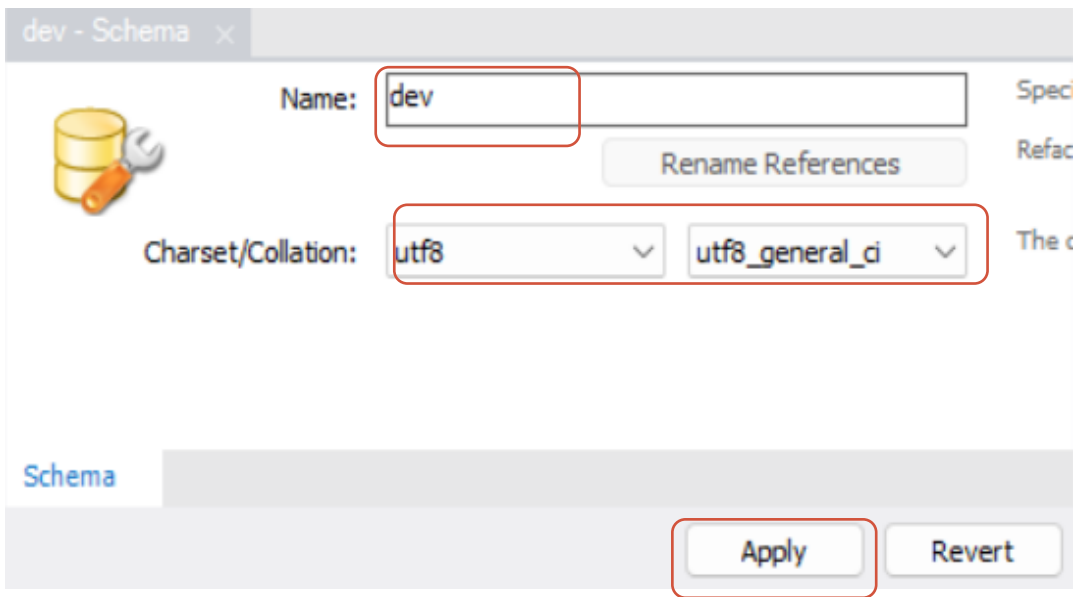


- `mysql`

- **관계형 데이터베이스 관리 시스템(RDBMS)**
- MySQL
  - **GUI** 관리툴은 내장되어 있지 않음. 별도로 설치
  - CLI 명령으로 데이터베이스를 관리하고, 데이터를 백업, 상태를 검사, 데이터베이스 구조를 생성, 또는 데이터 레코드를 작성하는 명령어를 이용
- **MySQL 워크벤치**
  - 공식적인 MySQL 프론트엔드 툴로서 오라클에 의해 개발되었으며, 자유롭게 사용할 수 있다



```
CREATE SCHEMA `dev` DEFAULT CHARACTER SET utf8 ;
```

customer - Table x



Table Name: customer

Schema: dev

Charset/Collation:

Default Charset

Default Collation

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
phone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
address	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```
CREATE TABLE `dev`.`customer` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `name` VARCHAR(45) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  `phone` VARCHAR(45) NOT NULL,  
  `address` VARCHAR(100) NULL,  
  PRIMARY KEY (`id`));
```

- 데이터 입력

```
insert into dev.customer(id, name, email, phone, address)
values(1, '홍길동', 'hong@gmail.com', '010-111-2222', '');
```

```
insert into dev.customer
set id=1,
    name='홍길동',
    email='hong@gmail.com',
    phone= '010-111-2222'
```

- 데이터 수정

```
update dev.customer
set name='홍길동',
    email='hong@gmail.com',
    phone= '010-111-2222'
where id = 1
```

- 데이터 삭제

```
delete from dev.customer where id = 1;
```

- 데이터 조회

```
select * from dev.customer
```

```
select * from dev.customer where id = 1
```

```
select * from dev.customer LIMIT 10;           // limit
```

```
select * from dev.customer LIMIT 10 offset 0    // offset, limit
```

```
select * from dev.customer LIMIT 0, 10         // offset, limit
```

Offset 값은 0부터 시작하므로 첫 번째 행 데이터를 가리키는 값은 0 이다.

- mysql 모듈 설치

```
> npm install mysql
```

- mysql 연결

- 사용자계정 생성

```
create user 'dev01'@'%' identified with mysql_native_password by '1234';
```

- 권한 부여

```
grant all privileges on dev.* to 'dev01'@'%' with grant option;  
flush privileges;
```

- 연결정보

```
// mysql 모듈 로드
const mysql = require("mysql");

// mysql 접속 정보
const conn = { host: "192.168.0.1",
               port: "3306",
               user: "dev01",
               password: "1234",
               database: "dev",
};

// DB 커넥션 생성
let connection = mysql.createConnection(conn);
```

- 연결문자열

```
let connection = mysql.createConnection("mysql://dev01:1234@localhost:3307/dev");
```



```
// 1. DB 접속 체크 (생략가능)
connection.connect((err) => {
  if (err) {
    console.log("error connection" + err.stack);
    return;
  }
});

// 2. SQL 실행
sql = "SELECT * FROM customer";
connection.query(sql, function (err, results, fields) {
  if (err) {
    console.log(err);
  }

  // 3. 결과 처리
  console.log(results);
});

// DB 접속 종료(비동기이지만 SQL이 모두 실행되면 종료)
connection.end();
```

- DataBase Connectopn Pool

- 데이터베이스에 연결된 Connection을 미리 만들어 둔후 Pool에 보관하였다가 필요할 때 Pool에서 Connection을 가져다 사용한 후, 다시 Pool에 반환하는 기법
- 새로운 connection을 구축할 때 생기는 오버헤드를 모두 피할 수 있음

```
const mysql = require("mysql");

// mysql 접속 정보
const conn = {
  host: "192.168.0.1",
  port: "3306",
  user: "dev01",
  password: "1234",
  database: "dev",
  connectionLimit: 10,
};

let pool = mysql.createPool(conn);
```

```
let pool = mysql.createPool(conn);

// 1. DB 접속(콜백함수)
pool.getConnection((err, connetion) => {

// 2. SQL 실행
  sql = "SELECT * FROM customer";
  connetion.query(sql, (err, results, fields) => {
    if (err) {
      console.log(err);
    }

    // 3. 결과 처리
    console.log(results);

    // 4. DB 접속 종료
    connetion.release();
  });
// pool.releaseConnection(conn); connetion을 직접 release 하거나 pool의 release를 사용
});
```

```
let pool = mysql.createPool(conn);

//1. 풀 connection은 생략 가능

// 2. SQL 실행
sql = "SELECT * FROM customer";
pool.query(sql, function (err, results, fields) {
  if (err) {
    console.log(err);
  }

  // 3. 결과 처리
  console.log(results);

  //4. 쿼리가 수행되면 connection은 자동으로 해제된다.
});
```

```
const mysql = require("mysql");

// mysql 접속 정보
const conn = {
  host: "localhost",
  ...
  connectionLimit: 10,
};

// DB 커넥션 생성
let pool = mysql.createPool(conn);

module.exports = pool;
```

```
const mysql = require("./mysql_pool");

sql = "SELECT * FROM customer";
mysql.pool.query(sql,
  function (err, results, fields) {
    if (err) {
      console.log(err);
    }
    console.log(results);
  });
```

