# DB_Final Project_Team22

## Main idea：

Our project aims to design and develop a web application based on Olympic historical data. By utilizing the “126 Years of Olympic History” dataset provided by Kaggle as our data source, we will create an intuitive, fully featured query platform. Users can easily access Olympic-related information such as athletes, countries, events, and medal statistics. Moreover, the platform will include functions to add, update, and delete data, meeting the diverse needs of users. By integrating a SQL-backed database with a front-end built using HTML and PHP, the system will effectively manage and display information, enabling users to quickly retrieve the data they require.

## Data

Table：

| | Athlete_ Biography | Athlete _Event _Details | Country_ Profile | Event_ Results | Games_ Summary | Medal_T allt_Hist ory |
|---|---|---|---|---|---|---|
| Athlete_id | ✓ | ✓ | | | | |
| Country | ✓ | | ✓ | | ✓ | ✓ |
| Noc | ✓ | ✓ | ✓ | | ✓ | ✓ |
| Edtion | | ✓ | | ✓ | ✓ | ✓ |
| Edition_id | | ✓ | | ✓ | ✓ | ✓ |
| Result_id | | ✓ | | ✓ | | |
| Sport | | ✓ | | ✓ | | |
| Year | | | | | ✓ | ✓ |
| | Name | Event | | Event_ title | Edition_url | Gold |
| | Sex | Pos | | Sport_ url | Country_flag_url | Silver |
| | Born | Medal | | Result_ date | Start_date | Bronze |
| | Heigh | | | Result_ location | End_date | Total |
| | Weight | | | Result_ Participant | Competition_date | |
| | Description | | | Result_ | | |

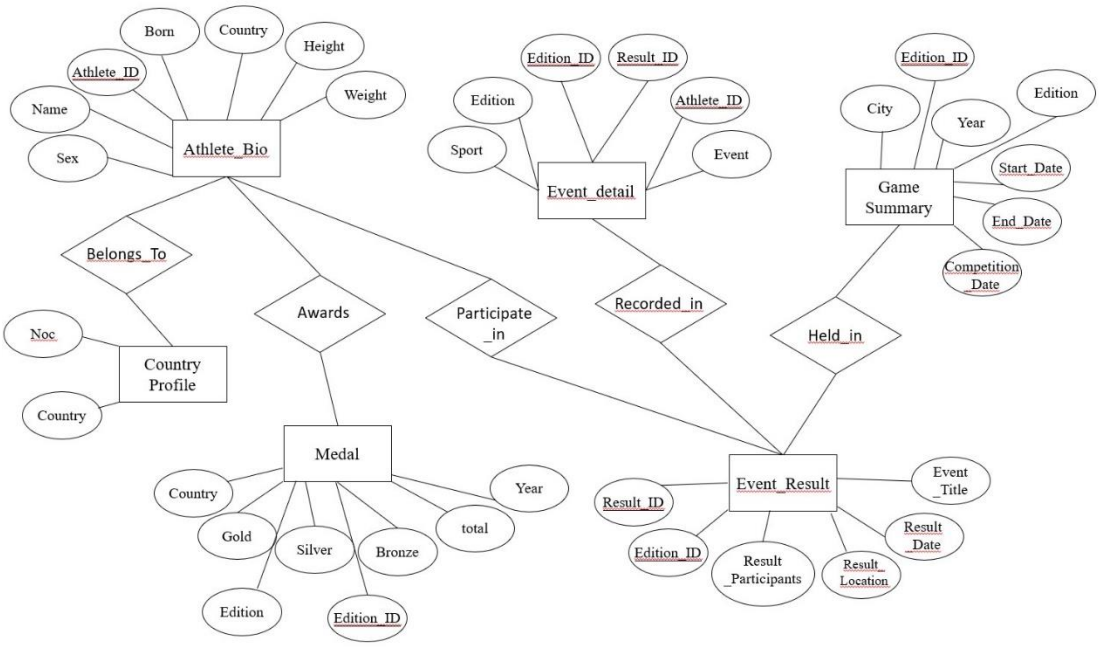
| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | format | | |
| | Special_ Notes | | | | | |

Primary Key：Athlete_id、Edition_id、Result_id

Addition：

- Description：Additional details or notable achievements of the athlete.
- Special_notes：Any special notes or remarks related to the athlete's career or personal background.
- Edition：The year or edition of the Olympic Games.
- Edition_id：A unique identifier for the Olympic Games edition.
- Edition_url：The URL for more information about the Olympic Games edition.
- NOC：The National Olympic Committee (NOC) code of the athlete's country.
- Result_id：A unique identifier for the athlete's result.
- Pos：The athlete’s position or rank in the event.
- Sport_url：The URL for more information about the sport.
- Country_flag_url：The URL of the country’s flag.

## ER model：



# Database

1) The database we use: In our project, we use SQL to maintain our database. Based on the following advantages:

1. Since the data of the Olympic are often structured. So, we can separate the above data by different tables based on their distinct characteristics (such as countries, athletes, or different games)
2. The SQL database supports the foreign key function and does not need other implementations, so they can easily handle the relation between two different tables.
3. When it comes to the Olympic games, we only need to update the data when the new game is held. So, the number of modifications are not too much. So, we do not need to use the NoSQL database for frequently writing. SQL is enough.

