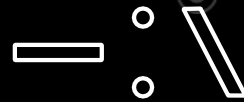


SHOUT OUR PASSION TOGETHER



PASSION

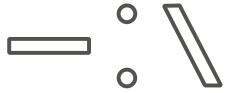
4th Seminar

EXPERIENCE

GROWTH

CHALLENGE





SHOUT OUR PASSION TOGETHER

조원

칠판

1조

2조

5조

3조

4조

6조

01 강영우 김민준 김해리 양승희 최영훈 최소영 황채연

02 김정환 남궁권 박시현 이소희 이재현 양희연 현주희

03 양시연 양희찬 유희수 이동훈 이시연 이재용

04 박경선 박승완 신윤재 심다은 조수민 지현이

05 박주연 손예지 양정훈 윤혁 이다현 이소연 허정민

06 김강희 김채린 박형모 신정아 조하담 제갈윤

01 Database

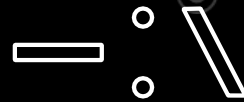
02 Nodejs연동

03 Database 실습

04 Database 실습

05 EC2에 올리기

SHOUT OUR PASSION TOGETHER



01

Database

Database

여러 사람에 의해 공유되어 사용될 목적으로 통합하여 관리되는 데이터의 집합

DBMS

데이터베이스를 관리하는 시스템

Database 정의

- 공유 데이터(Shared Data)
여러 응용 시스템들이 공동으로 소유하고 유지하는 자료
- 운영 데이터(Operational Data)
조직의 고유한 업무를 수행하는 데 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료
- 통합 데이터(Integrated Data)
자료의 중복을 배제한 데이터의 모임
- 저장 데이터(Stored Data)
컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료

Database 특징

- 데이터 중복의 최소화(Redundancy Minimize)
 - 지속적인 변화(Continuous Evolution)
데이터베이스의 상태는 동적이다. 즉 새로운 데이터의 삽입(Insert), 삭제(Delete), 갱신(Update)로 항상 최신의 데이터를 유지한다
 - 실시간 접근(Real Time Accessibility)
수시적이고 비정형적인 질의(조회)에 대하여 실시간 처리에 의한 응답이 가능해야 한다.
 - 동시 공유(Concurrent Sharing)
데이터베이스는 서로 다른 목적을 가진 여러 응용자들을 위한 것이므로 다수의 사용자가 동시에 같은 내용의 데이터를 이용할 수 있어야 한다.
- 내용에 의한 접근(Content Reference)
데이터베이스에 있는 데이터를 참조할 때 데이터 레코드의 주소나 위치에 의해서가 아니라, 사용자가 요구하는 데이터 내용으로 데이터를 찾는다.
- 뛰어난 자료간 연계성(Superb Data Combination)

SQL

(관계형 데이터베이스)

SQL은 구조화 된 쿼리 언어 (Structured Query Language)를 의미

1. 엄격한 스키마

스키마를 준수하지 않는 레코드는 추가할 수 없다.

