[DB Application Homework]

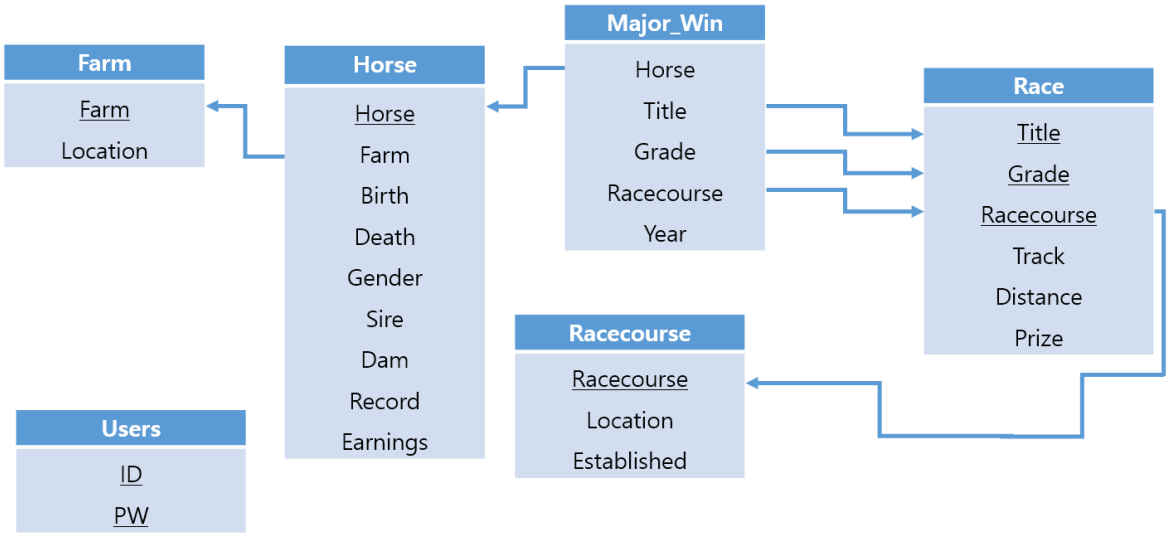
2017320139 서형진

JRA DataBase

1. 소개

JRA는 Japan Racing Association의 약자로 일본 경마 협회를 의미합니다. 과제로 제출하는 DB에는 JRA에 소속된 말과 목장, 경기, 경기장, 각 말의 주요 승리 경기에 대한 내용을 담고 있습니다.

2. Database Scheme



여기서 Horse Relation의 Sire와 Dam은 각각 부모의 이름을 뜻합니다.

3. 구현된 함수들

```
@app.route('/', methods = ["GET", "POST"])
def login():
    if request.method == "POST" :
        request_type = request.form['login_type']

        if request_type == "register" :
            new_id = request.form['id']
            new_pw = request.form['password']
            cur.execute("INSERT INTO Users VALUES ('{}', '{}')".format(new_id, new_pw))

        elif request_type == "update" :
            id = request.form['id']
            new_pw = request.form['password']
            cur.execute("UPDATE Users SET PW = CASE WHEN ID = '{}' THEN '{}' ELSE PW END".format(id, new_pw))

        elif request_type == "delete" :
            id = request.form['id']
            cur.execute("DELETE FROM Users where ID = '{}'".format(id))

    return render_template("login.html")
```

첫 페이지인 로그인 페이지를 불러옵니다. 로그인 페이지에서는 “계정 생성”, “비밀번호 변경”, “계정 삭제” 3 가지 기능을 할 수 있습니다. 이를 각각 INSERT, UPDATE, DELETE sql query 로 구현했습니다.

```
@app.route('/main', methods = ["GET", "POST"])
def main():
    #id = request.form["id"]
    #pw = request.form["password"]

    #cur.execute("SELECT ID FROM Users WHERE ID = '{}' and PW = '{}'".format(id, pw))
    #user = cur.fetchall()

    return render_template("main.html")
```

메인 페이지를 불러오는 함수입니다. 주석 부분은 본래 DB 에 저장된 ID 와 PW 를 이용해서 로그인 기능을 구현하려 했으나 역량 부족으로 실패한 내용입니다.

```
@app.route('/horses')
def print_horses_info():
    cur.execute("SELECT horse.horse FROM horse ORDER BY horse.horse;")
    result = cur.fetchall()

    return render_template("horses.html", horses_info = result)
```

