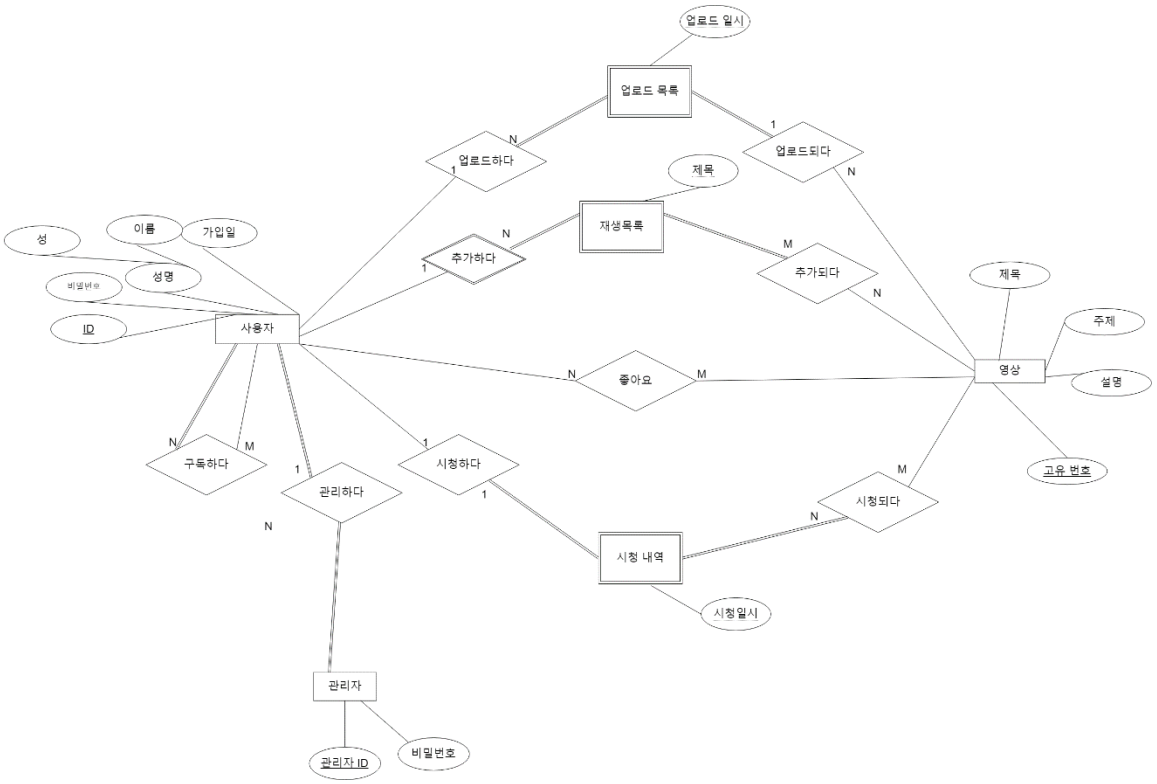
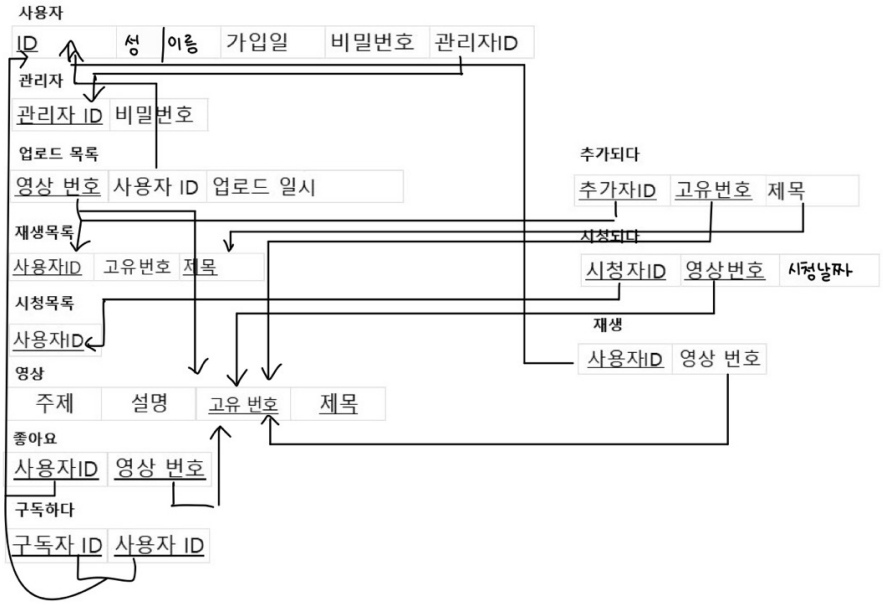
데이터베이스시스템 및 응용 #Project 2

2021037156 탁민주

1. 최종 E-R model 및 Relational model





2. 수정사항

* 삭제

- 사용자의 공유 기능, 영상의 자막, 화질 설정, 재생 속도, 추천 재생 목록, 영상 링크, 책갈피, 썸네일, 관리자의 가이드라인은 데이터베이스의 기능에 좀 더 초점을 맞추기 위해 삭제함.

* 수정

- 재생목록에는 동일한 영상이 담기지 못하도록 수정하였다.

- 관리자는 예외를 줄이기 위해 하나의 아이디와 비밀번호를 미리 설정하여 로그인하도록 하였다.

- 관리자는 관리를 위해 유저들의 모든 시청기록을 소유한다.