2. 관계

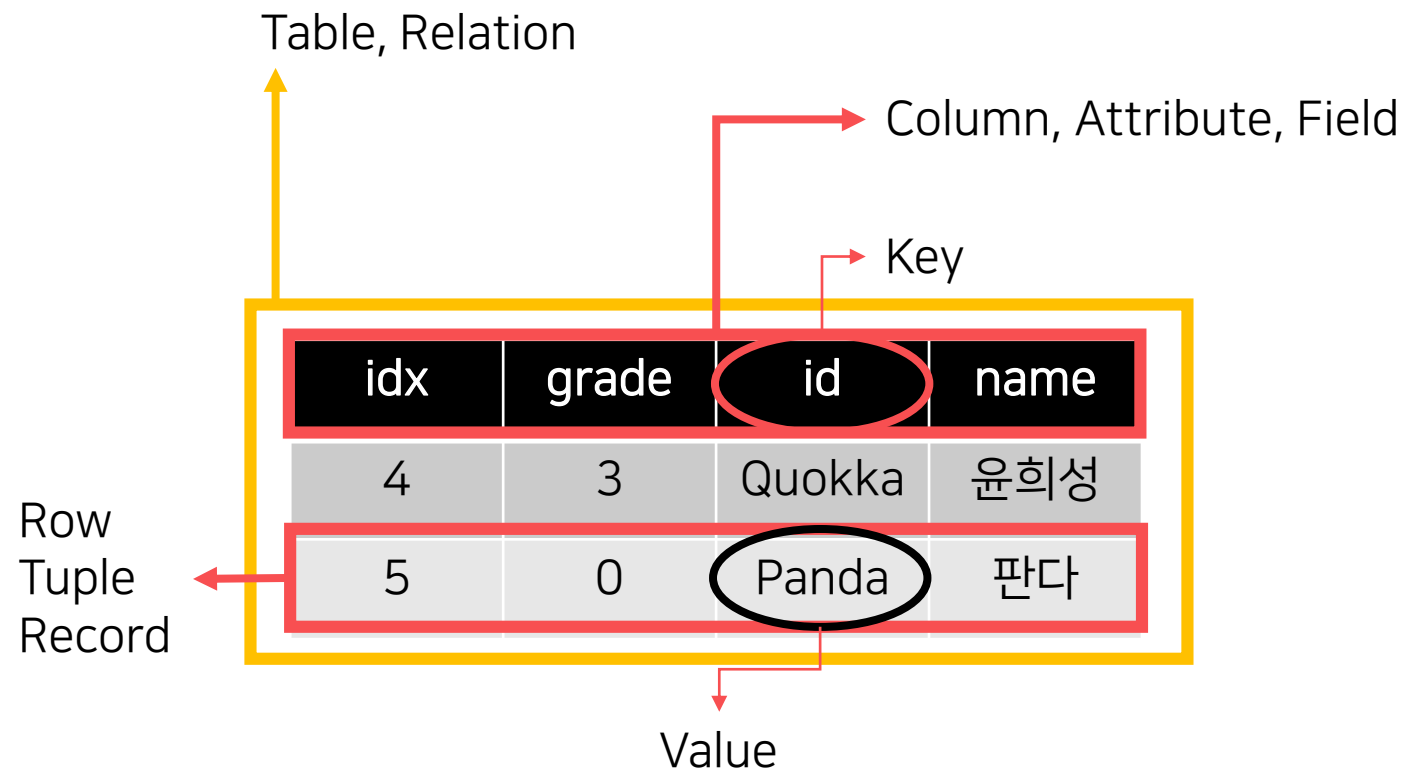
관계를 통해서 연결된 여러 개의 테이블에 분산

NoSQL

(비관계형 데이터베이스)

1. 스키마 없음

2. 관계 없음



1:1관계

한 명의 학생은
하나의 학번만 가질 수 있다.

하나의 글은
글쓴이가 한명이다.

1:N 관계

한 명의 학생은 하나의 파트에 속할 수 있지만,
파트는 여러 사람이 속할 수 있다

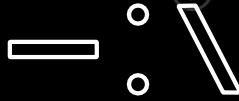
하나의 글은
댓글이 여러 개이다.

N:M 관계

한 명의 학생은 여러 학과를 선택할 수 있고,
하나의 학과는 여러 학생을 포함한다

한 명의 회원은 여러 글에 좋아요를 누를 수 있고,
하나의 글은 여러 명의 좋아요를 할 수 있다.

SHOUT OUR PASSION TOGETHER



03

Nodejs연동

promise-mysql

```
const mysql = require('promise-mysql')

const dbConfig = {
  host: database 주소,
  port: 포트번호,
  user: 접속 user id,
  password: 비밀번호,
  database: database 이름,
  dateStrings: 'date',
}

module.exports = mysql.createPool(dbConfig)
```

- 데이터 베이스에 접속하기 위한 설정값 Db접속을 위한 정보를 저장한다.

이는 보안을 유지해야하는 정보이다.

- promise-mysql에 의존
Nodejs에 mysql를 연동해주는 모듈

createPool method

```
export function createPool(config: mysql.PoolConfig | string): Bluebird<Pool>;
```

반환 타입이 Bluebird(Promise A+ 객체)이다.

