#cs #study

목차

1) 데이터베이스 개요

데이터베이스와 DBMS

- 데이터베이스 : 원하는 기능을 동작시키기 위해 마땅히 저장해야 하는 정보의 집합
- 데이터베이스 관리 시스템(DBMS) : 데이터베이스를 관리하는 프로그램
- DBMS 종류 : RDBMS, NoSQL DBMS
- 서버로서의 DBMS
  - DBMS는 사용자가 만든 프로그램과 상호작용하며 실행된다. 즉, OS와 응용 프로그램 사이에 위치해 있으며 응용 프로그램은 DBMS 쪽으로 자원(DB)을 요청한다는 점에서 서버와 유사하다.
  - DBMS 클라이언트는 DBMS에 쿼리(query)를 보내 DB를 조작하며 DBMS는 클라이언트가 쿼리를 사용할 수 있게 DB 언어를 제공한다.
- SQL
  - 데이터베이스 언어 중, SQL은 RDBMS에서 데이터를 관리하고 조작하기 위한 언어다.
  - SQL 언어 종류
    - DDL(Data Definition Language) : 데이터 정의
    - DML(Data Manipulation Language) : 데이터 조작
    - DCL(Data Control Language) : 데이터 제어
    - TCL(Transaction Control Language) : 트랜잭션 제어

파일 시스템 대신 DB를 이용하는 이유

- 데이터 일관성 및 무결성 제공이 어려움
  - 파일은 여차하면 여러 프로세스 또는 스레드가 공유 자원으로 접근할 경우, 레이스 컨디션이 발생할 수 있음
  - 파일에 명시된 데이터에 결함이 없는 걸 하나 하나 검사하는 것도 어려움
- 불필요한 중복 저장
  - DB에서는 각 테이블과 연관된 정보가 서로 유기적으로 연결되어 있다. 개별 파일은 독립적이므로 각 테이블의 원소값을 중심으로 다른 원소값을 나열하면 필시 중복 정보가 발생한다.
- 데이터 변경 시 연관 데이터 변경이 어려움

    - DB는 데이터 값 하나를 변경하면 연결된 모든 데이터에 이를 반영한다. 하지만 파일은 각자 개별로 값이 저장되어 있으므로 파일을 순회하며 다 바꿔줘야 한다.
- 정교한 검색이 어려움
    - DBMS가 제공하는 문법보다 별도의 스크립트를 작성해서 파일 내부 데이터를 검색하는 게 당연히 더 어렵다
- 백업 및 복구가 어려움

데이터베이스의 저장 단위와 트랜잭션

- 데이터베이스 저장 단위
    - 엔티티 : 독립적으로 존재할 수 있는 존재. RDBMS로 치면 테이블에 있는 하나의 행이고 NoSQL이면 하나의 키-값 묶음으로 볼 수 있다.
        - 속성 : 엔티티의 특성이며 RDBMS로 치면 테이블에 있는 개개의 열 항목이고 NoSQL이면 키에 해당한다
    - 엔티티 집합 : 엔티티의 추상화. 엔티티가 각 행에 실재하는 값들을 의미하면 이를 추상화한 하나의 엔티티를 엔티티 집합이라 한다. RDBMS에서는 Relation, NoSQL에서는 Collection이라 표현하는 경우가 많음
    - 각 데이터 항목을 부르는 이름 : RDBMS에서는 행을 Record, 열을 Field라 부른다. NoSQL에서는 개별 Record를 Document라 표현하고 개별 키-값을 Field라 부른다.
- 스키마(Schema)
    - 데이터베이스에 저장되는 레코드의 구조와 제약 조건을 정의한 것
    - RDBMS는 테이블 구조와 행과 열로 명확하게 DB 구조를 정의했다. 그래서 스키마 명확한 편이나 반대로 NoSQL은 명확한 스키마가 없다. 그래서 NoSQL 데이터베이스를 스키마-리스(Schema-less) 데이터베이스라고도 부른다.
- 트랜잭션과 ACID
    - 트랜잭션(Transaction) : 데이터베이스의 논리적인 상호작용 단위. 즉, 데이터베이스가 처리되는 작업의 단위다.
        - 초당 트랜잭션(TPS) : 단어의 이름 그대로 1초에 얼마나 트랜잭션을 처리하는 지 나타낸 수치이며 데이터베이스의 작업 성능을 나타낸다
    - ACID : 원자성, 일관성, 격리성, 지속성을 영어로 각 앞글자만 따서 합쳐놓은 단어. 트랜잭션이 지켜야 하는 성질이다.
        - 원자성(Atomicity) : 하나의 트랜잭션 결과가 모두 성공하거나 모두 실패하는 성질. 즉, 트랜잭션이 여러 쿼리로 이루어진 작업일 때, 중간 쿼리 작업 중에 문제가 생기면 처음에 성공한 쿼리들도 전부 무위로 돌아가야 한다
        - 일관성(Consistency) : 트랜잭션 전후로 데이터베이스 일관된 성질을 유지한다. 즉, 데이터 구조에 변형이 생기면 안 된다.
        - 격리성(Isolation) : 동시에 수행되는 여러 트랜잭션이 서로 간섭하지 않도록 보장하는 성질. 레이스 컨디션 방지용.
        - 지속성(Durability) : 트랜잭션이 성공적으로 끝난 후에 그 결과가 영구적으로 반영되어야 한다.