- 서로 다른 사용자는 같은 이름의 동영상을 올릴 수는 있으나 중복 방지를 위해 같은 사용자가 같은 이름의 동영상을 올리는 것은 금지하였다.

추가

- 재생 중에 좋아요, 구독, 플레이리스트에 추가 등의 부가적인 기능을 사용할 수 있다.

- 본인의 정보를 열람할 수 있는 함수를 추가하여 접속 정보를 알 수 있도록 하였다.

3. 응용 프로그램 코드 설명

(1) 변수 선언

```
private static final String Super_ID = "svr123";
private static final String Super_pwd = "svr456";
1개 사용 위치
private static String url = "jdbc:mysql://localhost:3306/db_project";
1개 사용 위치
private static String user = "root";
1개 사용 위치
private static String psw = "xkralswn39!";
196개 사용 위치
static PreparedStatement pst = null;
59개 사용 위치
static Connection connect = null;
121개 사용 위치
static ResultSet rs = null;
34개 사용 위치
private static Scanner sc = new Scanner(System.in);

88개 사용 위치
private static String sql;
32개 사용 위치
private static String db_user;
2개 사용 위치
private static boolean db_exit;
```

- Mysql을 연결하기 위한 변수를 미리 정의
- sql문을 위해 쓰는 변수 정의
- 현재 로그인한 사용자 변수

(2) 데이터베이스 연결 함수

```
1개 사용 위치
public static void DB_Connect() {
    try {
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        //driver 가져오기
        connect = DriverManager.getConnection( url, user, psw );
        //Connection 객체 생성 완료
    }
    catch (Exception e){
        e.printStackTrace();
    }
}
```

(3) main 실행문

```
public static void main(String[] args) throws SQLException, ClassNotFoundException {
    DB_Connect(); //DB 연결하기
    db_exit = false;
    Scanner sc = new Scanner(System.in);
    while(true) {
        if (db_exit) break;
        System.out.println("========== Video Site ==========");
        System.out.println("1. 회원가입");
        System.out.println("2. 로그인");
        System.out.println("3. 관리자 로그인");
        System.out.println("4. 종료");
        System.out.println("원하는 항목을 고르십시오.");
        int num = sc.nextInt();
        if (num == 1) {
            SignUp(sc);
        } else if (num == 2) {
            switch (Login(sc)) {
                case 1: //로그인 성공. 메뉴 화면으로 감
                    User_Menu();
                    continue;
                case 0: //비밀번호 다름.
                    continue;
                case -1:
                    //아이디 다름
                    continue;
                case -2: //DB 종료

                    System.exit( status: 1);
            }
        else if (num == 3){
            switch (supervisor_login()) {
                case 1: //로그인 성공. 메뉴 화면으로 감
                    supervisor_Menu();
                    continue;
                case 0: //비밀번호 다름.
                    continue;
                case -1:
                    //아이디 다름
                    continue;
                case -2: //DB 종료
                    System.exit( status: 1);
            }
        }
        else if (num == 4){
            break;

        }

    }
    if (rs != null) rs.close();
    if (pst != null) pst.close();
    if (!connect.isClosed()) connect.close();

}
```

- 전체 프로그램을 시작하는 메인이다.
- 총 4개의 메뉴 (회원가입, 로그인, 관리자 로그인, 종료) 의 메뉴가 있다. 사용자가 입력하는 번호의 메뉴가 선택된다. 마지막에는 사용자가 프로그램을 종료 시켰을 시 sql문과 DB 연결에 사용된 변수를 close 한다.

(3) – 1 회원가입 함수

```java
public static void SignUp() {
    try {
        System.out.println("---------- 회원가입 ----------");
        System.out.print("아이디(10자 이내): ");
        String new_ID = sc.next();
        System.out.print("비밀번호(12자 이내): ");
        String new_Pwd = sc.next();
        System.out.print("성(10자 이내): ");
        String new_Fname = sc.next();
        System.out.print("이름(10자 이내): ");
        String new_Lname = sc.next();
        java.util.Date utilDate = new java.util.Date();
        java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());
        //TODO 중복 있는지 , 글자 수 넘는지 check.

        //사용자 데이터베이스 새로운 계정 집어넣기
        pst = connect.prepareStatement( sql: "insert into user values (?,?,?,?,?,?)");
        pst.setString( parameterIndex: 1, new_ID);
        pst.setString( parameterIndex: 2, new_Pwd);
        pst.setString( parameterIndex: 3, new_Fname);
        pst.setString( parameterIndex: 4, new_Lname);
        pst.setDate( parameterIndex: 5, sqlDate);
        pst.setString( parameterIndex: 6, Super_ID);
        pst.executeUpdate();
    }
    //인서트 완료
    catch (SQLException e) {
        System.out.println("회원가입 도중 오류가 발생하였습니다. 다시 진행해주시길 바랍니다.");
    }
```

· 유저의 회원가입을 진행하는 함수이다. 아이디, 비밀번호, 성, 이름을 입력 받는다. 단, 만약에 아이디가 Primary key 이므로 기존의 유저와 아이디가 겹친다면 error가 발생하여 다시 시도하여야 한다. 모든 유저의 관리자 ID는 Super_ID로 insert 해준다.