Pool.js

```
const poolPromise = require('../config/dbConfig')
module.exports = {
  queryParam_None: async (...args) => {
    const query = args[0]
    let result
    const pool = await poolPromise;
    try {
      var connection = await pool.getConnection()
      result = await connection.query(query) || null
    } catch (err) {
      console.log(err)
      connection.rollback(() => {})
    } finally {
      pool.releaseConnection(connection)
      return result
    }
  },
}
```

- mysql-promise에서 만들어진 pool(Promise)를 가져온다.
- pool 객체에서 connection 객체 가져옴
- query문 실행
- connection을 pool에 반납한다.

queryParam_parse

```
queryParam_Parse: async (inputquery, inputvalue) => {  
  const query = inputquery  
  const value = inputvalue  
  let result  
  try {  
    var connection = await pool.getConnection()  
    result = await connection.query(query, value) || null  
    console.log(result)  
  } catch (err) {  
    console.log(err)  
    connection.rollback(() => {})  
    next(err)  
  } finally {  
    pool.releaseConnection(connection)  
    return result  
  }  
}
```

- Query문을 query와 value로 구분
- Query 문자열 내에서 ?부분에 차례대로 value 배열의 값이 들어간다.
- Ex)
query = 'INSERT INTO table values(?,?,?)'
value = [1, 'title', 'content']

queryParam_Arr

```
queryParam_Arr: async (...args) => {  
  const query = args[0]  
  const value = args[1]  
  let result  
  try {  
    var connection = await pool.getConnection()  
    result = await connection.query(query, value) || null  
  } catch (err) {  
    connection.rollback(() => {})  
    next(err)  
  } finally {  
    pool.releaseConnection(connection)  
    return result  
  }  
}
```

- queryParam_Parse의 Array 버전

Transaction이란? 여러 SQL문을 하나로 묶어서 하나라도 모든 쿼리가 성공한 경우에만 작업이 진행되도록 하는 것

Transaction

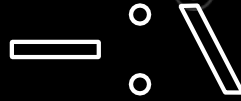
```
Transaction: async (...args) => {
  let result = "Success"
  try {
    var connection = await pool.getConnection()
    await connection.beginTransaction()
    await args[0](connection, ...args)
    await connection.commit()
  } catch (err) {
    await connection.rollback()
    console.log("mysql error! err log =>" + err)
    result = undefined
  } finally {
    pool.releaseConnection(connection)
    return result
  }
}
```

- beginTransaction로 트랜잭션 시작
- rollback: 이전 처리 무효화
- commit: 결과 반영

사용 예시

```
const insertTransaction = await db.Transaction(async(connection) => {
  const r1 = await connection.query(query1, [name, gender, part, univ]);
  const userIdx = r1.insertId;
  const r2 = await connection.query(query2, ["content", userIdx]);
});
```


SHOUT OUR PASSION TOGETHER



03

Database 실습



SHOUT OUR PASSION TOGETHER

Database 실습: 테이블 생성

1. 테이블 생성

Table Name: Schema:

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Express
userIdx	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
id	VARCHAR(50)	<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"/>	
name	VARCHAR(50)	<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"/>	
password	VARCHAR(200)	<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"/>	
salt	VARCHAR(200)	<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(50)	<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(50)	<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"/>	

Column Name:

Charset/Collation:

Comments:

[Columns](#) [Indexes](#) [Foreign Keys](#) [Triggers](#) [Partitioning](#) [Options](#)

Table Name: Schema:

Charset/Collation: Engine:

Comments:

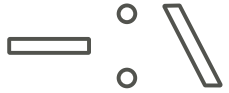
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Express
boardIdx	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
title	VARCHAR(50)	<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"/>	
content	VARCHAR(50)	<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"/>	
writerIdx	INT(11)	<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"/>	

Column Name:

Charset/Collation:

Comments:

[Columns](#) [Indexes](#) [Foreign Keys](#) [Triggers](#) [Partitioning](#) [Options](#)



SHOUT OUR PASSION TOGETHER

Database 실습

2. 외래키 지정

규칙 정의

1. **RESTRICT** : 개체를 변경/삭제할 때 다른 개체가 변경/삭제할 개체를 참조하고 있을 경우 변경/삭제가 취소 (제한)