2) Also, we use some methods to maintain the database in following situations:

1. Add new data: Before inserting a new (athlete attended) event, the server first determines whether the event "exists". That is, an event held in a new Olympic game, or a new sport category. This is done by checking if the user has selected the "(Other)" option when filling up the form.


If the said event does not exist, a new result ID is generated automatically (implemented by fetching the maximum result ID from the database, then plus 1).

```
$result_id = "";
$isTeamSport = "0";
if ($_POST["new"] == "true" || $_POST["new-year"] == "true") {
    $sql = "SELECT DISTINCT max(convert(result_id, SIGNED INT)) AS id
            FROM Details";
    $result_id = $conn->query(query: $sql)->fetch_assoc()["id"] + 1;
}
```

The same process happens as well if the Olympic games do not exist.

```php
if ($_POST["new-year"] == "true") {
    $year = $_POST["year"];
    $edition = $year." ".$_POST["season"]." Olympics";
    // generate new id
    $sql = "SELECT DISTINCT max(convert(edition_id, SIGNED INT)) AS id
            FROM Medal";
    $edition_id = $conn->query(query: $sql)->fetch_assoc()["id"] + 1;
}
```

This also triggers an insertion of a new game in the Games table, and a new medal tally in the Medal table.

```php
// [Games]
if ($_POST["new-year"] == "true") {
    $sql = "INSERT INTO Games (edition, edition_id, edition_url, year, city, country_flag_url, country_noc, start_date, end_date, competition_date, isHeld)
            VALUES
            ('$edition', '$edition_id', '', '$year', '', '', '', '', '', '', '1')";
    $conn->query(query: $sql);
}
```

```php
// [Medal]
if ($_POST["new-year"] == "true") {
    // check duplicates
    $check = $conn->query(query: "SELECT country_noc FROM Medal WHERE edition = '$edition' AND country_noc = '$noc'")->num_rows;
    if ($check == "0") {
        $sql = "INSERT INTO Medal (edition_id, edition, year, country, country_noc, gold, silver, bronze)
                VALUES
                ('$edition_id', '$edition', '$year', '$country', '$noc', 0, 0, 0)";
        $conn->query(query: $sql);
    }
}
```

Then, the new event is inserted in the Result table.

```php
// [Details]
$sql = "INSERT INTO Details (edition, edition_id, country_noc, sport, event, result_id, athlete, athlete_id, pos, medal, isTeamSport)
        VALUES
        ('$edition', '$edition_id', '$noc', '$sport', '$event', '$result_id', '$athlete', '$athlete_id', '', '', '$isTeamSport')";
$conn->query(query: $sql);
```

Finally, it checks if the Olympic record needs to be updated. First, it gathers the data of that sport, mainly the record score, and how to compare.

```php
$sql = "SELECT grade, ascend, s
        FROM (SELECT DISTINCT d.sport AS sport, d.event AS event, grade, ascend, a.sport AS s
              FROM AthleteRecords a, Details d
              WHERE a.result_id = d.result_id
              ORDER BY sport) AS Q,
             (SELECT DISTINCT d.sport, d.event
              FROM Details d
              WHERE d.result_id = '$result_id'
              ORDER BY sport) AS P
        WHERE Q.sport = P.sport AND Q.event = P.event";
$result = $conn->query(query: $sql)->fetch_assoc();
```

If the grade is a new record, the AthleteRecord table is updated

```php
// compare
if (($asc == 0 && $grade < $old_record) || ($asc == 1 && $grade > $old_record)) {
    $sql = "update AthleteRecords
            set athlete_id = '$athlete_id',
            result_id = '$result_id',
            country = '$noc',
            name = '$athlete',
            grade = '$grade$unit',
            year = '$year',
            ascend = '$asc'
            where sport = ".'"'.$sport.'"';
    $conn->query(query: $sql);
}
```

2. Delete data: When deleting the data, we will ensure that the information of the data in the other table is deleted or modified first by Backend, then delete the data in the end.

[Example]

When deleting an athlete, the following steps will occur:

1. The backend will first retrieve all competition records of the athlete from MySQL.
2. While deleting each record (Delete Details), it will check whether the record includes any medals won.
3. If so, the medal count of the athlete's country for that year will be decreased (Update Medal).
4. After all records have been deleted, the athlete will be removed (Delete Athlete).

```php
$player_id = $_POST["id"];

// get all events
$sql = "select result_id as id, if(medal != '', medal, 'None') as medal, edition_id, country_noc
        from Details
        where athlete_id = '$player_id'";
$events = $conn->query(query: $sql);

// delete all events
while ($event = $events->fetch_assoc()) {
    // Delete: Details
    // Update: Medal

    // [Details]
    $sql = "delete from Details where result_id = '{$event['id']}' and athlete_id = '$player_id'";
    $conn->query(query: $sql);

    // [Medal]
    $edition_id = $event["edition_id"];
    $noc = $event["country_noc"];
    $medal = $event["medal"];

    if ($medal != "None") {
        $sql = "UPDATE Medal
                SET $medal = $medal - 1
                WHERE edition_id = '$edition_id' AND country_noc = '$noc'";
        $conn->query(query: $sql);
    }
}

// delete player
// Delete: Athlete
$sql = "delete from Athlete where athlete_id ='$player_id'";
$conn->query(query: $sql);
```

