

데 이 터 의 수 집 및 활 용

데이터 분석을 위한 데이터베이스

CONTENTS

01

DataBase

- DataBase
- Database 특징
- SQL? NoSQL?

02

mySQL

- SQL이란?
- MySQL 설치
- SQL명령어

03

MongoDB

- MongoDB란?
- MongoDB 설치
- MongoDB 명령어



DataBase

01

DataBase

데이터베이스(DB: database)는 통합하여 관리되는 데이터의 집합체를 의미한다.

이는 중복된 데이터를 없애고, 자료를 구조화하여, 효율적인 처리를 할 수 있도록 관리된다.

따라서, 여러 업무에 여러 사용자가 데이터 베이스를 사용할 수 있다.

이러한 데이터베이스는 응용 프로그램과는 다른 별도의 미들웨어에 의해 관리된다.

데이터베이스를 관리하는 이러한 미들웨어를 데이터베이스 관리 시스템(DBMS: Database Management System)이라고 한다.

01

DataBase 특징

1. 사용자의 질의에 대하여 즉각적인 처리와 응답이 이루어진다.
2. 생성, 수정, 삭제를 통하여 항상 최신의 데이터를 유지한다.
3. 사용자들이 원하는 데이터를 동시에 공유할 수 있다.
4. 사용자가 원하는 데이터를 주소가 아닌 내용에 따라 참조 할 수 있다.
5. 응용프로그램과 데이터베이스는 독립되어 있으므로, 데이터의 논리적 구조와 응용프로그램은 별개로 동작된다.

01

SQL? NoSQL

	RDBMS	NoSQL
적합한 사용례	데이터 정합성이 보장되어야 하는 은행 시스템	낮은 지연 시간, 가용성이 중요한 SNS 시스템
데이터 모델	정규화와 참조 무결성이 보장된 스키마	스키마가 없는 자유로운 데이터 모델
트랜잭션	강력한 ACID 지원	완화된 ACID(BASE)
확장	하드웨어 강화(Scale up)	수평 확장 가능한 분산 아키텍처(Scale out)
API	SQL 쿼리	객체 기반 API 제공



MySQL

02

SQL이란?

SQL(Structured Query Language)은 데이터베이스에서 데이터를 정의, 조작, 제어하기 위해 사용하는 언어입니다.

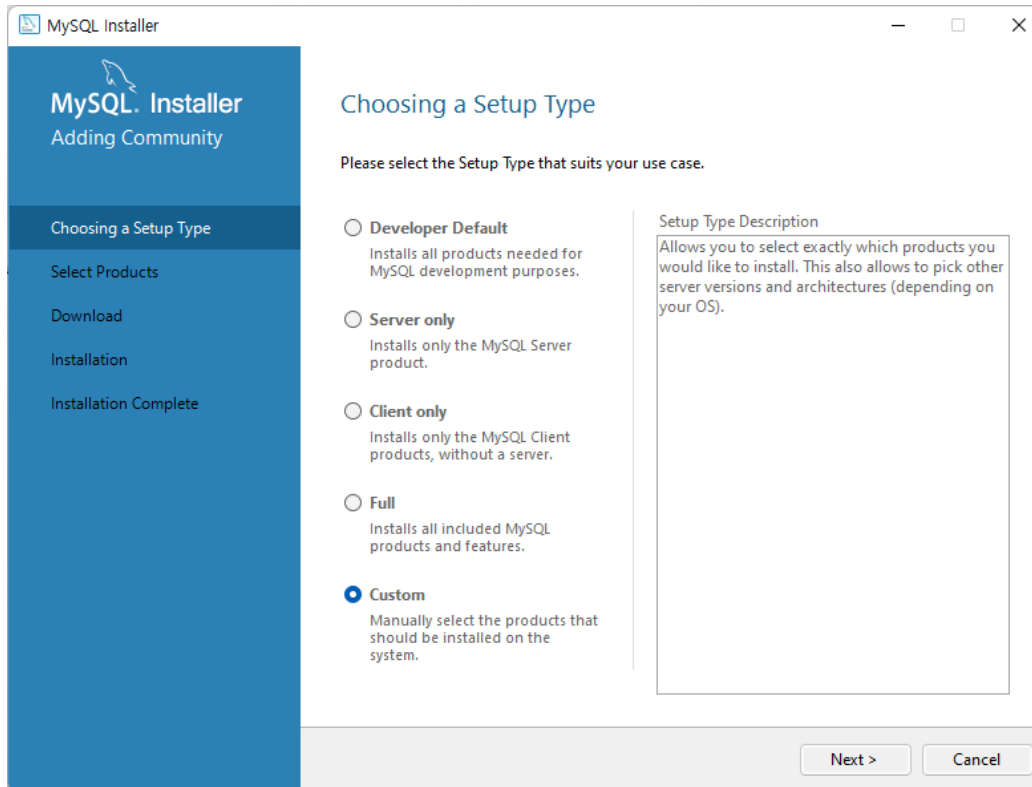
따라서 SQL 구문도 위의 목적에 맞게 크게 세 가지로 구분할 수 있습니다.

속성	설명	주요 명령어
DDL	데이터베이스나 테이블 등을 생성, 삭제하거나 그 구조를 변경하기 위한 명령어	CREATE, ALTER, DROP
DML	데이터베이스에 저장된 데이터를 처리하거나 조회, 검색하기 위한 명령어	INSERT, UPDATE, DELETE, SELECT 등
DCL	데이터베이스에 저장된 데이터를 관리하기 위하여 데이터의 보안성 및 무결성 등을 제어하기 위한 명령어	GRANT, REVOKE 등

