되듯 ~~

**MyISAM은 트랜잭션 지원이 안되기 때문에 여러 SQL문을 실행한 후 commit 또는 rollback 하는 기능이 없고, 테이블 단위로 락이 걸리기 때문에 테이블에 두 가지 이상의 데이터를 동시에 insert/update할 수 없습니다.**

대신 기준정보 테이블 처럼 데이터가 거의 고정되어 있고 읽기가 많이 발생하는 경우에는 그 성능 효과를 충분히 발휘할 수 있습니다. 회사의 조직도는 주로 LDAP에 저장하겠지만, MySQL을 사용한다면 InnoDB보다는 MyISAM엔진이 적합

다행인 것은, MySQL 인스턴스 안에 두 가지 엔진을 섞어 쓸 수 있습니다. 테이블을 생성할 때 지정해주면 되는데요,

```
CREATE TABLE company_org (org_code INT, org_name CHAR (100)) ENGINE=InnoDB;
```

물론 5.5 부터는 기본 엔진이 InnoDB 니까 안써줘도 되겠고, MyISAM은 "ENGINE = MYISAM"으로 지정하면 됩니다.

예시 :

#### Sample1. Creation Table

```
CREATE TABLE product (
  category VARCHAR(10) NOT NULL,
  id        VARCHAR(20) NOT NULL,
  price     INT,
  PRIMARY KEY(category, id)
) ENGINE=INNODB;

CREATE TABLE customer (
  id        VARCHAR(20) NOT NULL,
  name      VARCHAR(20) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=INNODB;

CREATE TABLE product_order (
  seq_no      INT NOT NULL AUTO_INCREMENT,
  prod_category VARCHAR(10) NOT NULL,
  prod_id     VARCHAR(20) NOT NULL,
  cust_id     VARCHAR(20) NOT NULL,
  PRIMARY KEY(seq_no),
  INDEX (prod_category, prod_id),
  FOREIGN KEY (prod_category, prod_id)
    REFERENCES product(category, id)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  INDEX (cust_id),
  FOREIGN KEY (cust_id)
    REFERENCES customer(id)
) ENGINE=INNODB;
```

### Sample2. Numeric constraints and comparisons

```
CREATE TABLE temp1 (  
    a INT CHECK (a>2),  
    b INT CHECK (b>2),  
    CONSTRAINT a_greater CHECK (a>b)  
);  
  
INSERT INTO temp1(a) VALUES (1);  
  
INSERT INTO temp1(a,b) VALUES (3,4);  
  
INSERT INTO temp1(a,b) VALUES (4,3);  
  
SELECT * FROM temp1;  
  
-- Dropping a constraint  
ALTER TABLE temp1  
    DROP CONSTRAINT b_greater CHECK (a<b);  
  
INSERT INTO temp1(a,b) VALUES (3,4);  
  
SELECT * FROM temp1;  
  
-- Adding a constraint;  
ALTER TABLE temp1  
    ADD CONSTRAINT different CHECK (a != b);  
  
INSERT INTO temp1(a,b) VALUES (7,7);  
INSERT INTO temp1(a,b) VALUES (6,9);  
  
SELECT * FROM temp1;
```

### 3. MariaDB 전체 운영 실습

#### 3.1 요구사항 분석과 시스템 설계 그리고 모델링

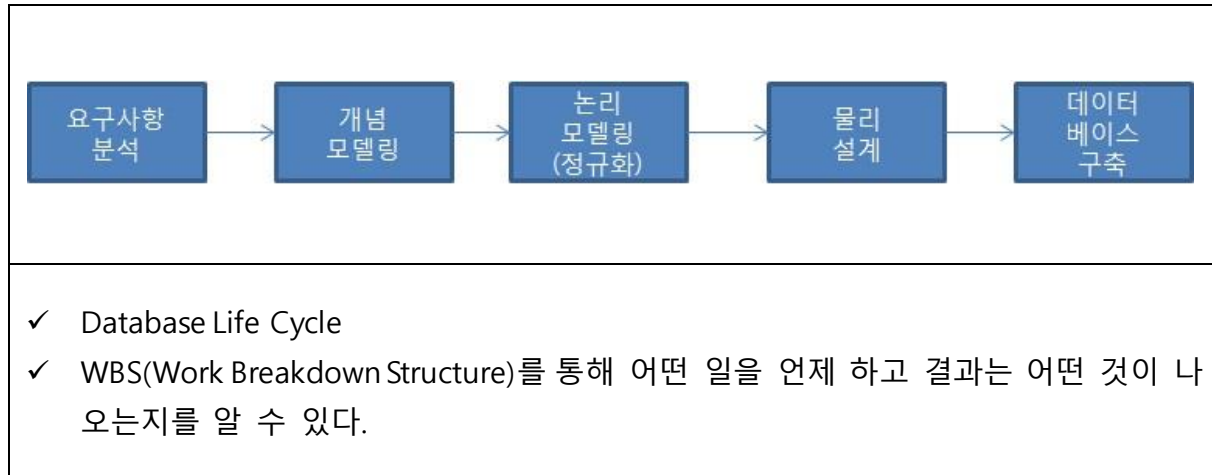
##### # 정보시스템 구축 절차

절차	내용	산출물
분석	시스템분석, 요구사항분석	DB설계서(개념모델링), AS-IS분석서
설계	시스템아키텍처, 데이터모델링, 프로그램설계, 프로토타입시안	DB설계서(논리모델링), 기능정의서, 화면설계서 (UI/UX)
구현	모듈개발 : Front-End, Back-End, Common 고려사항 : 환경셋팅, 운영관리, 배포관 리, 백업관리	DB설계서(물리모델링), 소스코드, 운영메뉴얼
테스트	유형: 단위, API, 통합, 시스템, 설치테스 트 테스트종료기준: 00%이상시 검수완료 → 고객과 합의 ( 통과TestCase/실행TestCase )x100 %	테스트 체크리스트, 테스트 케이스, 테스트 시나리오
유지보수	결함 수정, 성능 및 기타 다른 속성 개 선 변화환경에 적응을 목적으로 이루어지는 변경관리	버전관리, 형상관리

- 분석단계 : What을 결정, 설계단계:How를 정의
- 구현단계 : 이해하기 쉽고, 관리가 용이하고, 확장성을 고려
- 테스트단계 : 소프트웨어의 결함의 찾는 것
- 유지보수관리가 SW Life Cycle의 80% 이상을 차지

### 3.2 MariaDB를 이용한 데이터베이스 구축 절차

#### # 데이터베이스 구축 절차, Database Life Cycle



#### STEP1. 요구사항 분석 단계

- 데이터베이스에서 관리해야 하는 데이터를 도출하고 분석하는 단계
- 요구 사항은 업무를 수행하는데 필요한 데이터에 대한 요구/데이터 구조에 대한 요구/성능에 대한 요구 일수도 있다.
- 요구 사항은 사용자의 의견을 최우선으로 따른다.
- 현업과 인터뷰를 통해 도출 한다.

#### STEP2. 개념 모델링

- 개념 모델을 구축
- 개념 모델은 요구 사항을 분석하고 나서 도출되는 데이터 측면의 결과물이다.
- 개념 모델은 요구 사항을 개념적으로 반영한 모델이다.
- 개념 모델링 단계에서는 핵심 데이터를 대상으로 모델링을 수행해야 하며 통합된 모델이 도출돼야 한다.

#### STEP3. 논리모델링(정규화)

- 정규화는 데이터의 분해를, 일반화는 데이터의 통합을 의미한다.
- 논리 모델링 단계는 핵심 데이터를 포함한 모든 데이터를 대상으로 모델링을 수행하는 단계이다.

- 정규화는 함수 종속(Functional Dependency)에 의해 데이터를 분해하는 것이다. 이 단계에서는 더 분해할 수 없는 엔터티의 모습이 나타나게 된다.
- 개념 모델에서 도출된 엔터티는 실상 엔터티보다 더 큰 개념일 수도 있고 그대로 엔터티가 될 수도 있다.
- 정규화를 거쳐 분해된 엔터티는 엔터티 그 자체이다.
- 이 엔터티가 물리 설계 단계에서 목적에 의해 하나의 테이블로 합쳐지거나 두 개 이상의 테이블로 분리 될 수 있다.
- 엔터티는 식별자와 식별자가 아닌 속성 사이의 의존성에 의해서 분해 된다.
- 데이터가 통합된 모습에서 함수 종속에 의해 속성이 분해될 뿐이다.
- 이렇게 더 분해되지 않도록 최대한 분해된 모델을 정규형이라고 하고 정규화된 모델을 논리 모델(Logical Model) 이라고 한다.
- 엔터티를 정규화하면 데이터 무결성은 높아진다.

#### STEP4. 물리 설계

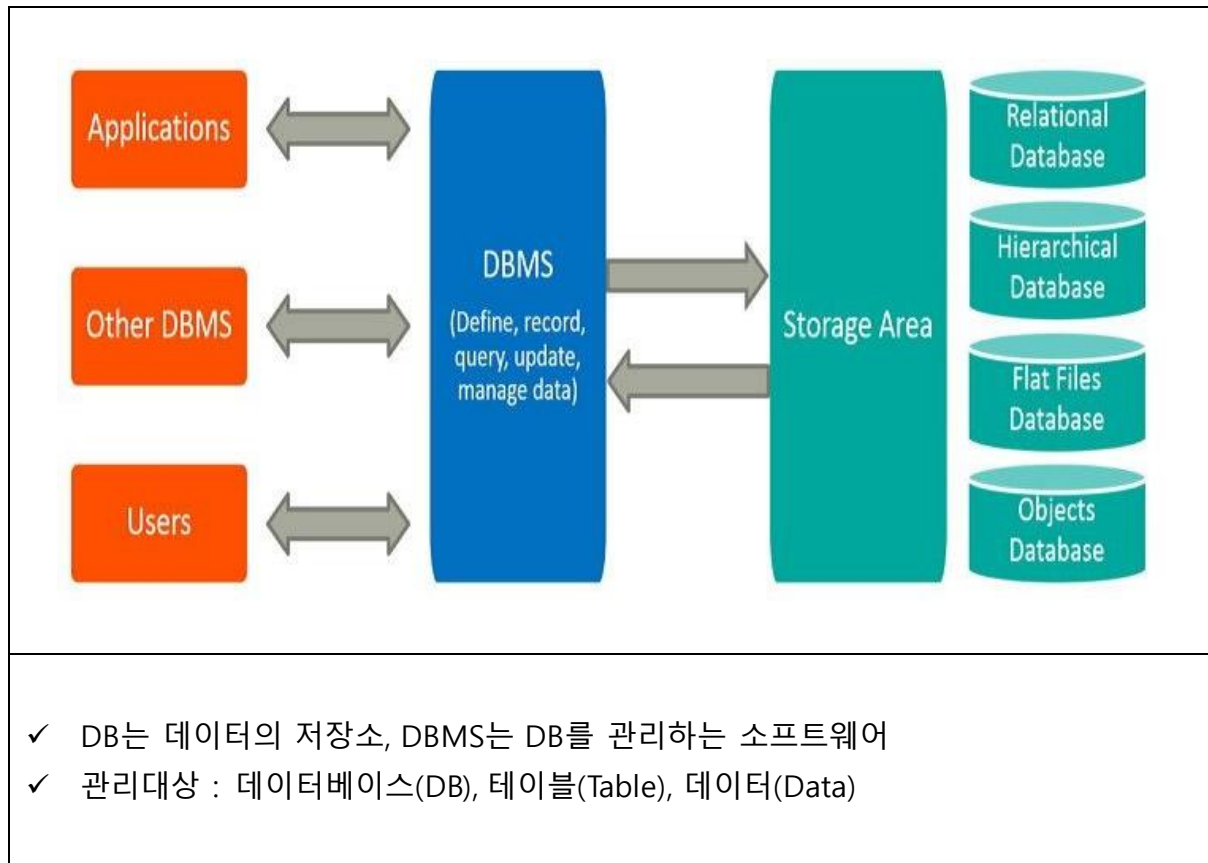
- 논리 모델을 물리 모델로 매핑하고 목적에 따라 테이블을 분해하거나 합치는 작업을 한다.
- 이 단계에서 가능한 성능을 최적화해야 한다.
- 성능을 위해서 중요한 작업이 비 정규화 이다.
- 정규화는 반드시 거쳐야 하는 필수 과정이고 정규화가 완전히 끝나야 비정규형을 고려할 수 있다.
- 인덱스 설계가 포함된다.
- 인덱스 설계는 대단히 중요한데 물리 설계 단계에서 완전하게 이루어지지 않고 파티셔닝 전략을 세우고 많은 테스트를 거쳐야 하며
- 클러스터링이나 IOT등의 테이블 타입도 시중하게 고려해야 한다.