3. Since the Records will always exist, we do not give the permission to delete the records to maintain all the records.

3) The process of linking a website to a database can be summarized as follows:

"Frontend" PHP <-> JavaScript preprocessing <-> (AJAX) <-> "Backend" PHP <-> MySQL server

[Detailed Process]

The "frontend" PHP is the page that users primarily see and interact with, consisting of pure HTML with attached JavaScript. When a user wants to utilize a specific feature, the attached JavaScript first collects and processes the data, then invokes the "backend" PHP using a function with AJAX capabilities.

AJAX is used here instead of directly calling PHP because we want to dynamically adjust the content on the website. This approach has three main advantages:

Reducing page reloads: It allows users to switch between multiple data entries (e.g., viewing athlete profiles) without clearing the current page data, facilitating better inspection and comparison.

Updating specific elements: It enables updating specific parts of the website (e.g., updating a menu) without affecting other parts.

Improved code maintainability: Separating frontend and backend functionalities makes the code easier to write and maintain.

After sending data to the "backend" PHP, the variables are appended to an SQL query string, which is then sent to the MySQL server for database read or update operations.

Note: Since our approach directly appends variables to the SQL query string, it is susceptible to SQL Injection attacks, which is an area for future improvement.

## Connecting to the Database

To connect to the database, the server requires four parameters: host, user, password, and database. In this project, we are connecting to a server set up on localhost (i.e., our own computer). Therefore, the host is localhost, the user is root, and the password is the one set during server setup.

## Data Flow

If the operation is only to update the database (e.g., updating an athlete's basic information), the process ends here.

However, if data retrieval is needed, the flow reverses.

The "backend" PHP receives the data returned by MySQL, processes each row (possibly preprocessing it), and appends it to an HTML string for output. AJAX then takes the returned data and uses JavaScript to display the HTML string in the "frontend" PHP.

[Example]

Suppose a user wants to search for male American athletes whose names include "Tom." The user would input the following criteria in player.php:


This action triggers the function shown below in player.js, which calls and sends parameters to player_query.php using AJAX:

```
// search player
$("#submit").on("click", function(event) {
    // load search query
    event.preventDefault();
    var name = $("#name").val();
    var country = $("#form .filter-country").val();
    var sex = $("#form .filter-sex").val();
    if (name.trim() != "") {
        search.name = name;
        search.country = country;
        search.sex = sex;
        // uncheck edit mode
        $("#edit-enable").prop("checked", false);
        $(".edit, .delete, .new").css("opacity", "0");

        var url = "player_query.php?m=search";
        $("#table-content").load(url, {
            "player": name,
            "country": country,
            "sex": sex
        });
        $("#default").remove();
    }
})
```

參數

"後端"php 網址

AJAX 函式

In player_query.php, the program embeds the parameters into an SQL query and calls the MySQL server:

```
// Search Player
case 'search':
    $name = $_POST['player'];
    $country = ($_POST['country'] == "all")? "" : $_POST['country'];
    $sex = ($_POST['sex'] == "all") ? "" : $_POST['sex'];
    $sql = "SELECT A.name, A.sex, A.born, A.height, A.weight, A.country, A.athlete_id
            FROM Athlete A
            LEFT JOIN Details E ON A.athlete_id = E.athlete_id
            WHERE A.name LIKE '%$name%' AND A.country LIKE '%$country%' AND A.sex LIKE '$sex%'
            GROUP BY A.athlete_id";
    $result = $conn->query(query: $sql);
```

參數

呼叫 MySQL

After MySQL returns the data, player_query.php embeds the data into an HTML string and outputs it:

```php
while ($row = $result->fetch_assoc()) {
    $rowString = '<tr class="row" data-value="'.$row["athlete_id"].'">
                    <td>'.$row["name"].'</td>
                    <td>'.$row["country"].'</td>
                    <td>'.formatDate(date: $row["born"]).'</td>
                    <td>'
                        .$row["sex"].'
                        <div style="position: relative">
                            <button class="edit">
                                <span class="material-symbols-outlined">
                                    edit
                                </span>
                            </button>
                            <button class="delete" ';
    if (is_int(value: array_search(needle: $row["athlete_id"], haystack: $arr))) {
        $rowString .= "disabled='disabled'";
    }
    $rowString .= '                 >
                                <span class="material-symbols-outlined">
                                    delete
                                </span>
                            </button>
                        </div>
                    </td>
                </tr>';
    echo $rowString;
```

輸出字串

Finally, player.js places the output string into the table in player.php:

table

```javascript
$("#table-content").load(url, {
    "player": name,
    "country": country,
    "sex": sex
});
```