(3)-2 로그인 함수

```java
public static int Login(Scanner s) throws SQLException {
    System.out.println("---------- Login ----------");
    System.out.print("ID: ");
    db_user = s.next();
    System.out.print("Password: ");
    String Login_Pwd = s.next();

    sql = "SELECT Password FROM USER WHERE userID = ?";

    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        if (rs.next()) {
            if (rs.getString( columnIndex: 1).contentEquals(Login_Pwd)) {
                System.out.println("로그인에 성공하였습니다!");
                return 1;
            } else {
                System.out.println("비밀번호가 다릅니다. 다시 시도해주세요.");
                return 0;
            }
        }
        System.out.println("아이디가 다릅니다. 다시 시도해주세요.");
        return -1;
    }

    catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

· 유저의 로그인을 진행하는 함수이다. ID와 Password를 각각 입력한 후 user 목록에 존재하면 현재 접속 유저를 해당 ID로 변경, 실패하면 아이디 혹은 비밀번호가 다르다는 문구를 출력한다. 로그인에 성공하면 유저 전용 메뉴로 화면이 변경된다.

(4) 유저 전용 메뉴

```java
public static void User_Menu(){
    while(true){
        System.out.println("========== 사용자 메뉴 ==========");
        System.out.println("0. 내 정보 출력");
        System.out.println("1. 내 동영상 목록");
        System.out.println("2. 동영상 업로드"); //비디오 등록하는 것
        System.out.println("3. 동영상 수정"); //자신이 업로드한 비디오 수정
        System.out.println("4. 동영상 삭제"); //자신이 업로드한 비디오 삭제
        System.out.println("5. 동영상 찾기"); //비디오 탐색
        System.out.println("6. 재생 목록");
        System.out.println("7. 구독 목록");
        System.out.println("8. 시청 기록");
        System.out.println("9. 로그아웃");
        int menu = sc.nextInt();
        switch (menu){
            case 0:
                showUserInf();
                continue;
            case 1:
                showMyVideo();
                continue;
            case 2:
                upload();
                continue;
            case 3:
                user_change();
                continue;
            case 4:
                video_delete();
                continue;
            case 5:
                video_search();
                continue;
            case 6:
                user_playlist();
                continue;
            case 7:
                // 구독목록
                user_sublist();
                continue;

            case 8: // 시청기록
                user_history();
                continue;

            case 9: return;
        }
    }
}
```

· 사용자가 로그인 했을 때의 메뉴들을 출력해주는 함수이다.

(4)-1 유저의 정보 출력

```java
sql = "select * from user where userID = ?";
try {
    pst = connect.prepareStatement(sql);
    pst.setString( parameterIndex: 1, db_user);
    rs = pst.executeQuery();
    while (rs.next()) {
        System.out.println( "아이디: "+ rs.getString( columnIndex: 1) + "\n" +
                "비밀번호: " + rs.getString( columnIndex: 2) + "\n" +
                "성: " + rs.getString( columnIndex: 3) + "\n" +
                "이름: " + rs.getString( columnIndex: 4) + "\n" +
                "가입일 : " + rs.getString( columnIndex: 5)
        );
    }
    System.out.println();
} catch (SQLException e) {
    throw new RuntimeException(e);
}
```

· 현재 접속 유저로 설정한 아이디를 테이블에서 찾아 기본정보(아이디, 비밀번호, 성명, 가입일)를 출력 시킨다.

(4) -2 동영상 업로드 목록 (유저)

```
public static void showMyVideo(){
    System.out.println("---------- 동영상 목록 ----------");
    System.out.println("현재 사용자가 업로드한 동영상 목록을 출력합니다.");
    sql = "select title,videonum from (select * from upload_list join video where ul_num = videonum)V where ul_ID = ?";
    int num = 0;
    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        while(rs.next()){
            num++;
            System.out.print(num + ". "+ rs.getString( columnIndex: 1) +" " + rs.getString( columnIndex: 2));
        }

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

· 접속한 유저가 업로드한 동영상의 목록 (제목, 고유번호)을 출력시킨다.

(4)-3 동영상 업로드 함수

```
public static void upload(){
    Scanner sc = new Scanner(System.in);
    //업로드할 영상의 고유번호는 중복 없이 랜덤번호로 설정된다.
    System.out.println("업로드할 동영상의 제목을 입력하세요.(30자 이내)");
    String title = sc.nextLine();
    System.out.println("동영상의 주제를 입력하세요.(10자 이내)");
    String topic = sc.nextLine();
    System.out.println("동영상에 대한 설명을 입력하세요.(500자 이내)");
    String description = sc.nextLine();
    //TODO 글자수가 넘을 경우를 체크... 안넘으면 안되나요
    String num = getRandomString( size: 5);
    java.util.Date utilDate = new java.util.Date();
    java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());
    try {
        pst = connect.prepareStatement( sql: "insert into video values (?,?,?,?)");
        //사용자의 영상 업로드
        pst.setString( parameterIndex: 1, topic);
        pst.setString( parameterIndex: 2, description);
        pst.setString( parameterIndex: 3, num);
        pst.setString( parameterIndex: 4, title);
        pst.executeUpdate();
        System.out.println("동영상 업로드를 완료하였습니다.");

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
```

```
    try {
        pst = connect.prepareStatement( sql: "insert into upload_list values (?,?,?)");
        pst.setString( parameterIndex: 1, num);
        pst.setString( parameterIndex: 2, db_user);
        pst.setDate( parameterIndex: 3, sqlDate);
        pst.executeUpdate();
    }
     catch(SQLException e) {
        throw new RuntimeException(e);
     }

}
```

· 동영상의 제목, 주제, 설명을 유저가 입력한 후 video 테이블과 유저가 가지고 있는 upload_list에 삽입한다.

