**데이터베이스**

**신효섭 교수님**

**<기말 프로젝트 선행 과제>**

**네이버 현재 상영 영화 정보 크롤링**

컴퓨터공학부

201911278

정 경 은

**목차**

1. 과제 개요

2. 소스 코드

3. 프로그램 수행 결과 snapshot

4. 요약 및 결론

**1. 과제 개요**

텍스트이(가) 표시된 사진

자동 생성된 설명 과제는 네이버 영화 사이트의 현재 상영 영화의 모든 데이터를 크롤링하는 과제이다. 모든 영화 정보의 포맷은 아래와 같다.

각 영화마다 총 11개의 정보를 가져온다. 그것은 각각 title (영화 제목), movie\_rate (영화 등급), netizen\_rate (네티즌 평점), netizen\_count (네티즌 평점 참여자 수), journalist\_score (기자평론가 평점), journalist\_count (기자평론가 참여자 수), scope (개요), playing\_time (상영 시간), opening\_date (개봉 날짜), director (감독), image (영화 대표 이미지 주소)이며, 이 11개는 naverMovie table의 애트리뷰트 명이 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명영화 제목은 그 길이가 다양하여 최대 길이를 길게 하여 varchar를 이용하고, 영화 등급은 15세 이상, 전체 관람가 등 길이 않기 때문에 최대 길이를 10으로 한다. 네티즌 평점이나 기자평론가 평점은 소수점 두 자리까지 나타나기 때문에 float형으로, 네티즌/기자평론가 참여자 수는 인원 수만 나타내면 되므로 int형, 개요는 영화에 따라 여러 가지가 존재할 수 있으므로 varchar형을 이용하였다. 영화 상영 시간은 몇 분 간 상영되는지 정수로 나타낼 수 있으므로 int형, 개봉 날짜는 년, 월, 일로 나타낼 수 있어 datetime으로, 감독은 영화마다 차이가 크기 때문에 varchar로 하였다. 대표 이미지 링크의 경우 긴 주소가 저장되므로 varchar로 하되, 최대 길이를 크게 설정하였다. 기본 키를 설정하기 위하여 임의의 int형 id값을 auto\_increment로 애트리뷰트를 추가하였으며, 각 투플, 즉, 영화 정보가 언제 저장되었는지는 datatime default now()를 사용하였다. 이를 반영한 create table sql문은 아래와 같다.

(각 정보를 추출한 방법은 아래 소스 코드에 각 애트리뷰트에 대한 주석과 함께 첨부합니다.)

**2. 소스코드**

(user 이름과 password는 임의의 값으로 수정하여 제출합니다.)

*import* pymysql

*from* selenium *import* webdriver

*from* selenium.webdriver.common.by *import* By

*import* requests *as* rq

*from* bs4 *import* BeautifulSoup

*# db 연결*

def open\_db():

    conn = pymysql.connect(host='localhost', user='user',

                           password='password', db='naverMovie')

    cur = conn.cursor(pymysql.cursors.DictCursor)

*return* conn, cur

*# 네이버 현재 상영 영화 정보 크롤링 함수*

def movieCrawling():

    conn, cur = open\_db()

*# 네이버 현재 상영 영화 페이지 url*

    url = "https://movie.naver.com/movie/running/current.naver"

*# BeautifulSoup 사용하여 위 url 페이지의 정보 읽어오기*

    soup = BeautifulSoup(urllib.request.urlopen(url).read(), 'html.parser')

*# 현재 상영 영화 페이지의 각 영화 정보 가져오기*

    data = soup.select(

        '#content > div.article > div.obj\_section > div.lst\_wrap > ul > li')

    cnt = 0

*# 각 영화에 대하여 11개 애트리뷰트 값 추출*

*for* tempData *in* data:

        cnt += 1

*# 프로그램 실행 시 터미널에서의 결과 확인을 위한 출력*

        print("{0} 번째 영화 :".format(cnt))

        print()

*# 1. title: 영화 제목*

        titleData = tempData.select\_one('dl > dt > a')

        title = titleData.string

        print("title:", title)

*# 2. movie\_rate: 영화 등급*

        movie\_rateData = tempData.select\_one('dl > dt > span')

*if* movie\_rateData is not None:

            movie\_rate = movie\_rateData.string

*else*:

*# 영화 등급 데이터가 존재하지 않는 경우*

            movie\_rate = None  *# null로 넣기*

        print("movie\_rate:", movie\_rate)

*# 3. netixen\_rate: 네티즌 평점*

        netizen\_rateData = tempData.select\_one(

            'dl > dd.star > dl > dd:nth-child(2) > div > a > span.num')

*if* netizen\_rateData is not None:

            netizen\_rate = float(netizen\_rateData.string)

*else*:

*# 네티즌 평점 데이터가 존재하지 않는 경우*

            netizen\_rate = None  *# null로 넣기*

        print("netizen\_rate:", netizen\_rate)

*# 4. netizen\_count: 네티즌 평점 참여자 수*

        netizen\_count = tempData.select\_one(

            'dl > dd.star > dl > dd > div > a > span.num2 > em')

*if* netizen\_count is not None:

            temp = netizen\_count.string

            netizen\_count = temp.replace(",", "")

*else*:

*# 네티즌 평점 참여자 수 데이터가 존재하지 않는 경우*

            netizen\_count = 0  *# 네티즌 평점이 null인 경우 \_ 참여자 0명*

        print("netizen\_count:", netizen\_count)