#### STEP5. 데이터베이스 구축

- 물리 설계에서 도출된 여러 객체를 생성하는 단계
- 물리 설계에서 스크립트가 나오므로 이 단계에서는 모델러 보다는 보통 DBA가 수행하게 된다.
- 데이터베이스가 구축되고 나서는 데이터가 적재되고 운영된다
- 운영하면서 문제점이 발생하면 방안을 찾아 대처해야 하며 요구 사항이 추가되거나 변경되면서 모델 변경 관리를 하게 된다.

## Review. 데이터베이스 용어

### # DBMS(Database Management System)



### # 데이터모델링 : 개념적 / 논리적 / 물리적 데이터 모델

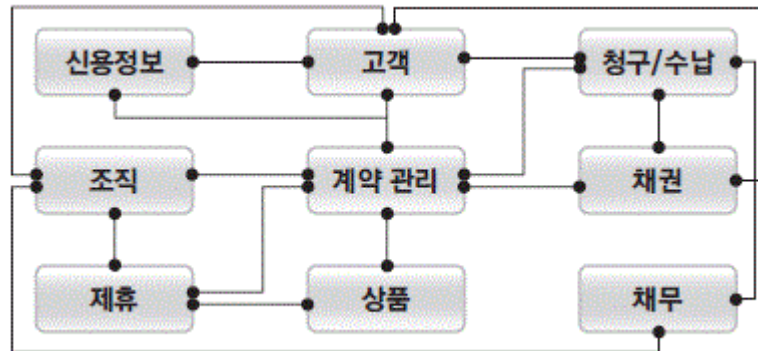
#### 1. 데이터모델링의 개념

- DB를 구축하고자 하는 대상이 되는 기관에서 사용되는 데이터를 분석하여 제약 조건을 체계적으로 정의하고 개념적인 도구를 이용하여 간결하고 이해하기 쉽게 표현하는 것
- 디테일한 설계도라기 보다는 나중에 건물이 어떤 식으로 보일까 하는 조감도라고 보는게 적절
- 개념 -> 논리 -> 물리적 데이터모델은 단순한 밑그림에서 부터 구체적으로 데이터베이스에 데이터를 넣는 마지막 단계까지 플로우에 따라 진행이 된다.

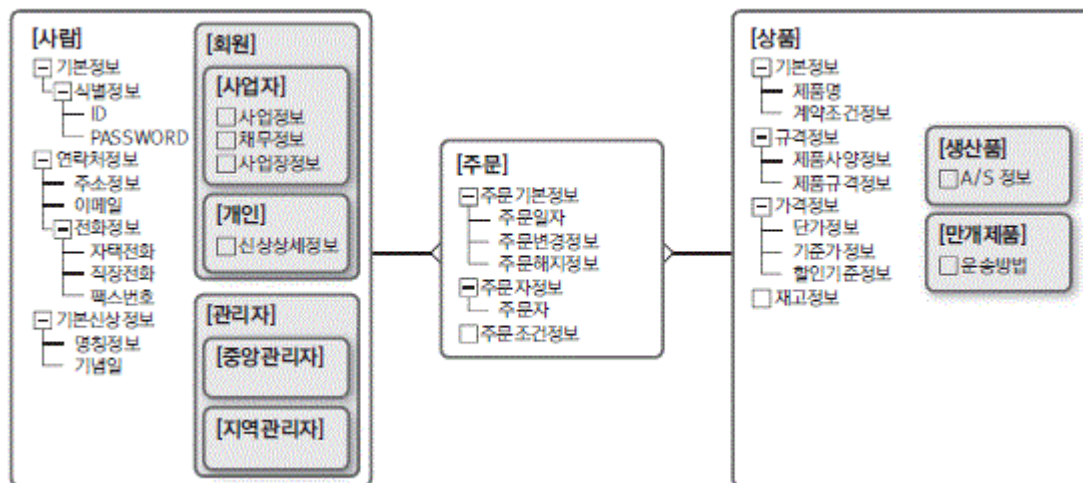
### 1) 개념적 데이터 모델

개념 데이터 모델이란 업무 요건을 충족하는 데이터의 주제 영역과 핵심 데이터 집합을 정의하고 관계를 정의한 모델을 의미한다.

개괄 데이터 모델 예시



개념 데이터 모델 예시

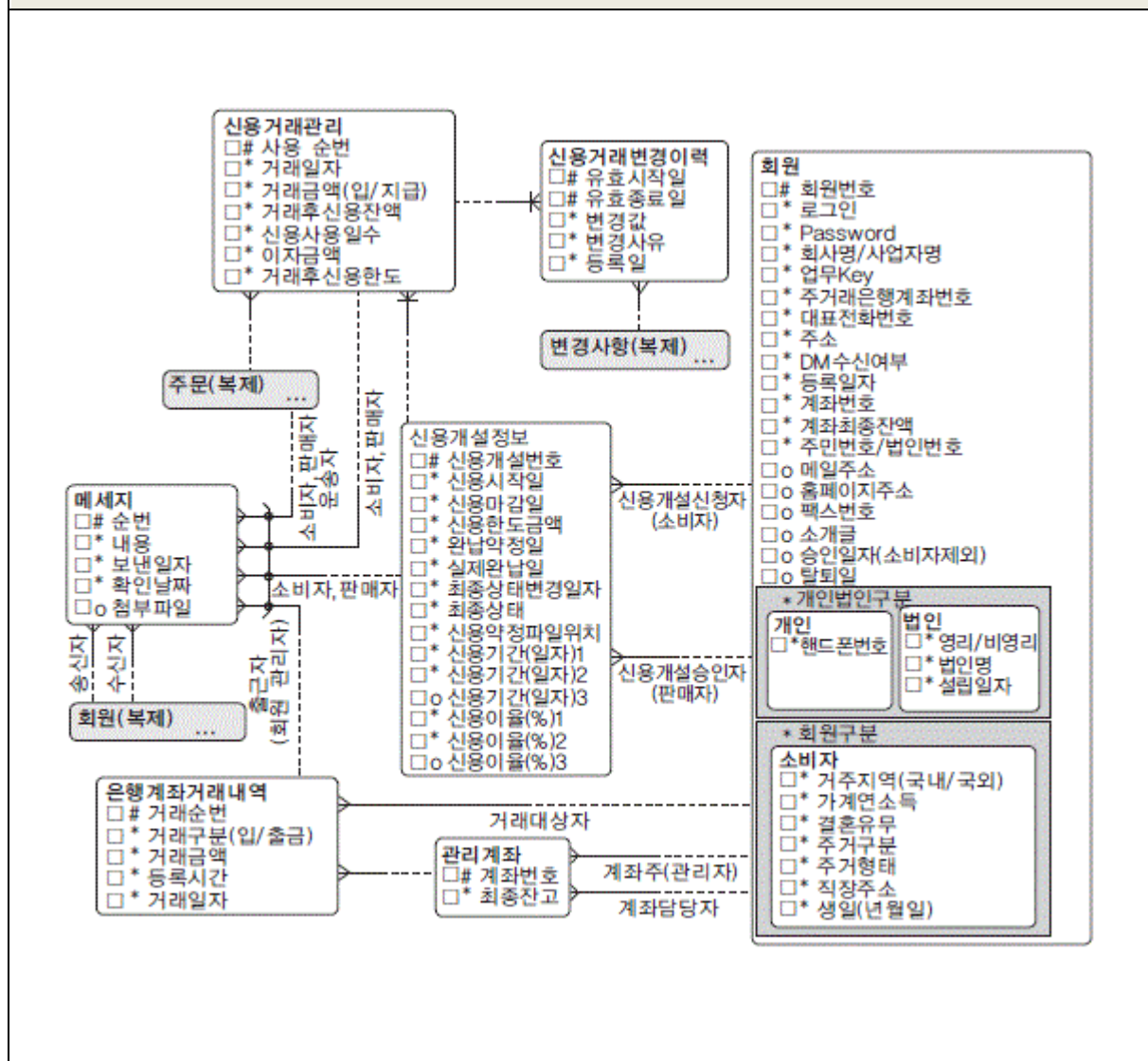




## 2) 논리적 데이터 모델

논리 데이터 모델이란 개념 데이터 모델을 상세화 하여 논리적인 데이터 집합, 관리 항목, 관계를 정의한 모델을 말한다. 논리 데이터 모델은 전체 데이터 구조에서 가장 핵심을 이루는 모델로서 전체 업무 범위와 업무 구성요소를 확인할 수 있다.