02

MySQL 설치

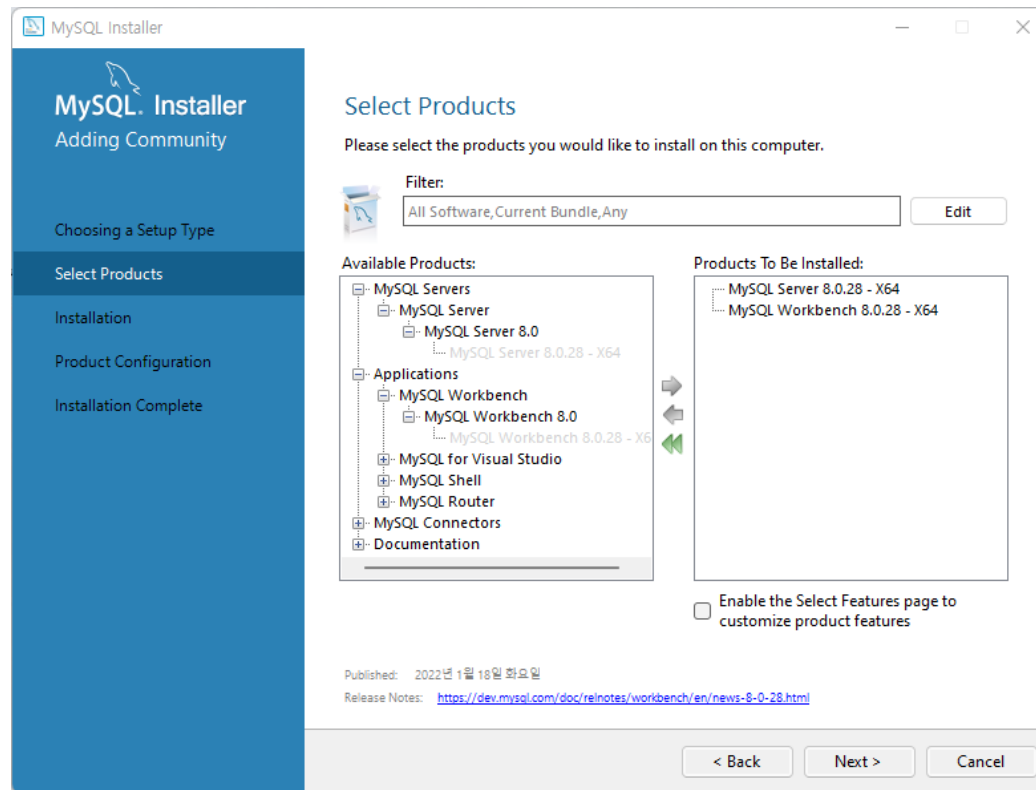
<https://dev.mysql.com/downloads/mysql/>
링크를 통해 인스톨 파일을 다운로드
파일 실행 후 Custom 체크 - Next