(4)-4 동영상 정보 수정 함수

```
public static void user_change(){
    System.out.println("---------- 동영상 수정 ----------");
    System.out.println("현재 사용자가 업로드한 동영상 목록을 출력합니다.");
    String sql = "select title,videonum from (select * from upload_list join video where ul_num = videonum)V where ul_ID = ?";
    int num = 0;
    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        while(rs.next()){
            num++;
            System.out.println("동영상 ["+num +"] "+ rs.getString( columnIndex: 1) +" " + rs.getString( columnIndex: 2));
        }

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }

    Scanner sc = new Scanner(System.in);
    System.out.println("수정할 동영상의 고유번호를 입력해주세요.");
    String v_num = sc.nextLine();
    System.out.println("무엇을 수정하시겠습니까?");
    System.out.println("1.제목     2.주제     3.설명 ");
    int menu = sc.nextInt();
    sc.nextLine();
```

```
switch (menu){
    case 1:
        System.out.println("변경할 제목을 입력하세요.");
        String new_title = sc.nextLine();
        try {
            pst = connect.prepareStatement( sql: "update video set title = ? where videonum = ?");
            pst.setString( parameterIndex: 1, new_title);
            pst.setString( parameterIndex: 2, v_num);
            pst.executeUpdate();
            break;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }

    case 2:
        System.out.println("변경할 주제를 입력하세요.");
        String new_topic= sc.nextLine();
        try {
            pst = connect.prepareStatement( sql: "update video set topic = ? where videonum = ?");
            pst.setString( parameterIndex: 1, new_topic);
            pst.setString( parameterIndex: 2, v_num);
            pst.executeUpdate();
            break;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
```

(이하 생략)

- 유저가 동영상의 정보를 수정할 수 있다. 우선 유저의 업로드 목록을 출력한 후 수정을 원하는 동영상의 고유번호를 입력한다. 그 후 제목, 주제, 설명 중에 무슨 정보를 수정할 것인지 입력하고 수정한다.

(4)-5 동영상 삭제 함수

```
public static void video_delete(){
    System.out.println("---------- 동영상 삭제 ----------");
    System.out.println("현재 사용자가 업로드한 동영상 목록을 출력합니다.");
    String sql = "select title,videonum from (select * from upload_list join video where ul_num = videonum)V where ul_ID = ?";
    int num = 0;
    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        while(rs.next()){
            num++;
            System.out.println("동영상 ["+num +"] "+ rs.getString( columnIndex: 1) +" " + rs.getString( columnIndex: 2));
        }

        if (num == 0){
            System.out.println("업로드한 영상이 없습니다. 메뉴화면으로 돌아갑니다.");
            return;
        }

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
```

- 유저가 업로드한 영상 목록을 출력한 후 삭제할 동영상의 고유번호를 입력하여 삭제시킨다.

(4)-6 동영상 찾기 함수

```
public static void video_search(){
    Scanner sc = new Scanner(System.in);

    while(true){
        System.out.println("---------- 동영상 찾기 ----------");
        System.out.println("찾으시는 동영상과 관련된 키워드를 입력하세요. 해당 키워드를 포함하는 동영상을 나열합니다.");
        String keyword = sc.nextLine();

        sql = "SELECT title, videoNum FROM video WHERE title Like ?";
        int num = 0;
        try {
            pst = connect.prepareStatement(sql);
            pst.setString( parameterIndex: 1,  x: "%"+keyword+"%");
            rs = pst.executeQuery();

            while(rs.next()){
                num++;
                System.out.println("결과 ["+num +"] "+ rs.getString( columnIndex: 1) +" " + rs.getString( columnIndex: 2));
            }
            if (num == 0) {
                System.out.println("해당 키워드를 포함하는 동영상이 존재하지 않습니다.");
                return;
            }
```

```
    else{
        System.out.println("동영상 검색을 완료하였습니다.");
        System.out.println("동영상 재생을 원하시면 동영상의 번호를, 그렇지 않으면 '0'을 눌려주세요.");
        while(true){
        String func = sc.nextLine();
        if(func.equals('0')){
            System.out.println("메뉴화면으로 돌아갑니다.");
            return;
        }
        else {
            sql = "select count(*) from video where videonum = ?";
            try {
                pst = connect.prepareStatement(sql);
                pst.setString( parameterIndex: 1, func);
                rs = pst.executeQuery();
                while (rs.next()) {
                    if (rs.getInt( columnIndex: 1) == 0) {
                        System.out.println("동영상의 고유번호를 잘못 입력하셨습니다. 다시 시도해주세요.");
                    } else {
                        play_Video(func);
                        return; // 재생 후 기능을 이용 후에는 메뉴화면으로 갑니다.
                    }
                }
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }
        }
    }
```

- 유저가 특정 키워드를 입력하면 그 특정 키워드를 포함하는 동영상의 목록을 출력시킨다. 그 후 재생을 원하면 재생 함수를 , 그렇지 않으면 반환한다.