*# 5. journalist\_score: 기자 평론가 평점*

        journalist\_scoreData = tempData.select\_one(

            'dl > dd.star > dl > dd:nth-child(4) > div > a > span.num')

*if* journalist\_scoreData is not None:

            journalist\_score = float(journalist\_scoreData.string)

*else*:

*# 기자평론가 평점 데이터가 존재하지 않는 경우*

            journalist\_score = None  *# null로 넣기*

        print("journalist\_score:", journalist\_score)

*# 6. journalist\_count: 기자 평론가 참여자 수*

        journalist\_count = tempData.select\_one(

            'dl > dd.star > dl > dd:nth-child(4) > div.star\_t1 > a > span.num2 > em')

*if* journalist\_count is not None:

            temp = journalist\_count.string

            journalist\_count = temp.replace(",", "")

*else*:

*# 기자평론자 참여자 수 데이터가 존재하지 않는 경우*

            journalist\_count = 0  *# 기자 평론가 평점이 null인 경우 \_ 참여자 0명*

        print("journalist\_count:", journalist\_count)

*# 7. scope: 개요*

        scope = ""

        scopeList = tempData.select(

            'dl > dd:nth-child(3) > dl > dd:nth-child(2) > span.link\_txt > a')

*if* scopeList is not None:

*for* temp *in* scopeList:

                scope += temp.string + ", "

            scope = scope[:-2]

*else*:

*# 개요 데이터가 존재하지 않는 경우*

            scope = None  *# null로 넣기*

        print("scope:", scope)

*# 8. playing\_time: 상영 시간*

        playing\_timeData = tempData.select\_one(

            'dl > dd:nth-child(3) > dl > dd:nth-child(2)')

        count = 0

        playing\_time = int(list(playing\_timeData)[4].string.strip()[:-1])

        print("playing\_time:", playing\_time)

*# 9. opening\_date: 개봉 날짜*

        opening\_dateData = tempData.select\_one(

            'dl > dd:nth-child(3) > dl > dd:nth-child(2)')

        opening\_date = list(opening\_dateData)[6].string.strip()[:-3]

        print("opening\_date:", opening\_date)

*# 10. director: 감독*

        director = ""

        directorList = tempData.select(

            'dl > dd:nth-child(3) > dl > dd:nth-child(4) > span > a')

*if* directorList is not None:

*for* temp *in* directorList:

                director += temp.string + ", "

            director = director[:-2]

*else*:

*# 감독 데이터가 존재하지 않는 경우*

            director = None  *# null로 넣기*

        print("director:", director)

*# 11. image: 영화 대표 이미지 주소*

        image = tempData.select(

            'div > a > img')[0]['src']

        print(image)

*# 터미널에 데이터 출력시 영화 간 구분*

        print()

        print()

        print("--------------------------------------------------------------")

        print()

        print()

*# movie 테이블에 추출한 값을 넣는 insert문*

        insert\_sql = """

            insert into movie(title, movie\_rate, netizen\_rate, netizen\_count, journalist\_score, journalist\_count, scope, playing\_time, opening\_date, director, image)

            values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)

        """

*# sql 문 수행, 각 데이터 매칭*

        cur.execute(insert\_sql, (title, movie\_rate, netizen\_rate, netizen\_count,

                    journalist\_score, journalist\_count, scope, playing\_time, opening\_date, director, image))

*# 결과 반영*

        conn.commit()

    driver.close()

    driver.quit()

    close\_db(conn, cur)

*# db 연결 닫기*

def close\_db(conn, cur):

    cur.close()

    conn.close()

*# main 에서 크롤링 함수 실행*

*if* \_\_name\_\_ == '\_\_main\_\_':

    movieCrawling()

**3. 프로그램 수행 결과 snapshot**

- python 프로그램 수행 (값을 출력하도록 코드를 작성 후, 일부만 캡쳐하였습니다.)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

- 워크 벤치에서 select \* from naverMoive; 결과

텍스트, 실내, 스크린샷, 장식이(가) 표시된 사진

자동 생성된 설명

(중략)

테이블이(가) 표시된 사진

자동 생성된 설명

**4. 요약 및 결론**

Beautiful Soup를 이용하여 현재 상영 중인 모든 영화 정보를 가져와, movie 테이블에 저장하였다.

데이터를 저장하는 과정에서 일부 애트리뷰트에 해당하는 데이터 값의 존재 유무에 따른 예외 처리를 해주었다. 예를 들어, 영화 등급(ex. 전체 관람가) 데이터가 존재하지 않는 경우 python에서 null을 나타내는 None 값을 저장하여, database에 null 값으로 insert 되게 하였다. 또, 네티즌 또는 기자평론가 평점의 경우에도 마찬가지로 None 값을 통해 null 값으로 저장하였으며, 이 경우 참여자 수는 0으로 저장하였다.

처음 코드를 작성했을 때는 셀레니움을 사용하였다. Beautiful Soup은 동적 페이지의 정보를 가져올 때 사용하고, 셀레니움은 정적 페이지는 물론이고, 동적 페이지의 정보를 크롤링할 때 사용한다. 셀레니움 사용 여부에 따라 방대한 양의 데이터를 크롤링함에 있어서 속도의 차이가 큰데, Beautiful Soup만 하용하면 셀레니움을 사용할 때보다 속도가 월등히 빠르다. 이를 고려하여 셀레니움을 사용하지 않고 네이버의 현재 상영 영화 정보를 크롤링하도록 수정한 바 있다.

또, 교수님의 조언에 따라 execute가 아닌 executemany를 사용하여 database에 값을 추가하도록 코드를 수정할 것이다.