02

MySQL 설치

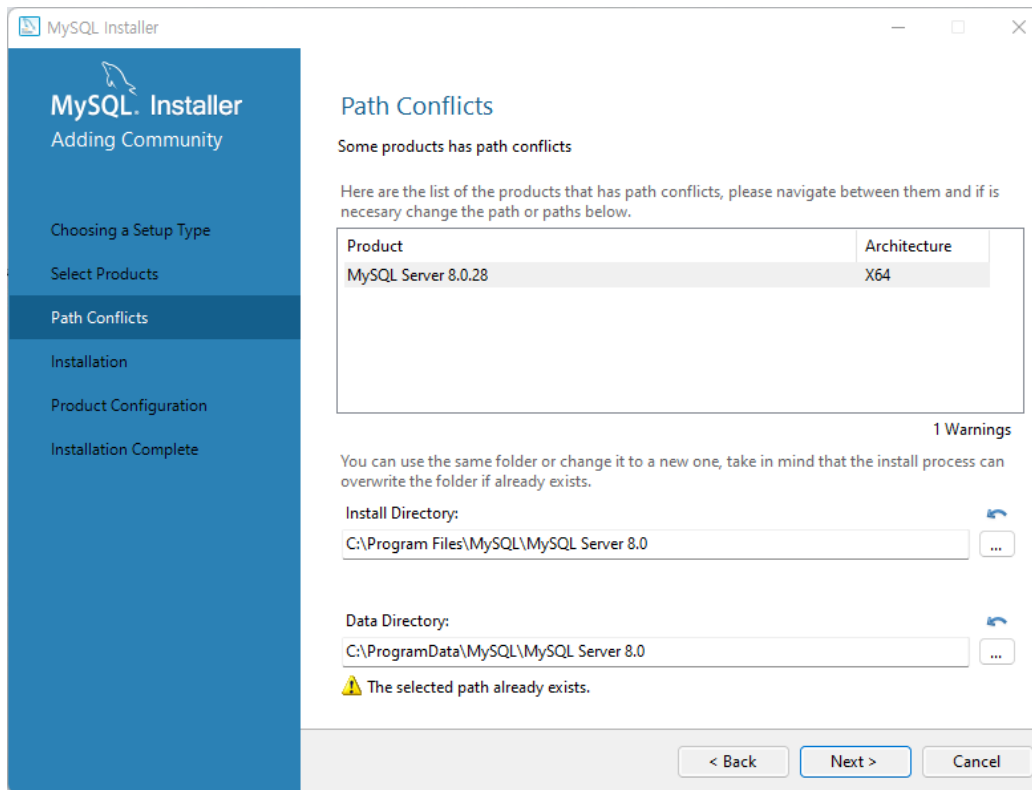
MySQL Server와 MySQL Workbench 추가 후 Next



02

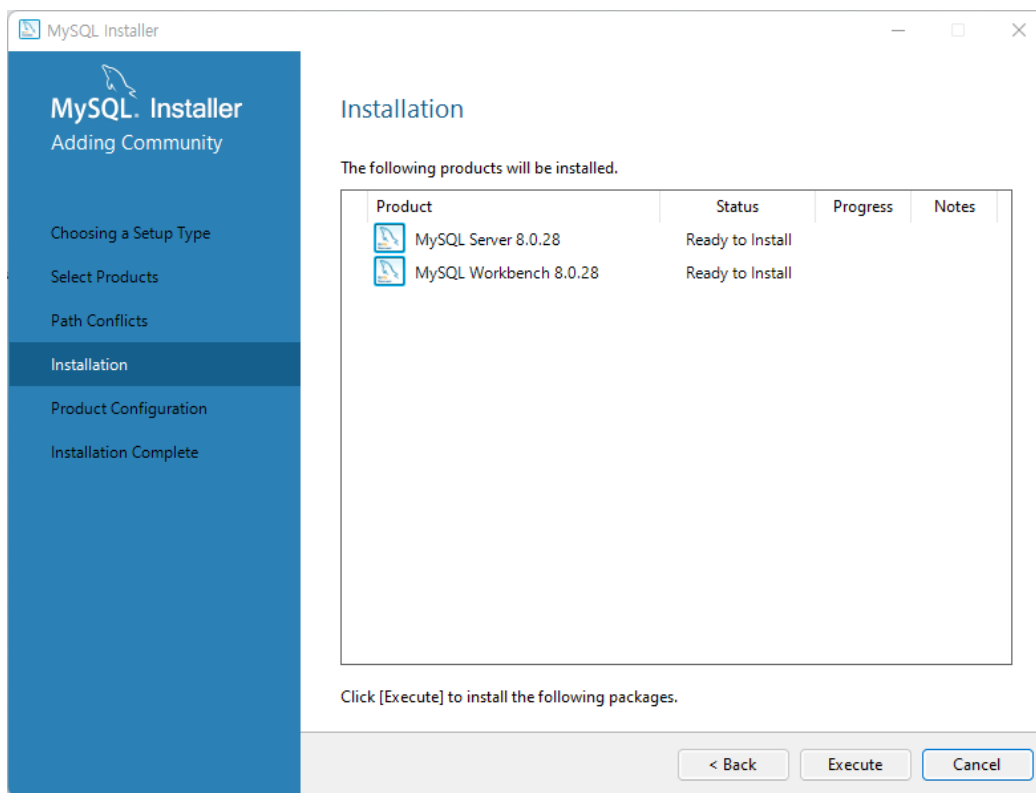
MySQL 설치

설치 위치 설정 페이지 바로 Next



MySQL 설치

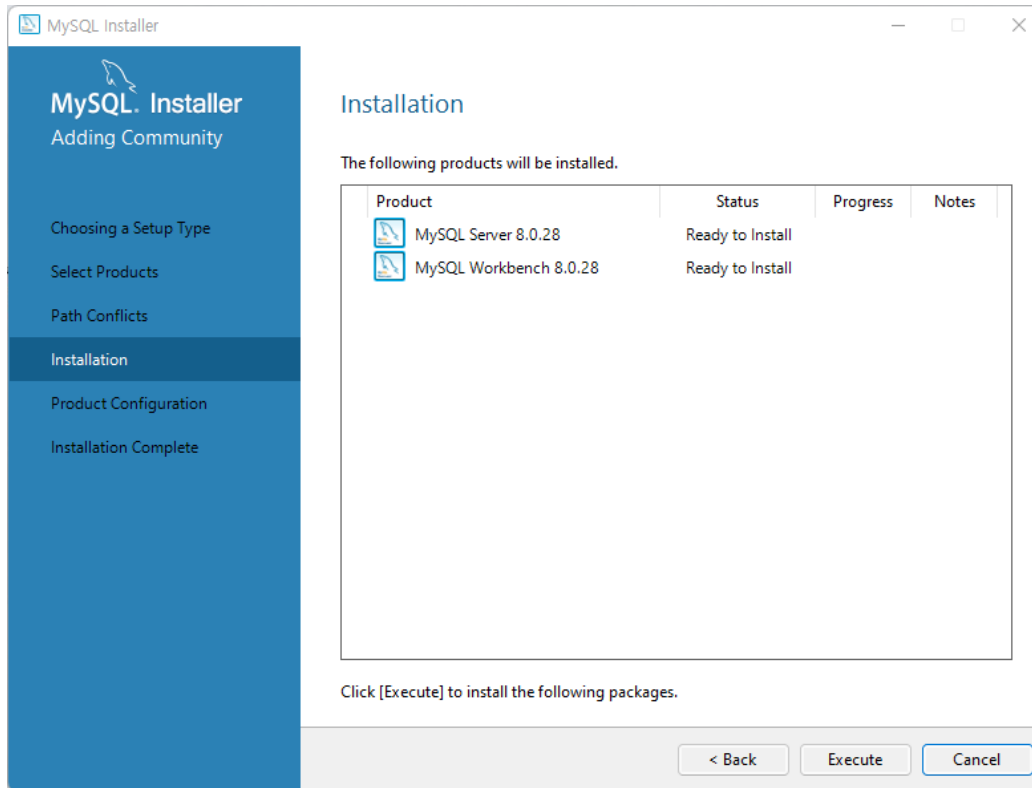
설치 목록 확인 Execute



02

MySQL 설치

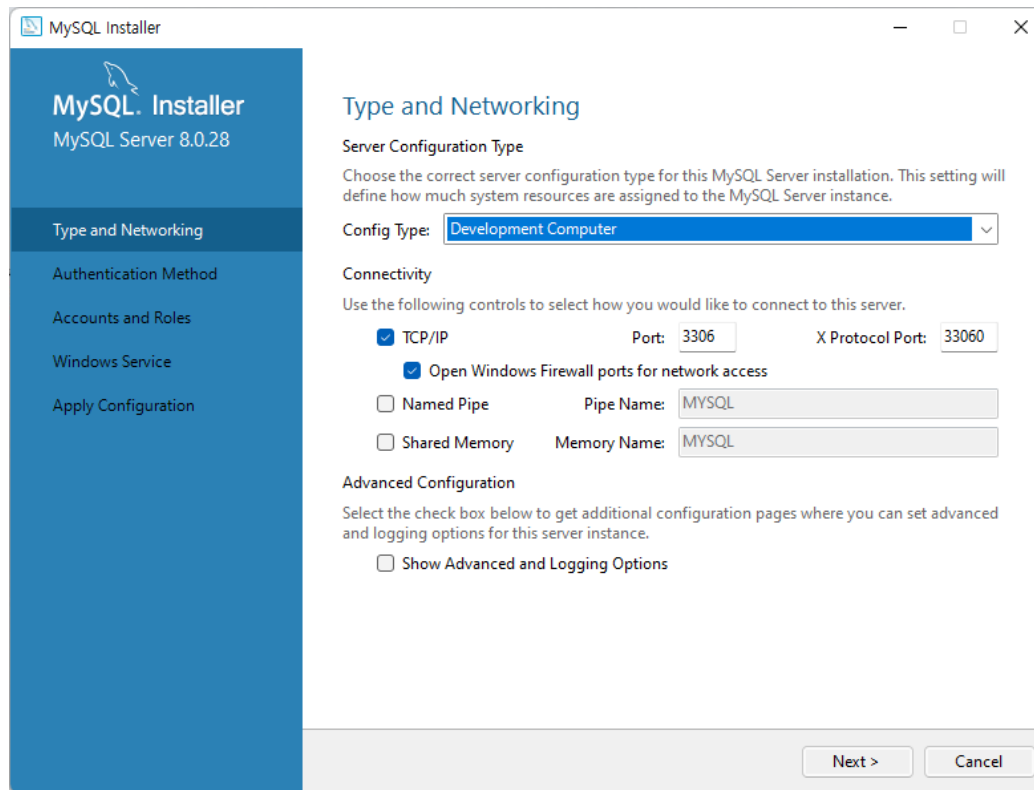
설치 목록 확인 Execute → 설치가 완료 되면 Next



02

MySQL 설치

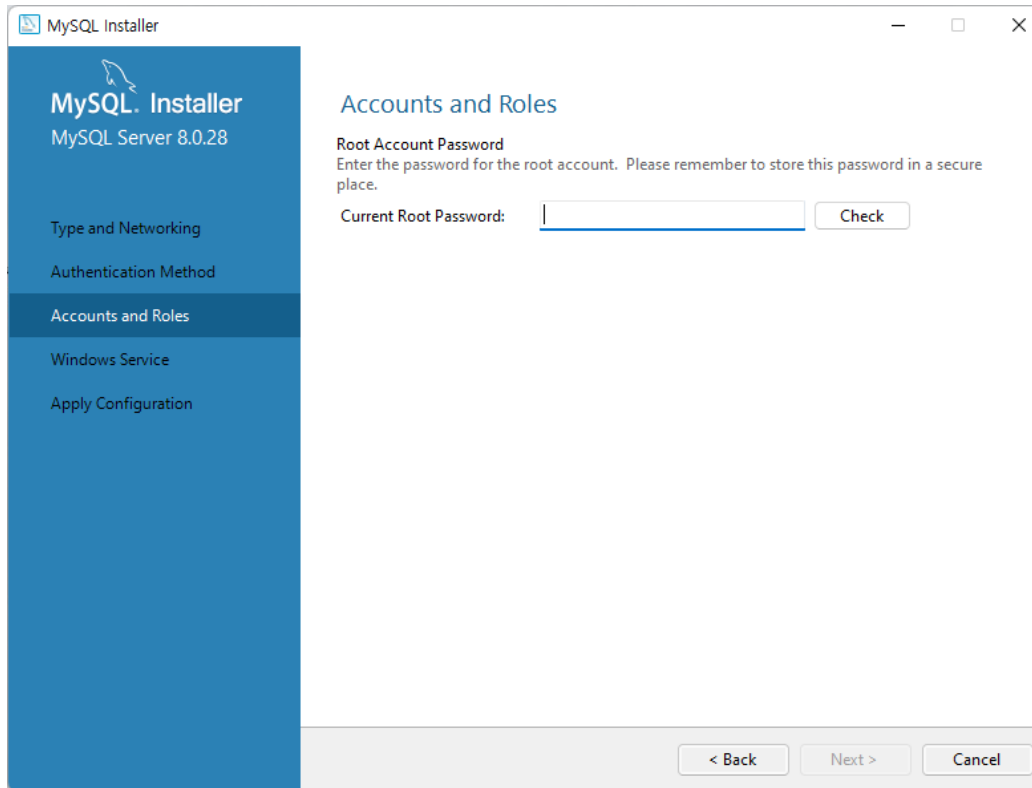
네트워크 타입 설정 부분 Next



02

MySQL 설치

SQL의 Password 설정 부분 각자 비밀번호를 입력 후 Check → Next



02

CREATE 문

- 새로운 데이터베이스 생성

```
CREATE DATABASE 데이터베이스이름;
```

```
새로운 데이터베이스 선택  
USE 데이터베이스이름;
```

- 테이블 생성

```
CREATE TABLE 테이블이름  
(  
    필드이름1 필드타입1,  
    필드이름2 필드타입2,  
    ...  
);
```


02

데이터의 필드 타입

유형	표현	byte	표현 범위	비고
숫자형	TINYINT[(M)]	1	-128 ~ 127 0 ~ 255	