Quiz

- 아래와 같이 구성된 RDBMS가 있다. 여기서 철수가 영희에게 5000원을 이체하는 작업은 하나의 트랜잭션으로 볼 수 있는가?

이름	잔액
철수	10000
영희	10000

- 둘의 차이점을 얘기하고 스키마 형식에 대해 논해보시오.

```
1.
| 이름 | 성별 |
| JJH |  M  |
| BGM |  M  |

2.
{
    {
        "이름" : "JJH",
        "성별" : "M"
    },
    {
        "이름" : "BGM",
        "성별" : "M"
    }
}
```

2) RDBMS 기본

테이블의 구성: 필드와 레코드

- RDBMS는 스키마가 명확하다고 배운 게 기억나는가? 그렇기에 RDBMS는 필드의 자료형은 각 필드마다 정해져 있다. 즉, 특정 필드의 자료형을 바꾸면 해당 필드에 저장된 레코드들을 전부 자료형에 맞는 데이터로 바꿔줘야 하는 것이다.
- 필드 타입 : 필드의 자료형은 크게 숫자형, 문자형, 날짜/시간형, 기타로 나누어 진다.
  - 숫자형 : TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL
  - 문자형 : CHAR, VARCHAR, BLOB, TEXT
  - 날짜/시간형 : DATE, TIME, DATETIME, TIMESTAMP
  - 기타 : ENUM, CEOMETRY, XML, JSON
- 키 : 테이블의 특정 레코드를 식별하기 위한 특정 필드를 의미한다. 해당 필드값을 통해 테이블에서 특정 위치의 레코드를 불러올 수 있다.
  - 키의 종류 : 후보 키, 복합 키, 슈퍼 키, 기본 키, 대체 키, 외래 키

    - 후보 키(Candidate Key) : 테이블의 한 레코드를 식별하기 위한 필드의 최소 집합. 유일성(특정 레코드를 유일하게 식별 가능), 최소성(최소한의 정보로 레코드 식별)을 갖춘 키
    - 복합 키(Composite Key) : 2개의 필드 이상으로 레코드를 식별할 수 있는 키. 가령, 한 대학교의 학생들이 들은 수강 과목들을 열거한다 치면 각 레코드를 구별하기 위해서는 [학번, 과목명] 둘 다 알 필요가 있다. 만약, 재수강 과목까지 포함하면 [학번, 과목명, 수강 연도]까지 알아야 식별할 수 있을 것이다.
    - 기본 키(Primary Key) : 하나의 레코드를 식별하도록 선정되어 테이블당 하나만 존재하는 키. 자료구조에서 배열의 첫번째 원소 주소가 배열을 대표하는 주소이듯 테이블 하나를 대표하는 특정 키다. 가령, 대학교의 학생 명부를 데이터베이스에 기입했을 때, 학생 명부라는 테이블의 기본 키는 '학번'이 될 것이다.
    - 외래 키(Foreign Key) : 다른 테이블의 기본 키를 참조하는 필드. 테이블 간 참조 관계를 형성할 때 사용한다. 학생들 수강 과목 테이블이 있고 수강과목 정보 테이블이 있으면 학생 테이블 -> 과목 테이블로 참조하기 위해서는 과목명 필드가 외래 키가 될 것이다.