DB 에 저장된 모든 말들의 정보를 표시하는 페이지를 불러오는 함수입니다. SELECT 를 이용해 DB 로부터 모든 말의 이름을 가져옵니다.

```python
@app.route('/horse_info', methods = ['POST', 'GET'])
def show_horse_info() :
    horse_name = request.form['horse']
    horse_info = horse_name
    cur.execute("SELECT farm.farm, farm.location FROM farm, horse WHERE horse.horse = '{}' and horse.farm = farm.farm;".format(horse_name))
    farm_info = cur.fetchall()
    cur.execute("SELECT horse.birth FROM horse WHERE horse.horse = '{}';".format(horse_name))
    birth_info = cur.fetchall()
    cur.execute("SELECT horse.death FROM horse WHERE horse.horse = '{}';".format(horse_name))
    dead_info = cur.fetchall()
    cur.execute("SELECT horse.gender FROM horse WHERE horse.horse = '{}';".format(horse_name))
    gender_info = cur.fetchall()
    cur.execute("SELECT horse.sire FROM horse WHERE horse.horse = '{}';".format(horse_name))
    sire_info = cur.fetchall()
    cur.execute("SELECT horse.dam FROM horse WHERE horse.horse = '{}';".format(horse_name))
    dam_info = cur.fetchall()
    cur.execute("SELECT horse.record FROM horse WHERE horse.horse = '{}';".format(horse_name))
    record_info = cur.fetchall()

    cur.execute("SELECT horse.earnings FROM horse WHERE horse.horse = '{}';".format(horse_name))
    earned_info = cur.fetchall()
    earned_info = str(earned_info[0])
    characters = "(,)"
    for x in range(len(characters)) :
        earned_info = earned_info.replace(characters[x], "")
    earned_info = int(earned_info)

    cur.execute("SELECT title, year, racecourse, grade, distance, prize FROM major_win NATURAL JOIN race WHERE horse = '{}' ORDER BY year ASC;".format(horse_name))
    win_info = cur.fetchall()

    image_path = "image/" + horse_info + ".jpg"

    return render_template("horse_info.html", horse=horse_info, farm=farm_info[0], birth=birth_info[0], dead=dead_info[0], gender=gender_info[0],
            sire=sire_info[0], dam=dam_info[0], record=record_info[0], earned=earned_info, win=win_info, image_path = image_path)
```

모든 말의 이름을 보여주는 페이지에서 말의 이름 옆에 있는 버튼을 클릭하면 말에 대한 상세 정보가 표시되는 페이지를 불러오는 함수입니다. 말에 대한 모든 정보를 SELECT 를 이용해 가져오고 있습니다. 또한 농장 정보를 말의 이름을 이용해 가져오기 위해 horse relation 과 farm relation 을 Cartesian product 하고 있으며, 각 말이 승리한 경기 이름과 년도를 major_win relation 에서 가져오기 위해 Natural Join 이 사용되었습니다.
참고로 말의 상세 정보가 적혀있는 페이지에서 테이블 중 Won Match 의 경기 이름에 마우스를 올리면, 해당 경기와 관련된 정보(경기장, 등급, 거리, 1 등 상금)이 팝업으로 나오는 것을 확인하실 수 있습니다.

```python
@app.route('/farms', methods = ["GET"])
def print_farms() :
    cur.execute("SELECT farm.farm, farm.location, (SELECT COUNT(horse.horse) FROM horse WHERE horse.farm = farm.farm) FROM farm;")
    farms_info = cur.fetchall()

    return render_template("farms.html", farms_info = farms_info)

@app.route('/farm_info', methods = ["GET", "POST"])
def print_farm_info() :
    farm_name = request.form['farm']
    cur.execute("SELECT horse.horse FROM horse WHERE horse.farm = '{}';".format(farm_name))
    horses = cur.fetchall()
    cur.execute("SELECT location FROM farm WHERE farm = '{}';".format(farm_name))
    loc = cur.fetchall()

    return render_template("farm_info.html", horses = horses[0], location=loc[0])
```

