SKU x LIKELION

Official Team
Official **LIKELION** at **SKU**

RDB 설계 및 실습

DB 설계

데이터베이스 설계(모델링)이란 ?

현실에 있는 정보를 데이터화 하는 것

시스템 내에서 어떻게 표현할지 결정

-> 컴퓨터에게 알려주기 위함

데이터베이스 설계(모델링)의 순서

1. 요구사항 분석

2. 개념적 데이터 모델링

3. 논리적 데이터 모델링

4. 물리적 데이터 모델링
   = 구현

요구사항 파악

: 요구사항 이해

만들고자 하는 것 → 게시글과 댓글을 쓸 수 있는 블로그

↓

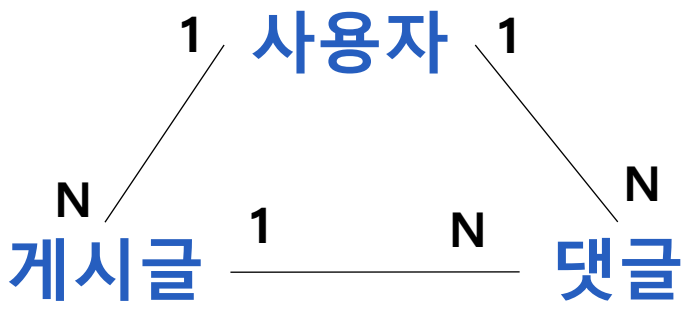
= 요구사항

개념적 데이터 모델링

: 핵심 개체(Entity) 도출 + ERD 작성

1. 게시글
2. 댓글
3. 사용자

Entity- **R**elationship **D**iagram
: 엔티티들 간의 관계를 나타낸 다이어그램



논리적 데이터 모델링

:ERD 상세 속성 정의

이전 단계에서 만들었던 ERD를 기반으로 상세 속성(필드)을 정의하는 단계

Member(사용자)		
int	id	PK
String	name	
String	password	

1

N

Comment(댓글)		
int	id	PK
String	Content	
int	Article_id	FK
int	Member_id	FK

1

N

Article(게시글)		
int	id	PK
String	Title	
String	Content	
DateTime	CreateDate	
DateTime	UpdatedDate	
String	Writer	FK