테이블 간의 관계

- 일대일 대응
    - 하나의 레코드가 다른 테이블의 레코드 하나에만 대응하는 경우
    - 가령 국민의 인적 정보가 들어간 테이블과 주민등록증 번호와 발급일이 들어가 있는 경찰 행정 테이블이 있으면 두 테이블은 일대일 대응이다. 왜냐하면 국민이 행정상 주민등록증 2개를 등록해서 들고 다닐 수 없기 때문이다.
- 일대다 대응
    - 하나의 테이블이 다른 테이블의 여러 레코드와 대응하는 경우
    - 가령 국민의 인적 정보 테이블과 전입이력을 관리하는 테이블이 있으면 두 테이블은 일대다 대응이다. 왜냐하면 한 사람이 이사 여러 번 다닐 수도 있는 것이니까.
- 다대다 대응
    - 하나의 테이블의 여러 레코드가 다른 테이블의 여러 레코드와 대응하는 경우
    - 테이블 간 다대다 대응은 보통 서로 다대다로 연결된 레코드를 쉽게 식별하기 위해 중간 테이블을 하나 거친다. 이 때 거치는 테이블은 서로 다대다로 연결된 필드들이 나열된 테이블.
    - 가령 대학교의 학생 인적 정보 테이블과 동아리 목록 테이블을 생각해 보자. 학생은 여러 동아리를 가입할 수 있고 또한 동아리에 속한 학생들도 여러 명이다. 그렇기에 서로가 가리키고 있는 레코드를 식별하기 위해서는 서로 어느 레코드가 매칭되었는 지 나열되어 있는 중간 테이블이 필요하다.

무결성 제약 조건

- 무결성 제약 조건 : 데이터베이스에 저장된 데이터의 일관성과 유효성을 유지하기 위해 마땅히 지켜야 하는 조건
- 도메인 제약 조건 : 테이블이 가질 수 있는 필드 유형과 범위에 대한 규칙. 필드 데이터는 지정된 필드 타입 준수, 원자 값, NULL 허용 여부에 따른 NULL 값 존재 등. 주어진 제약 조건을 준수한다.
    - 원자 값(atomic value) : 더는 쪼개질 수 없는 단일값. 해당 필드의 데이터가 복수의 값을 가지고 있으면 안된다

- 키 제약 조건 : 레코드를 고유하게 식별할 수 있는 키로 지정된 필드에 중복값이 존재하면 안 됨
- 엔티티 무결성 제약 조건 : 기본 키로 지정한 필드는 고유한 값이어야 하며, NULL이 되어서는 안 된다는 규칙
- 참조 무결성 제약 조건 : 외래 키를 통해 다른 테이블을 참조할 때, 데이터의 일관성을 지키기 위한 제약 조건

참조 테이블 삭제/수정 문제

- 참조하고 있는 테이블에서 키 값이 사라지거나 변경되어 동일한 값을 추적할 수 없는 경우, 다음 4가지 동작을 취할 수 있다.
- 연산 제한(restrict) : 주어진 수정 및 삭제 연산 자체를 거부한다.
- 기본값 설정(set default) : 참조하는 레코드를 미리 지정한 기본값으로 설정
- NULL 값 설정(set NULL) : 참조하는 레코드를 NULL로 설정
- 연쇄 변경(cascade) : 참조하는 레코드도 함께 수정하거나 삭제한다

Quiz

- 영희는 DB를 아래와 같이 구성했다.

```
TABLE 1 : Personal Info
| 번호 | 성명 |  운전면허증(FK) |
|  1  | ANN |  1012021A8E  |
|  2  | GYK |  2032132B7A  |
|  3  | TYU |  7683827C2B  |

TABLE 2 : Location INFO
|  운전면허증(PK) | 권한 |
|  1012021A8E  |  1  |
|  2032132B7A  |  1  |
|  2032132B7A  |  2  |
|  7683827C2B  |  2  |
|  7683827C2B  |  7  |
```

    - 위와 같은 관계로 TABLE이 이루어져 있을 때, 무슨 관계인 지, 어떻게 관계를 추측했는 지 논하시오.