모든 농장의 이름을 보여주는 페이지를 불러오는 함수와, 농장 이름 옆의 버튼을 클릭하면 농장에 대한 정보와 해당 농장 출신 말의 이름을 보여주는 페이지를 불러오는 함수입니다, SELECT 를 이용해 DB 로부터 정보를 가져오고 있습니다. 또한 농장의 정보 중 해당 농장 출신 말의 수를 보여주기 위해 COUNT aggregation function 과 nested query 가 사용되고 있습니다.

```python
@app.route('/racecourses', methods = ["GET"])
def print_racecourses() :
    cur.execute("SELECT * FROM racecourse")
    racecourses_info = cur.fetchall()

    return render_template("racecourses.html", racecourses_info = racecourses_info)

@app.route('/racecourse_popup', methods = ["POST", "GET"])
def print_racecourse_popup() :
    course_name = request.form["formData"]

    cur.execute("SELECT race.title FROM race WHERE race.racecourse = '{}';".format(course_name))
    races = cur.fetchall()

    return render_template("racecourse_popup.html", races=races, course_name=course_name)
```

모든 경기장의 이름을 보여주는 페이지를 불러오는 함수와, 경기장 이름 옆의 버튼을 클릭하면 해당 경기장에서 개최하는 경기 이름을 보여주는 페이지를 불러오는 함수입니다, SELECT 를 이용해 DB 로부터 정보를 가져오고 있습니다.

```python
@app.route('/races', methods = ["GET"])
def print_races() :
    cur.execute("SELECT * FROM race ORDER BY grade DESC;")
    races_info = cur.fetchall()

    return render_template("races.html", races_info=races_info)

@app.route('/race_winner', methods = ["GET", "POST"])
def print_race_winner() :
    race_name = request.form["race"]

    cur.execute("SELECT horse, year FROM major_win WHERE title = '{}' ORDER BY year ASC;".format(race_name))
    winners = cur.fetchall()

    return render_template("race_winner.html", winners=winners, race_name=race_name)
```

모든 경기의 이름을 보여주는 페이지를 불러오는 함수와, 경기 이름 옆의 버튼을 클릭하면 해당 경기에서 승리한 말의 이름과 연도를 보여주는 페이지를 불러오는 함수입니다, SELECT 를 이용해 DB 로부터 정보를 가져오고 있습니다.

```python
@app.route('/data_modify', methods = ["POST", "GET"])
def print_data_modify() :
    #password = request.form["password"] => 작동을 안함...

    #if password == "admin" :
    #    return render_template("data_modify.html")
    #else :
    #   return redirect(url_for("main"))

    return render_template("data_modify.html")
```

위 함수는 일종의 admin password 를 받아서 DB 를 변경할 수 있도록 작동하기 위해 만든 함수이지만, 이상하게도 html 의 form 을 통해서 password 의 전달이 정상적으로 진행되지 않는 것을 확인했습니다. 따라서 이 함수는 사실상 구현이 되지 않은 것이니, 넘어가시면 됩니다.