2. **CASCADE** : 개체를 변경/삭제할 때 다른 개체가 변경/삭제할 개체를 참조하고 있을 경우 함께 변경/삭제

3. **NO ACTION** : 개체를 변경/삭제할 때 다른 개체가 변경/삭제할 개체를 참조하고 있을 경우 변경/삭제할 개체만 변경/삭제되고 참조하고 있는 개체는 변동이 없음

4. **SET NULL** : 개체를 변경/삭제할 때 다른 개체가 변경/삭제할 개체를 참조하고 있을 경우 참조하고 있는 값은 NULL로 세팅

<https://h5bak.tistory.com/125>

Table Name: board Schema: with-sopt-server

Charset/Collation: utf8 utf8_unicode_ci Engine: InnoDB

Comments:

Foreign Key Name	Referenced Table
FOREIGN_KEY_USER_ID	with-sopt-server`.`user`

외래키 이름 참조 테이블

Column	Referenced Column
<input type="checkbox"/> boardIdx	
<input type="checkbox"/> title	
<input type="checkbox"/> content	
<input checked="" type="checkbox"/> writerIdx	userId

참조 칼럼

Foreign Key Options

On Update: RESTRICT

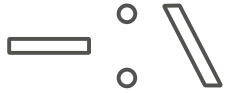
On Delete: RESTRICT

☐ Skip in SQL generation

Foreign Key Comment

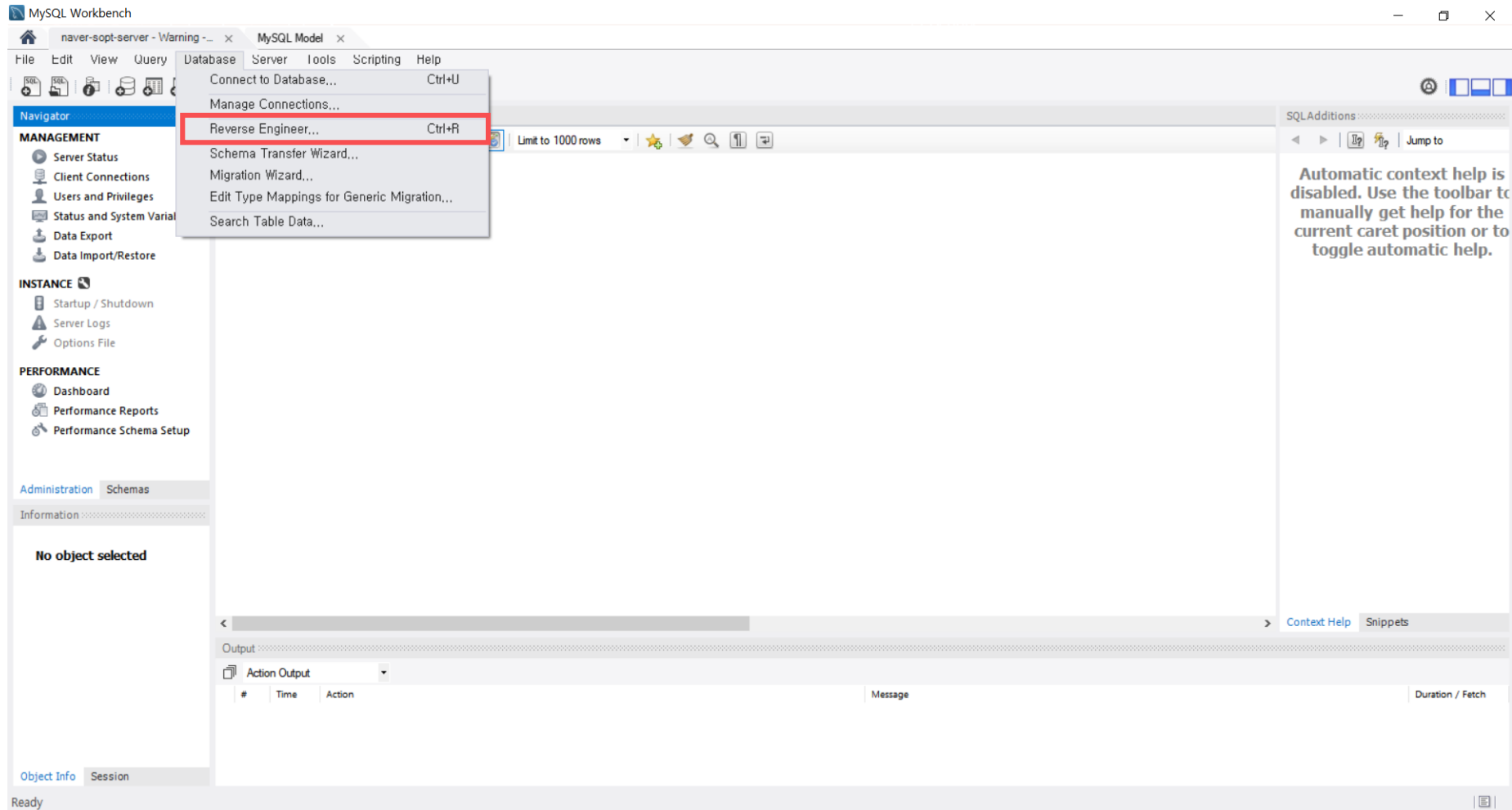
Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert



