Database

강의날짜	@2022/10/04
작성일시	@2022년 10월 4일 오후 1:34
편집일시	@2022년 10월 4일 오후 4:00
분야	database
공부유형	강의
복습	☐
태그	

Database

데이터베이스의 등장

- 파일을 이용한 데이터 관리
  - 우리는 일반적으로 데이터를 **파일**에 저장한다
  - 장점
    - 운영체제에 관계없이 어디에서나 쉽게 사용가능
    - 이메일이나 메신저를 이용해 간편하게 전송가능
  - 단점
    - 성능과 보안적 측면에서 한계가 명확
    - 대용량 데이터를 다루기에 적합하지 않음
    - 데이터를 구조적으로 정리하기에 어려움
    - 확장이 불가능한 구조
- **스프레드 시트**를 이용한 데이터 관리
  - 스프레드 시트(엑셀 시트)을 사용

    - 스프레드 시트는 컬럼(열)을 통해 데이터의 유형을 지정하고 레코드(행)을 통해 구체적인 데이터 값을 포함
    - 스프레드 시트 자체를 데이터베이스라고 부를 수는 없지만 데이터베이스로 가는 길목정도로 볼 수 있음
- 스프레드 시트와 달리 프로그래밍 언어를 사용해 작동시킬 수 있음
- 데이터베이스는 많은 형태가 있지만 실제 가장 많이 쓰이는 유형은 RDB(Relational Database)라고 부르는 **관계형 데이터베이스**
- RDB는 각각의 데이터를 테이블에 기입함 (마치 스프레드 시트에 작성하는 것처럼)
- 쉽게 생각하면 스프레드시트 파일 모음을 관계형 RDB라고 생각하자

Database 학습 목표

- 데이터베이스를 사용하면 데이터를 안전하고 편리하고 빠르게 보관하고 사용할 수 있음
- 거대하고 복잡한 데이터를 다루기 위해서 고안된 도구이기 때무에 매우 많은 기능을 제공
    - 기능이 많다 == 데이터 관련해서 할 수 있는 일들이 많다
    - 모든 기능을 학습하는 것은 불필요함 → 우선 기초적인 부분에 집중하자
- DB 학습의 기초
    - 데이터베이스에 데이터를 어떻게 **입력**하고, 어떻게 **출력**하는가
- 즉 데이터베이스에서의 CRUD와 여러 키워드들 위주로 학습할 예정

Database 정의

- 체계화된 데이터의 모임
- 여러 사람이 공유하고 사용할 목적으로 통합 관리되는 정보의 집합
- 검색, 구조화 같은 작업을 보다 쉽게 하기 위해 조직화된 데이터를 수집하는 저장 시스템
    - 내용을 고도로 구조화함으로써 검색과 갱신의 효율화를 꾀한 것
    - 즉, 자료 파일을 조직적으로 통합하여 자료 항목의 중복을 없애고 구조화하여 기억시켜 놓은 자료의 집합체
- 이러한 Database를 조작하는 프로그램 = DBMS(Database Magnagment System)
    - 한번쯤 들어봤을 Oracle, MySQL, SQLite … 등이 모두 DBMS
    - DBMS에서 Database를 조작하기 위해 사용하는 언어를 SQL이라 함

- 웹 개발에서 대부분의 데이터베이스는 **관계형 데이터베이스 관리 시스템(RDBMS)**을 사용하여 SQL로 데이터와 프로그래밍을 구성

RDB

RDB란

- Relational Databse (관계형 데이터베이스)
- 데이터를 테이블, 행, 열 등으로 나누어 구조화하는 방식
- 자료를 여러 테이블로 나누어서 관리하고, 이 테이블간 관계를 설정해 여러 데이터를 쉽게 조작할 수 있다는 장점이 있음
- SQL을 사용하여 데이터를 조회하고 조작

스키마(Schema)