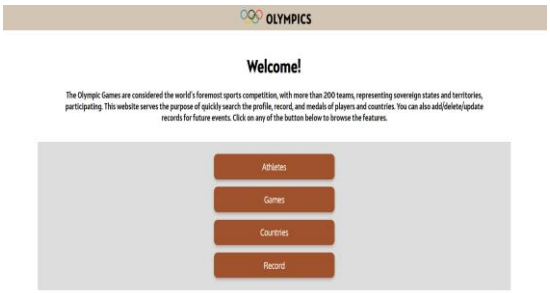
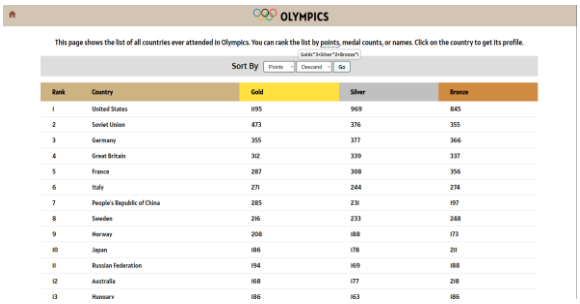
Result:

Enter Name Tom Search

Filter United States Male

| Athlete | Country | Born | Sex |
|---|---|---|---|
| Tommy Reidy | United States | 1968-11-26 | Male |
| Mike Bantom | United States | 1951-12-03 | Male |
| Tommy Burleson | United States | 1952-02-24 | Male |
| Tom Henderson | United States | 1952-01-26 | Male |
| Tom LaGarde | United States | 1955-02-10 | Male |
| Tom McMillen | United States | 1952-05-26 | Male |
| Ron Tomsic | United States | 1933-04-03 | Male |
| Tom Kirby | United States | 1904-12-21 | Male |
| Tommy Lown | United States | 1904-04-05 | Male |
| Tom Southworth | United States | 1944-04-12 | Male |
| Tom Montemage | United States | 1927-01-21 | Male |

# Application

| Home Page | Athletes Page |
|---|---|
|  |  |
| **Games Page** | **Countries Page** |
|  |  |
| **Record Page** | |
|  | |

## Home page

The homepage is the entrance to the application and provides simple navigation.

1. Quick navigation button: Fixed in the center of the page , there are four buttons that provide quick jumps to other main pages : Athletes, Games , Countries , Record

## Athletes page

Users can search for athletes' Profile, and can add, delete athletes, edit profiles

Page Overview and Query :

1. Search field: User can enter keywords of athlete names, select countries, and genders to search for athletes matching conditions **(READ)**


Query:

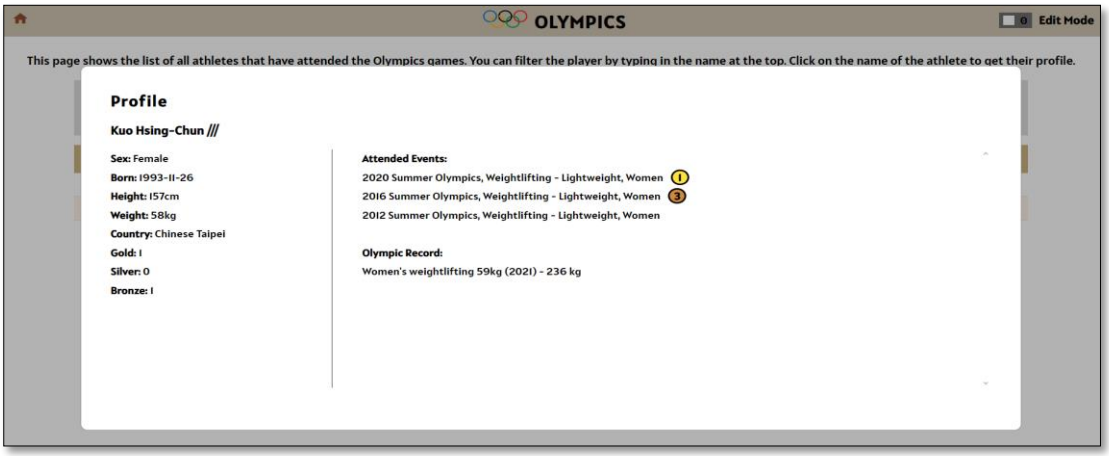
```
SELECT A.name, A.sex, A.born, A.height, A.weight, A.country, A.athlete_id
FROM Athlete A
LEFT JOIN Details E ON A.athlete_id = E.athlete_id
WHERE A.name LIKE '%$name%' AND A.country LIKE '%$country%' AND A.sex LIKE '$sex%'
GROUP BY A.athlete_id";
= $conn->query($sql);

ord = "SELECT DISTINCT a.athlete_id
        FROM AthleteRecords a
        WHERE a.name LIKE '%$name%'";
```

2. Result list (Column: Athlete, Country , Born, Sex): Displays eligible athletes **(READ)**


3. Athlete Profile: Click on any row in the list to display the athlete profile **(READ)**

As shown in the figure, the profile displays the basic information of the players, number of medals, participation records, and Olympic records.