3) SQL

데이터 정의 언어(DDL)

- CREATE
    - 데이터베이스에서 관리될 수 있는 다양한 대상(DB, TABLE, VIEW, USER 등)을 정의한다.

```
-- 데이터베이스 조회
SHOW DATABASES;
```

```
-- 해당 데이터베이스에서 작업 시작
USE database_name;
-- 테이블 구조 조회
DESCRIBE table_name;

-- 데이터베이스 생성
CREATE DATABASE database_name; // 데이터베이스 생성
-- 특정 필드 구조를 가진 table 형성
CREATE TABLE table_name (
    field_name1 fleld_type1,
    field_name2, field_type2,
    [CONSTRAINT 제약_조건_이름] PRIMARY KEY (field_name),
    [CONSTRAINT 제약_조건_이름] FOREIGN KEY (field_name) REFERENCES
table_name2 (field_name),
    [CONSTRAINT 제약_조건_이름] UNIQUE (field_name)
);
```

  - CREATE TABLE문 하단에 별도의 키워드를 명시해서 제약 조건을 걸 수 있다
    - PRIMARY KEY : 특정 필드를 기본 키로 지정
    - UNIQUE : 특정 필드가 고유한 값을 갖도록 설정
    - FOREIGN KEY : 특정 필드를 외래 키로 지정
    - DEFAULT 기본값 : 기본값 지정
    - NULL/NOT NULL : 특정 필드에 NULL 값을 허용/허용하지 않음
- ALTER
  - CREATE TABLE 문을 통해 생성된 테이블에 새로운 필드를 추가하거나 기존의 필드를 수정/삭제, 제약 조건 또한 새롭게 추가하거나 수정/삭제 가능한 명령어

```
-- 새로운 필드 추가
ALTER TABLE table_name ADD COLUMN field_name field_type [제약_조건];
-- 기존 필드 수정
ALTER TABLE table_name CHANGE COLUMN origin_field_name new_field_name
[제약_조건];
-- 기존 필드 삭제
ALTER TABLE table_name DROP COLUMN field_name;
-- 외래 키 제약 조건 추가
ALTER TABLE table_name [ADD CONSTRAINT 제약_조건_이름] ADD FOREIGN KEY
(field_name) REFERENCES 참조_테이블_이름;
-- UNIQUE 제약 조건 추가
ALTER TABLE table_name MODIFY field_name field_type NOT NULL;
-- 기본 키 설정(PRIMARY KEY로 사용 중인 필드가 없는 경우)
ALTER TABLE table_name ADD PRIMARY KEY (field_name);
```

- DROP
  - 테이블이나 데이터베이스를 삭제할 수 있는 명령어

```
-- 데이터베이스 삭제
DROP DATABASE database_name;
-- 테이블 삭제
DROP TABLE table_name;
```

- TRUNCATE
  - 테이블의 구조를 유지한 채로 테이블의 모든 레코드를 삭제한다

```
-- 테이블의 모든 레코드 삭제
TRUNCATE TABLE table_name;
```

데이터 조작 언어(DML)

- SELECT
  - 기존에 생성된 테이블 내 레코드를 조회할 때 쓰인다

```
-- SELECT 조회문(WHERE 부터는 선택적으로 들어간다)
SELECT field1, field2, ...
    FROM table_name
    WHERE 조건식
    GROUP BY 그룹으로 묶을 필드
    HAVING 필터 조건
    ORDER BY 정렬할 필드
    LIMIT 레코드 제한;
-- 테이블에 속한 모든 필드와 레코드 조회하기
SELECT * FROM table_name;
```

