DB 프로젝트4
2013011800 구장회

[기존 설계와 바뀐점]

<1> 요구사항 변경

기존 : 각 장르 별 매니저가 존재한다.(즉, 장르가 같으면 매니저도 같다.)
수정 : 각 장르 별 매니저는 다를 수 있다.
이유 : 장르는 앨범의 특성이다.(ALBUM은 GENRE를 애트리뷰트로 가진다). 또 각 앨범은 한명의 관리자에게 관리되므로 매니저의 ID(MID)를 외래키로 가지고 있어야 한다. 기존의 전제라면 , ALBUM의 GENRE와 MGRID는 중복이 되고 실제의 상황에 맞추려는 노력으로 기존의 전제를 수정한다.(매니저의 key attributute는 id이므로)

<2> 이름 변경
기존 : ADD
수정 : ADDSONG
이유 : 유저가 플레이리스트에 음악을 추가하는 관계타입에서 이름을 적절하게 수정한다.

[프로그램 코드에 대한 자세한 설명]

- **메서드 설명**
  - LoadingDriver() : maria db driver를 load한다.
  - ConnectDB() : 로컬 db(이름 : MUSIC)와 Connection을 만든다.
  - CreateStatement() : 만들어진 Connection에 Statement를 만든다.
  - StoreTableName() : 현재 MUSIC DB의 테이블의 이름을 ArrayList에 저장한다.
  - CloseConnection() : 로컬 db와 Connection을 종료한다.
  - mainMenu() : 메인 메뉴를 출력하고 사용자의 입력을 받는다.
  - process_main() : 메인 메뉴에서 입력받은 것에 따른 처리를 한다.
    - [0:Exit] : 프로그램 종료
    - [1:Show Table List] : 현재 db의 모든 테이블의 목록을 출력한다.
    - [2:Create New Table] : 새로운 테이블을 만든다.
      - 테이블의 이름을 입력받아서
      - 같은 이름의 테이블이 존재하면 break
      - 아니면, CreateTableQuery() 메서드에 테이블 이름을 인자로 넣어 호출한다.
      - 생성된 테이블의 이름을 테이블의 이름이 담긴 ArrayList에 저장한다.
    - [3:Delete Table]: 테이블을 삭제한다.
      - 테이블의 이름을 입력 받아서
      - 같은 이름의 테이블이 존재하지 않으면 break
      - 아니면, DeleteTableQuery() 메서드에 테이블의 이름을 인자로 넣어 호출한다.
      - 제거된 테이블의 이름을 테이블의 이름이 담긴 ArrayList에서 제거한다.
    - [4:Choose Table]: 해당 테이블에 대한 메뉴로 간다.
      - 테이블의 이름을 입력받아서
      - 같은 이름의 테이블이 존재하지 않으면 break
      - 아니면, tablemenu() 메서드에 테이블의 이름을 인자로 넣어서 호출한다.
      - 그 리턴값을 process_table() 메서드에 테이블의 이름과 함께 인자로 넣어서 호출한다.
    - [default] : break
  - tablemenu() : 해당 테이블을 사용자가 선택했을때의 메뉴가 출력되고, 사용자의 입력을 받는다.
  - process_table() : 테이블 메뉴에서 입력받은 것에 따른 처리를 한다.
    - [0:Return to previous menu] : 메인 메뉴로 간다.
    - [1:View Table] : 해당 테이블의 모든 정보를 출력한다.
    - [2:Insert] : InsertQuery() 메서드에 테이블의 이름을 넣어 호출한다.
    - [3:Update] : UpdateQuery() 메서드에 테이블의 이름을 넣어 호출한다.
    - [4:Delete] : DeleteQuery() 메서드에 테이블의 이름을 넣어 호출한다.
    - [5:Select] : SelectQuery() 메서드에 테이블의 이름을 넣어 호출한다.
    - [default] : break
  - CreateTableQuery() : 테이블의 이름을 인자로 받아서 테이블을 생성하는 메서드
    - **[주석에서 추출]**
    - // while loop 돌면서 에러가 없으면 탈출
    - // 애트리뷰트의 갯수를 입력받는다
    - // 애트리뷰트의 갯수만큼 문자열 배열을 만든다. 애트리뷰트마다 한줄씩 저장하기 위함
    - // 애트리뷰트 갯수만큼 for-loop
    - // 애트리뷰트 이름 , 데이터형 , NOT NULL 제약조건을 입력받아서 배열에 저장
    - // 애트리뷰트 이름 입력
    - // 애트리뷰트 타입 입력
    - // NOT NULL 제약조건 여부 확인