논리 데이터 모델 예시



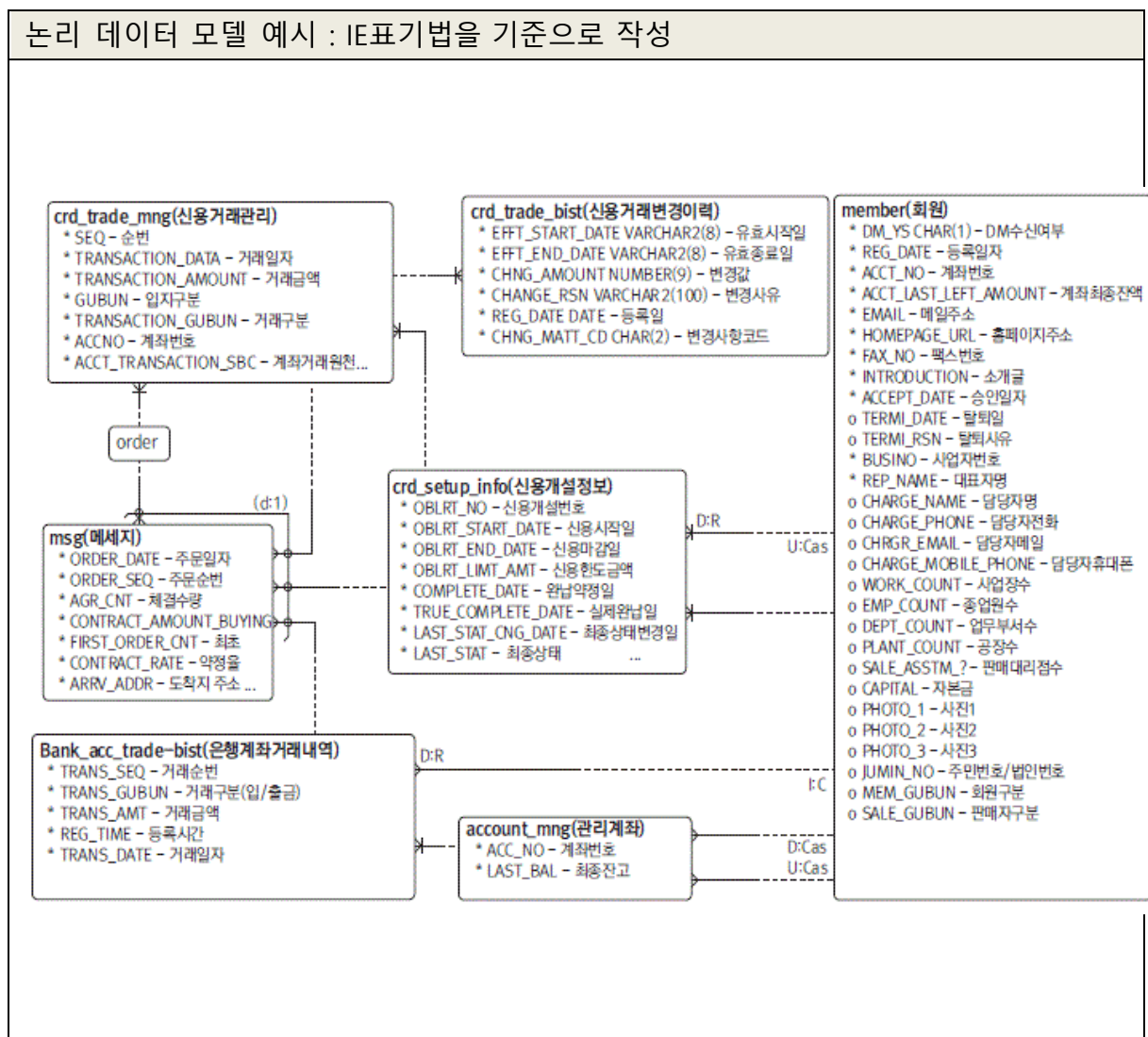


### 3) 물리적 데이터 모델

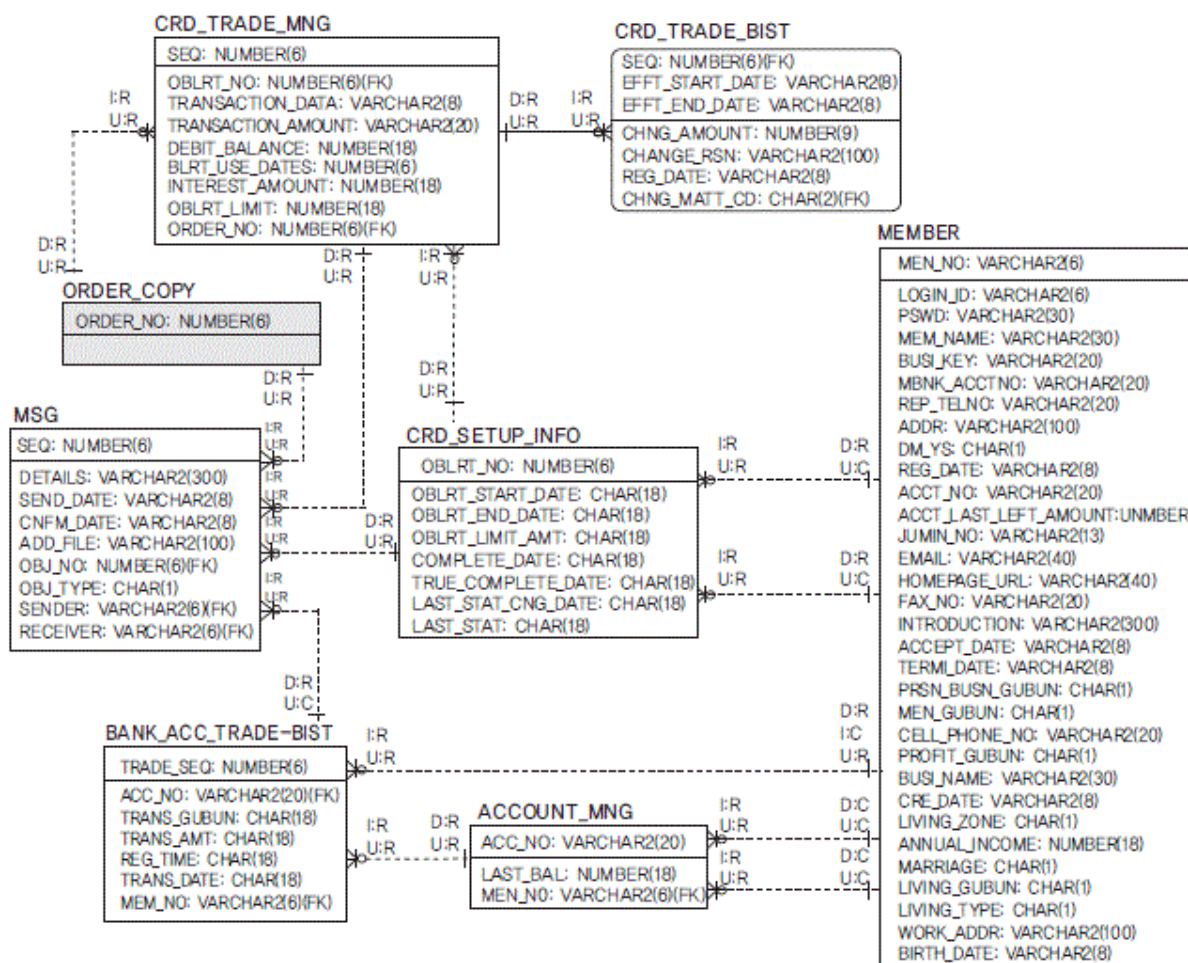
물리 데이터 모델이란 논리 데이터 모델을 DBMS의 특성 및 성능을 고려하여 구체화시킨 모델을 말한다. 물리 데이터 모델은 DBMS 선정 이후에 해당 DBMS 상에서 최상의 성능을 보장하도록 논리 데이터 모델에서 저장하는 데이터의 물리적 특성을 최대한 반영하여 설계하고 이를 관리한다.

논리 데이터 모델이 1:1로 데이터베이스의 객체로 대응되어 생성되지 않으므로 DBMS의 성능을 최대한 살릴 수 있고 저장되는 데이터의 특성을 충분히 반영할 수 있다.

논리 데이터 모델 예시 : IE표기법을 기준으로 작성



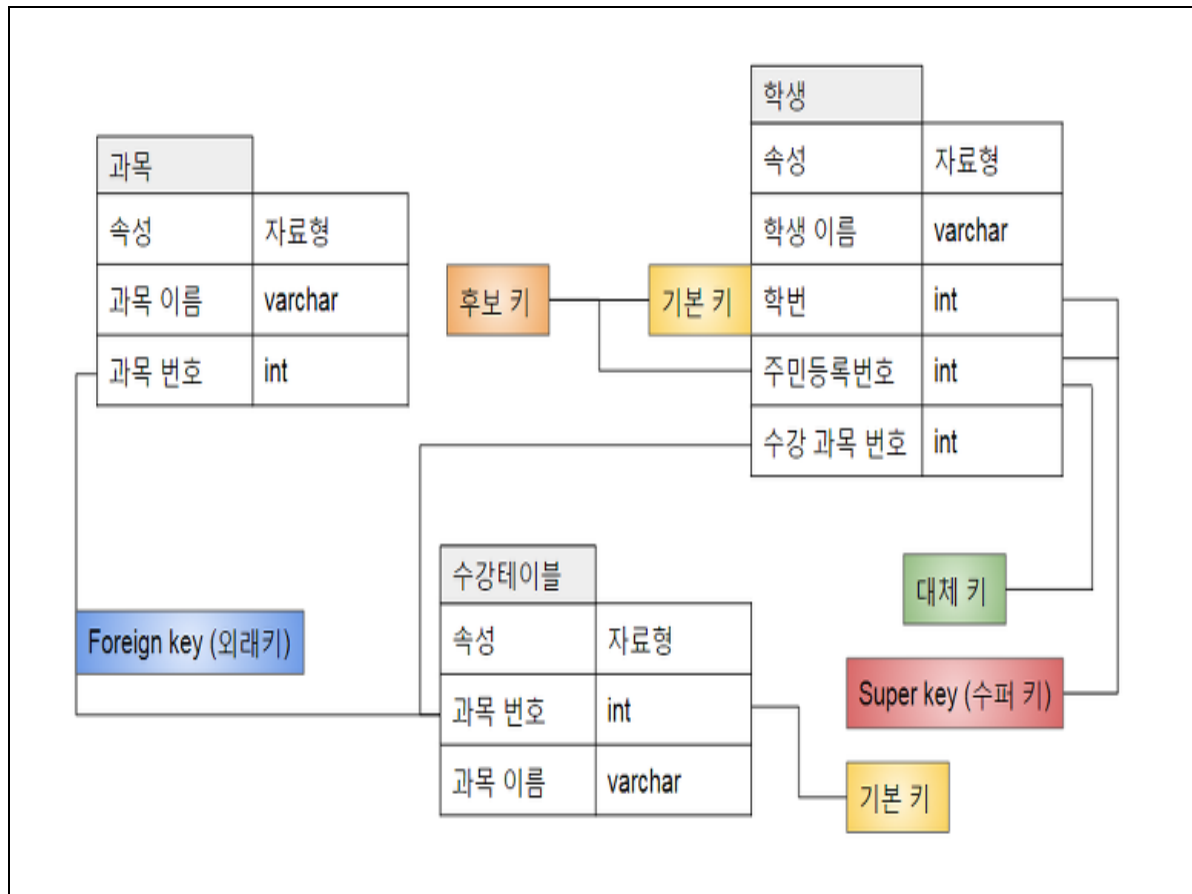
## 물리 데이터 모델 예시 : IE표기법을 기준으로 작성



## # 관계형 데이터베이스(Relational Database, RDB) ERD 표기

### 1) ERD(Entity Relationship Diagram)

- 개체 속성과 개체 간 관계를 도표로 표현한 것을 의미한다.