SHOUT OUR PASSION TOGETHER

Database 실습: ER diagram 보기





SHOUT OUR PASSION TOGETHER

Database 실습: ER diagram 보기

Reverse Engineer Database

Connection Options

- Connect to DBMS
- Select Schemas
- Retrieve Objects
- Select Objects
- Reverse Engineer
- Results

Set Parameters for Connecting to a DBMS

대상 서버 선택

Stored Connection: local Select from saved connection settings

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

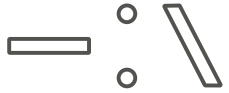
Parameters SSL Advanced

Hostname: 127.0.0.1 Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: root Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Back Next Cancel



SHOUT OUR PASSION TOGETHER

Database 실습: ER diagram 보기

Reverse Engineer Database

Connection Options

Connect to DBMS

Select Schemas


Retrieve Objects

Select Objects

Reverse Engineer

Results

Select Schemas to Reverse Engineer

 Select the schemas you want to include:

☐ sopt-server

☐ test

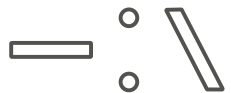
☐ with-sopt-server

Database 선택

Back

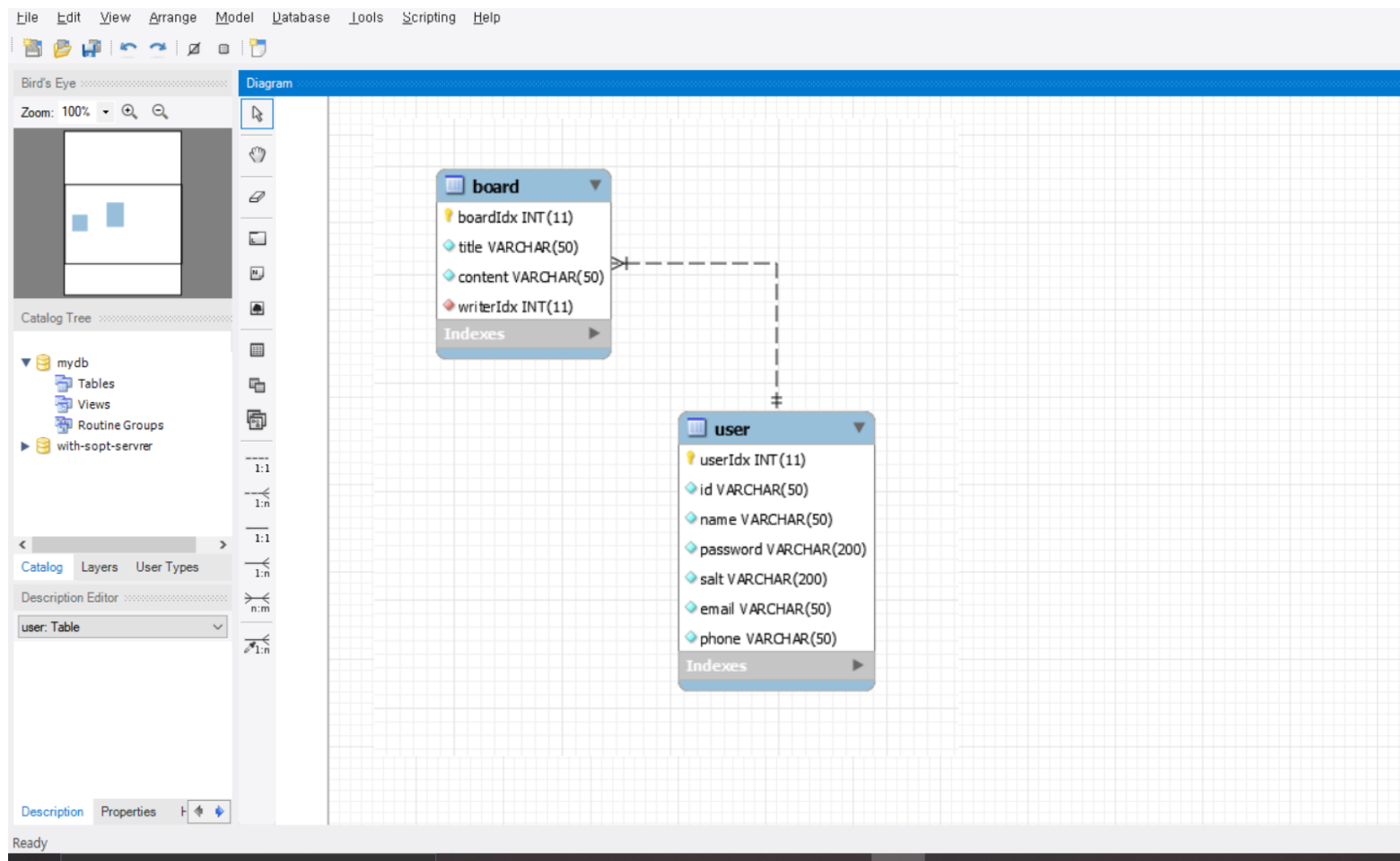
Next

Cancel



SHOUT OUR PASSION TOGETHER

Database 실습: ER diagram 보기



Pool.js 수정

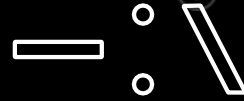
PoolAsync.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/module/poolAsync.js>