	INT[(M)]	4	-2147483648 ~ 2147483647 0 ~ 4294967295	
문자형	CHAR(M)	고정	1~255 까지 고정형, 남은 공간 공백 채움	
	VARCHAR(M)	가변	1~255 까지 가변길이, 문자길이 + 1byte	
	TEXT	가변	~ 65535	문자 데이터 저장

02

제약 조건

제약 조건(constraint)이란 데이터의 무결성을 지키기 위해 데이터를 입력받을 때 실행되는 검사 규칙을 의미한다.

이러한 제약 조건은 CREATE 문으로 테이블을 생성할 때나, ALTER 문으로 필드를 추가할 때도 설정할 수도 있다.

CREATE TABLE 문에서 사용할 수 있는 제약 조건은 다음과 같다.

1. NOT NULL : 해당 필드는 NULL 값을 저장할 수 없게 된다.
2. UNIQUE : 해당 필드는 서로 다른 값을 가져야만 한다.
3. PRIMARY KEY : 해당 필드가 NOT NULL과 UNIQUE 제약 조건의 특징을 모두 가지게 된다.
4. FOREIGN KEY : 하나의 테이블을 다른 테이블에 의존하게 만든다.
5. DEFAULT : 해당 필드의 기본값을 설정한다.

02

제약 조건

AUTO_INCREMENT 키워드를 사용하면 해당 필드의 값을 1부터 시작하여 새로운 레코드가 추가될 때마다 1씩 증가된 값을 저장한다. 이때 AUTO_INCREMENT 키워드 다음에 대입 연산자(=)를 사용하여 시작 값을 변경할 수 있다.

