USER GUIDE ON THE OLYMPICS DATABASE

Final Assessment

STUDENT ID: 19501204
LAB: MONDAY 2-4PM
DUE DATE: 27/10/2021
UNIT: DATABASE SYSTEMS (ISYS1001)

Contents

Introduction

The purpose of this user guide is for users to learn how to implement and use the database designed for the final assessment of ISYS1001 Database Systems based on the Olympics 2020.

Assumptions

This guide assumes the following:

- The scripts will be run in the Curtin lab VM environment.
- The user has basic knowledge on Linux terminal commands.
- The database is in an archived state.

Logging into the server

Open the terminal (PRESS+ALT+T)

Navigate to the directory with the scripts: '…\19501204_Final_Assessment\'

Enter the following command in the terminal and press enter:

```
mysql -u me -p
```

You will be prompted to enter a password for the server.

Password is: 'myUserPassword'.

Once connected to the MySQL server, the terminal prompt will be changed to

```
mysql >
```

Building the database

Run the *build.sql* script using the following command to build the database:

```
mysql > SOURCE build.sql;
```

The build.sql script creates a fresh new database called **'olympics_19501204'** and runs the below scripts to create its respective table:

Table	**Script file**
Country	createcountry.sql
Team	createteam.sql
Athletes	createathletes.sql
Games	creategames.sql
AthleteParticipation	createathleteparticipation.sql

Once the tables have been created, the build.sql executes scripts as shown in the table below to populate the tables within the database using the values in the respective csv files.

Table	Script	CSV file
Country	inscountry.sql	Country.csv
Athletes	insathletes.sql	Athletes.csv
Team	insteams.sql	Teams.csv
Games	insgames.sql	Games.csv
AthleteParticipation	insathpar.sql	AthleteParticipation.sql

Table Structure

Country Table

Fields	Data Type	Size	Null	Description
country_code (Primary Key)	CHAR	3	No	County Iso 3166-1 code, unique
name	VARCHAR	27	No	Name of country

Athletes Table

Fields	Data Type	Size	Null	Description
athlete_id (primary key)	CHAR	6	No	Athlete number, unique
firstname	VARCHAR	12	No	First name
lastname	VARCHAR	15	No	Last name
sex	CHAR	1		M=male, F=female, O=other
birthyear	YEAR	4		Year of birth
country_code (foreign key)	CHAR	3	No	County Iso 3166-1 code

Teams Table

Fields	Data Type	Size	Null	Reference Table
team_id (primary key)	CHAR	6	No	Team number, unique
discipline	VARCHAR	36		Sport name
country_code (foreign key)	CHAR	3	No	County Iso 3166-1 code
gender	VARCHAR	5	No	Team's gender

Games Table

Fields	Data Type	Size	Null	Reference Table
game_id (primary key)	CHAR	6	No	Game number, unique
event	VARCHAR	40	No	Name of the event
discipline	VARCHAR	24	No	Sport type
sex	CHAR	1	No	M=Mens, W=Womens, O=Mixed
eventdate	DATE			Start date of games
teamgame	TINYINT	1		If the game is a team game; 1=true, 0= false

AthleteParticipation Table

Fields	Data Type	Size	Null	Description
athlete_id (foreign key)	CHAR	6	No	Athlete number
game_id (foreign key)	CHAR	6	No	Game number
team_id (foreign key)	CHAR	6		Team number for team-based games
medal	CHAR	1		G=gold, S=silver, B=bronze, null=no medals

Combination of athlete_id and game_id is unique for rows of this table.

Queries

All queries are stored in command.sql. To run these queries, run the command below:

```
mysql > SOURCE command.sql;
```

The SQL queries are detailed in the following section.

Query 1

The query below displays all games within the Games table where the date of the event is after 26-07-2021.

```
SELECT *
FROM Games
WHERE eventdate >= '2021-07-27';
```

Query 2

The query below displays all tuples in the Games table with an additional column called "DaysDifference" that shows the difference in days from when each game occurred to the moment the query is executed.

```
SELECT *,
    DATEDIFF(NOW(), eventdate) AS DaysDifference
FROM Games;
```

Query 3

The query below joins the Athletes, AthleteParticipation and Games tables to display all American athletes and the events they participated in.

```
SELECT  Athletes.athlete_id,
    CONCAT(Athletes.firstname, " ", Athletes.lastname) AS FullName,
    AthleteParticipation.game_id,
    Games.event
FROM Athletes
LEFT OUTER JOIN AthleteParticipation ON Athletes.athlete_id=AthleteParticipation.athlete_id
LEFT OUTER JOIN Games ON Games.game_id=AthleteParticipation.game_id
WHERE Athletes.country_code='USA'
ORDER BY game_id;
```

Query 4

The query below displays all athletes who won a gold medal.

```
SELECT AthleteParticipation.athlete_id,
    Athletes.firstname,
    Athletes.lastname,
    count(medal) as total_gold_medals,
    country_code
FROM AthleteParticipation
INNER JOIN Athletes ON Athletes.athlete_id=AthleteParticipation.athlete_id
WHERE medal='G'
GROUP BY AthleteParticipation.athlete_id;
```

Query 5

The query below displays gold medallists of individual games.

```
SELECT athlete_id,
       game_id,
       count(medal)
FROM AthleteParticipation
WHERE medal='G' AND team_id IS NULL
group by athlete_id, game_id;
```

