FLASK WEB STUDY - SESSION 01

웹 개발의 시작
데이터베이스(DB) & Security

데이터 관리부터 SQL Injection 실습까지

Presented by
APS_김영빈

Today's Agenda (120min)

PART 1. DB 이론 심화 (40분)

- DB vs File System
- SQL 4대장 상세 (DDL, DML, DCL, TCL)
- 데이터 타입과 제약조건

PART 2. 스키마 설계 트레이닝 (40분)

- 연습 1: 카페 주문 목록
- 연습 2: 게임 캐릭터 정보
- 연습 3: 회원 정보
- Main: 게시판 설계

PART 3. Python 연동 (20분)

- DB 연결 및 커서
- 데이터 저장 (`INSERT` & `commit`)
- 데이터 조회 (`SELECT`)

PART 4. [CTF] SQL Injection (20분)

- 로그인 우회 챌린지
- **Hint Only**

PART 1. 데이터베이스 기초 이론

왜 DB를 쓰는가?

- ✓ **데이터 무결성:** 중복 없고 정확한 데이터
- ✓ **동시성 제어:** 여러 명이 동시에 써도 OK
- ✓ **보안:** 사용자별 권한 관리
- ✓ **표준 언어(SQL):** 체계적인 관리

SQL 명령어 4대장

DDL: CREATE, DROP (구조 정의)

DML: SELECT, INSERT (데이터 조작)

DCL: GRANT, REVOKE (권한 제어)

TCL: COMMIT, ROLLBACK (트랜잭션)

SQL 명령어의 4가지 축 (상세)

1. DDL (Data Definition Language)

데이터베이스의 **구조(뼈대)**를 정의합니다.

- **CREATE:** 테이블, 뷰 등을 생성
- **ALTER:** 기존 구조(컬럼 추가 등) 변경
- **DROP:** 테이블 자체를 삭제 (데이터 포함)
- **TRUNCATE:** 구조는 남기고 데이터만 싹 비움

2. DML (Data Manipulation Language)

실제 **데이터(내용물)**를 다룹니다.

- **SELECT:** 데이터 조회 (가장 중요 ★)
- **INSERT:** 데이터 삽입
- **UPDATE:** 데이터 수정
- **DELETE:** 데이터 삭제

3. DCL (Data Control Language)

데이터 접근 **권한(보안)**을 관리합니다.

- **GRANT:** 특정 유저에게 권한 부여
- **REVOKE:** 권한 회수

4. TCL (Transaction Control Language)

작업의 **완료 여부**를 결정합니다.

- **COMMIT:** 작업 결과를 DB에 영구 반영
- **ROLLBACK:** 작업 취소 (이전 상태로 복구)
- **SAVEPOINT:** 저장 지점 설정

데이터의 종류 (Data Types)

SQLite는 5가지 핵심 저장 클래스를 사용합니다.

INTEGER

정수

`1, 42, -100`

사용: 개수, ID, 레벨

TEXT

문자열

`'Hello', '홍길동'`

사용: 아이디, 닉네임, 본문

REAL

실수 (부동소수점)

`3.14, 99.9`

사용: 학점, 온도, 확률

BLOB

Binary Large Object

이진 데이터 덩어리

사용: 이미지 파일, 동영상, 암호화된 데이터 자체를 저장할 때

NULL

값이 '없음(Unknown)'을 의미

※ 0이나 공백('')과는 다릅니다. "아직 모른다"는 뜻.