- 테이블의 구조(Structure)
- 데이터베이스에서 자료의 구조, 표현 방법, 관계 등 전반적인 명세를 기술한 것

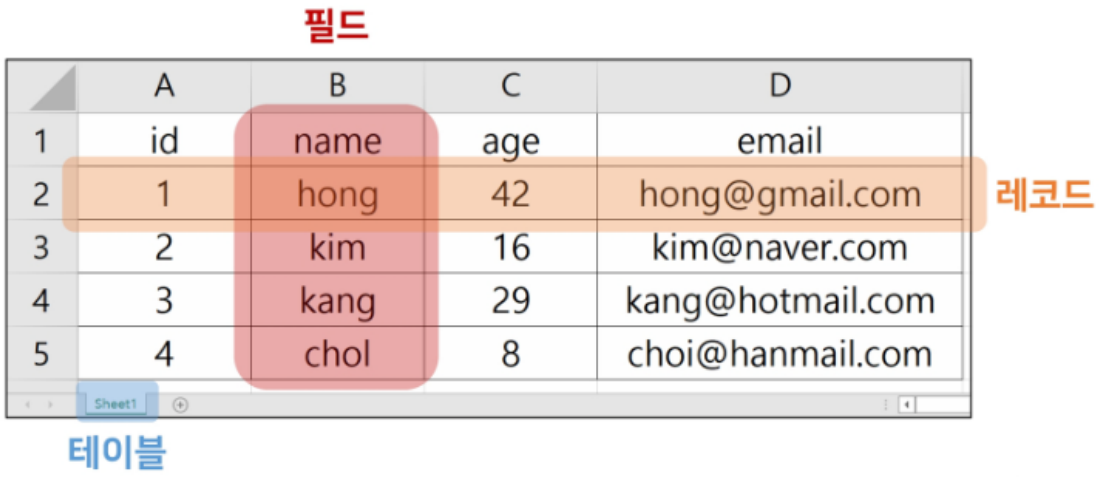
column	datatype
id	INTEGER
name	TEXT
address	TEXT
age	INTEGER

테이블(Table)

- 필드와 레코드를 사용해 조직된 데이터 요소들의 집합

- 관계(Relation)라고도 부름

1. 필드(field)
    - 속성, 컬럼(Column)
    - 테이블의 데이터는 레코드에 저장됨
2. 레코드(record)
    - 튜플, 행(Row)



PK (Primary Key)

- 기본 키
- 각 레코드의 고유한 값
    - 각각의 데이터를 구분할 수 있는 고윳값
- 기술적으로 다른 항목과 절대로 중복될 수 없는 **단일값(unique)**
- 데이터베이스 관리 및 테이블

관계형 데이터베이스의 이점

- 데이터를 직관적으로 표현할 수 있음
- 관련한 각 데이터에 쉽게 접근할 수 있음
- 대량의 데이터도 효율적으로 관리 가능

RDBMS

- Relational Database Management System (관계형 데이터베이스 관리 시스템)

- 관계형 데이터베이스를 만들고 업데이트하고 관리하는데 사용하는 프로그램
- **SQLite**

SQLite

- 응용 프로그램에 파일 형식으로 넣어 사용하는 비교적 가벼운 데이터베이스
- 안드로이드, iOS, macOS에 기본적으로 탑재되어 있으며 임베디드 소프트웨어에서도 많이 활용됨
- 오픈 소스 프로젝트이기 때문에 자유롭게 사용가능
- SQLite 단점
  - 대규모 동시 처리 작업에는 적합하지 않음
  - 다른 RDMBS에서 지원하는 SQL 기능을 지원하지 않을 수 있음
- 학습하는 이유
  - 어떤 환경에서나 실행 가능한 호환성
  - 데이터 타입이 비교적 적고 강하지 않기 때문에 유연한 학습 환경을 제공
  - Django Framework의 기본 데이터베이스

SQL

SQL이란

- **Structured Query Language**
- RDBMS의 데이터를 관리하기 위해 설계된 **특수 목적의 프로그래밍 언어**
- RDBMS에서 데이터베이스 스키마를 생성 및 수정할 수 있으며, 테이블에서의 자료 검색 및 관리도 할 수 있음
- 데이터베이스 객체에 대한 처리를 관리하거나 접근 권한을 설정하여 허가된 사용자만 RDBMS를 관리할 수 있도록 할 수 있음
- 많은 데이터베이스 관련 프로그램들이 SQL을 표준으로 채택하고 있음

SQL 정리

- SQL은 **데이터베이스와 상호작용하는 방법**
- 따라서 SQL을 배우면서 데이터베이스의 동작원리 또한 익힐 수 있음

SQL Commands 종류

- 명령어는 특성에 따라 다음 세가지 그룹으로 분류

1. DDL (Data Definition Language)
    a. 데이터 정의 언어
    b. 관계형 데이터베이스 구조(테이블, 스키마)를 정의(생성, 수정 및 삭제)하기 위한 명령어
    c. CREATE, DROP, ALTER
2. DML (Data Manipulation Language)
    a. 데이터 조작 언어
    b. 데이터를 조작(추가, 조회, 변경, 삭제)하기 위한 명령어
    c. INSERT, SELECT, UPDATE, DELETE
3. DCL (Data Control Language)

SQL Syntax

```sql
-- SQL Syntax 예시

SELECT column_name FROM table_name;
```

- 모든 SQL문(statement)는 SELECT, INSERT, UPDATE 등과 같은 키워드로 시작하고, 하나의 statement는 **세미콜론(;)**으로 끝남
    - 세미콜론은 각 SQL 문을 구분하는 표준 방법
- SQL 키워드는 대소문자를 구분하지 않음
    - 즉, SELECT와 select는 SQL문에서 동일한 의미
    - 하지만 대문자로 작성하는 것을 권장
- SQL에 대한 세부적인 문법 사항들은 이어지는 DDL, DML을 진행하며 익혀볼 것]