02

ALTER 문

- 데이터베이스 수정

ALTER DATABASE 데이터베이스이름 CHARACTER = 문자집합이름

ALTER DATABASE 데이터베이스이름 COLLATE = 콜레이션이름

- 테이블 수정

ALTER TABLE 테이블이름 OPTION

02

ALTER 문 (테이블 수정)

ALTER TABLE 문은 테이블에 필드를 추가, 삭제, 수정을 할수 있게 해준다.

- ADD

ALTER TABLE 테이블이름 ADD 필드이름 필드타입;

- DROP

ALTER TABLE 테이블이름 DROP 필드이름

- MODIFY COLUMN

ALTER TABLE 테이블이름 MODIFY
COLUMN 필드이름 필드타입

DROP 문

DROP문을 사용하면 데이터베이스나 테이블을 삭제 할 수 있다.

1. DROP DATABASE 데이터베이스이름
2. DROP TABLE 테이블 이름

02

INSERT 문

INSERT INTO문을 사용하여 테이블에 새로운 값을 추가 할 수 있다.

```
INSERT INTO 테이블이름(필드이름1, 필드이름2, 필드이름3, ...)  
VALUES (데이터값1, 데이터값2, 데이터값3, ...);
```

```
INSERT INTO 테이블이름 VALUES (데이터값1, 데이터값2, 데이  
터값3, ...);
```

두번째 문법과 같이 사용을 하면 테이블의 필드에 순서대로 자동 대입이 된다.

02

UPDATE 문

UPDATE문은 테이블의 내용을 수정할 수 있다.

UPDATE 테이블이름 SET 필드이름1 = 데이터값1, 필드이름2 = 데이터값2, WHERE 필드이름 = 데이터값;

WHERE절의 조건에 맞는 테이블의 값을 수정한다.
WHERE절이 없다면 테이블의 모든 값을 수정한다.

02

DELETE 문

DELETE문은 테이블의 데이터 값을 삭제 할 수 있다.

DELETE FROM 테이블이름 WHERE 필드이름 = 데이터값;

WHERE절에 만족하는 테이블의 값만 삭제한다.

UPDATE문과 마찬가지로 WHERE절이 없으면 테이블의 모든 데이터가 삭제된다.

(DROP문과의 차이는 DROP문은 테이블을 삭제 DELETE문은 테이블 내의 데이터만 삭제한다.)

02

SELECT 문

SELECT문은 테이블의 데이터를 선택 할 수 있다.

SELECT 필드이름 FROM 테이블이름 [WHERE 조건식];

SELECT * FROM 테이블이름

필드이름에 *을 넣게 되면 모든 필드 값의 데이터를 선택한다.

02

WHERE절

SQL에도 프로그래밍 언어에서의 IF문과 같은 조건문이 존재합니다.

```
WHERE {COLUMN} = {VALUE};
```

"=" 위치에 "<>", "!=", ">", ">=", "<", "<=" 등의 비교연산자 사용

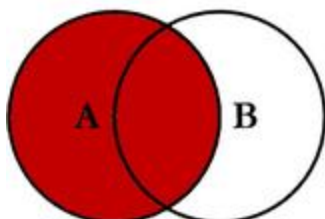
```
WHERE {COLUMN} IN (VALUE_LIST);
```

리스트에 없는 값을 선택할 땐 NOT IN 사용

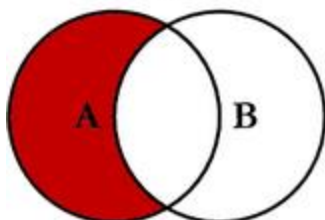
02

JOIN

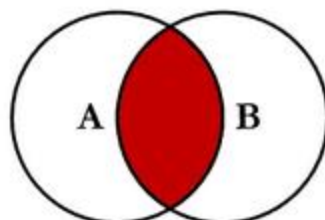
SQL JOINS



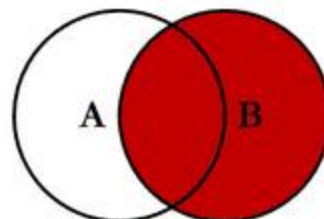
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



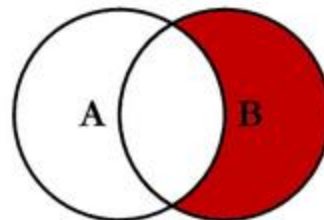
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



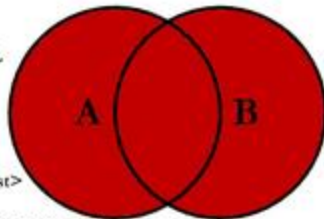
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



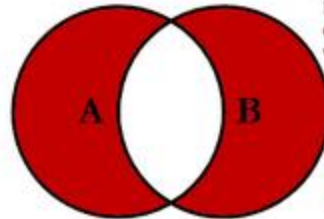
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

02

JOIN

JOIN은 데이터베이스 내의 여러 테이블에서 가져온 컬럼 값들을 조합하여 하나의 집합으로 표현해준다.
즉 서로 다른 테이블에서 데이터를 가져올 때 사용한다.

02

LEFT JOIN

A, B 테이블 중 A값의 전체와 A의 key 값과 B의 Key 값이 같은 결과를 출력한다.

```
SELECT A.컬럼명1, A.컬럼명2, B.컬럼명1, B.컬럼명2 FROM  
A LEFT JOIN B ON A.Key = B.Key;
```

RIGHT JOIN은 대상 테이블만 다르게 설정하면 된다.

02

INNER JOIN

A, B 테이블 중 두 테이블의 교집합인 데이터를 출력한다.

```
SELECT A.컬럼명1, A.컬럼명2, B.컬럼명1, B.컬럼명2 FROM  
A INNER JOIN B ON A.key = B.Key
```

02

FULL OUTER JOIN

MySQL에서는 FULL OUTER JOIN를 지원하지 않는다.
LEFT JOIN과 RIGHT JOIN을 이용하여 FULL OUTER JOIN을 사용 할 수 있다.

02

ORDER BY

SELECT 문으로 데이터를 조회할 때 ORDER BY를 추가하여 사용하면 지정된 컬럼을 기준으로 정렬이 가능하다.