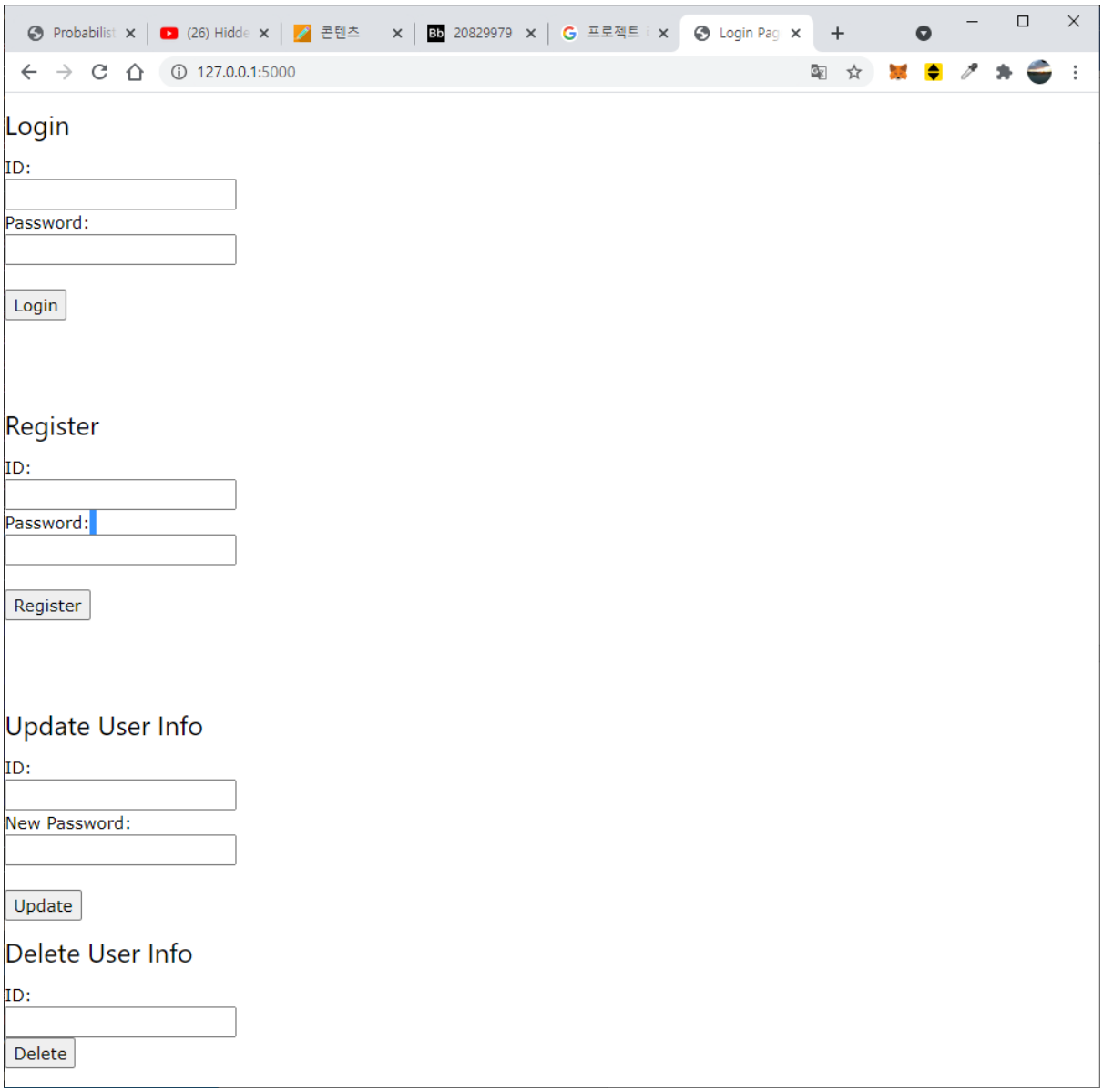
```python
@app.route('/register', methods = ["GET", "POST"])
def print_register() :
    job_type = request.form['job_type']

    return render_template("register.html", job_type = job_type)
```

메인 페이지에서 admin 으로 로그인하면 INSERT, UPDATE, DELETE 3 가지 작업을 고를 수 있습니다. form 을 통해 어떤 작업을 선택했는지 정보를 받아오고, render template 를 이용해 어떤 작업을 선택했는지 알려주는 역할을 하는 함수입니다.

```python
@app.route('/register_progress', methods=["POST", "GET"])
def register_progress() :
    job_type = request.form['job_type']

    if job_type == "insert_farm" :
        farm_name = request.form['farm_name']
        farm_location = request.form['farm_location']
        cur.execute("INSERT INTO farm VALUES ('{}', '{}')".format(farm_name, farm_location))
        connect.commit()

    elif job_type == "insert_horse" :
        horse_image = request.files.get('horse_image', None)
        horse_name = request.form['horse_name']
        horse_farm = request.form['horse_farm']
        horse_birth = request.form['horse_birth']
        horse_death = request.form['horse_death']
        horse_gender = request.form['horse_gender']
        horse_sire = request.form['horse_sire']
        horse_dam = request.form['horse_dam']
        horse_record = request.form['horse_record']
        horse_earnings = request.form['horse_earnings']
        cur.execute("INSERT INTO horse VALUES ('{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}', {})".format(horse_name, horse_farm, horse_birth, horse_d
        connect.commit()
        if horse_image :
            path = secure_filename(horse_image.filename)
            horse_image.save(os.path.join(app.config['UPLOAD_FOLDER'], path))

    elif job_type == "insert_win" :
        win_horse = request.form['win_horse']
        win_title = request.form['win_title']
        win_grade = request.form['win_grade']
        win_racecourse = request.form['win_racecourse']
        win_year = request.form['win_year']
        cur.execute("INSERT INTO major_win VALUES ('{}', '{}', '{}', '{}', '{}')".format(win_horse, win_title, win_grade, win_racecourse, win_year))
```

DB 에 말의 정보를 추가/업데이트/삭제 하거나, 농장 정보를 추가/삭제 하거나, 주요 승리 정보를 추가하는 기능을 가진 함수입니다. Register 페이지로부터 어떤 작업을 하는지 form 을 통해 정보를 받아오고, 각 작업에 따라 분기문을 작성해 요구한 작업을 수행하는 함수입니다. 추가에는 INSERT, 업데이트에는 UPDATE, 삭제에는 DELETE 를 사용하고 있습니다.