PoolSync.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/module/poolSync.js>

SHOUT OUR PASSION TOGETHER



04

패스워드 암호화

암호 저장하는 방법

Request (signup)

```
{  
  "id": "아이디"  
  "pwd": "1234"  
  ...  
}
```

Salt 생성

암호화

User Table

useridx	Password	salt
...	해시값	해시할 때 사용한 salt값

암호 확인하는 방법

Request (signin)

```
{  
  "id": "아이디"  
  "pwd": "1234"  
  ...  
}
```

Salt 추가

암호화

비교

성공 or 실패

User Table

userIdx	Password	salt
...	해시값	해시할 때 사용한 salt값

Model/user.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/model/user.js>

Model/board.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/model/board.js>

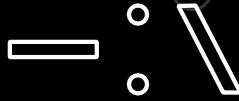
routes/boards.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/routes/boards.js>

routes/users.js:

<https://github.com/WITH-SOPT-SERVER/SOPT-SERVER-SEMINAR/blob/develop/seminar4/practice4/routes/users.js>

SHOUT OUR PASSION TOGETHER



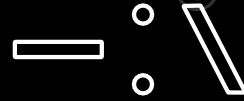
05

EC2에 올리기

EC2에 업로드 실습

1. Git clone repository
2. Config 파일 복사 붙여넣기
3. npm install
4. npm start

SHOUT OUR PASSION TOGETHER



05

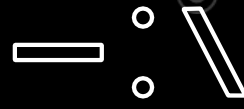
과제안내

3주차 과제에서 아래 두가지 기능 및 라우팅을 추가해주세요.

- blogIdx에 해당하는 Article 보기
- ArticleIdx에 해당하는 Comment 보기

그리고 이를 ec2 서버에 올려주세요.

SHOUT OUR PASSION TOGETHER



PASSION

Thank You :)

EXPERIENCE

GROWTH

CHALLENGE