(4)-6-1 동영상 재생 함수

```
public static void play_Video (String num) {
        System.out.println("재생을 시작합니다.");
        System.out.println();
        System.out.println("@@@@@@@@@@" + num+ " 재생중 @@@@@@@@@@");
        System.out.println();
        sql = "select count(*) from history_list where hs_ID = ?";
        try {
            pst = connect.prepareStatement(sql);
            pst.setString( parameterIndex: 1, db_user);
            rs = pst.executeQuery();
            while (rs.next()) {
                if (rs.getInt( columnIndex: 1) == 0) { //시청기록에 아이디가 없는 경우 추가
                    sql = "insert into history_list values (?)";
                    pst = connect.prepareStatement(sql);
                    pst.setString( parameterIndex: 1, db_user);
                    pst.executeUpdate();
                } // 시청기록에 아이디가 있는 경우
                sql = "select count(*) from view_video where view_ID = ? and view_num = ?";
                pst = connect.prepareStatement(sql);
                pst.setString( parameterIndex: 1, db_user);
                pst.setString( parameterIndex: 2, num);
                rs = pst.executeQuery();
                java.util.Date utilDate = new java.util.Date();
                java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());
```

```
while(true) {
    try {
        Thread.sleep( millis: 1000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    System.out.println("원하는 기능을 입력해주세요.");
    System.out.println("1. 좋아요    2. 좋아요 취소   3. 게시자 구독하기   4. 게시자 구독 취소  5. 재생 목록에 추가   6. 재생 종료");
    int func = sc.nextInt();
    switch (func){
        case 1 : like_video(num); break;
        case 2 : dislike_video(num); break;
        case 3 : sql = "select ul_ID from upload_list where ul_num = ?";
                 try {
                     pst = connect.prepareStatement(sql);
                     pst.setString( parameterIndex: 1, num);
                     rs = pst.executeQuery();
                     String n_ID;
                     while(rs.next()) {
                         n_ID = rs.getString( columnLabel: "ul_ID");
                         System.out.println("해당 동영상의 게시자는 " + n_ID + "입니다.");
                         subscribe_(n_ID);

                     }break;
                 } catch (SQLException e) {
                     throw new RuntimeException(e);
                 }
```

(이하 생략)

- 해당 동영상을 재생하고 있다는 문구를 출력한다. 약 1초동안 시스템을 멈추었다가 재생 중 사용자가 원하는 기능 (좋아요, 좋아요 취소, 구독, 구독 취소, 플레이리스트에 추가)

  1. 좋아요 (like_video)

  기존에 좋아요를 누른 데이터가 있으면 이미 눌렀다는 문구를 출력, 그렇지 않으면 진행한다.

  2. 좋아요 취소 (dislike_video)

  기존에 좋아요를 누른 데이터가 있으면 취소, 없으면 좋아요를 누르지 않았다는 문구를 출력

3. 구독 (subscribe_)

재생중이던 동영상의 고유번호를 인자로 넘겨 해당 동영상의 게시자를 출력 후 구독한다.

단, 자기자신이 게시자일 경우 구독은 불가능하다.

4. 구독 취소 (cancel_subscribe)

재생중이던 동영상의 고유번호를 인자로 넘겨 해당 동영상의 게시자를 찾고 이미 구독중이었다면 구독 취소 처리를 , 유저의 구독 목록이 비었다면 해당 문구를 출력한다.

5. 플레이리스트에 추가

해당 사용자의 플레이리스트 목록을 출력하고 추가를 원하는 플레이리스트의 제목을

입력하여 동영상을 추가한다.

(4)-7 재생목록(유저)

```
public static void user_playlist() {
    print_playlist();
    try {
        System.out.println("원하는 기능을 입력해주세요.");
        System.out.println("1. 재생목록 만들기 2. 재생목록 삭제 3. 재생목록 선택 4. 메뉴로 돌아가기");
        int func_ = sc.nextInt();
        sc.nextLine();
        switch (func_) {
            case 1: //재생목록 만들기
                System.out.println("만들 재생목록의 제목을 입력해주세요.");
                String pl_ttl = sc.nextLine();
                try {

                    sql = "insert into playlist values (?,?)";
                    pst = connect.prepareStatement(sql);
                    pst.setString( parameterIndex: 1, db_user);
                    pst.setString( parameterIndex: 2, pl_ttl);
                    pst.executeUpdate();
                    System.out.println("재생목록을 생성하였습니다.");
                    System.out.println();
                    return;
                } catch (SQLException e) {
                    throw new RuntimeException(e);
                }
```

· 현재 가지고 있는 재생목록을 추가한 후 원하는 기능을 입력한다. (재생목록 만들기, 삭제, 선택) 재생목록을 선택할 경우 재생목록 내의 동영상들을 전체 재생, 하나의 영상 재생, 영상 삭제 등을 할 수 있다. 단, 한 유저가 한 재생 목록에 동일한 동영상을 추가하는 것은 금지한다.

(4) – 8 구독 목록 (유저)

```
public static void user_sublist(){
    sql = "select count(*) from subscribe where subscriber = ?";
    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        while (rs.next()) {
            if (rs.getInt( columnIndex: 1) == 0) {
                System.out.println("구독중인 게시자가 없습니다.");
                return;
            } else {
                sql = "select sub_ID from subscribe where subscriber = ?";
                pst = connect.prepareStatement(sql);
                pst.setString( parameterIndex: 1, db_user);
                rs = pst.executeQuery();
                int num = 0;
                while (rs.next()) {
                    num++;
                    System.out.println("["+num +"] "+ rs.getString( columnIndex: 1)
                }
            }
                        }
        System.out.println("원하는 기능을 입력해주세요.");
        System.out.println("1. 게시자 구독 취소    2. 메뉴 화면으로 돌아가기");
        int func = sc.nextInt();
        sc.nextLine();
        if (func == 1){
            System.out.println("구독 취소를 원하는 게시자의 ID를 입력하세요.");
            String cancel_ID = sc.nextLine();
```

- 해당 접속 유저의 구독 목록 (구독은 재생 중에 가능) 을 출력한다. 만약 구독자가 없는 경우는 없다는 문구를 출력하고 있을 경우 특정 구독(하고 있는)자를 더 이상 구독할지 하지 않을지 선택한다.