ASC : 오름차순 (기본값)

DESC : 내림차순

오름차순 정렬

```
SELECT * FROM 테이블명 ORDER BY 컬럼명 ASC;
```

내림차순 정렬

```
SELECT * FROM 테이블명 ORDER BY 컬럼명 DESC;
```

02

GROUP BY

GROUP BY는 유형별로 개수를 알고 싶을 때 사용한다.

GROUP BY : 특정 컬럼을 그룹화

HAVING : 그룹화한 결과에 조건을 건다

HAVING과 WHERE의 차이는 WHERE은 그룹화 하기 전 조건이고, HAVING은 그룹화 후의 조건이다.

02

GROUP BY

그룹화

```
SELECT * FROM 테이블명 GROUP BY 그룹화할 컬럼명;
```

조건 처리 후 그룹화

```
SELECT * FROM 테이블명 WHERE 조건식 GROUP BY  
그룹화할 컬럼명;
```

그룹화 후 조건 처리

```
SELECT * FROM 테이블명 GROUP BY 그룹화할 컬럼명  
HAVING 조건식;
```

02

GROUP BY

조건 처리 후 그룹화 후 조건 처리

```
SELECT * FROM 테이블명 WHERE 조건식 GROUP BY  
그룹화할 컬럼명 HAVING 조건식;
```

ORDER BY절 포함

```
SELECT * FROM 테이블명 WHERE 조건식 GROUP BY  
그룹화할 컬럼명 HAVING 조건식 ORDER BY 컬럼명;
```

The MongoDB logo is a large, solid dark blue circle. Inside this circle, the word "MongoDB" is written in a white, bold, sans-serif font. The text is centered horizontally and vertically within the circle. Above the main circle, there is a smaller, semi-circular dark blue shape that appears to be part of a larger design element.

MongoDB

MongoDB란?

몽고DB(MongoDB←HUMONGOUS)는 크로스 플랫폼 도큐먼트 지향 데이터베이스 시스템이다.

NoSQL 데이터베이스로 분류되는 몽고DB는 JSON과 같은 동적 스키마형 도큐먼트들(몽고DB는 이러한 포맷을 BSON이라 부름)을 선호함에 따라 전통적인 테이블 기반 관계형 데이터베이스 구조의 사용을 하지 않는다.

이로써 특정한 종류의 애플리케이션을 더 쉽고 더 빠르게 데이터 통합을 가능케 한다.

아페로 GPL과 아파치 라이선스를 결합하여 공개된 몽고DB는 자유-오픈 소스 소프트웨어이다.

03

MongoDB 설치

<https://www.mongodb.com/try/download/enterprise>

링크를 통해 인스톨 파일 다운로드



03

MongoDB 설치

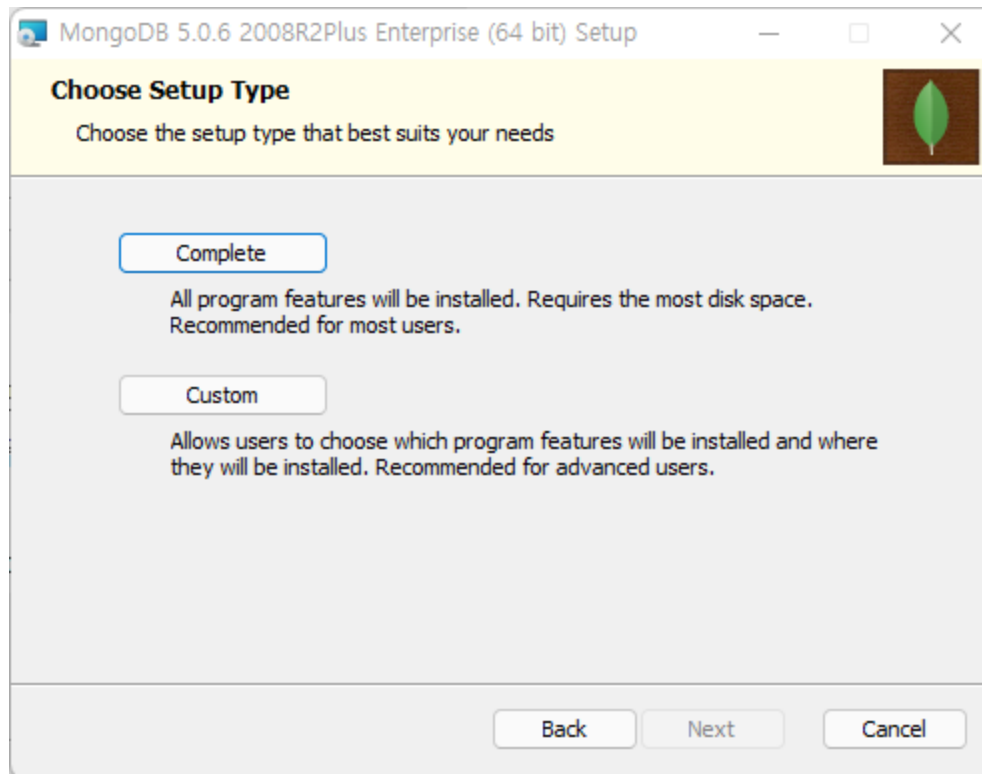
I accept the terms in the License Agreement 체크 후 Next



03

MongoDB 설치

Complete 버튼 클릭



03

MongoDB 설치

설치 위치 설정 Next

MongoDB 5.0.6 2008R2Plus Enterprise (64 bit) Service Cus...

Service Configuration

Specify optional settings to configure MongoDB as a service.