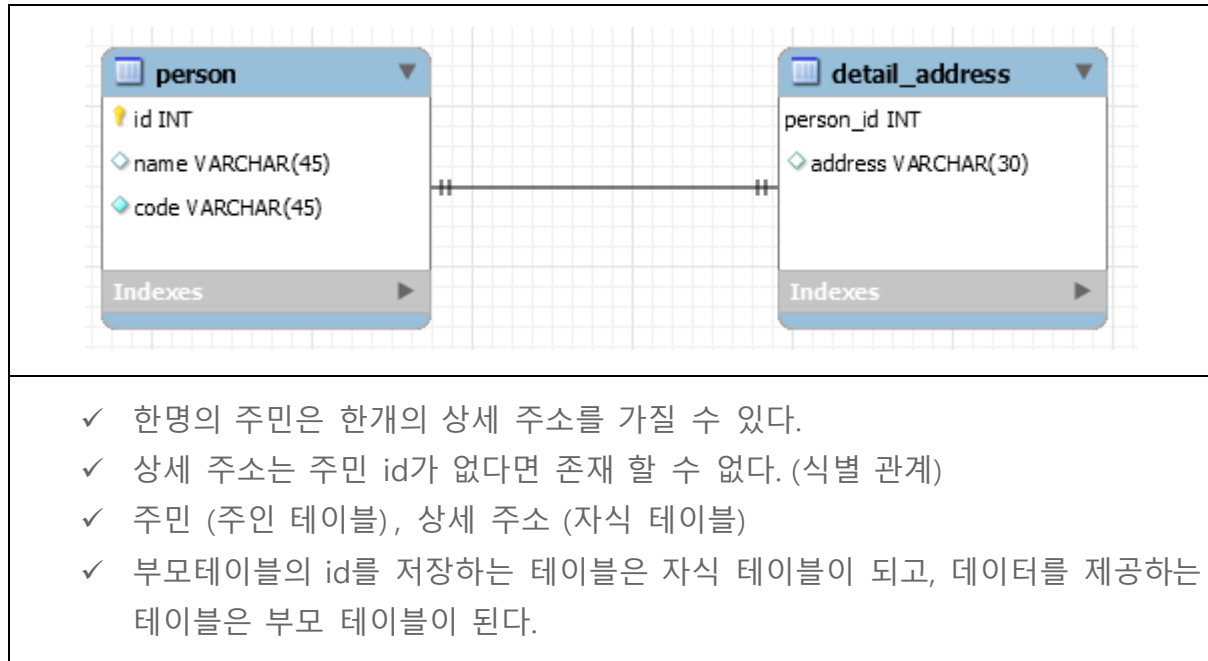


### 2) ERD 관계 표현법

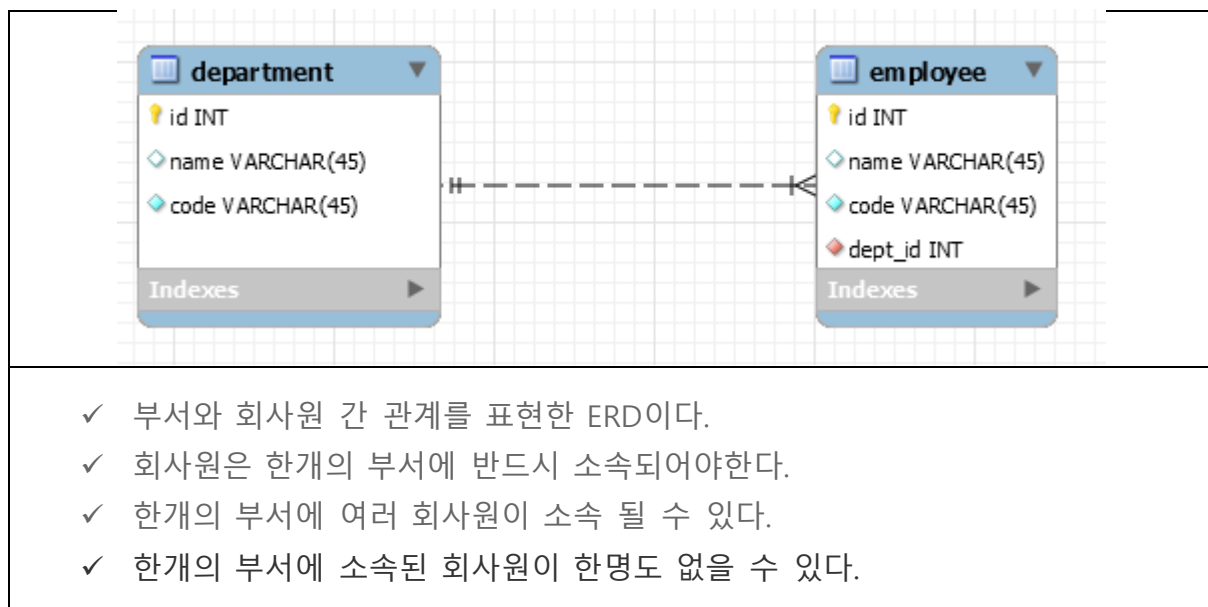


### 가. 일대 일 (식별 관계)

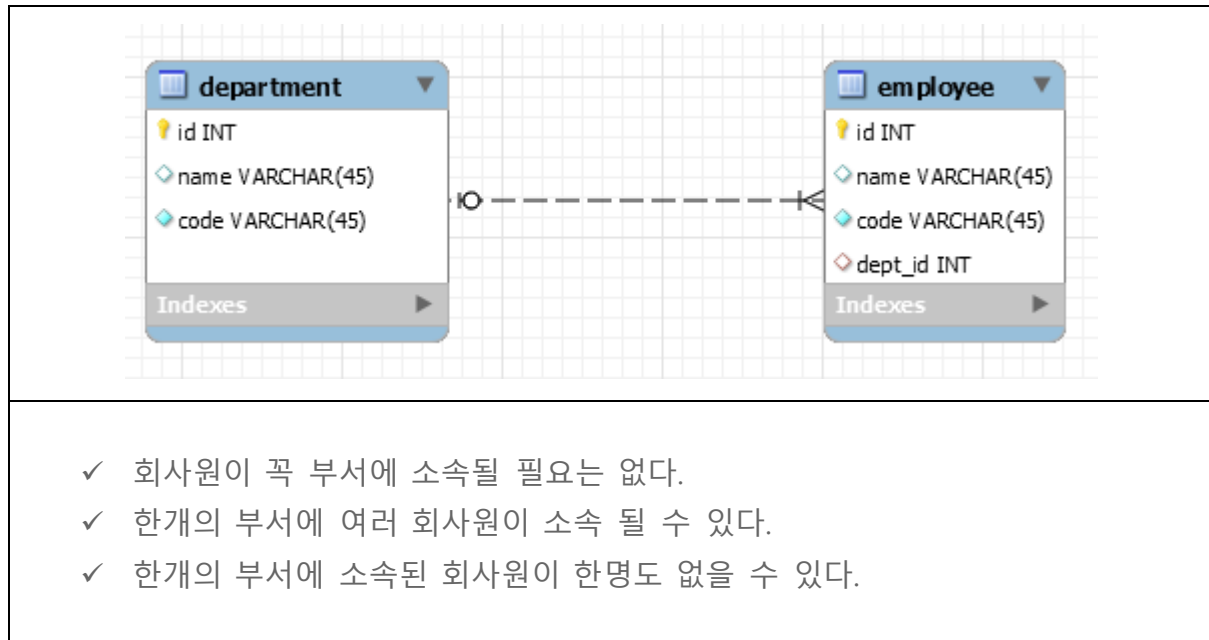
- 주민과 상세 주소 간 관계를 표현한 ERD 이다.
- 상세 주소 테이블은 person\_id를 기본키로 사용하고 있고, person\_id를 통해 person 테이블을 참조 하고 있다.



### 나. 일대 다 (참조 필수)



다. 일대 다(참조 Null 허용)

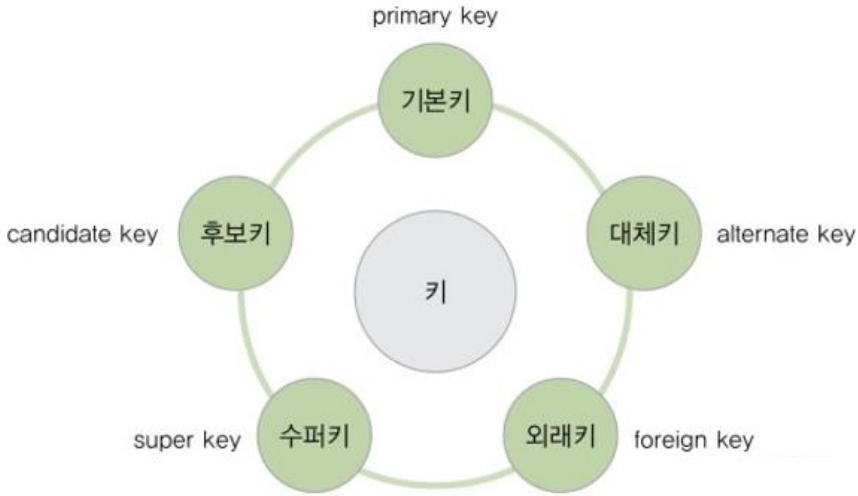


3) 비 식별관계와 식별관계

관계유형	표기법	내용
비식별 관계		- 비식별관계는 점선으로 표현한다. - 기본키에 외래키가 포함되어 있지 않다면 비식별 관계라고 한다.
식별 관계		- 식별관계는 실선으로 표현한다. - 기본키에 외래키가 포함되어 있다면 이를 식별 관계라고한다.



## # 키의 종류

	
<b>수퍼키</b>	레코드들을 식별할 수 있는 '필드의 집합' (유일성) 테이블은 적어도 1개의 슈퍼키를 가져야 한다.
<b>후보키</b>	슈퍼키에서 레코드를 식별할 수 있는 최소한의 필드만 남겨놓은 집합 (유일성, 최소성) Ex. 슈퍼키 { 학번, 학생이름, 학과 } 집합은 학번만으로도 레코드를 식별할 수 있으므로 부분집합 중 { 학번 } 집합만 후보키가 될 수 있다.
<b>기본키</b>	설계자가 여러 후보키 중 하나를 선택하여 정의한 식별자 (유일성, 최소성) 기본키의 모든 필드의 값은 null 이 될 수 없다.
<b>외래키</b>	다른 테이블의 기본키를 참조한다. 외래키의 모든 필드는 참조하는 기본키와 동일한 도메인(값의 종류&범위)을 갖는다. 외래키의 모든 필드의 값은 참조하는 기본키와 동일하거나 null 이다.
<b>대체키</b>	기본키로 선택되지 못한 후보키들이다. 이름에서 알 수 있듯이 대체키는 기본키를 대신할 수 있지만 기본키가 되지 못하고 탈락한 이유가 있을 수 있다.

기본키를 선택할 때 고려할 사항을 하나씩 따져보면 기본키의 [주소 속성이 추가된 릴레이션의 예]의 고객 릴레이션은 고객아이디 속성을 기본키로 선택하는 것이 무난하다. 따라서 기본키로 선택되지 못한 (고객이름, 주소) 속성 집합이 대체키가 된다.



## 실습 : Database Creation & Table Creation

### PRIMARY KEY

```
-- shopdb 데이터베이스 구조 내보내기
DROP DATABASE IF EXISTS dm_test;

CREATE DATABASE IF NOT EXISTS dm_test /* DEFAULT CHARACTER SET utf8 */;

use dm_test;

DROP TABLE IF EXISTS Persons ;
CREATE TABLE Persons (
    PersonID    int          NOT NULL,
    LastName    varchar(255)  NOT NULL,
    FirstName    varchar(255),
    Age         int
);

DESC Persons;

ALTER TABLE Persons
ADD PRIMARY KEY (PersonID);

DROP TABLE IF EXISTS Persons ;
CREATE TABLE Persons (
    PersonID    int          NOT NULL PRIMARY KEY ,
    LastName    varchar(255)  NOT NULL,
    FirstName    varchar(255),
    Age         int
);

DROP TABLE IF EXISTS Persons ;
CREATE TABLE Persons (
    PersonID    int          NOT NULL,
    LastName    varchar(255)  NOT NULL,
    FirstName    varchar(255),
    Age         int,
    PRIMARY KEY (PersonID)
);

ALTER TABLE Persons
DROP PRIMARY KEY;
```

```
DROP TABLE IF EXISTS Persons ;
CREATE TABLE Persons (
    PersonID    int          NOT NULL,
    LastName    varchar(255) NOT NULL,
    FirstName    varchar(255),
    Age         int,
    CONSTRAINT PK_Person PRIMARY KEY (PersonID, LastName)
);

ALTER TABLE Persons
DROP PRIMARY KEY;

ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (PersonID, LastName);

DESC Persons;
```

#### FOREIGN KEY

```
DROP TABLE IF EXISTS Orders ;
CREATE TABLE Orders (
    OrderID      int    NOT NULL,
    OrderNumber  int    NOT NULL,
    PersonID     int,
    PRIMARY KEY (OrderID)
);

ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

DROP TABLE IF EXISTS Orders ;
CREATE TABLE Orders (
    OrderID      int    NOT NULL,
    OrderNumber  int    NOT NULL,
    PersonID     int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

## 실습 : Database Creation & Table Creation

---

```
-- -----  
-- 호스트      : 127.0.0.1  
-- 서버 버전   : 10.5.5-MariaDB - mariadb.org binary distribution  
-- 서버 OS     : Windows64  
-- DBMS 툴     : ToadEdge 2.1.5  
-- -----  
  
-- shopdb 데이터베이스 구조 내보내기  
DROP DATABASE IF EXISTS shopdb;  
  
CREATE DATABASE IF NOT EXISTS shopdb /* DEFAULT CHARACTER SET utf8 */;  
  