(4) – 8 시청목록 (유저)

```
public static void user_history(){
    sql = "select count(*) from view_video where view_ID = ?";
    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        while (rs.next()) {
            if (rs.getInt( columnIndex: 1) == 0) {
                System.out.println("시청했던 동영상이 없습니다.");
            } else {
                sql = "select view_Num, view_Date from view_video where view_ID = ?";
                pst = connect.prepareStatement(sql);
                pst.setString( parameterIndex: 1, db_user);
                rs = pst.executeQuery();
                int num = 0;
                while (rs.next()) {
                    num++;
                    System.out.println("["+num +"] "+ rs.getString( columnIndex: 1) + " " + rs.getString( columnIndex: 2));
                }
            }
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

- 시청목록에 데이터가 없는 경우 없다는 문구를 출력하고 있는 경우 시청 동영상의 고유번호와 제목을 출력한다.

(5) 관리자 로그인

```
public static int supervisor_login(){
    System.out.println("---------- 관리자 로그인 ----------");
    System.out.print("아이디: ");
    db_user = sc.next();
    System.out.print("비밀번호: ");
    String Login_Pwd = sc.next();

    sql = "SELECT Password FROM supervisor WHERE super_ID = ?";

    try {
        pst = connect.prepareStatement(sql);
        pst.setString( parameterIndex: 1, db_user);
        rs = pst.executeQuery();
        if (rs.next()) {
            if (rs.getString( columnIndex: 1).contentEquals(Login_Pwd)) {
                System.out.println("로그인에 성공하였습니다!");
                System.out.println();
                return 1;
            } else {
                System.out.println("비밀번호가 다릅니다. 다시 시도해주세요.");
                return 0;
            }
        }
        System.out.println("아이디가 다릅니다. 다시 시도해주세요.");
        return -1;
    }
```

- sql script에서 설정해둔 관리자 ID 와 비밀번호와 동일하면 관리자 로그인에 성공한다.

(6) 관리자 메뉴

```
public static void supervisor_Menu(){
    while(true){
        System.out.println("========= 관리자 메뉴 ==========");
        System.out.println("0. 관리자 정보 출력");
        System.out.println("1. 유저 목록");
        System.out.println("2. 시청 기록 열람"); //비디오 등록하는 것
        System.out.println("3. 업로드 목록 열람");
        System.out.println("4. 로그아웃");
        int menu = sc.nextInt();
        switch (menu){
            case 0:
                showUserInf();
            case 1:
                showUsers();
                continue;
            case 2:
                svr_history();
                continue;
            case 3:
                svr_uploadlist();
                continue;
            case 4:
                return;
        }
    }
}
```

- 관리자는 관리자 정보를 열람, 유저 목록을 출력, 특정 유저의 시청 기록과 업로드 목록을 열람할 수 있다.

4. SQL문 상세 설명

(1) 회원가입

```
( sql: "insert into user values (?,?,?,?,?,?)");
```

- user 테이블에 입력한 아이디와 비밀번호 삽입

(2) 로그인

```
sql = "SELECT Password FROM USER WHERE userID = ?";
```

- user 테이블의 ID가 입력받은 아이디 일 때 password를 받아옴

(3) 유저 정보 출력

```
sql = "select * from user where userID = ?";
```

- user 테이블에서 userID가 현재 접속 유저인 테이블 선택

(4) 유저의 업로드 목록 출력

```
sql = "select title,videonum from (select * from upload_list join video where ul_num = videonum)V where ul_ID = ?";
```

- 업로드한 ID가 해당 접속 유저이고 ul_num과 video_num이 같은 video와 upload_list의 행들을 조인한 테이블에서 title과 videonum을 선택

(5) 유저의 동영상 업로드

```
sql: "insert into video values (?,?,?,?)");
```

- video 테이블에 입력받은 동영상의 정보를 insert, upload_list에 동영상의 번호와 접속 아이디, 날짜를 insert

(6) 유저의 동영상 정보 수정

```
( sql: "update video set title = ? where videonum = ?");
```

- videonum이 선택한 동영상의 번호일 때 video 테이블에 있는 title을 입력 받은 제목으로 변경.

(7) 유저의 동영상 삭제

```
( sql: "delete from video where videoNum = ?");
```

- videonum이 선택한 동영상의 번호인 video 테이블의 row를 삭제

(8) 동영상 찾기

```
sql = "SELECT title, videoNum FROM video WHERE title Like ?";
```

- 키워드를 입력받고 해당 키워드를 포함하는 title이 있는 video 테이블의 row 에서 title과 videonum을 선택

(8)-1 동영상 재생

```
"select count(*) from video where videonum = ?";
```

- 재생을 원하는 동영상의 번호가 있는 video 테이블의 row 개수 파악

```
"select count(*) from history_list where hs_ID = ?";
```

- history table의 hs_ID가 있는 row 개수를 확인한 후 없으면 ID를 추가