SQLite Data Types

1. NULL
    a. NULL value

    b. 정보가 없거나 알 수 없음을 의미 (missing information or unknown)
2. INTEGER
    a. 정수
    b. 크기에 따라 0, 1, 2, 3, 4, 6, 또는 8바이트와 같은 가변 크기를 가짐
3. REAL
    a. 실수
    b. 8바이트 부동 소수점을 사용하는 10진수 값이 있는 실수
4. TEXT
    a. 문자 데이터
5. BLOB (Binary Large Object)
    a. 입력된 그대로 저장된 데이터 덩어리 (대용타입없음)
    b. 바이너리 등 멀티미디어 파일
    c. 이미지 데이터

SQLite Datatypes 특징

- SQLite 는 다른 모든 SQL 데이터베이스 엔진(MySQL, PostgreSQL 등)의 정적이고 엄격한 타입(static, rigid typing)이 아닌 **동적 타입 시스템(dynamic type system)**을 사용
    - 컬럼에 선언된 데이터 타입에 의해서가 아니라 **컬럼에 저장된 값에 따라 데이터 타입이 결정됨**
- 또한 테이블을 생성할 때 컬럼에 대해 특정 데이터 타입을 선언하지 않아도 됨
    - 다만 다른 데이터베이스와의 호환성 문제가 있기 때문에 테이블 생성 시 **데이터 타입을 지정하는 것을 권장**

Type Affinity

- **타입 선호도**
- 특정 컬럼에 저장된 데이터에 권장되는 타입
- 데이터 타입 작성 시 SQLite의 5가지 데이터 타입이 아닌 다른 데이터 타입을 선언한다면, 내부적으로 각 타입의 지정된 선호도에 따라 5가지 선호도로 인식됨

1. INTEGER

2. TEXT
3. BLOB
4. REAL
5. NUMERIC

- 타입 선호도 존재 이유
  - 다른 데이터베이스 엔진 간의 **호환성**을 최대화
  - 정적이고 엄격한 타입을 사용하는 데이터베이스의 SQL문을 SQLite에서도 작동하도록 하기 위함

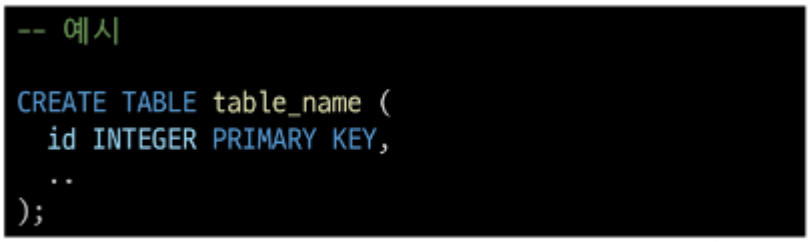
Constraints

데이터 무결성

- 데이터 베이스 내의 데이터에 대한 정확성, 일관성을 보장하기 위해 데이터 변경 혹은 수정 시 여러 제한을 두어 데이터의 정확성을 보증하는 것
  - 무결성이란 데이터의 정확성, 일관성을 나타냄
- 데이터베이스에 저장된 데이터의 무결성을 보장하고 데이터베이스의 상태를 일관되게 유지하는 것이 목적

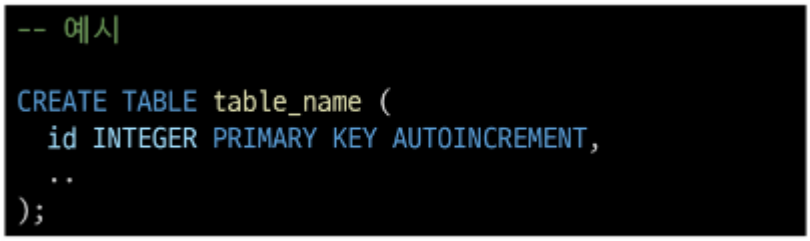
Constraints 종류

1. NOT NULL
   a. 컬럼이 NULL 값을 허용하지 않도록 지정
   b. 기본적으로 테이블의 모든 컬럼은 NOT NULL 제약 조건을 명시적으로 사용하는 경우를 제외하고는 NULL 값을 허용함
2. UNIQUE
   a. 컬럼의 모든 값이 서로 구별되거나 고유한 값이 되도록 함
3. PRIMARY KEY
   a. 테이블에서 행의 고유성을 식별하는데 사용되는 컬럼
   b. 각 테이블에는 하나의 기본 키만 있음
   c. 암시적으로 NOT NULL 제약 조건이 포함되어 있음