☒ Install MongoDB as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

Service Name:

Data Directory:

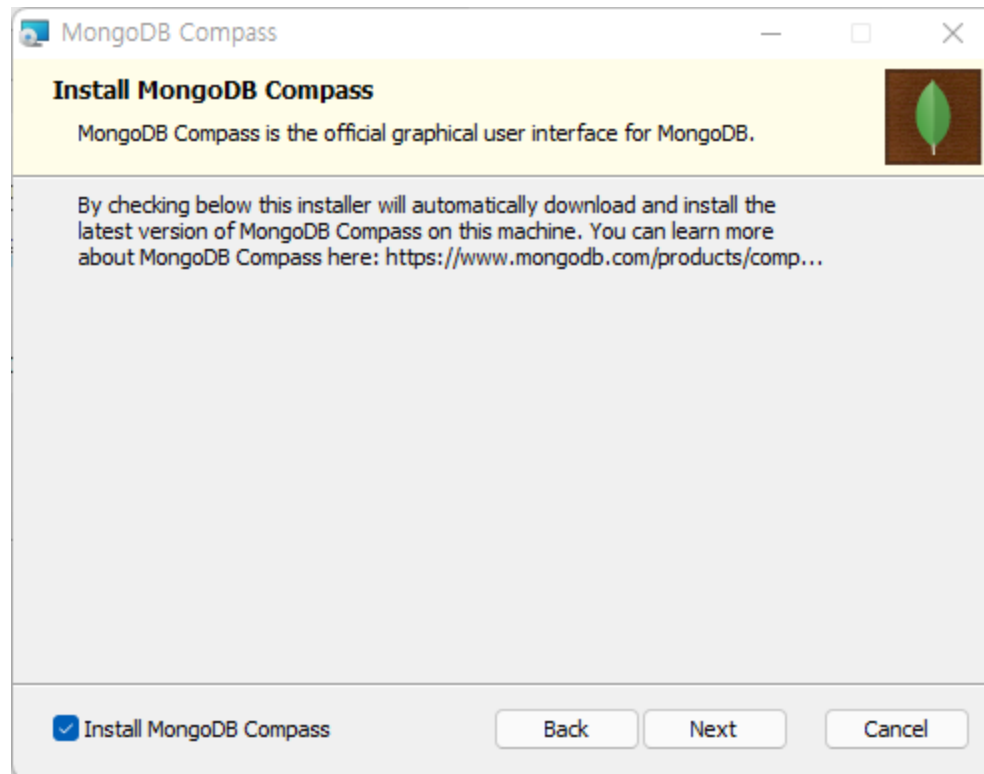
Log Directory:

< Back Next > Cancel

03

MongoDB 설치

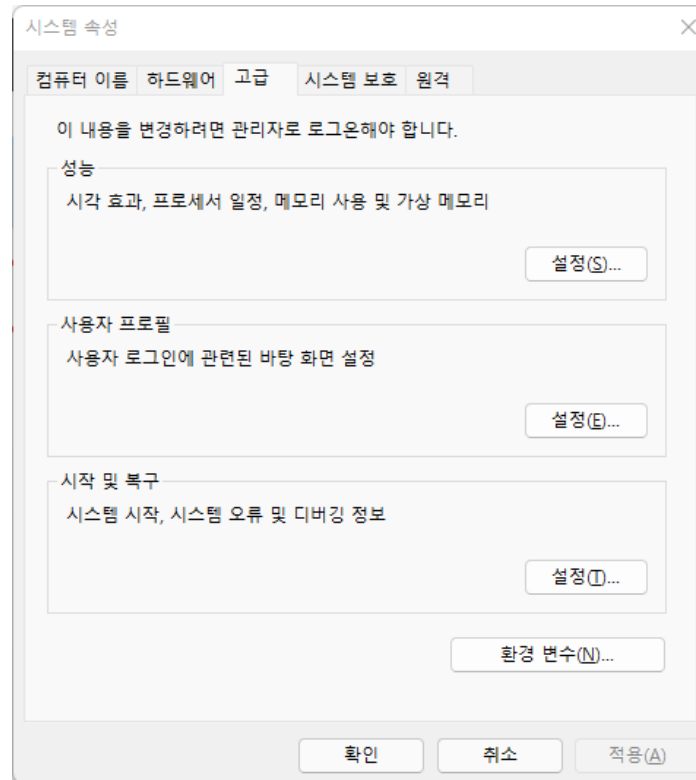
Install MongoDB Compass 체크가 되어 있지 않다면 체크 후 Next