```
sql = "select count(*) from view_video where view_ID = ? and view_num = ?";
```

- view_video 에 접속 아이디와 동영상 번호가 동시에 있는 row의 개수 파악

  만약 없다면 그냥 추가, 있다면 시청 날짜를 수정

```
 sql = "select count(*) from video_like where lk_ID = ? and lk_num = ? ";
sql = "insert into video_like values (?,?)";
```

- video_like 테이블에서 ID와 동영상 번호가 동시에 있는 table이 있는지 확인하고 없으면 video_like 테이블에 삽입

```
 sql = "select count(*) from video_like where lk_ID = ? and lk_num = ? ";
sql = "delete from video_like where lk_ID = ? and lk_num = ?";
```

- video_like 테이블에서 ID와 동영상 번호가 동시에 있는 table이 있는지 확인하고 있으면 video_like 테이블에서 삭제

```
sql = "insert into subscribe values (?,?)";
```

- subscribe 테이블에 선택한 동영상의 게시자와 접속 유저 아이디 삽입

```
sql = "delete from subscribe where sub_ID = ? and subscriber = ?";
```

- subscirbe 테이블에서 구독한 사람의 ID 와 구독된 사람의 ID가 동시에 있는 row 삭제

```
sql = "insert into add_video values (?,?,?)";
```

- add_video 테이블에 추가할 재생목록 제목, 접속 유저 아이디, 동영상 고유번호를 삽입

(9) 재생목록

```
sql = "insert into playlist values (?,?)";
```

- playlist에 원하는 재생목록 제목과 접속 아이디 삽입

```
sql = "delete from playlist where pl_title = ?";
```

- playlist에서 원하는 재생목록 제목과 접속 아이디 삭제

```
sql = "select title from video where videonum = (select add_num from add_video where add_Title = ?)";
```

- 추가된 재생목록이 입력한 재생목록인 add_video 테이블의 add_num이 videonum인 video의 제목을 선택

```
sql = "select add_num from add_video where add_title = ?";
pst = connect.prepareStatement(sql);
pst.setString( parameterIndex: 1, title);
rs = pst.executeQuery();
while (rs.next()) {
    play_Video(rs.getString( columnIndex: 1));
```

- 추가한 제목이 입력한 재생목록의 제목인 add_video의 add_num을 선택 후 모두 재생

```
sql = "delete from add_video where add_ID = ? and add_num = ? and add_Title = ? ";
```

- 재생목록 내의 원하는 영상 삭제

(10) 구독목록

```
sql = "select count(*) from subscribe where subscriber = ?";
```

```
sql = "select sub_ID from subscribe where subscriber = ?";
```

- 구독하고 있는 사람이 접속 유저인 row가 존재하는지 확인 후 있다면 누구를 구독하고 있는지 파악

(11) 시청목록

```
sql = "select count(*) from view_video where view_ID = ?";
```

```
sql = "select title,videonum from (select * from view_video join video where view_num = videonum)V where view_ID = ?";
```

- 시청목록이 있는지 확인 후 시청 동영상과 기존 동영상의 고유번호가 같은 row를 join한 테이블에서 view_ID가 접속 아이디 같은 title과 videonum 선택

(12) 관리자의 유저 목록

```
sql = "select count(*) from user";
```

```
sql = "select userID from user where superID = ?";
```

- user의 개수 파악 후 회원가입한 유저가 있으면 관리자 ID가 미리 설정해둔 관리자 ID라면 user의 ID를 출력한다.

```
sql: "delete from user where userID = ?")
```

- userID가 입력한 ID 일시 user 테이블에서 삭제

(13) 관리자의 시청 기록 관리

```
"select view_Num, view_Date from view_video where view_ID = ?";
```

- view_ID가 입력한 ID일시 view_video 테이블에서 view_Num, view_Date 를 파악

```
sql = "delete from view_video where view_ID = ? and view_num = ?";
```

- view_ID와 view_num이 입력한 값일 경우 view_video에서 해당 row 삭제

(14) 관리자의 업로드 목록 관리

```
sql = "select count(*) from upload_list where ul_ID = ?";
```

```
sql = "select title,videonum, ul_Date from (select * from upload_list join video where ul_num = videonum)V where ul_ID = ?";
```

- 특정 유저가 업로드한 영상의 title, videonum, ul_Date를 파악

```
sql = "delete from  upload_list where ul_ID = ? and ul_num = ?";
```

- ul_ID 와 ul_num이 선택한 유저라면 upload_list에서 삭제

5. 실행 화면

```
D:₩2-2₩데베시₩2021037156_탁민주_P2>javac video_db.java

D:₩2-2₩데베시₩2021037156_탁민주_P2>java video_db
Enter ID: root
Enter PW: xkralswn39!
========== Video Site ==========
1. 회원가입
2. 로그인
3. 관리자 로그인
4. 종료
원하는 항목을 고르십시오.
```

자바 파일과 jar 파일을 한 폴더에 두고 컴파일 후 실행시켰습니다.