4. Application 작동 사진



첫 로그인 페이지입니다. User table 을 통해 계정이 관리됩니다. 하지만 로그인 기능은 구현되지 않았기 때문에, Login 버튼을 누르면 바로 메인 페이지로 이동합니다.

Hello!

Horse List

Race Course List

Race List

Farm List

Password for modify data :

Login as Admin

메인 페이지입니다. 모든 다른 페이지에서 Back to Main Page 버튼으로 이 페이지로 돌아올 수 있도록 설계되어 있으며, 위에서부터 4 개의 버튼은 각각 말, 경기장, 경기, 농장 목록을 볼 수 있는 페이지로 이동합니다.

Horse List

Name	
aaa	more info
Agnes Tachyon	more info
Air Groove	more info
Gold Ship	more info
Haru Urara	more info
Mejiro McQueen	more info
Rice Shower	more info
Silence Suzuka	more info
Special Week	more info
Symboli Rudolf	more info
Tokai Teio	more info
Twin Turbo	more info

Name	Agnes Tachyon
Farm	Shadai Farm
Birth	1998_04_13
Dead	2009_06_22
Gender	male
Sire	Sunday Silence
Dam	Agnes Flora
Record	4_4 (Total_Win)
Earned	222082000 Yen
Won Match	**Race** **Year** Radio Tampa Hai Sansai Stakes 2000 Yayoi Sho 2001 Satsuki Sho 2001

To get Race Information, Hover your mouse cursor on the race name

Back to Main Page

Back to Horse List

말의 목록을 보여주는 페이지입니다. 맨 위의 aaa 는 제가 사이트를 통해서 임의로 추가한 내용입니다. More info 버튼을 누르면 해당 말의 자세한 정보를 볼 수 있습니다.

More info 를 통해 들어간 사이트에서는 다시 말 리스트 페이지로, 그리고 메인 페이지로 돌아갈 수 있습니다.

참고로 Won Match 행의 경기 이름 위에 마우스를 올리면, 해당 경기와 관련된 정보(경기장, 등급, 거리, 1 등 상금)을 보실 수 있습니다.

Racecourse List

Racecourse	Location	Established Year
Tokyo	Japan 183-0024 Tokyo, Fuchu, Hiyoshicho, 穐묅닋1	1933
Kyoto	612-8265 Kyoto, Fushimi Ward, Yoshijima Watashibajimacho, 32	1925
Nakayama	1 Chome-1-1 Kosaku, Funabashi, Chiba 273-0037 Japan	1907
Hanshin	1-1 Komanocho, Takarazuka, Hyogo 665-0053 Japan	1949
Sapporo	16 Chome-1-1 Kita 16 Jonishi, Chuo Ward, Sapporo, Hokkaido 060-0016 Japan	1907
Hakodate	12-2 Komabacho, Hakodate, Hokkaido 042-0935 Japan	1896
Fukushima	9-23 Matsunamicho, Fukushima, 960-8114 Japan	1918
Niigata	3490 Sasayama, Kita Ward, Niigata, 950-3301 Japan	1965
Chukyo	Shikita-1225 Magomecho, Toyoake, Aichi 470-1132 Japan	1994
Kokura	4 Chome-5-1 Kitagata, Kokuraminami Ward, Kitakyushu, Fukuoka 802-0841 Japan	1994

Back to Main Page

Races from Tokyo

- Tenno Sho (Fall)
- Japan Cup
- Japanese Oaks
- Japanese Derby
- Mainichi Okan
- Kyodo News Service Hai혻

경기장 목록과, 이름을 클릭하면 해당 경기장에서 진행하는 경기 이름을 보실 수 있습니다.