❖ 주의) INTEGER 타입에만 사용가능 (INT BIGINT 등 불가능)

4. AUTOINCREMENT
    a. 사용되지 않은 값이나 이전에 삭제된 행의 값을 재사용하는 것을 방지
    b. INTEGER PRIMARY KEY 다음에 작성하면 해당 rowid를 다시 재사용하지 못하도록 함
    c. Django에서 테이블 생성 시 id 컬럼에 기본적으로 사용하는 제약조건



5. 그외 기타 Constraints

rowid의 특징

- 테이블을 생성할 때마다 rowid라는 암시적 자동 증가 컬럼이 자동으로 생성됨
- 테이블의 행을 고유하게 식별하는 63비트 부호 있는 정수 값
- 테이블에 새 행을 삽입할 때 마다 정수 값을 자동으로 할당
    - 값은 1에서 시작
    - 데이터 삽입 시에 rowid 또는 INTEGER PRIMARY KEY 컬럼에 명시적으로 값이 지정되지 않은 경우, SQLite는 테이블에서 가장 큰 rowid보다 하나 큰 다음 순차 정수를 자동으로 할당 (AUTOINCREMENT와 관계없이)
- 만약 INTEGER PRIMARY KEY 키워드를 가진 컬럼을 직접 만들면 이 컬럼은 rowid 컬럼의 별칭(alias)이 됨
    - 즉, 새 컬럼 이름으로 rowid에 액세스 할 수 있으며 rowid 이름으로도 여전히 액세스 가능

- 데이터가 최대 값에 도달하고 새 행을 삽입하려고 하면 SQLite는 사용되지 않는 정수를 찾아 사용
- 만약 SQLite가 사용되지 않은 정수를 찾을 수 없으면 SQLite_FULL 에러가 발생
    - 또한 일부 행을 삭제하고 새 행을 삽입하면 SQLite는 삭제된 행에서 rowid 값을 재사용하려고 시도

ALTER TABLE

개요

- **Modify the structure of an existing table**
- 기존 테이블의 구조를 수정(변경)
- SQLite의 ALTER TABLE 문을 사용하면 기존 테이블을 다음과 같이 변경할 수 있음
    1. **Rename** a table
    2. **Rename** a column
    3. **Add** a new column to a table
    4. **Delete** a column
- 예시

```sql
-- 1. Rename a table
ALTER TABLE table_name RENAME TO new_table_name;


-- 2. Rename a column
ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;


-- 3. Add a new column to a table
ALTER TABLE table_name ADD COLUMN column_definition;


-- 4. Delete a column
ALTER TABLE table_name DROP COLUMN column_name;
```

- ALTER TABLE ADD COLUMN

- 현재 과정에서는 일어나지 않지만 만약 테이블에 기존 데이터가 있을 경우 다음과 같은 에러가 발생

```
Cannot add NOT NULL column with default value NULL
```

- 이전에 이미 저장된 데이터들은 새롭게 추가되는 컬럼에 값이 없기 때문에 NULL이 작성됨
- 그런데 새로 추가되는 컬럼에 NOT NULL 제약조건이 있기 때문에 기본 값 없이는 추가될 수 없다는 에러가 발생한 것
- 다음과 같이 **DEFAULT** 제약 조건을 사용하여 해결할 수 있음

```
ALTER TABLE new_contacts ADD COLUMN address TEXT NOT NULL DEFAULT 'no address';
```

- 이렇게 하면 address 컬럼이 추가되면서 기존에 있던 데이터들의 address 컬럼값은 **no address**가 됨

- ALTER TABLE DROP COLUMN
  - 단, 삭제하지 못하는 경우가 있음
    - 컬럼이 다른 부분에서 참조되는 경우
      - foreign key(외래 키) 제약조건에서 사용되는 경우
    - PRIMARY KEY인 경우
    - UNIQUE 제약 조건이 있는 경우

```
ALTER TABLE new_contacts DROP COLUMN email;
```

```
Cannet drop UNIQUE column: "email"
```

DROP TABLE

- **remove a table from the database**
- 데이터베이스에서 테이블을 제거

```
DROP TABLE table_name;
```

- 존재하지 않는 테이블을 제거하면 SQLite에서 오류가 발생

```
no such table: table_name
```

- 한번에 하나의 테이블만 삭제할 수 있음
- 여러 테이블을 제거하려면 여러 DROP TABLE 문을 실행해야 함
- DROP TABLE 문은 실행 취소하거나 복구할 수 없음
    - 따라서 각별히 주의하여 수행해야 한다

DML

개요

- DML을 통해 데이터를 조작하기 (CRUD)
- INSERT, SELECT, UPDATE, DELETE