물리적 데이터 모델링

: 데이터베이스의 상세한 설정

실제 데이터 베이스에 데이터가
어떻게 저장될지 결정하는 단계

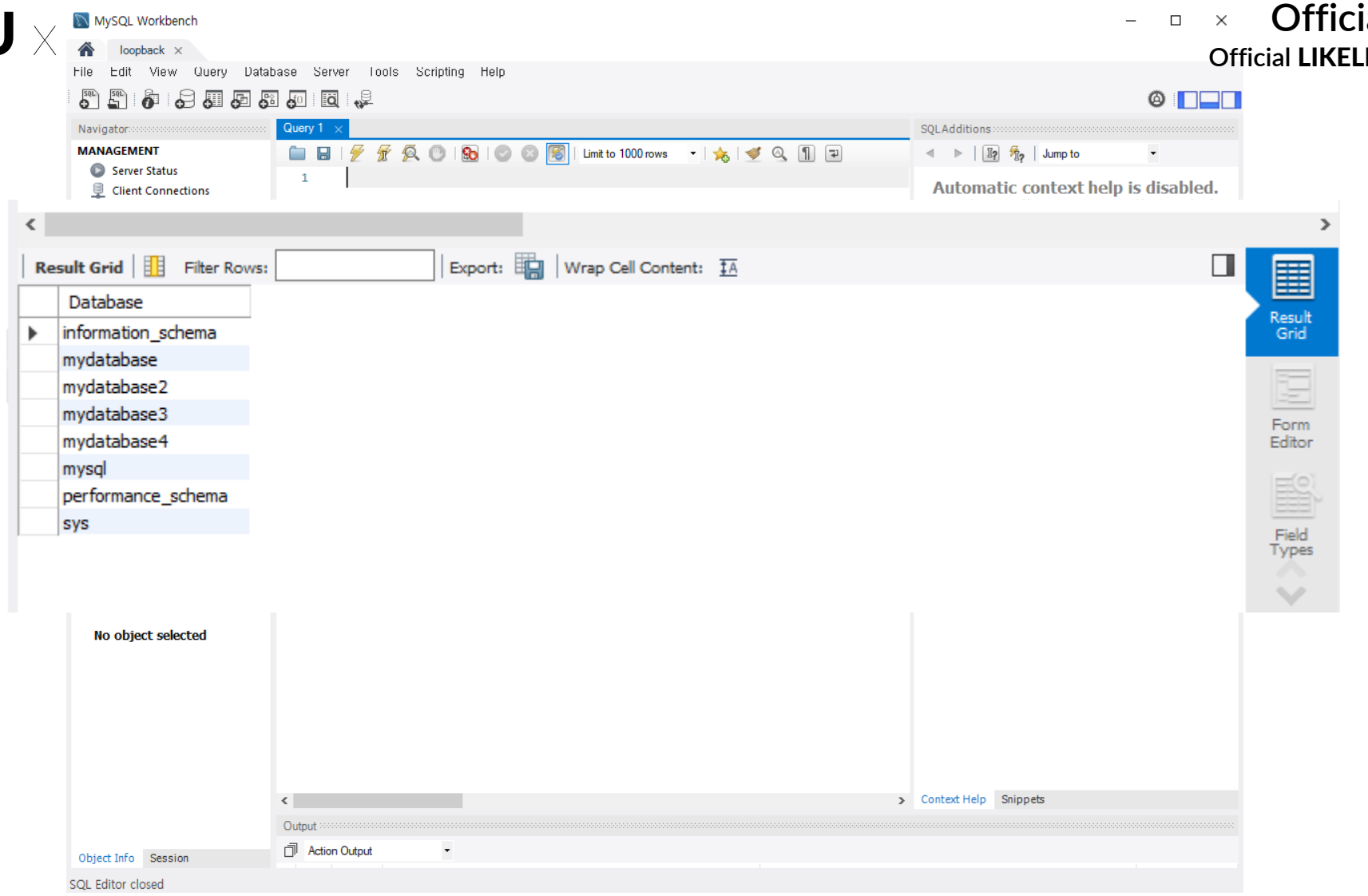
MySql을 이용한 구현

간단한 Sql문으로 설계한 DB 구현 및 확인

SKU x

Official Team

Official **LIKELION** at **SKU**



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Database
information_schema
mydatabase
mysql
performance_schema
sys

Result Grid
Form Editor
Field Types

29

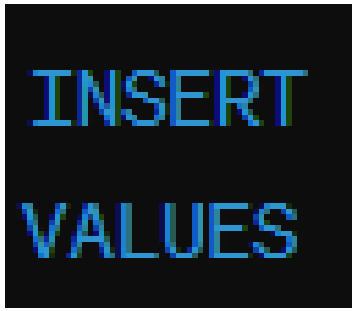
```
USE mydatabase;
```

```sql
-- comment 테이블 생성
CREATE TABLE comment (
    id INT AUTO_INCREMENT PRIMARY KEY,
    content TEXT NOT NULL,
    article_id INT,
    member_id INT,
    FOREIGN KEY (article_id) REFERENCES article(id) ON DELETE CASCADE,
    FOREIGN KEY (member_id) REFERENCES member(id) ON DELETE CASCADE
);
```

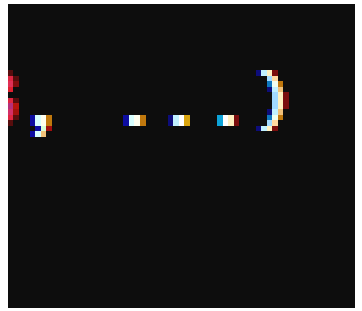
```
SHOW COLUMNS FROM member;
SHOW COLUMNS FROM article;
SHOW COLUMNS FROM comment;

DESC member;
DESC article;
DESC comment;
```

데이터 삽입




```
INSERT INTO member(name)
VALUES ("likelion"),
       ("hi"),
       ("bye");
```



데이터 조회

```
SELECT * FROM 테이블명 WHERE 조건;
```

```
SELECT * FROM member WHERE name = "hi";
```

```
SET SQL_SAFE_UPDATES = 0;
```

```
DELETE FROM 테이블명 WHERE 조건;
```


```
TRUNCATE TABLE 테이블명;
```

실습 (p.14 ~ p.16)

1. **Member** 테이블에 **5**개의 행 삽입
2. **Member** 전체 행 조회
3. **Member** 테이블 중 **id**가 **3**인 행 조회
4. **Member** 테이블 중 **id**가 **4**인 행 삭제

1.
```
INSERT INTO member(name)
VALUES ("likelion"),
       ("hi"),
       ("bye"),
       ("hojoo"),
       ("sku");
```

2.
```
select * from member;
```

3.
```
select * from member WHERE id = 3;
```

4.
```
delete from member where id = 4;
```

테이블 결합(join)

두 개 이상의 테이블에서 관련된 데이터를 결합

두 테이블 간 공통으로 존재하는 칼럼
(= 외래 키와 기본 키)
기준으로 연결

테이블 결합(join)

1. INNER JOIN
2. LEFT(RIGHT) JOIN

INNER JOIN

가장 많이 사용되는 join으로
두 테이블 간 교집합을 반환

-> 양쪽 테이블에 모두 일치하는 데이터만 선택

INNER JOIN

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

LEFT(RIGHT) JOIN

왼쪽(오른쪽) 테이블의 모든 데이터와
오른쪽(왼쪽) 테이블에서
일치하는 데이터 반환

오른쪽(왼쪽) 테이블에 일치하는 데이터가 없으면
NULL값으로 대체

LEFT(RIGHT) JOIN

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.common_column = table2.common_column;
```

실습 (p.18 ~ p.24)

1. inner join
2. Left join
3. Right join

실습파일을 받아서
데이터 세팅 후 진행 !

1.
```
SELECT *
FROM member
INNER JOIN article
ON member.id = article.writer_id;
```

2.
```
SELECT *
FROM member
left JOIN article
ON member.id = article.writer_id;
```

3.
```
SELECT *
FROM member
right JOIN article
ON member.id = article.writer_id;
```

SKU x LIKELION

Official Team
Official LIKELION at SKU

RDB 설계 및 실습

데이터베이스 설계

DBMS로 데이터 삽입, 조회, 삭제, 결합