  - 패턴 검색과 연산/집계 함수
    - 패턴 검색 : 문자열 데이터에서 특정 패턴을 찾는 기능. WHERE과 LIKE 연산자그리고 와일드 문자(%, _)를 써서 찾는다
      - LIKE : 연산자 뒤에 붙는 내용물과 같은 지 찾는 연산자
      - % : 0개 이상의 임의의 문자와 일치
      - _ : 1개인 임의의 문자와 일치

```
-- 학생 명부에서 전공이 컴퓨터공학과인 학생을 찾고 학생의 이름과 성 그리고 전공을 표시
SELECT first_name, last_name, major FROM students WHERE major = 'CS';
-- 학생 명부에서 전공 명칭에 과학이 들어가는 학생을 찾고 학생의 성명, 전공 표시
SELECT first_name, last_name, major FROM students WHERE major LIKE '%Science%';
-- 학생 명부에서 전공 명칭에서 두번째 글자에 a가 들어가는 학생을 찾고 학생의 성명, 전공 표시
```

```
SELECT first_name, last_name, major FROM students WHERE major
LIKE '_a%';
```

- 집계 함수 : 조회한 레코드에 특저 연산을 수행하거나 집계하는 함수
  - COUNT : 조회한 레코드 개수 반환
  - SUM : 조회한 레코드의 총합 반환
  - AVG : 조회한 레코드의 평균 반환
  - MAX : 조회한 레코드의 최댓값을 반환
  - MIN : 조회한 레코드의 최솟값을 반환

```
-- 학생 명부에서 학생 인원수, 평균 학점, 최고 학점, 최저 학점을 조회한다
SELECT COUNT(*), AVG(gpa), MAX(gpa), MIN(gpa) FROM students;
```

- INSERT
  - 존재하는 테이블에 레코드를 삽입한다
  - INSERT 구문 사용 시 꼭 주의해야 할 점은 무결성 제약 조건을 위배하지 않았는 지 확인하는 것이다.

```
-- 테이블에 레코드를 추가로 하나 생성한다
INSERT INTO table_name(field_name1, field_name2) VALUES (field_value1,
field_value2);
-- 테이블에 여러 레코드를 삽입한다
INSERT INTO table_name(field_name1, field_name2, field_name3) VALUES
    (value1, value2, value3),
    (value1, value2, value3),
    ...
```

- UPDATE와 DELETE
  - 각각 레코드르 수정하고 삭제하는 명령어

```
-- 해당 테이블에서 조건식에 준수하는 레코드만 각 필드에 들어간 값을 수정한다
UPDATE table_name
    SET field1 = value1, field2 = value2, ...
    WHERE 조건식;
-- 해당 테이블에서 조건식에 준수하는 레코드만 삭제한다
DELETE FROM table_name
    WHERE 조건식;
```

- WHERE 구문 지원 연산자
  - 비교 연산자 : =, >, <, >=, <=
    - <> : 다를 경우 참
  - 논리 연산자 : AND, OR, NOT
    - IN : IN 뒤에 명시되는 값과 하나 이상이 일치할 경우 참

    - 외래 키 참조 상황에서 레코드 수정/삭제 시, CREATE 또는 ALTER 구문에서 미리 제약 조건을 걸어줘야 한다
        - CASCADE : 참조하는 데이터도 함께 수정/삭제
        - SET NULL : 참조하는 데이터를 NULL로 변경
        - SET DEFAULT : 참조하는 데이터를 기본값으로 변경
        - RESTRICT : 수정/삭제를 허용하지 않음
        - NO ACTION : (MySQL의 경우)사실상 RESTRICT와 동일하게 동작함
- DML에서 사용 잘하는 정렬 연산자
    - GROUP BY
        - 특정 필드를 기준으로 필드를 그룹화하기 위해 사용

```
-- 전공별 학생수를 조회한다. AS 뒤에 붙는 건 COUNT(*)의 필드명을 붙인 것. 안 붙이면 COUNT(*)이 필드명이 된다
SELECT major, COUNT(*) AS student_count
    FROM students
    GROUP BY major;
-- 나이별 학생수 조회
SELECT age, COUNT(*) AS student_count
    FROM students
    GROUP BY age;
```