USE shopdb;  
  
-- 테이블 shopdb.deletedmembertbl 구조 내보내기  
DROP TABLE IF EXISTS deletedmembertbl ;  
  
CREATE TABLE IF NOT EXISTS deletedmembertbl (  
    memberID      char(8)      DEFAULT NULL,  
    memberName    char(5)      DEFAULT NULL,  
    memberAddress char(20)     DEFAULT NULL,  
    deletedDate   date         DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
DESC deletedmembertbl;  
  
-- 테이블 데이터 shopdb.deletedmembertbl:~1 rows (대략적) 내보내기  
INSERT INTO deletedmembertbl ( memberID , memberName , memberAddress ,  
deletedDate )  
VALUES ( 'Dang', '당탕이', '경기 부천시 중동', '2018-12-25' );  
  
SELECT * FROM deletedmembertbl;
```

```
-- 테이블 shopdb.indextbl 구조 내보내기
DROP TABLE IF EXISTS indextbl ;

CREATE TABLE IF NOT EXISTS indextbl (
    first_name  varchar(14) DEFAULT NULL,
    last_name   varchar(16) DEFAULT NULL,
    hire_date   date        DEFAULT NULL,
    KEY idx_indexTBL_firstname ( first_name )
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- sqlcmd -S myServer\instanceName -i C:\Temp\shopdb_reg_data.sql;
source "C:\Temp\shopdb_reg_data.sql";

SELECT * FROM indextbl ;

-- 테이블 shopdb.membertbl 구조 내보내기
DROP TABLE IF EXISTS membertbl;

CREATE TABLE IF NOT EXISTS membertbl (
    memberID    char(8)    NOT NULL,
    memberName  char(5)    NOT NULL,
    memberAddress char(20) DEFAULT NULL,
    PRIMARY KEY (memberID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 테이블 데이터 shopdb.membertbl:~3 rows (대략적) 내보내기
INSERT INTO membertbl (memberID, memberName, memberAddress)
VALUES
    ('Han',  '한주연', '인천 남구 주안동'),
    ('Jee',  '지운이', '서울 은평구 증산동'),
    ('Sang', '상길아', '경기 성남구 분당구');

SELECT * FROM membertbl ;
```

## 실습 : Stored Procedure

```
-- 프로시저 shopdb.myProc 구조 내보내기
DROP PROCEDURE IF EXISTS myProc;

DELIMITER //
CREATE DEFINER=root@localhost PROCEDURE myProc()
BEGIN
    SELECT * FROM memberTBL WHERE memberName = '당탕이' ;
    SELECT * FROM productTBL WHERE productName = '냉장고' ;
END//
DELIMITER ;

CALL myProc();

-- 테이블 shopdb.producttbl 구조 내보내기
DROP TABLE IF EXISTS producttbl;

CREATE TABLE IF NOT EXISTS producttbl (
    productName    char(4)    NOT NULL,
    cost           int(11)    NOT NULL,
    makeDate       date       DEFAULT NULL,
    company        char(5)    DEFAULT NULL,
    amount         int(11)    NOT NULL,
    PRIMARY KEY (productName)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 테이블 데이터 shopdb.producttbl:~3 rows (대략적) 내보내기
INSERT INTO producttbl (productName, cost, makeDate, company, amount)
VALUES
    ('냉장고', 5, '2019-02-01', '대우', 22),
    ('세탁기', 20, '2018-09-01', 'LG', 3),
    ('컴퓨터', 10, '2017-01-01', '삼성', 17);

SELECT * FROM producttbl p;

CALL myProc();

INSERT INTO membertbl (memberID, memberName, memberAddress)
VALUES ('Dang', '당탕이', '서울 강남구 역삼동');

CALL myProc();
```

```
-- 프로시저 shopdb.myProc 구조 내보내기
DROP PROCEDURE IF EXISTS myProc2;
DELIMITER //
CREATE DEFINER=root@localhost PROCEDURE myProc2()
BEGIN
    SELECT * FROM membertbl m, producttbl p
    WHERE m.memberName = '당탕이'
    AND p.productName = '냉장고';
END//
DELIMITER ;

CALL myProc2();

SELECT * FROM membertbl m;
SELECT * FROM producttbl p;
SELECT * FROM membertbl m, producttbl p;

SELECT * FROM membertbl m, producttbl p
WHERE m.memberAddress LIKE '서울%'
AND p.company = '대우'
;
```

## 실습 : View Table

---

```
-- 뷰 shopdb.uv_membertbl 구조 내보내기
DROP VIEW IF EXISTS uv_membertbl;

-- VIEW 종속성 오류를 극복하기 위해 임시 테이블을 생성합니다.
CREATE TABLE uv_membertbl (
    memberName    CHAR(5)    NOT NULL COLLATE utf8_general_ci,
    memberAddress CHAR(20)    NULL COLLATE utf8_general_ci
) ENGINE=MyISAM;

SELECT * FROM uv_membertbl;

-- 뷰 shopdb.uv_membertbl 구조 내보내기
DROP VIEW IF EXISTS uv_membertbl;

CREATE ALGORITHM=UNDEFINED DEFINER=root@localhost VIEW uv_membertbl
AS SELECT memberName, memberAddress FROM memberTBL ;

SELECT * FROM uv_membertbl;
```

## 실습 : Trigger

---

```
-- 트리거 shopdb.trg_deletedMemberTBL 구조 내보내기
DROP TRIGGER IF EXISTS trg_deletedMemberTBL;

SET                                     @OLDTMP_SQL_MODE=@@SQL_MODE,
SQL_MODE='STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
DELIMITER //

CREATE TRIGGER trg_deletedMemberTBL    -- 트리거명
    AFTER DELETE                       -- 삭제후에 작동하게 지정
    ON membertbl                      -- 트리거를 부착할 테이블
    FOR EACH ROW                      -- 각 행마다 적용시킴
BEGIN
    -- OLD 테이블의 내용을 백업테이블에 삽입
    INSERT INTO deletedMemberTBL
    VALUES (OLD.memberID, OLD.memberName, OLD.memberAddress, CURDATE() );
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;

SELECT * FROM deletedMemberTBL;

SELECT * FROM membertbl ;

DELETE FROM membertbl
WHERE memberName = '상길이';

DELETE FROM membertbl
WHERE memberAddress LIKE '서울%';

SELECT * FROM membertbl ;

SELECT * FROM deletedMemberTBL;
```



## 4. 데이터베이스 모델링

### 4.1 데이터베이스 모델링

dbForge Studio 모델링 툴

다운로드 : <https://www.devart.com/dbforge/>

## 실습 : Foreign Table

```
USE shopdb;
```

```
CREATE TABLE userTbl -- 회원 테이블
```

```
(
    userID      CHAR(8) NOT NULL PRIMARY KEY, -- 사용자 아이디(PK)
    name        VARCHAR(10) NOT NULL, -- 이름
    birthYear   INT NOT NULL, -- 출생년도
    addr        CHAR(2) NOT NULL, -- 지역(경기,서울,경남 식으로 2글자만입력)
    mobile1     CHAR(3), -- 휴대폰의 국번(011, 016, 017, 018, 019, 010 등)
    mobile2     CHAR(8), -- 휴대폰의 나머지 전화번호(하이픈제외)
    height      SMALLINT, -- 키
    mDate       DATE -- 회원 가입일
);
```

```
CREATE TABLE buyTbl -- 회원 구매 테이블
```

```
(
    num          INT AUTO_INCREMENT NOT NULL PRIMARY KEY, -- 순번(PK)
    userID       CHAR(8) NOT NULL, -- 아이디(FK)
    prodName     CHAR(6) NOT NULL, -- 물품명
    groupName    CHAR(4) NULL, -- 분류
    price        INT NOT NULL, -- 단가
    amount       SMALLINT NOT NULL, -- 수량
    FOREIGN KEY (userID) REFERENCES userTbl(userID)
);
```

```
INSERT INTO userTbl VALUES('LSG', '이승기', 1987, '서울', '011', '11111111', 182, '2008-8-8');
INSERT INTO userTbl VALUES('KBS', '김범수', 1979, '경남', '011', '22222222', 173, '2012-4-4');
INSERT INTO userTbl VALUES('KKH', '김경호', 1971, '전남', '019', '33333333', 177, '2007-7-7');
INSERT INTO userTbl VALUES('JYP', '조용필', 1950, '경기', '011', '44444444', 166, '2009-4-4');
INSERT INTO userTbl VALUES('SSK', '성시경', 1979, '서울', NULL, NULL, 186, '2013-12-12');
INSERT INTO userTbl VALUES('LJB', '임재범', 1963, '서울', '016', '66666666', 182, '2009-9-9');
INSERT INTO userTbl VALUES('YJS', '윤종신', 1969, '경남', NULL, NULL, 170, '2005-5-5');
INSERT INTO userTbl VALUES('EJW', '은지원', 1972, '경북', '011', '88888888', 174, '2014-3-3');
INSERT INTO userTbl VALUES('JKW', '조관우', 1965, '경기', '018', '99999999', 172, '2010-10-10');
INSERT INTO userTbl VALUES('BBK', '바비킴', 1973, '서울', '010', '00000000', 176, '2013-5-5');
```

```
INSERT INTO buyTbl VALUES(NULL, 'KBS', '운동화', NULL, 30, 2);
INSERT INTO buyTbl VALUES(NULL, 'KBS', '노트북', '전자', 1000, 1);
INSERT INTO buyTbl VALUES(NULL, 'JYP', '모니터', '전자', 200, 1);
INSERT INTO buyTbl VALUES(NULL, 'BBK', '모니터', '전자', 200, 5);
INSERT INTO buyTbl VALUES(NULL, 'KBS', '청바지', '의류', 50, 3);
INSERT INTO buyTbl VALUES(NULL, 'BBK', '메모리', '전자', 80, 10);
INSERT INTO buyTbl VALUES(NULL, 'SSK', '책', '서적', 15, 5);
INSERT INTO buyTbl VALUES(NULL, 'EJW', '책', '서적', 15, 2);
INSERT INTO buyTbl VALUES(NULL, 'EJW', '청바지', '의류', 50, 1);
INSERT INTO buyTbl VALUES(NULL, 'BBK', '운동화', NULL, 30, 2);
INSERT INTO buyTbl VALUES(NULL, 'EJW', '책', '서적', 15, 1);
INSERT INTO buyTbl VALUES(NULL, 'BBK', '운동화', NULL, 30, 2);
```

```
SELECT * FROM userTbl
ORDER BY userID;
```

```
SELECT * FROM buyTbl
ORDER BY userID;
```

```
SELECT *
FROM usertbl u, buytbl b
WHERE u.userID = b.userID
ORDER BY b.userID;
;
```

```
SELECT *
FROM usertbl u, buytbl b
WHERE u.userID = b.userID
AND b.price > 100
ORDER BY b.userID;
;
```

## 5. MariaDB 유틸리티 사용법

### 5.1 HeidiSQL 사용 방법

### 5.2 외부 MariaDB 서버 관리하기

### 5.3 사용자 관리하기

## 6. SQL 기본

# SELECT의 구문 형식

SELECT select\_expr

FROM table\_references

WHERE where\_condition

GROUP BY { col\_name | expr | position }

HAVING where\_condition

ORDER BY { col\_name | expr | position }

# GROUP BY와 함께 사용하는 집계함수

함수명	설명
AVG()	평균을 구한다
MIN()	최소값
MAX()	최대값
COUNT()	행의 개수
COUNT(DISTINCT)	행의 개수 (중복은 제외)
STDEV()	표준편차
VAR_SAMP()	분산

## Review : Data Handling in Python

### # 미션 : 도서 목록 입력 및 출력

#### 1. 다양한 데이터 구조들을 활용하는 데에 초점

- 데이터 담기
- 여러 데이터 만들어 보기
- 소스 코드 단순화 하기

#### 2. 데이터 구조 예시

제목	출판연도	출판사	쪽수	추천유무
파이썬 프로그램	2016	A	200	X
플랫폼 비즈니스	2013	B	584	O
빅데이터 마케팅	2014	A	296	O
외식경영 전문가	2010	B	526	X
십억만 벌어보자	2013	A	248	O

## 3. 데이터형과 데이터구조에서 배운 내용을 토대로 담기

- 여러 책을 담을 수 있는 리스트형인 books라는 변수를 선언 및 초기화
- Books의 각 항목에 한 권의 책을 담기 위한 사전형 데이터를 추가
- 사전형 데이터는 변수 선언 없이 리스트에 바로 추가
- 사전형 데이터에는 4개의 키와 값의 쌍이 존재하며, 키는 컬럼명이고 값은 각 책에 해당하는 컬럼 값이 된다.

➤

books