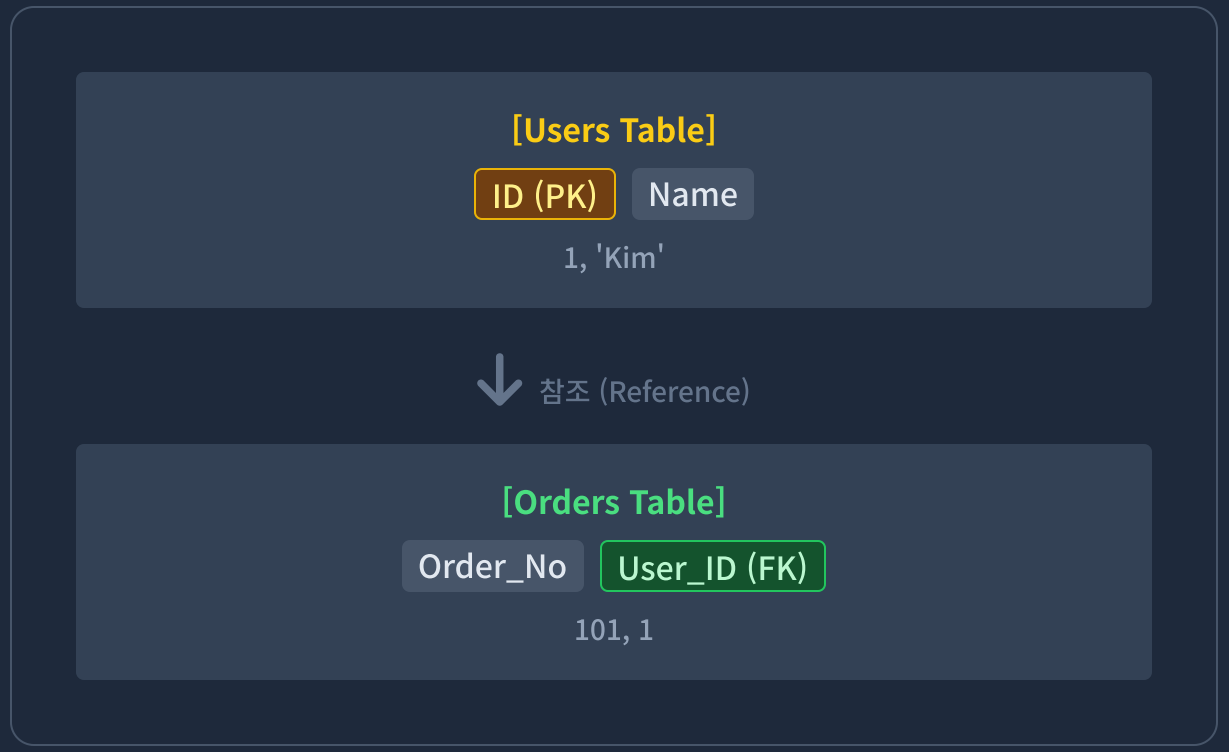
테이블의 연결고리: Key

Primary Key (기본키)

- 주민등록번호처럼 **나를 식별하는 유일한 값**
- 테이블당 1개만 존재
- 중복 불가, 빈 값 불가

Foreign Key (외래키)

- **다른 테이블의 PK를 가리키는 값**
- 데이터 간의 관계(Relationship)를 형성
- 부모 테이블에 없는 값은 넣을 수 없음 (참조 무결성)



Practice 1. 카페 주문 목록

설계 요구사항

카페 키오스크에 들어갈 메뉴판입니다.

- 메뉴마다 고유 번호 필요
- 메뉴 이름 (한글/영어)
- 가격 (숫자)
- 카테고리 (Coffee, Tea, Ade...)

* 테이블 컬럼: name, price, category

데이터 추가 SQL 작성하기

```sql
INSERT INTO menus (name, price, category)
VALUES ('아메리카노', 3000, 'Coffee');

-- 나머지 메뉴도 넣어보세요!
```

Practice 2. 게임 캐릭터 정보

설계 요구사항

RPG 게임의 캐릭터 저장소입니다.

- 캐릭터 ID (PK)
- 닉네임 (중복되면 안됨 - UNIQUE)
- 레벨 (기본값 1 - DEFAULT)
- 직업 (전사, 마법사...)

* 테이블 컬럼: nickname, job, level

데이터 추가 SQL 작성하기

```sql
INSERT INTO characters (nickname, job)
VALUES ('전설의용사', 'Warrior');

-- 레벨은 DEFAULT 1이라서 생략 가능!
```

Practice 3. 회원 정보

설계 요구사항

가장 기본적인 회원가입 DB입니다.