Query:

```
SELECT A.name,
       A.sex,
       A.born,
       A.height,
       A.weight,
       A.country,
       GROUP_CONCAT(DISTINCT E.sport SEPARATOR '  ') AS sport,
       COUNT(E.athlete_id) AS total_events, -- Total events participated in
       COUNT(CASE WHEN E.medal = 'Gold' THEN 1 END) AS gold_medals, -- Gold medal count
       COUNT(CASE WHEN E.medal = 'Silver' THEN 1 END) AS silver_medals, -- Silver medal count
       COUNT(CASE WHEN E.medal = 'Bronze' THEN 1 END) AS bronze_medals, -- Bronze medal count
       GROUP_CONCAT(DISTINCT E.event SEPARATOR '  ') AS events -- List of distinct events
FROM Athlete A
LEFT JOIN Details E ON A.athlete_id = E.athlete_id
WHERE (A.sex = :sex OR :sex IS NULL)
  AND (A.name LIKE CONCAT('%', :name, '%') OR :name IS NULL)
  AND (A.country LIKE CONCAT('%', :country, '%') OR :country IS NULL)
GROUP BY A.athlete_id;
```

```
"select d.edition as edition , d.sport as sport ,d.event as event , if(d.medal != '', d.medal, 'None') as medal
 from Details d
 inner join Athlete a on a.athlete_id = d.athlete_id
 where a.athlete_id = '$playerId'
 order by edition desc , event";
->query($query_event);
```

```
"select ar.sport,ar.year,ar.grade
 from AthleteRecords ar
 inner join Athlete a on a.athlete_id = ar.athlete_id
 where a.athlete_id = '$playerId'
 order by ar.year desc , ar.sport";
```

4. Add/Delete Athlete Functionality **(CREATE, DELETE)**


Turn on the edit mode button,in the upper right corner of the page , and the +New/Edit/Delete button will appear on the far right of the list.

Query:

Add athlete: (Give athlete_id automatically)

```
INSERT INTO Athlete (athlete_id, name, sex, born, height, weight, country, country_noc)
VALUES ('$athlete_id', '$name', '$sex', '$born', '$height', '$weight', '$countries[$noc]', '$noc')";
```

```
'SELECT DISTINCT max(convert(athlete_id, SIGNED INT)) AS id
 FROM Athlete";
```

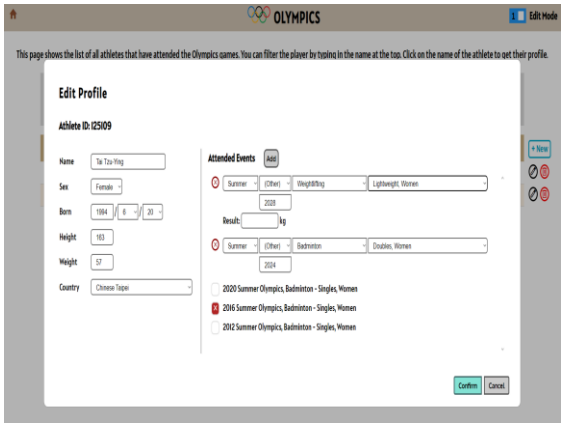
Delete athlete(Can't delete athlete if he/she is olympic record holder)

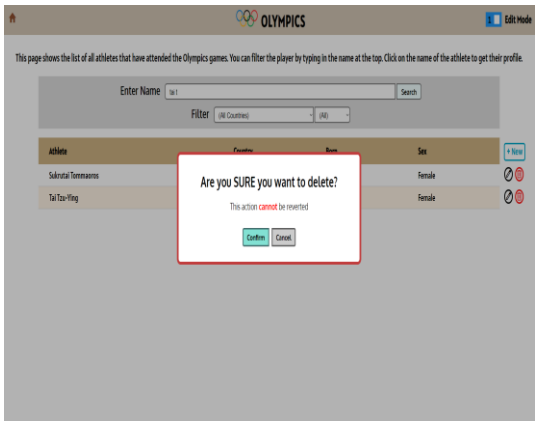
```
"delete from Athlete where athlete_id ='$player_id'";
```

5. Edit Athlete Profile

Add/Delete Athlete Participation Records **(CREATE, DELETE)**

Update athlete details, Olympic records, and medal counts **(UPDATE)**




Query:

Display Edit

```
"SELECT A.name, A.sex, A.born, A.height, A.weight, A.country_noc,
 GROUP_CONCAT(DISTINCT E.event) AS events
 FROM Athlete A
 LEFT JOIN Details E ON A.athlete_id = E.athlete_id
 WHERE A.athlete_id = '$playerId'
 GROUP BY A.athlete_id";
t = $conn->query($sql);

vent = "select d.edition as edition , d.sport as sport ,d.event as event , d.result_id as id
        from Details d
        inner join Athlete a on a.athlete_id = d.athlete_id
        where a.athlete_id = '$playerId'
        order by edition desc , event";
s = $conn->query($sql_event);

ecord = "SELECT DISTINCT a.year, d.sport AS sport, d.event AS event
        FROM AthleteRecords a, Details d
        WHERE a.result_id = d.result_id AND a.athlete_id = '$playerId'
        ORDER BY sport";
```