```
[{'제목': '파이썬 프로그램', '출판연도': '2016', '출판사': 'A', '쪽수': 200, '추천유무': False},
 {'제목': '플랫폼 비즈니스', '출판연도': '2013', '출판사': 'B', '쪽수': 584, '추천유무': True},
 {'제목': '빅데이터 마케팅', '출판연도': '2014', '출판사': 'A', '쪽수': 296, '추천유무': True},
 {'제목': '외식경영 전문가', '출판연도': '2010', '출판사': 'B', '쪽수': 526, '추천유무': False},
 {'제목': '십억만 벌어보자', '출판연도': '2013', '출판사': 'A', '쪽수': 248, '추천유무': True}]
```

## 4. 작성한 소스의 books를 활용하여 아래 질문에 답하기 추가하기

- 1) 내가 읽은 책 중 250쪽이 넘는 책의 제목으로 이루어진 리스트형 변수 `many_page`를 만든다.
- 2) 내가 읽은 책 중 추천하고 싶은 책의 제목으로 이루어진 리스트형 변수 `recommends`를 만든다.
- 3) 내가 읽은 책의 전체 쪽수를 담은 숫자형 변수 `all_pages`를 만든다.
- 4) 내가 읽은 책의 출판사를 위한 세트 형 변수 `pub_company`를 만든다.

```
print(result.format(dot_lint, many_page, recommends, all_pages, pub_companies))
```

## ■ 질문에 대한 결과값

- 1) 쪽수가 250 쪽 넘는 책 리스트 : ['플랫폼 비즈니스', '빅데이터 마케팅', '외식경영 전문가']
- 2) 내가 추천하는 책 리스트 : ['플랫폼 비즈니스', '빅데이터 마케팅', '십억만 벌어보자']
- 3) 내가 읽은 책 전체 쪽수 : 1854 page
- 4) 내가 읽은 책의 출판사 목록 : ['A', 'B']

cf. 책목록 등록

```
# 책 목록
books = list()

books.append({ '제목': '파이썬 프로그램', '출판연도': '2016', '출판사': 'A', '쪽수': 200, '추천유무': False })
books.append({ '제목': '플랫폼 비즈니스', '출판연도': '2013', '출판사': 'B', '쪽수': 584, '추천유무': True })
books.append({ '제목': '빅데이터 마케팅', '출판연도': '2014', '출판사': 'A', '쪽수': 296, '추천유무': True })
books.append({ '제목': '외식경영 전문가', '출판연도': '2010', '출판사': 'B', '쪽수': 526, '추천유무': False })
books.append({ '제목': '십억만 벌어보자', '출판연도': '2013', '출판사': 'A', '쪽수': 248, '추천유무': True })

print(books)
```



## ■ 내가 읽은 책 목록

[book\_list.py]

```

# 책 목록
books = list()

books.append({ '제목': '파이썬 프로그램', '출판연도': '2016', '출판사': 'A', '쪽수': 200, '추천유무': False })
books.append({ '제목': '플랫폼 비즈니스', '출판연도': '2013', '출판사': 'B', '쪽수': 584, '추천유무': True })
books.append({ '제목': '빅데이터 마케팅', '출판연도': '2014', '출판사': 'A', '쪽수': 296, '추천유무': True })
books.append({ '제목': '외식경영 전문가', '출판연도': '2010', '출판사': 'B', '쪽수': 526, '추천유무': False })
books.append({ '제목': '십억만 벌어보자', '출판연도': '2013', '출판사': 'A', '쪽수': 248, '추천유무': True })

# 변수 선언
many_page = list() # 결과값 1. 책제목
recommends = list() # 결과값 2. 책제목
all_pages = int() # 결과값 3. 전체 쪽수
pub_companies = set() # 결과값 4. 출판사

# 로직 구현
for book in books:

    if book['쪽수'] > 250:
        many_page.append(book['제목'])

    if book['추천유무']:
        recommends.append(book['제목'])

    all_pages = all_pages + book['쪽수']

    pub_companies.add(book['출판사'])

# 결과값 출력
print(' 1) 쪽수가 250 쪽 넘는 책 리스트 :', many_page)
print(' 2) 내가 추천하는 책 리스트 :', recommends)
print(' 3) 내가 읽은 책 전체 쪽수 :', all_pages)
print(' 4) 내가 읽은 책의 출판사 목록 :', pub_companies)

```

Tip. Pythonic Code, 파이썬스러운 코드

```
# 로직 구현
many_page    = [ book['제목'] for book in books if book['쪽수'] > 250 ]
recommends   = [ book['제목'] for book in books if book['추천유무'] ]
all_pages    = sum([ book['쪽수'] for book in books ])
pub_companies = { book['출판사'] for book in books }
```

## 실습 : SQL Practice in MariaDB

---

```
-- -----  
-- 호스트      : 127.0.0.1  
-- 서버 버전   : 10.5.5-MariaDB - mariadb.org binary distribution  
-- 서버 OS     : Windows64  
-- DBMS 툴     : ToadEdge 2.1.5  
-- -----  
  
show DATABASES ;  
  
use bigpy;  
  
-- 테이블 shopdb.deletedmembertbl 구조 내보내기  
DROP TABLE IF EXISTS my_books ;  
  
CREATE TABLE my_books (  
    title text,  
    published_date text,  
    publisher text,  
    pages integer,  
    recommendation integer  
);  
  
DESC my_books;  
  
INSERT INTO my_books VALUES ('인더스트리 4.0', '2016.07.09', 'B', 584, 1);  
INSERT INTO my_books VALUES ('유니콘 스타트업', '2011.07.15', 'A', 248, 1);  
INSERT INTO my_books VALUES ('빅데이터 마케팅', '2012.08.25', 'A', 296, 1);  
INSERT INTO my_books VALUES ('사물인터넷 전망', '2013.08.22', 'B', 526, 0);  
INSERT INTO my_books VALUES ('메가트랜드', '2002.03.02', 'A', 200, 0);  
  
SELECT * FROM my_books m;  
  
SELECT title, publisher, published_date FROM my_books;  
  
SELECT published_date, title, publisher FROM my_books  
WHERE pages > 250;  
  
SELECT publisher  
FROM my_books m;  
  
SELECT DISTINCT publisher  
FROM my_books m;
```

```
INSERT INTO my_books VALUES ('머신러닝', '2019.08.22', 'BPC', 600, 0);
INSERT INTO my_books VALUES ('딥러닝', '2019.08.25', 'BPC', 500, 0);
INSERT INTO my_books VALUES ('빅데이터', '2019.07.15', 'BPC', 400, 0);
```

```
SELECT * FROM my_books m;
```

```
SELECT published_date, title, publisher
FROM my_books
WHERE publisher = 'BPC'
AND pages > 450;
```

```
SELECT * FROM my_books
WHERE title='빅데이터';
```

```
UPDATE my_books SET recommendation=1
WHERE title='빅데이터';
```

```
SELECT * FROM my_books
WHERE recommendation!=1;
```

```
SELECT * FROM my_books
WHERE recommendation<>1;
```

```
SELECT * FROM my_books
WHERE recommendation=0;
```

```
DELETE FROM my_books
WHERE recommendation=0;
```

```
SELECT * FROM my_books
WHERE recommendation=0;
```

## 실습 : DB Handling in Python

### # 데이터베이스 연결

```
import pymysql

# Open database connection
conn = pymysql.connect(host='localhost', port=3306, user='root',
passwd='password', db='proj20_iitp_hk', charset='utf8', autocommit=True)

# prepare a cursor object using cursor() method
cursor = conn.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print(type(data), len(data))
print(data[0])

conn.close()
```

### # 공통패키지에 DB Connection 함수 생성

```
import pymysql

def get_connection(db_name):
    conn = pymysql.connect(host='localhost', port=3306, user='root',
passwd='password',
                           db=db_name, charset='utf8', autocommit=True)

    return conn
```

## # 테이블 생성

```

import pymysql

def create_table(db_name, db_sql):
    """
    데이터베이스 테이블을 생성하는 함수
    Args:
        db_name : Database Name
        db_sql : Query for creating Table
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = pymysql.connect(host='localhost', port=3306, user='root',
                                passwd='password', db=db_name, charset='utf8',
                                autocommit=True)

        # 커서 확보
        cur = conn.cursor()

        # 테이블 생성
        cur.execute(db_sql)

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        # print('Finish process of function.')
        conn.close()

    return is_success

```

## # 데이터 등록

```

# 데이터 입력 함수
def insert_books(db_name):
    """
    데이터베이스 테이블에 데이터를 등록하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 데이터 입력 SQL1
        db_sql = "INSERT INTO my_books VALUES ('메가트랜드', '2002.03.02','A',
200, 0)"
        cur.execute(db_sql)

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        conn.close()

    return is_success

if insert_books(db_name):
    print('데이터가 성공적으로 등록되었습니다.')
else :
    print('데이터가 등록되지 않았습니다')

```

```

# 데이터 입력 함수 2
def insert_books2(db_name):
    """
    데이터베이스 테이블에 데이터를 등록하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 데이터 입력 SQL2
        db_sql = 'INSERT INTO my_books VALUES (%s, %s, %s, %s, %s)'
        cur.execute(db_sql, ('인더스트리 4.0', '2016.07.09', 'B', 584, 1))

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        conn.close()

    return is_success

if insert_books2(db_name):
    print('데이터가 성공적으로 등록되었습니다.')
else:
    print('데이터가 등록되지 않았습니다')

```



```

def insert_books3(db_name):
    """
    데이터베이스 테이블에 데이터를 등록하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)
        cur = conn.cursor()

        # 데이터 입력 SQL3
        db_sql = 'INSERT INTO my_books VALUES (%s, %s, %s, %s, %s)'
        books = [
            ('유니콘 스타트업', '2011.07.15', 'A', 248, 1),
            ('빅데이터 마케팅', '2012.08.25', 'A', 296, 1),
            ('사물인터넷 전망', '2013.08.22', 'B', 526, 0)
        ]
        cur.executemany(db_sql, books)

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        conn.close()

    return is_success

if insert_books3(db_name):
    print('데이터가 성공적으로 등록되었습니다.')
else:
    print('데이터가 등록되지 않았습니다')

```

## # 데이터 조회

```
import pandas as pd

def getBooksDF(books):
    ret_df = pd.DataFrame()

    title = list()
    published_date = list()
    publisher = list()
    pages = list()
    recommendation = list()

    column_name = ['title', 'published_date', 'publisher', 'pages',
'recommendation']
    for book in books:
        title.append(book[0])
        published_date.append(book[1])
        publisher.append(book[2])
        pages.append(book[3])
        recommendation.append(book[4])

    data = {
        'title': title,
        'published_date': published_date,
        'publisher': publisher,
        'pages': pages,
        'recommendation': recommendation
    }

    ret_df = pd.DataFrame(data, columns=column_name)

    return ret_df
```

```

def select_all_books(db_name):
    """
    전체 데이터를 조회하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books"
        cur.execute(db_sql)