    - // 기본키 절 사용 여부 확인
    - // 기본키 절를 담을 문자열
    - // 기본키 절 사용
    - // 기본키 제약조건 이름을 정할건지 여부 확인
    - // 넣는다면, 문자열에 추가
    - // 기본키 입력 받고, 문자열에 추가
    - // 기본키 절 안쓸거면 아무것도 안한다
    - // 외래키 절 사용 여부 확인
    - // 외래키 절 사용
    - // 외래키 절 갯수 입력
    - // 그 갯수만큼 문자열 배열 생성
    - // 외래키 제약조건 이름 사용 여부 확인
    - // 이름 넣을거면 문자열 배열에 추가
    - // 외래키 입력
    - // 참조할 테이블 이름 입력
    - // 참조할 테이블의 애트리뷰트 입력
    - // 쿼리문 만들어서 문자열 배열에 저장
    - // ON DELETE 옵션 사용한다면 option을 입력받고 아니면 N을 입력
    - // 문자열 배열에 옵션 추가
    - // ON UPDATE 옵션 사용한다면 option을 입력받고 아니면 N을 입력
    - // 문자열 배열에 옵션 추가
    - // 외래키 절 사용 안하면 아무것도 안한다.
    - /* 쿼리문을 한 문자열로 만드는 과정 */
    - // 쿼리 실행
    - // 실행한 쿼리가 에러가 있을경우(Syntax error 혹은 제약조건 위배 등등) 다시 입력받도록(while loop 재실행)
- DeleteTableQuery() : 테이블의 이름을 인자로 받아서 테이블을 삭제하는 메서드
    - **[주석에서 추출]**
    - // 에러가 없으면 while문 종료
    - // 테이블 삭제시 옵션이 있는지 확인하고,
    - // 옵션이 있으면,
    - // 추가하고,
    - // 없으면 그냥 DROP TABLE + 테이블이름
    - // 쿼리 실행
    - // 실행한 쿼리가 에러가 있을경우(Syntax error 혹은 제약조건 위배 등등) 다시 입력받도록(while loop 재실행)

- InsertQuery() : 테이블의 이름을 인자로 받아서 Insert query 수행
    - **[주석에서 추출]**
    - // 쿼리문 준비
    - // 해당 테이블의 column 정보를 가져온다.
    - // 각 column 마다 값을 입력받는다.(INTEGER,VARCHAR,DATE 구분)
    - // 쿼리문 실행

- UpdateQuery() : 테이블의 이름을 인자로 받아서 update query 수행
    - **[주석에서 추출]**
    - // 쿼리문 준비
    - // 해당 테이블의 column 정보를 가져온다.

    - // SET 절
    - // 각 column 마다 값을 입력받는다.(INTEGER,VARCHAR,DATE 구분)
    - // WHERE 절
    - // 각 column 마다 값을 입력받는다.
    - // 쿼리문 실행
- DeleteQuery() : 테이블의 이름을 인자로 받아서 update query 수행
    - **[주석에서 추출]**
    - // 쿼리문 준비
    - // 해당 테이블의 column 정보를 가져온다.
    - // WHERE 절
    - // 각 column 마다 값을 입력받는다.
    - // 쿼리문 실행
- SelectQuery() : 테이블의 이름을 인자로 받아서 update query 수행
    - **[주석에서 추출]**
    - // 쿼리문 준비
    - // 해당 테이블의 column 정보를 가져온다.
    - // SELECT 절
    - // 각 column 마다 값을 입력받는다.
    - // FROM 절
    - // 테이블의 이름을 그대로
    - // WHERE 절
    - // 각 column 마다 값을 입력받는다.
    - // 쿼리문 실행하고 결과를 출력

[사용되는 sql 문 명세]

- InsertQuery() 메서드에서 : “INSERT INTO [TABLE NAME] VALUE (_____) ”
- UpdateQuery() 메서드에서 : “UPDATE [TABLE NAME] SET _____ WHERE _____”
- DeleteQuery() 메서드에서 : “DELETE FROM [TABLE NAME] WHERE _____ ”
- SelectQuery() 메서드에서 :
  - “SELECT * FROM [TABLE NAME] ”
  - “SELECT _____ FROM [TABLE NAME] WHERE _____”
- CreateTableQuery() 에서 :
  - “CREATE TABLE [TABLE NAME] (attr,type,not null , , , , , CONSTRAINT _____ PRIMARY KEY(___) , CONSTRAINT _____ FOREIGN KEY(___) REFERENCES [TABLE NAME] (_____), ON DELETE [OPTION] ON UPDATE [OPTION]”
- DeleteTableQuery() 에서 :
  - “DROP TABLE [TABLE NAME] [OPTION]”

[스크린샷-jpg 파일로 첨부]

[생각하지 않기로 한 경우]

<1> 데이터형
사용자의 입력을 받아 테이블에 대한 쿼리문 실행시
데이터형은 INTEGER,VARCHAR,DATE 3가지만 고려하였다.

<2> 참조-트리거 된 동작
제약조건의 이름을 명시하는 기능은 구현했으나
그 이름을 바꾸거나 삭제하는 등의 기능은 구현하지 않았다.