Update edit

```
'update Athlete
 set name = '".$_POST["name"]."',
     sex = '".$_POST["sex"]."',
     born = '".$_POST["birthday"]."',
     height = '".$_POST["height"]."',
     weight = '".$_POST["weight"]."',
     country = '".$countries[$_POST['noc']]."',
     country_noc = '".$_POST["noc"]."'
 where athlete_id = '".$_POST["id"]."'";
```

Add athlete participation record

```
SELECT DISTINCT max(convert(edition_id, SIGNED INT)) AS id
FROM Medal";
```

```
'SELECT edition
 FROM Medal
 WHERE edition_id = '{$_POST['year']}'

"SELECT DISTINCT event
 FROM Details AS d
 WHERE d.result_id = '{$_POST['event']}'"

SELECT DISTINCT max(convert(result_id, SIGNED INT)) AS id
FROM Details";

'SELECT isTeamSport
 FROM Details
 WHERE result_id = '{$_POST['event']}''
```

```
INSERT INTO Games (edition, edition_id, edition_url, year, city, country_flag_url, country_noc, start_date, end_date, competition_date, isHeld)
VALUES
('$edition', '$edition_id', '', '$year', '', '', '', '', '', '', '1')";

'INSERT INTO Details (edition, edition_id, country_noc, sport, event, result_id, athlete, athlete_id, pos, medal, isTeamSport)
VALUES
('$edition', '$edition_id', '$noc', '$sport', '$event', '$result_id', '$athlete', '$athlete_id', '', '', '$isTeamSport')";
```

```
'INSERT INTO Medal (edition_id, edition, year, country, country_noc, gold, silver, bronze)
VALUES
('$edition_id', '$edition', '$year', '$country', '$noc', 0, 0, 0)";
```

Update Olympic Record

```
"SELECT grade, ascend, s
 FROM (SELECT DISTINCT d.sport AS sport, d.event AS event, grade, ascend, a.sport AS s
       FROM AthleteRecords a, Details d
       WHERE a.result_id = d.result_id
       ORDER BY sport) AS Q,
      (SELECT DISTINCT d.sport, d.event
       FROM Details d
       WHERE d.result_id = '$result_id'
       ORDER BY sport) AS P
 WHERE Q.sport = P.sport AND Q.event = P.event";
```

```
"update AthleteRecords
 set athlete_id = '$athlete_id',
 result_id = '$result_id',
 country = '$noc',
 name = '$athlete',
 grade = '$grade$unit',
 year = '$year',
 ascend = '$asc'
 where sport = ".'"'.$sport.'"';
```

Delete Participation Record

```
SELECT if(medal != '', medal, 'None') as medal, edition_id, country_noc
FROM Details
WHERE athlete_id = '$player_id' AND result_id = '$result_id'";

 UPDATE Medal
 SET $medal = $medal - 1
 WHERE edition_id = '$edition_id' AND country_noc = '$noc''

select result_id as id, if(medal != '', medal, 'None') as medal, edition_id, country_noc
from Details
where athlete_id = '$player_id'";

delete from Details where result_id = '{$event['id']}' and athlete_id = '$player_id''
 "UPDATE Medal
  SET $medal = $medal - 1
  WHERE edition_id = '$edition_id' AND country_noc = '$noc'"
```

## Games page

Allow users to use a dropdown menu to view medalists of all subcategories for a particular sport in a given year.

Page Overview and Query :

1. Search Dropdown: Filter based on Summer/Winter Olympic, Year, and Sport **(READ)**


Query:

```
= "SELECT DISTINCT year FROM Games WHERE edition LIKE '%Summer%' ORDER BY year ASC";
type === 'Winter') {
= "SELECT DISTINCT year FROM Games WHERE edition LIKE '%Winter%' ORDER BY year ASC";

= "SELECT DISTINCT year FROM Games ORDER BY year ASC";

"SELECT DISTINCT sport FROM Details D INNER JOIN Games G ON D.edition_id = G.edition_id WHERE G.year = ? ORDER BY sport ASC"
```

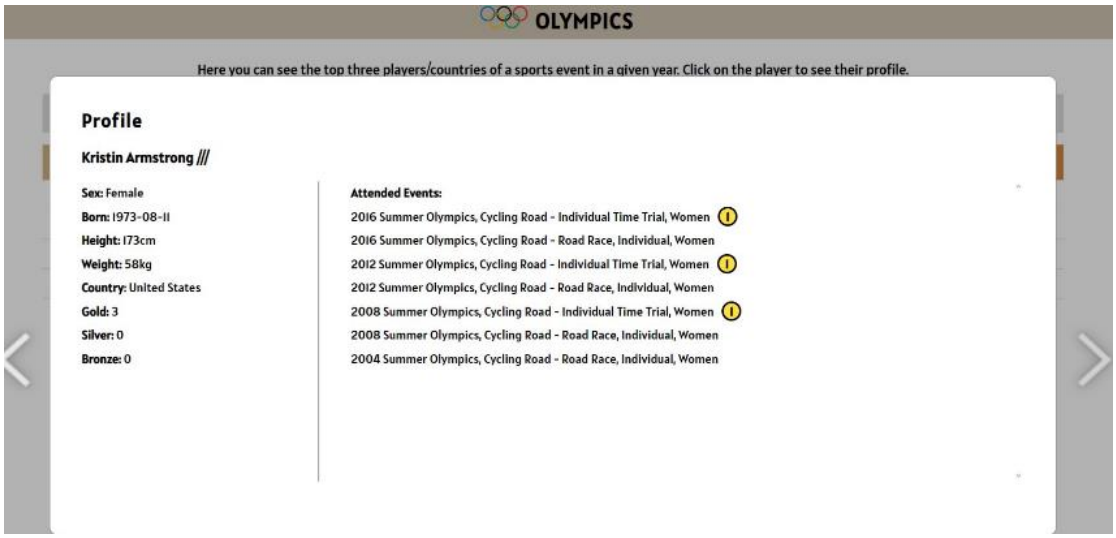
2. Result list: Display the top three players of the sport (the team event displays the country code, the individual event displays the athlete name)