    - HAVING
        - GROUP BY 절로 그룹화된 결과에 조건을 적용하기 위해 사용
        - WHERE은 그룹화 되기 전 개별 레코드에 대한 조건식이다

```
-- 평균 GPA가 3.6 이상인 전공을 조회
SELECT major, AVG(gpa)
    FROM students
    GROUP BY major
    HAVING AVG(gpa) >= 3.6;
-- 평균 나이가 21세 이상인 전공
SELECT major, AVG(gpa)
    FROM students
    GROUP BY major
    HAVING AVG(age) >= 21;
```

    - ORDER BY
        - 특정 필드를 기준으로 데이터를 정렬

```
-- 학점 기준으로 내림차순 정렬
SELECT first_name, last_name, gpa
    FROM students
    ORDER BY gpa DESC;
-- 성 기준으로 오름차순 정렬
```

```
SELECT first_name, last_name, gpa
    FROM students
    ORDER BY gpa ASC;
```

    - LIMIT
        - 조회할 레코드 수를 제한
        - SELECT 문에서 가장 마지막 순서에 적는다.
- SELECT에서 문법 실행 순서
    - FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY -> LIMIT

트랜잭션 제어 언어(TCL)

- COMMIT
    - 데이터베이스에 작업을 반영한다
    - 기본적으로 대부분의 RDBMS에서는 DDL 언어 사용 시, 별도로 구문을 넣지 않아도 자동 커밋을 한다.
- ROLLBACK
    - 작업 이전의 상태로 되돌린다

```
-- 트랜잭션 시작
START TRANSACTION;

-- 두 개의 레코드 확인
SELECT * FROM account;
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;

-- account_id가 1인 레코드의 balance가 100 감소되었음을 확인
SELECT * FROM accounts;
-- 위의 감소된 결과물들이 다시 취소되고 원상태로 복귀한다
ROLLBACK;

-- 다시 레코드 확인, 원래대로 돌아온 걸 확인할 수 있다
SELECT * FROM accounts
```

- SAVEPOINT
    - 롤백의 기준점 설정

```
START TRANSACTION;

-- 세이브포인트 생성
SAVEPOINT sp1;

UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;
```

```
-- 세이브포인트 생성
SAVEPOINT sp2;

UPDATE accounts SET account_name = 'new_Kim' WHERE account_id = 1;
UPDATE accounts SET account_name = 'new_Lee' WHERE account_id = 2;

-- 세이브 포인트 생성
SAVEPOINT sp3;

SELECT * FROM accounts;

-- 특정 세이브포인트로 롤백
ROLLBACK TO SAVEPOINT sp2;
ROLLBACK TO SAVEPOINT sp1;
```

데이터 제어 언어(DCL)

- 사용자 권한과 관련된 명령어
- GRANT : 사용자에게 권한 부여
- REVOKE : 사용자로부터 권한 회수

Quiz

- 다음 테이블 작성 구문을 해석하시오

```
CREATE TABLE users (
    post_id INT AUTO_INCREMENT,
    user_id INT,
    title VARCHAR(50) NOT NULL,
    content VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY_KEY (post_id),
    CONSTRAINT FK_user_id FOREIGN_KEY (user_id) REFERENCES users(user_id),
    CONSTRAINT UQ_title UNIQUE (title)
);
```

- 아래와 같이 명령어를 사용했을 때, 테이블의 형태와 레코드를 유추하시오

```
INSERT INTO users (username, email, birthdate) VALUES
    ('lee', 'lee@example', '1994-03-22'),
    ('dong', 'dong@example', '2000-01-30')
```

추천 문제

- 조건에 맞는 회원수 구하기, https://school.programmers.co.kr/learn/courses/30/lessons/131535
- 재구매가 일어난 상품과 회원 리스트 구하기, https://school.programmers.co.kr/learn/courses/30/lessons/131536
- 이름에 el이 들어가는 동물 찾기, https://school.programmers.co.kr/learn/courses/30/lessons/59047
- 루시와 엘라 찾기, https://school.programmers.co.kr/learn/courses/30/lessons/59046