03

MongoDB 환경 변수 편집

window키 + S키를 누른 후 시스템 환경 변수 실행
환경 변수 클릭



03

MongoDB 환경 변수 편집

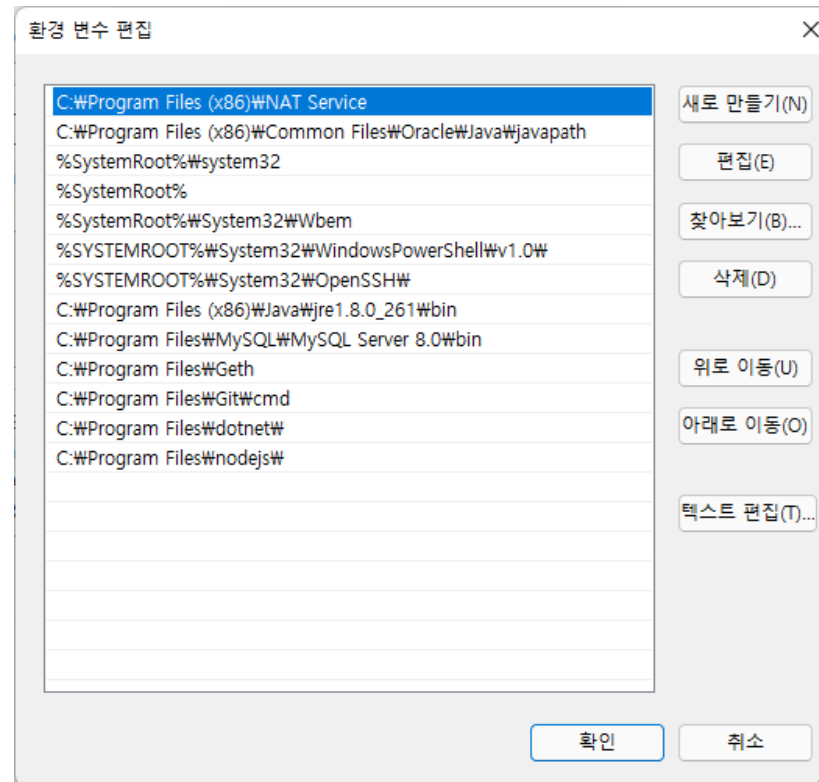
시스템 변수에서 Path 더블 클릭



03

MongoDB 환경 변수 편집

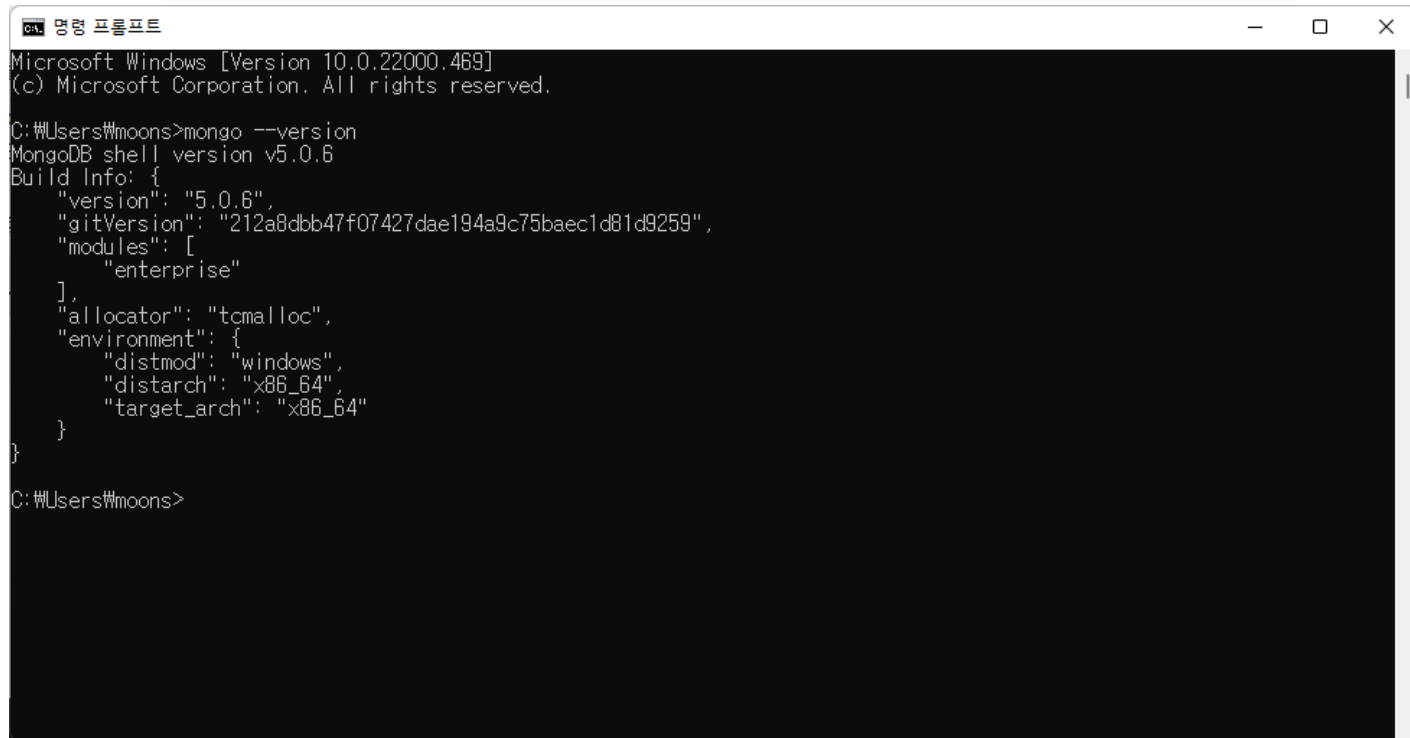
하단 빈 칸에 MongoDB를 설치한 디렉토리(C:\Program Files\MongoDB\Server\5.0\bin)를 추가한 후 확인



03

MongoDB 환경 변수 편집

명령 프롬프트 실행 후 `mongo --version` 를 입력하여 환경 변수 확인



```
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\moons>mongo --version
MongoDB shell version v5.0.6
Build Info: {
  "version": "5.0.6",
  "gitVersion": "212a8dbb47f07427dae194a9c75baec1d81d9259",
  "modules": [
    "enterprise"
  ],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

03

데이터 구조

DataBase : Collection의 물리적 컨테이너이다.

Collection : MySQL의 table과 같다고 생각하면 된다. Document의 그룹이다.

Document : 한 개 이상의 Key-value 쌍으로 이루어진 구조이다.

Key/Field : MySQL에서 Column과 같다고 생각하면 된다.

03

기본 명령어

show dbs : 데이터베이스 리스트 출력

db : 현재 사용중인 데이터베이스 이름 출력

db.stats() : 현재 사용중인 데이터베이스의 정보 출력

use 데이터베이스명 : 사용하려는 데이터베이스를 바꿔준다.
(데이터베이스가 존재하지 않는다면
데이터베이스를 생성한다.)

db.dropDatabase() : 데이터베이스를 삭제

03

기본 명령어

```
db.createCollection("Collection명", {capped:true, size:6142800,  
max:10000})
```

capped : bool타입, true로 설정 시 활성화 되며 사이즈를 초과하게 되면 가장 오래된 데이터를 덮어쓴다.

size : capped가 true인 경우 필수로 설정. 해당 Collection의 최대 사이즈

max : 해당 Collection에 추가할 수 있는 최대 document 개수

03

CRUD

```
db.(Collection명).insert({ key : value }, single
```

key/field 값을 입력한다.

Collection이 존재하지 않는다면 Collection을 자동으로 생성한다.

```
db.(Collection명).insert([ { key1 : value1 }, { key2 : value2 } ])
```

key/field 값을 대괄호로 묶어서 멀티로 입력이 가능하다.

03

CRUD

```
db.(Collection명).drop()
```

해당하는 Collection을 제거한다.
제거 전에 데이터베이스를 설정해야 된다.

```
db.(Collection명).find()
```

Document 리스트를 확인한다.

```
db.(Collection명).remove({key1 : value1})
```

Document에서 해당하는 리스트를 삭제한다.

**THANK
YOU**