Query:

```
SELECT
    D.event AS event,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Gold' THEN
                        CASE WHEN D.isTeamSport = '1' THEN CONCAT('[ ', C.country, ' ]') ELSE A.name END
                    ELSE NULL END SEPARATOR ', ') AS gold,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Silver' THEN
                        CASE WHEN D.isTeamSport = '1' THEN CONCAT('[ ', C.country, ' ]') ELSE A.name END
                    ELSE NULL END SEPARATOR ', ') AS silver,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Bronze' THEN
                        CASE WHEN D.isTeamSport = '1' THEN CONCAT('[ ', C.country, ' ]') ELSE A.name END
                    ELSE NULL END SEPARATOR ', ') AS bronze,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Gold' THEN
                        CASE WHEN D.isTeamSport = '1' THEN '' ELSE A.athlete_id END
                    ELSE NULL END SEPARATOR ', ') AS gold_id,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Silver' THEN
                        CASE WHEN D.isTeamSport = '1' THEN '' ELSE A.athlete_id END
                    ELSE NULL END SEPARATOR ', ') AS silver_id,
    GROUP_CONCAT(DISTINCT CASE
                    WHEN D.medal = 'Bronze' THEN
                        CASE WHEN D.isTeamSport = '1' THEN '' ELSE A.athlete_id END
                    ELSE NULL END SEPARATOR ', ') AS bronze_id

  FROM
      Details D
  INNER JOIN
      Games G ON D.edition_id = G.edition_id
  LEFT JOIN
      Athlete A ON D.athlete_id = A.athlete_id
  LEFT JOIN
      Country C ON D.country_noc = C.noc
  WHERE
      G.year = ? AND D.sport = ? AND G.edition LIKE ?
  GROUP BY
      D.event
  ORDER BY
      D.event ASC;
```

3. Athlete Profile: There is a button on each athlete's name, users can click it to see athlete's profile, same as athletes page.



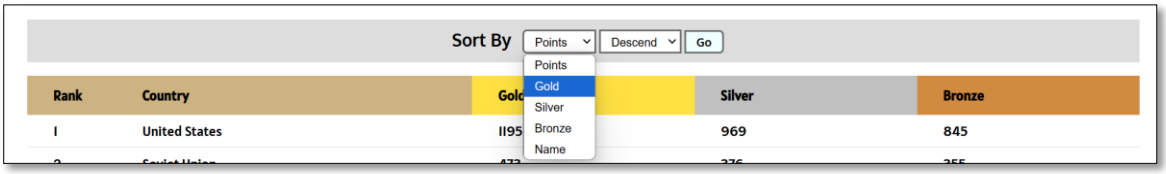
## Countries page

List the Olympic medal rankings of all countries. Users can change the sorting conditions.

Page Overview and Query :

1. Condition Dropdown: Sort by ascend or descend according to points (Gold*3+Sliver*2+Bronze*1), Gold, Sliver, Bronze, names (initial setting is points, descend)

**(READ)**




Query:

```
= "SELECT country, country_noc,
        SUM(gold) AS gold, SUM(silver) AS silver, SUM(bronze) AS bronze,
        SUM(gold * 3 + silver * 2 + bronze) AS points
 FROM Medal
 GROUP BY country, country_noc";
```

2. Result list (Column: Rank, Country , Gold, Sliver, Bronze):

| Rank | Country | Gold | Silver | Bronze |
|---|---|---|---|---|
| 1 | United States | 1195 | 969 | 845 |
| 2 | Soviet Union | 473 | 376 | 355 |
| 3 | Germany | 355 | 377 | 366 |
| 4 | Great Britain | 312 | 339 | 337 |
| 5 | France | 287 | 308 | 356 |

3. Profile of Country: Click on any row in the list to display the specific years and events of all medals in that country.