        # 조회한 데이터 불러오기
        print('[1] 전체 데이터 출력하기')
        books = cur.fetchall()

        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally:
        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success, ret_df

is_success, books_df = select_all_books(db_name)
if is_success:
    print('조회된 데이터는 총 %d 건 입니다.' % len(books_df))
else :
    print('데이터를 조회하지 못했습니다')

books_df

```

```

# 일부 조회용 함수
def select_some_books(db_name, number):
    """
    일부 데이터를 조회하는 함수
    Args:
        db_name : Database Name
        number : Count of data to query
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books"
        cur.execute(db_sql)

        # 조회한 데이터 일부 불러오기
        print('[2] 데이터 일부 출력하기')
        books = cur.fetchmany(number)

        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally:
        conn.close()

    return is_success, ret_df

# select_some_books(db_name, number=3)

is_success, books_df = select_some_books(db_name, number=3)
if is_success:
    print('조회된 데이터는 총 %d 건 입니다.' % len(books_df))
else:
    print('데이터를 조회하지 못했습니다')

books_df

```

```

# 1 개 조회용 함수
def select_one_book(db_name):
    """
    최상단 하나의 데이터를 조회하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = "SELECT * FROM my_books "
        cur.execute(db_sql)

        # 데이터 한개 출력하기
        print('[3] 1 개 데이터 출력하기')
        book = cur.fetchone()
        books = [book]
        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally:
        conn.close()

    return is_success, ret_df

is_success, books_df = select_one_book(db_name)
if is_success:
    print('하나의 데이터를 성공적으로 조회하였습니다.')
else :
    print('데이터를 조회하지 못했습니다')

books_df

```

```

# 쪽수 많은 책 조회용 함수
def find_big_books(db_name):
    """
    조건에 맞는 데이터를 조회하는 함수
    조건 : 페이지수가 300 쪽보다 큰 데이터
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
        ret_df : DataFrame of books
    """
    ret_df = pd.DataFrame()
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)
        cur = conn.cursor()

        # 조회용 SQL 실행
        db_sql = " SELECT * FROM my_books "
        db_sql += " WHERE pages > 300 "
        cur.execute(db_sql)

        # 조회한 데이터 불러오기
        print('[4] 페이지 많은 책 출력하기')
        books = cur.fetchall()
        ret_df = getBooksDF(books)

    except:
        is_success = False
        print("Database Error!")

    finally:
        conn.close()

    return is_success, ret_df

is_success, books_df = find_big_books(db_name)
if is_success:
    print('조건에 맞는 데이터는 총 %d 건 입니다. (조건:pages>300)' % len(books_df))
else:
    print('데이터를 조회하지 못했습니다')

books_df

```

## # 데이터 수정

```

def update_books(db_name):
    """
    데이터를 수정하는 함수
    Args:
        db_name : Database Name
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 데이터 수정 SQL ( 제목이 ? 인 책의 추천 유무를 ? 로 변경하라 )
        db_sql = "UPDATE my_books SET recommendation=%s WHERE title=%s "

        # 수정 SQL 실행
        cur.execute(db_sql, (1, '메가트랜드'))

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success

```

```
is_success, books_df1 = select_one_book(db_name)

if update_books(db_name):
    print('데이터가 성공적으로 수정되었습니다.')
else:
    print('데이터가 수정되지 않았습니다')

is_success, books_df2 = select_one_book(db_name)

books_df = pd.concat([books_df1, books_df2], axis=0)
books_df['update'] = ['수정전', '수정후']
books_df.set_index('update', inplace=True)
books_df
```



## # 데이터 삭제

```

# 데이터 삭제용 함수
def delete_books_by_title(db_name, title):
    """
    책제목에 해당하는 데이터를 삭제하는 함수
    Args:
        db_name : Database Name
        title : Title of the book to be removed
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()

        # 데이터 삭제 SQL
        db_sql = "DELETE FROM my_books "
        db_sql += "WHERE title = %s "

        # 수정 SQL 실행
        cur.execute(db_sql, (title,))

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success

```

```
title = '메가트랜드'
if delete_books_by_title(db_name, title):
    print('데이터가 성공적으로 삭제되었습니다.')
else :
    print('데이터가 삭제되지 않았습니다')

is_success, books_df = select_all_books(db_name)
books_df
```

```

def delete_books(db_name, col_name, col_val):
    """
    조건에 맞는 데이터를 삭제하는 함수
    Args:
        db_name : Database Name
        col_name : Column Name
        col_val : Column Value
    Returns :
        is_success : Boolean
    """
    is_success = True

    try:
        # 데이터베이스 커넥션 생성
        conn = get_connection(db_name)

        # 커서 확보
        cur = conn.cursor()
        # 데이터 삭제 SQL
        db_sql = 'DELETE FROM my_books '
        db_sql += 'WHERE {} = %s '
        db_sql = db_sql.format(col_name)

        # 수정 SQL 실행
        cur.execute(db_sql, (col_val,))

    except:
        is_success = False
        print("Database Error!")

    finally:
        if is_success:
            # 데이터베이스 반영
            conn.commit()
        else:
            # 데이터베이스 철회
            conn.rollback()

        # 데이터베이스 커넥션 닫기
        conn.close()

    return is_success

is_success, books_df = select_all_books(db_name)
books_df

```

```
col_name = 'publisher'
col_val = 'A'
if delete_books(db_name, col_name, col_val):
    print('데이터가 성공적으로 삭제되었습니다.')
else :
    print('데이터가 삭제되지 않았습니다')

is_success, books_df = select_all_books(db_name)
books_df

col_name = 'title'
col_val = '사물인터넷 전망'
if delete_books(db_name, col_name, col_val):
    print('데이터가 성공적으로 삭제되었습니다.')
else :
    print('데이터가 삭제되지 않았습니다')