- 사용자 ID (PK)
- 로그인 아이디 (문자열, 필수, 중복불가)
- 비밀번호 (문자열, 필수)
- 가입일자

* 테이블 컬럼: user_id, password

데이터 추가 SQL 작성하기

```sql
INSERT INTO members (user_id, password)
VALUES ('user1', '1234');
```

MAIN MISSION: 게시판 완성하기

설계된 테이블에 데이터를 넣어 완성해봅시다.

설계 요구사항

id: 자동 증가하는 번호 (PK, AUTOINCREMENT)

title: 제목 (필수, NOT NULL)

content: 내용 (긴 글, TEXT)

user_id: 작성자 (숫자, FK 개념)

created_at: 작성일 (DEFAULT CURRENT_TIMESTAMP)

데이터 입력 테스트

```
-- 데이터 테스트 (INSERT)
INSERT INTO posts (title, content, user_id)
VALUES ('DB 완전 정복', '이제 스키마 짤 수 있어요', 1);

INSERT INTO posts (title, content, user_id)
VALUES ('C언어 점수', '다들 몇점이야?', 2);

SELECT * FROM posts;
```

PART 3. Python과 DB 연동 (1/3)

기본 연결 구조: Connect → Cursor

```python
import sqlite3

conn = sqlite3.connect('test.db')
cur = conn.cursor()

# DB 연결 성공!
print("DB connected")

conn.close()
```

1. Connect

파이썬 프로그램과 DB 파일 사이의 '전화선' 연결

2. Cursor

SQL 명령어를 배달하고 결과를 받아오는 '심부름꾼'

PART 3. 데이터 저장하기 (2/3)

INSERT 문을 실행하고 반드시 COMMIT을 해야 합니다.

```
try:
    # 다음 파트(해킹)를 위한 관리자 계정 생성
    sql = "INSERT INTO users (id, pw, name) VALUES ('admin',
'1234', '관리자')"
    cur.execute(sql)

    # 확정 도장 쾅! (필수)
    conn.commit()
    print("저장 성공!")

except Exception as e:
    conn.rollback() # 에러나면 취소
    print("에러 발생:", e)
```

주의: conn.commit()

INSERT, UPDATE, DELETE 후에는 반드시 커밋을 호출해야 DB 파일에 실제로 저장됩니다.
(안 하면 프로그램 종료 시 다 날아감!)

실습 포인트

Part 4에서 해킹할 **'admin'** 계정을 지금 만들어둡니다.

PART 3. 데이터 가져오기 (3/3)

데이터를 하나만 볼 것인가, 몽땅 가져올 것인가?

fetchone()

```
cur.execute("SELECT * FROM users WHERE id='admin'")
row = cur.fetchone()
print(row)
# (1, 'admin', ...)
```

- 딱 **한 줄**만 가져옵니다.
- 로그인 체크할 때 (ID는 하나니까) 주로 사용
- 결과는 **Tuple** 형태

fetchall()

```
cur.execute("SELECT * FROM users")
rows = cur.fetchall()
for row in rows:
    print(row)
# [(1, 'A'), (2, 'B')...]
```

- 조건에 맞는 **모든 줄**을 가져옵니다.
- 게시판 목록, 상품 목록 뿌릴 때 사용
- 결과는 **List of Tuples** 형태

PART 4. SQL Injection Challenge

Vulnerable Code (Python f-string)

```
user_id = input("ID: ")
user_pw = input("PW: ")

# f-string을 사용한 위험한 쿼리
query = f"SELECT * FROM users WHERE id = '{user_id}' AND pw =
'{user_pw}'"

cur.execute(query)
```

여러분의 목표는 **비밀번호 없이** 로그인하는 것입니다.
ID 입력창에 마법의 문자열을 넣어보세요.

MISSION START

쿼리의 뒷부분 `AND pw = '...'` 을 무력화시키세요.

Hints

- SQL에서 **주석(Comment)**처리는 어떻게 하나요?
- 문자열을 닫으려면 무엇이 필요한가요?
- 조건을 무조건 **참(True)**으로 만들려면?

* 검색을 통해 답을 찾아보세요!

Session 01 Clear!

"이제 여러분은 DB를 설계하고,
Python으로 조작하며,
기본적인 SQL 취약점까지 이해했습니다."

Next Session: Network, Server, Web

Any Questions?