Query:

```
"SELECT DISTINCT edition, sport, event, edition_id, result_id, medal
 FROM Details
 WHERE country_noc = '$noc' AND medal != ''
 ORDER BY edition, sport, event";
```

## Record page

Show Olympic records

1. List: Lists Olympic records for sports **(READ)**

| Event | Athlete Name | Country | Grade | Date |
|---|---|---|---|---|
| Men's track and field 100 meters | Usain Bolt | Jamaica | 9.63 s | 2012 |
| Men's track and field 200 meters | Usain Bolt | Jamaica | 19.3 s | 2008 |
| Men's track and field 400 meters | Wayde van Niekerk | South Africa | 43.03 s | 2016 |
| Men's track and field 800 meters | David Rudisha | Kenya | 41.91 s | 2012 |
| Men's track and field 1500 meters | Jakob Ingebrigtsen | Norway | 28.32 s | 2021 |
| Men's Marathon | Sammy Wanjiru | Kenya | 2 h, 6 m, 32 s | 2008 |
| Men's 3,000 metres Steeplechase | Conseslus Kipruto | Kenya | 3.28 s | 2016 |
| Men's 400m hurdles | Karsten Warholm | Norway | 45.94 s | 2021 |
| Men's high jump | Charles Austin | United States | 2.39 m | 1996 |
| Men's pole vault | Thiago Braz | Brazil | 6.03 m | 2016 |
| Men's long jump | Bob Beamon | United States | 8.9 m | 1968 |
| Men's triple jump | Kenny Harrison | United States | 18.09 m | 1996 |
| Men's shot put | Ryan Crouser | United States | 23.3 m | 2021 |

Query:

```
"SELECT sport AS event, c.country AS country, name, grade, year
 FROM AthleteRecords AS a, Country AS c
 WHERE a.country = c.noc";
```

**CRUD**

| | | |
|---|---|---|
| **Create** | Add new athletes , participation records | |
| **Read** | Athletes | ( profile , participation records , medals , olympic records ) + filter name , country , gender |
| | Games | (top three in event, winner/country) + selection year, winter/summer, sport |
| | Countries | ( number of awards won over the years , details ) + sorted by medals , points , names |
| | Record | ( items , holders , nationalities , results , years ) |
| **Update** | Update athletes' profile , Records , Medals | |
| **Delete** | Delete athletes and participation records | |

## Other：

Github Repository：https://github.com/ZHEN-HONG/Database-Team22

YouTube：https://www.youtube.com/watch?v=0wX515MbXHY

Progress：

| | |
|---|---|
| 10/24 | Discuss the topic together and make a Final Project Proposal. |
| 10/31 | Discussion suspended during midterm exam week. |
| 11/07 | Already done：<br>Create: Add Olympic data.<br>Read: You can manually select data such as country, Olympian, Olympic year, number of medals, etc.<br>Update: Update world records, number of medals, etc.<br>Delete: Delete player information or national entries.<br>Expected completion：<br>✧ Familiar with Table data, design and reference of conceptual variables.<br>✧ Design interface layout and planning filtering function. |
| 11/12 | Already done：<br>✧ Database design (SQL)<br>✧ Web front-end infrastructure<br>✧ Preliminary API design<br>Expected completion：<br>✧ It is expected that the interface will have 4 buttons: "Country", "Annual", "Athletes", "Sports and Records"<br>✧ CRUD function keys.<br>✧ SQL side: connect to the required database.<br>✧ Web page: Set up the buttons and links required for the web page. |
| 11/21 | Already done：<br>✧ Table creation of countries, years and players on the SQL side and implements some filtering functions.<br>✧ The web page has presented the basic structure, the presentation of the web interface, the links of buttons and the filtering menu.<br>Problem & Solution：<br>✧ The Olympic records are incomplete and require additional DB |

| | |
|---|---|
| | or Table to complete.<br>✧ If the data conflicts (for example, the same country has different data before and after Russia is removed from the list), it can be resolved by simple processing.<br>Expected completion：<br>✧ Confirm whether the basic operations of CRUD are functioning normally, including adding, querying, updating, and deleting player and competition information.<br>✧ The basic front-end interface is completed and can input and display database data.<br>✧ Add error handling, such as data input errors, connection failures, invalid requests, etc.<br>✧ Send clear error messages back to the front end to ensure users understand the problem. |
| 11/28 | Already done：<br>✧ Country interface (lists medals and rankings)<br>✧ All player details<br>✧ Add(Table)<br>✧ Olympic Records(Table)<br>Not done：<br>✧ Click on country to open details -> Winning Year/Gold, Silver and Bronze Medals. (Table missing)<br>✧ Year, Olympic record. (Table has been uploaded to the web page for processing)<br>✧ RUD(Table)<br>Expected completion：<br>✧ Ensure that the front-end page can interact smoothly with the back-end API to achieve complete data circulation.<br>✧ Fix minor issues between API and front-end to ensure correct data transfer. |
| 12/05 | Already done：<br>User experience optimization.<br>Expected completion：<br>✧ Player screening optimization.<br>✧ Automatically enter Athlete_ID or Result_ID when adding new data.<br>✧ Add and delete functions. |
| 12/12 | Discussion suspended for final exam. |

| | |
|---|---|
| 12/19 | Final discussion, fine-tuning and division of labor for report writing |
| 12/27 | Complete assignment requirements and submit. |

Contribution：

| | |
|---|---|
| 劉真宏 | Responsible for detailed planning and assisting both parties in integrating plans. |
| 林啟堯、嚴偉哲 | Responsible for writing and processing web page HTML and PHP programs |
| 彭叡楀、駱巍文 | Responsible for writing and processing web SQL programs |