```
========== Video Site ==========
1. 회원가입
2. 로그인
3. 관리자 로그인
4. 종료
원하는 항목을 고르십시오.
1
---------- 회원가입 ----------
아이디(10자 이내): tmj5574
비밀번호(12자 이내): password
성(10자 이내): tak
이름(10자 이내): minju
```

```
---------- Login ----------
ID: tmj5574
Password: password
로그인에 성공하였습니다!
========= 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
```

```
========== 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
0
아이디: tmj5574
비밀번호: password
성: tak
이름: minju
가입일 : 2022-12-10
```

```
========== 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
1
---------- 동영상 목록 ----------
현재 사용자가 업로드한 동영상 목록을 출력합니다.
1. funny video 77D93
```

```
========== 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
2
업로드할 동영상의 제목을 입력하세요.(30자 이내)
funny video
동영상의 주제를 입력하세요.(10자 이내)
dog
동영상에 대한 설명을 입력하세요.(500자 이내)
this video is funny
동영상 업로드를 완료하였습니다.
```

```
---------- 동영상 수정 ----------
현재 사용자가 업로드한 동영상 목록을 출력합니다.
동영상 [1] funny video 77D93
수정할 동영상의 고유번호를 입력해주세요.
77D93
무엇을 수정하시겠습니까?
1.제목     2.주제     3.설명
1
변경할 제목을 입력하세요.
boring
수정이 완료되었습니다.
========== 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
1
---------- 동영상 목록 ----------
현재 사용자가 업로드한 동영상 목록을 출력합니다.
1. boring 77D93
```

```
---------- 동영상 삭제 ----------
현재 사용자가 업로드한 동영상 목록을 출력합니다.
동영상 [1] boring 77D93
삭제할 동영상의 고유번호를 입력해주세요.
77D93
동영상 삭제가 완료되었습니다.
```

```
---------- 동영상 찾기 ----------
찾으시는 동영상과 관련된 키워드를 입력하세요. 해당 키워드를 포함하는 동영상을 나열합니다.
서가 방금에 마지막으로 저장되었습니다
결과 [1] Cat Video W68V6
동영상 검색을 완료하였습니다.
동영상 재생을 원하시면 동영상의 번호를, 그렇지 않으면 '0'을 눌려주세요.
W68V6
재생을 시작합니다.

@@@@@@@@@@W68V6 재생중 @@@@@@@@@@
```

```
원하는 기능을 입력해주세요.
1. 좋아요    2. 좋아요 취소   3. 게시자 구독하기   4. 게시자 구독 취소  5. 재생 목록에 추가   6. 재생 종료
1
좋아요를 하셨습니다.
```

```
원하는 기능을 입력해주세요.
1. 좋아요    2. 좋아요 취소   3. 게시자 구독하기   4. 게시자 구독 취소  5. 재생 목록에 추가   6. 재생 종료
2
좋아요를 취소하셨습니다.
```

```
원하는 기능을 입력해주세요.
1. 좋아요    2. 좋아요 취소   3. 게시자 구독하기   4. 게시자 구독 취소  5. 재생 목록에 추가   6. 재생 종료
3
해당 동영상의 게시자는 b입니다.
구독을 완료하였습니다.
```

```
원하는 기능을 입력해주세요.
1. 좋아요    2. 좋아요 취소   3. 게시자 구독하기   4. 게시자 구독 취소  5. 재생 목록에 추가   6. 재생 종료
4
해당 동영상의 게시자는 b입니다.
구독을 취소하였습니다.
```

```
6
소유하고 있는 플레이리스트가 없습니다.
원하는 기능을 입력해주세요.
1. 재생목록 만들기 2. 재생목록 삭제 3. 재생목록 선택 4. 메뉴로 돌아가기
1
만들 재생목록의 제목을 입력해주세요.
playlist1
재생목록을 생성하였습니다.
```

```
6
재생목록 [1] playlist1
원하는 기능을 입력해주세요.
1. 재생목록 만들기 2. 재생목록 삭제 3. 재생목록 선택 4. 메뉴로 돌아가기
3
원하는 재생 목록을 선택해주세요.
playlist1
재생목록 내에 추가된 영상이 없습니다.
```

```
========== 사용자 메뉴 ==========
0. 내 정보 출력
1. 내 동영상 목록
2. 동영상 업로드
3. 동영상 수정
4. 동영상 삭제
5. 동영상 찾기
6. 재생 목록
7. 구독 목록
8. 시청 기록
9. 로그아웃
8
[1] cat video W68V6
```

```
삭제할 유저의 아이디를 입력하세요. 종료를 원하시면 '0'을 입력하세요.
mj5574
유저 삭제가 완료되었습니다.
========== 관리자 메뉴 ==========
0. 관리자 정보 출력
1. 유저 목록
2. 시청 기록 열람
3. 업로드 목록 열람
4. 로그아웃

모든 유저의 정보들을 출력합니다.
[1] a
[2] b
[3] h
[4] ?????
[5] ???
유저 삭제를 원하시면 1, 종료를 원하면 0을 입력하세요.
```

```
========== 관리자 메뉴 ==========
0. 관리자 정보 출력
1. 유저 목록
2. 시청 기록 열람
3. 업로드 목록 열람
4. 로그아웃
2
모든 유저의 정보들을 출력합니다.
[1] a
[2] b
[3] h
[4] ?????
[5] ???
시청기록을 열람할 유저의 아이디를 입력하세요. 종료를 원하시면 '0'을 입력하세요.
a
시청했던 동영상이 없습니다.
```

```
========== 관리자 메뉴 ==========
0. 관리자 정보 출력
1. 유저 목록
2. 시청 기록 열람
3. 업로드 목록 열람
4. 로그아웃
3
모든 유저의 정보들을 출력합니다.
[1] a
[2] b
[3] h
[4] ?????
[5] ???
업로드 목록을 열람할 유저의 아이디를 입력하세요. 종료를 원하시면 '0'을 입력하세요.
a
업로드한 동영상이 없습니다.
```