Back to Racecourse List Page

Back to Main Page

Race List

Title	Grade	Racecourse	Track Type	Distance	1st Prize	Who won this race?
Mermaid Stakes	G3	Hanshin	Handicap/Turf	2000m	36000000 Yen	more info
Yayoi Sho	G3	Nakayama	Turf	2000m	52000000 Yen	more info
All Comers	G3	Nakayama	Turf	2200m	67000000 Yen	more info
Kokura Daishoten	G3	Kokura	Turf	1800m	41000000 Yen	more info
Kyodo News Service Hai휴	G3	Tokyo	Turf	1800m	38000000 Yen	more info
Tulip Sho	G3	Hanshin	Turf	1600m	52000000 Yen	more info
Kisaragi Sho	G3	Chukyo	Turf	2000m	38000000 Yen	more info
Kisaragi Sho	G3	Kyoto	Turf	2000m	38000000 Yen	more info
Tanabata Sho	G3	Fukushima	Handicap/Turf	2000m	41000000 Yen	more info

---

Tenno Sho (Spring) Winners

Horse	Year
Symboli Rudolf	1985
Mejiro McQueen	1991
Mejiro McQueen	1992
Rice Shower	1993
Rice Shower	1995
Special Week	1999
Gold Ship	2015

모든 경기 리스트와, 해당 경기에서 승리한 전적이 있는 말과 년도에 대한 정보를 볼 수 있습니다.

Back to Race List

Back to Main Page

Farm List

Farm	Location	Horses_Num	To get Horse Info, Click button
Symboli Stud	Hidaka-Cho, Hokkaido, Japan, JPN	1	horses info
Shadai Farm	Abira, Hokkaido, JPN	2	horses info
Deguchi Bokujo	Hidaka, Hokkaido, JPN	1	horses info
Nagahama Bokujo	Niikappu, Hokkaido, JPN	1	horses info
Katashi Yoshida	null	1	horses info
Inahara Bokujo	Biratori, Hokkaido, JPN	1	horses info
Hidaka Taiyo Bokujo	Hidaka-Cho, Hokkaido, JPN	1	horses info
Utopia Bokujo	Noboribetsu-Shi, Hokkaido, JPN	1	horses info
Nobuta Bokujo	null	1	horses info
Toshihiro Fukuoka	null	1	horses info
aaa	aaa	1	horses info

Back to Main Page

Location : Abira, Hokkaido, JPN

모든 농장 리스트와 해당 농장 출신의 말이 몇 마리인지, 그리고 이름은 무엇인지 알 수 있습니다.

Horses from this farm

Air Groove	more info

Back to Farm List

Back to Main Page

Password for modify data :

Login as Admin

Login as Admin 을 통해 DB 내용을 바꿀 수 있습니다.

Select your job

위의 버튼을 누르면 이 화면이 나오고, INSERT/DELETE/UPDATE 작업을 선택할 수 있습니다.

INSERT

DELETE

UPDATE

Back to Main Page

Farm

Farm Name	
Location	

Submit

Horse

Horse Image (Only .jpg & File name should be same as Horse name)	파일 선택 선택된 파일 없음
Horse Name	
Farm	
Birth	
Death	
Gender	
Sire	
Dam	
Record	
Earnings	

Submit

Major Win

Horse	
Title	
Grade	
Racecourse	
Year	

Submit

INSERT 작업은 농장, 말, 주요 승리 경기 relation 에 새로운 tuple 을 넣을 수 있습니다. 경기와 경기장 같은 경우 추가/삭제/변경이 일어나는 경우가 없다고도 볼 수 있기 때문에 이 DB application 에서는 구현을 제외했습니다.

Farm

Farm Name	

Submit

Horse

Horse Name	

Submit

Back to Main Page

DELETE 작업은 Farm 과 Horse relation 의 한 tuple 을 각각 이름을 이용하여 삭제할 수 있습니다. 이때 Horse 의 farm 은 Farm relation 을 참조하고 있다는 것을 고려해서 삭제해야 합니다.

Horse

Horse Name	
New Death	
New Record	
New Earnings	

Submit

Back to Main Page

말을 제외한 다른 relation 의 경우 update 가 일어나지 않습니다. 따라서 Horse, 그 중에서도 update 가 일어나는 내용인 Death, Record, Earnings (죽은 날짜, 전적, 총 상금)만을 update 할 수 있습니다.

원하는 말의 이름으로 검색 후 값을 update 할 수 있습니다.

* 오류 해결 내용

바이트로 조합된 문자(인코딩: "UHC")와 대응되는 문자 코드가 "UTF8" 인코딩에는 없습니다

```
postgres=# drop database jra;
DROP DATABASE
postgres=# create database jra with encoding = 'UTF8';
CREATE DATABASE
postgres=#
```

이걸로 해결했습니다.