is_success, books_df = select_all_books(db_name)
books_df
```

## Tip. Analysis for Survey

다음의 4단계를 따라 설문조사 결과를 더 효과적으로 계산하세요.

- 1) 가장 핵심적인 연구조사 질문 살펴보기
- 2) 결과 교차분석 및 필터링하기
- 3) 통계 수치 계산하기
- 4) 결론 도출하기

### S1. 가장 핵심적인 연구조사 질문 살펴보기

우선, 가장 핵심적인 질문으로부터 확보한 설문조사 결과를 분석하는 방법부터 살펴보겠습니다. 경험적인 연구 질문도 포함시키셨나요? 확률적 표본추출도 고려해 보셨나요? 설문조사 목표를 정했을 때 핵심적인 연구조사 질문에 대한 윤곽도 정했어야 합니다.

예를 들어, 교육 컨퍼런스를 주최하고 참석자들에게 행사 후 피드백 설문조사를 보냈다고 가정하죠. 여기서 가장 핵심적인 연구조사 질문 중 하나는 '컨퍼런스에 대한 전반적인 참석자들의 평가'일 것입니다. 참석자들이 행사를 전반적으로 어떻게 평가했습니까? 다음과 같은 특정 설문조사 질문에 대해 수집된 답변을 살펴보면 이 핵심적인 연구조사 질문에 대한 답을 얻을 수 있습니다.

내년에 이 컨퍼런스에 참석할 계획입니까?

보기		
예	71%	852
아니요	18%	216
확실하지 않음	11%	132
총계		1,200

응답을 살펴보면 백분율(71%, 18%)이 있고 처리되지 않은 수치(852, 216)가 있습니다.

백분율은 특정 답변을 제공한 사람들에 대한 백분율입니다. 이를 다른 말로 설명하면 백분율은 곧 각 답변을 제공한 사람들의 수를 해당 질문에 답변을 제공한 사람들의 수에 대한 비율로서 나타내는 것입니다. 이 표에서는 설문조사 응답자의 71%(설문조사 대상 1,200명 중 852명)가 내년에 다시 참석할 의향을 나타내고 있습니다.

18%의 사람들은 다시 참석할 의향이 없고 11%는 아직 결정하지 않았음을 보여줍니다.

## S2. 결과 교차분석 및 필터링하기

설문조사 목표를 정하고 분석 계획을 개발했을 때 어떤 하위 그룹을 분석하고 비교할 것인지에 대해 생각해 두셨던 것을 기억하시나요? 예를 들어, 내년에 열릴 컨퍼런스에 대한 질문에 교사, 학생 및 행정인들의 답변을 서로 비교한다고 가정하죠. 이를 알아내기 위해 컨퍼런스 질문에 대한 결과를 하위 그룹별로 나타낼 수 있는 교차분석을 통해 응답률을 살펴볼 수 있습니다.

	예	아니요	확실하지 않음	총계
교사	80% 320	7% 28	13% 52	400
행정인	46% 184	40% 160	14% 56	400
학생	86% 344	8% 32	6% 24	400
총 응답자 수	852	216	132	1,200

이 표를 미루어 보아 학생 및 교사들의 대부분(각각 86% 및 80%)이 내년에 다시 참석할 계획임을 알 수 있습니다. 하지만 이와는 달리, 올해 컨퍼런스에 참석했던 행정인들 중 절반에 미치지 못하는(46%) 행정인들이 내년에 참석할 계획임을 보여주고 있습니다! 다른 질문들을 통해 어째서 이런 현상이 벌어졌는지를 알아보고 행정인들을 위해 컨퍼런스를 개선하여 내년에 더 많은 행정인들이 다시 참석하도록 할 수 있습니다.

필터는 데이터 모형화에 사용할 수 있는 또 다른 유용한 도구입니다. **필터링이란 다른 요소들을 모두 제거하고 한 개의 특정 하위 그룹에만 초점을 맞추는 것입니다.** 즉 하위 그룹들을 서로 비교하는 것이 아니라, 단 한 개의 하위 그룹이 어떻게 답변했는지를 살펴보게 됩니다. 예를 들어, 여성 행정인, 여성 교사, 여학생들을 비교하기 위해 여성(또는

남성)에만 중점을 두어 참석자 유형별 교차분석을 다시 실행할 수 있습니다. 이렇게 결과를 여러 가지 면으로 분석할 때 주의해야 할 점은 필터 또는 교차 분석을 적용할 때마다 표본 크기가 줄어든다는 것입니다. 통계적으로 유의성 있는 결과를 확보하기 위해 표본 크기 계산기를 사용할 수 있습니다.

### 벤치마크, 동향 및 비교 데이터

컨퍼런스 피드백 설문조사에서 핵심 질문 중 하나가 '컨퍼런스에 대해 전반적으로 얼마나 만족'했는지를 묻는 질문이었다고 가정하죠. 여기서 75%의 참석자들이 컨퍼런스에 만족했다고 답변했습니다. 꽤 높은 만족도죠? 하지만 이 수치가 좀 더 의미를 가지도록 하는 것은 어떨까요? 다른 요소와 비교해 보는 것은요? 작년보다 더 나아진 수치인가요, 아니면 더 하락된 수치인가요? 다른 컨퍼런스와 비교했을 때는 어떨까요?

또한, 이 질문을 작년 컨퍼런스 후의 컨퍼런스 피드백 설문조사에서도 물었다고 가정하죠. 이 경우엔 동향 비교를 할 수 있습니다. 여론조사 전문가들이 하나같이 이구동성으로 하는 말은 '동향을 알아야 길이 보인다'라는 것입니다.

작년 만족도가 60%였다면 올해에는 15%나 그 수치가 상승했습니다! 만족도를 상승시킨 요인은 무엇일까요? 설문조사의 다른 질문들에서 이에 대한 답변을 찾아볼 수 있습니다.

작년 컨퍼런스 데이터가 없는 경우, 올해가 각 컨퍼런스마다 피드백을 수집하기 시작하는 시점이 됩니다. 이를 벤치마킹이라고 합니다. **벤치마크나 기준치를 수립하고 나면 다음부터는 이러한 수치에 변화가 생겼는지, 어째서 변화가 생겼는지를 알 수 있게 됩니다.** 참석자들의 만족도뿐만 아니라 다른 질문들도 벤치마크하여 참석자들이 컨퍼런스에 대해 어떻게 생각하는지 매년 확인할 수 있습니다. 이 방법을 **종단적 데이터 분석**이라고 합니다.

서로 다른 하위 그룹에 대한 데이터도 추적할 수 있습니다. 예를 들어, 학생과 교사의 만족도는 매년 계속해서 상승되나 행정인들의 만족도는 그렇지 않다고 가정하죠. 이런 경우엔 행정인들이 답변을 제공한 다른 여러 가지 질문들을 연구하여 어째서 행정인들이 다른 참석자들보다 덜 만족해 하는지에 대한 통찰력을 확보할 수 있습니다.

### S3. 통계 수치 계산하기

다시 참석할 의향이 있다고 밝힌 사람들의 수는 알고 있지만 설문조사로부터 확보한 답변이 얼마나 신뢰성 있으며 향후 의사결정에 확신을 가지고 사용할 수 있는지 어떻게 알 수 있을까요? 데이터 질에 주의를 기울이고 통계적 유의성에 대한 요소를 이해하는 것은 매우 중요합니다.

일상적 대화에서 '유의성'이라는 말은 중요하거나 의미가 있다는 뜻으로 사용됩니다. **설문조사 분석이나 통계학에서의 유의성은 '정확성에 대한 평가'를 뜻합니다.** 바로 여기서 설문조사 작업에 '플러스 또는 마이너스'가 사용되기 시작합니다. 구체적으로 말하면 이는 설문조사 결과가 특정 신뢰 수준 내에서 정확성이 있으며 그 정확성이 임의적인 요소로 인한 결론이 아님을 뜻합니다. 부정확한 결과(즉, 통계적 유의성이 없는 결론)를 기반으로 도출한 결론은 위험성을 지닙니다. 모든 통계적 유의성 평가에서 첫 번째로 고려해야 하는 요소는 표본의 대표성입니다. 즉, 설문조사에 포함된 그룹의 사람들이 결론을 도출하고자 하는 대상인 사람들 전체를 얼마나 잘 나타내고 있는지를 고려하는 것입니다.

설문조사를 완성한 컨퍼런스 참석자들 중 90%가 남성이나 컨퍼런스 참석자들의 15%만이 남성이라면 여기서 결론되는 수치를 확신할 수 없습니다. 연구하고자 하는 인구 집단에 대해 더 많이 알면 알수록 이러한 수치와 설문조사가 일치할 때 설문조사로부터 확보하는 결과에 확신을 가지게 됩니다. 이 예에서 성별로 보자면 남성이 설문조사 응답자의 15%를 차지했다면 문제가 될 일이 없습니다.

설문조사 표본이 이미 파악되어 있는 인구 집단의 무작위 선택 표본이라면 통계적 유의성은 간단한 방법으로 계산될 수 있습니다. 여기서 **가장 중요한 요소는 표본 크기**입니다. 예를 들어, 컨퍼런스에 참석한 1,000명 중 50명이 설문조사에 응답했다고 가정하죠. 이 50명은 작은 표본 크기로 인해 넓은 오차 한계를 가지게 됩니다. 간단히 말해, 이 결과의 의미는 그다지 신뢰성이 없게 되는 것입니다.

이번엔 설문조사 응답자들에게 컨퍼런스 중 진행된 10개의 세션 중 몇 개에 참석했는지 물었고 다음과 같은 결과를 확보하게 되었다고 가정해 보죠.

	1	2	10	총계	평균 등급
참석한 세션 수	10%	0%	1%		
	100	0	10	1,000	6.1



여기서는 평균을 분석하고자 합니다. 평균에는 평균값, 중앙값, 최빈값의 세 가지 종류가 있습니다.

위의 표에서 보자면 세션에 참석한 평균 수는 6.1개입니다. 표에 나와 있는 평균은 평균값으로 대부분의 사람들에게 익숙한 종류의 평균입니다. 평균값을 계산하려면 데이터를 모두 합한 후, 더한 수치의 수로 나눕니다. 이 예에서는 100명이 1개의 세션에 참석했고, 50명은 4개, 100명은 5개에 참석했음을 알 수 있습니다. 참석 세션 수와 인원 수 쌍을 서로 곱하고, 이 곱에서 나온 수치를 합하여 총 인원 수로 나눕니다.

중앙값은 평균의 또 다른 종류입니다. 중앙값은 중간에 있는 값, 즉 50% 지점을 뜻합니다. 위의 표에서는 500명이 세션 수의 왼쪽에 있고 다른 500명은 오른쪽에 있는 세션 수를 파악하고자 합니다. 이 경우 중앙값은 세션 6개입니다. 이렇게 하면 데이터에 부정적인 영향을 주는 이상치로부터의 영향을 제거할 수 있습니다.

마지막 종류의 평균은 최빈값입니다. 최빈값은 가장 빈번한 응답입니다. 여기서 최빈값은 6입니다. 왜냐하면 6개의 세션에 참석한 응답 수가 다른 어떠한 참석 세션 수보다도 더 많았기 때문입니다.

다른 종류의 평균인 평균값은 결과가 리커트 척도에 기반하는 경우에도 사용될 수 있습니다.

#### S4. 결론 도출하기

설문조사 결과에 대해 보고할 때에는 데이터가 전달하고자 하는 의미를 생각해야 합니다.

컨퍼런스가 전반적으로 그다지 좋지 않은 평가를 받았다고 가정하죠. 그렇다면 왜 그런 평가를 받게 되었는지 좀 더 심층적으로 살펴보아야 합니다. 데이터를 살펴보면 세션, 수업, 사교 행사, 호텔 등 컨퍼런스의 모든 측면에서 참석자들이 매우 높은 등급을 주었으나 컨퍼런스가 열린 도시를 좋아하지 않았을 수도 있습니다. 예를 들어 겨울철 1월에 가장 추운 도시에서 컨퍼런스가 개최되어 참석자들이 밖으로 나가기에 너무 추웠을 수 있습니다. 바로 이러한 사항이 데이터가 전달하고자 하는 의미입니다. 컨퍼런스 자체는 매우 훌륭했으나 장소 면에서 좋지 않은 선택을 한 것입니다. 이러한 의미로부터 겨울에는 따뜻한 곳에서 컨퍼런스를 개최하는 개선점을 만들 수 있습니다.

데이터 분석 및 보고 요소들 중 고려해야 하는 한 가지는 **인과관계와 상관관계**입니다.

## APPENDIX

### 설문조사 데이터 수집

설문조사 데이터 수집은 설문조사를 이용하여 특정 응답으로부터 정보를 수집하는 것입니다. 설문조사 데이터 수집은 인터뷰, 포커스 그룹 등을 포함한 다른 데이터 수집 유형을 대체하거나 보완할 수 있습니다. 설문조사로부터 수집된 데이터는 직원 몰입도를 증가시키고, 구매자 행동 유형을 이해하며, 고객 체험을 향상하는 데 이용할 수 있습니다.

### 종단적 분석

종단적 데이터 분석(흔히 '동향 분석'이라고 함)은 기본적으로 **특정 질문에 대해 알게 된 사항들이 시간의 경과에 따라 어떻게 변하는지를 추적하는 것**입니다. 일단 벤치마크를 수립하고 나면 수치가 변하는지, 어떻게 변하는지 측정할 수 있습니다. 컨퍼런스에 대한 만족도가 3년 전에는 50%였고, 2년 전에는 55%, 작년에는 65%, 올해는 75%였다고 가정하죠. 이 경우엔 종단적 데이터 분석이 만족도 면에서 계속 상승 추세를 보여주므로 바람직한 변화를 나타내고 있습니다.

### 상관관계와 인과관계의 차이

인과관계는 한 가지 요소가 다른 요소의 원인이 되는 것이며 상관관계는 두 가지 변수가 함께 움직이지만 서로 영향을 미치거나 서로 다른 변수의 원인이 되지는 않습니다. 예를들어, 겨울철에 핫초콜릿을 마시고 병어리 장갑을 끼는 것은 상관관계가 있는 두 가지 변수입니다. 즉, 함께 상승하거나 함께 하락합니다. 하지만 핫초콜릿과 병어리 장갑이 서로의 원인이 되지는 않으며 둘 다 모두 추운 날씨와 같이 제3의 요소가 그 원인이 됩니다. 추운 날씨는 핫초콜릿 수요와 병어리 장갑을 끼게 되는 확률을 모두 높입니다. 추운 날씨는 독립 변수이며 핫초콜릿 수요와 병어리 장갑을 끼게 되는 확률은 종속 변수입니다. 여기서 설명한 컨퍼런스 의견 설문조사에서는 추운 날씨가 컨퍼런스가 열린 도시와 컨퍼런스 전반에서 참석자들의 불만족에 영향을 미쳤습니다. 최종적으로, 설문조사의 변수 간 관계를 더 심층적으로 조사하기 위해 회귀 분석을 실행해야 할 수도 있습니다.

## 회귀 분석

회귀 분석은 고급 데이터 분석 방법으로 두 가지 이상의 변수들 간의 관계를 연구할 수 있습니다. 회귀 분석에는 여러 가지 종류가 있으며 설문조사 과학자가 선택하는 분석법은 조사하고자 하는 변수에 달려 있습니다. 하지만 모든 종류의 회귀 분석이 가지고 있는 공통점은 **종속 변수에 미치는 한 가지 이상의 독립 변수의 영향을 파악**한다는 것입니다. 설문조사 데이터 분석을 통해 세션 수, 기조연설자, 또는 사고 행사나 컨퍼런스 장소 등과 같이 컨퍼런스 참석자들의 만족도에 가장 큰 영향을 미치는 요소가 무엇인지를 알아보려고 할 수도 있습니다. 세션 수가 영향을 미쳤나요? 기조연설자였나요? 사고 행사였나요? 아니면 행사 장소였나요? 설문조사 과학자는 여기에서 회귀 분석을 사용하여 컨퍼런스의 서로 다른 이러한 요소들이 전체적인 만족도에 영향을 미쳤는지와 어느 정도로 영향을 미쳤는지를 측정할 수 있습니다.

이러한 측정을 통해 다음 번 컨퍼런스에서는 어떤 면을 개선할 것인지를 알게 됩니다. 예를 들어, 오프닝 세션에 최고의 기조연설자를 고용하기 위해 매우 높은 사례금을 지불했다고 가정하죠. 참석자들이 이 연설자와 컨퍼런스에 전체적으로 높은 점수를 주었습니다. 이 두 가지만을 기반으로 하자면 비싸고 인기 있는 기조연설자를 고용하는 것이 컨퍼런스의 성공을 좌지우지한다고 결론을 내릴 수도 있습니다. 회귀 분석을 사용하여 이러한 결론이 정확한지를 알아볼 수 있습니다. 기조연설자의 인기도가 컨퍼런스 만족도를 결정 짓는 주요 요소였다는 사실을 발견하게 될 수 있습니다. 이런 경우엔, 내년에도 다시 최고의 기조연설자를 고용할 필요가 있습니다. 하지만 회귀 분석에서 참석자들이 연설자를 좋아하는 했지만 이 사실이 참석자들의 만족도에 크게 영향을 미치지 않았다는 사실을 알게 되었습니다. 이런 경우엔 기조연설자 고용에 소요한 높은 지출 비용을 다른 곳에 사용해야 할 수 있습니다. 설문조사 데이터가 제시하는 의미를 신중하게 분석함으로써 답변을 사용하여 충분한 정보에 입각한 결정을 내리는 데 큰 도움을 받을 수 있습니다.

- 끝 -