Query 6

The query below displays the teams that won a gold medal in team games.

```
SELECT DISTINCT game_id,
    AthleteParticipation.team_id,
    country_code
FROM AthleteParticipation
LEFT OUTER JOIN Team ON AthleteParticipation.team_id=Team.team_id
WHERE medal='G' AND AthleteParticipation.team_id IS NOT NULL;
```

Query 7

The query below displays the major events to declare the gold medallists.

```
SELECT DISTINCT AthleteParticipation.game_id,
        event,
        sex,
        eventdate
FROM AthleteParticipation
LEFT OUTER JOIN Games on AthleteParticipation.game_id=Games.game_id
WHERE medal='G'
GROUP BY game_id;
```

Query 8

The query below displays all gold medals a country has won. Winning a gold medal in a team event will increase the total by one.

```
SELECT COUNT(DISTINCT game_id) as goldmedals,
       Athletes.country_code
FROM AthleteParticipation
INNER JOIN Athletes ON Athletes.athlete_id=AthleteParticipation.athlete_id
WHERE medal='G'
GROUP BY Athletes.country_code;
```

Query 9

The final query displays all the medal types for athletes who have won more than one medal.

```
SELECT athlete_id,
    SUM(CASE WHEN medal = 'B' THEN 1 ELSE 0 END) AS BronzeMedals,
    SUM(CASE WHEN medal = 'S' THEN 1 ELSE 0 END) AS SilverMedals,
    SUM(CASE WHEN medal = 'G' THEN 1 ELSE 0 END) AS Goldmedals,
    count(medal) as TotalMedals
FROM AthleteParticipation
GROUP BY athlete_id
HAVING TotalMedals > 1;
```

Procedures

To view all procedures currently active inside the database, run the following command:

```
mysql> SHOW PROCEDURE STATUS WHERE Db='olympics_19501204';
```

Medal Counter

The medal counter procedure returns the number of medals a country has won. The procedure imports a size 3 char type for the ISO 3166 code of a country and exports an integer.

This procedure is stored in the script: medalcounter.sql.

Execute the following command to store the procedure into the database:

```
mysql> SOURCE medalcounter.sql;
```

To execute the procedure:

```
mysql> CALL countMedals('AUS', @result);
```

‘AUS’ is interchangeable for any country_code in the Country table. ‘@result’ can be replaced by any other variable name.

To view the results:

```
mysql> SELECT @result;
```

The medalcounter.sql script is compromised of the code below:

```
DROP PROCEDURE IF EXISTS countMedals;
CREATE PROCEDURE countMedals(
    IN code CHAR(3),
    OUT total INT)
COMMENT 'Count the number of medals a country has. Team games are counted as one medal. Provided the ISO-3166 code.'
SELECT COUNT(DISTINCT game_id)
FROM AthleteParticipation
INNER JOIN Athletes
    ON Athletes.athlete_id=AthleteParticipation.athlete_id
WHERE medal IS NOT NULL
    AND Athletes.country_code = code
GROUP BY Athletes.country_code INTO total;
```

Insert New Athlete

The procedure to insert a new athlete accepts all values except for the athlete_id. The procedure will find the highest athlete_id, increment it by 1 and assign this value to the new athlete.

This procedure is stored in the script: insathproc.sql.

Execute the following command to store the procedure into the database:

```
mysql> SOURCE insathproc.sql;
```

An example on how to execute the procedure:

```
mysql> CALL insNewAthlete(‘<firstname>’, ‘<lastname>’, ‘F’, ‘2000’, ‘AUS’);
```

insathproc.sql is compromised of the code below:

```
DROP PROCEDURE IF EXISTS insNewAthlete;
DELIMITER //
CREATE PROCEDURE insNewAthlete(
    f VARCHAR(12),   -- firstname
    l VARCHAR(15),   -- lastname
    s CHAR(1),       -- sex
    y YEAR,          -- year
    c CHAR(3)        -- country_code
    )
    COMMENT 'Insert a new Athlete into Athletes table.'
    BEGIN
        DECLARE newid CHAR(6);
        SELECT MAX(athlete_id)+1 FROM Athletes INTO newid;
        INSERT INTO Athletes
            VALUES(newid, f, l, s, y, c);
    END//
DELIMITER ;
```

Views

The following command is to view all views currently active in the database:

```
mysql> SHOW FULL TABLES IN olympics_19501204 WHERE TABLE_TYPE LIKE 'VIEW';
```

Athlete Medals

The athlete medal view contains a table to display how many bronze, silver and gold medals an athlete has won.

The view is created in the script: athmedview.sql.

Execute the following command to store the view into the database.

```
mysql> SOURCE athmedview.sql;
```

To view the newly created view, execute this command:

```
mysql> SELECT * FROM Athletemedals;
```

Athlete Details

The athlete details view simply contains a table to show all the names of athletes competing in each game.

This view is created in the script: athdetview.sql.

Execute the following command to store the view into the database.

```
mysql> SOURCE athdetview.sql;
```

To view the newly created view, execute this command:

```
mysql> SELECT * FROM Athletedetails;
```

Python3

1. Open the python3 console by entering the following command in the Linux terminal that is in the same directory as the olympic_19501204 database created:

```
Python3
```

2. Type in the code below to check if the mysql connector is installed:

```
>>> import mysql.connector
```

3. If it is not installed then exit the python3 console and install the connector:

```
>>> exit()
> pip3 install mysql-connector-python
```

4. Open the python3 console once the connector has been installed and run the following command to execute query 1 in the section above:

```
>>> exec(open("query1.py").read())
```

5. Run the following command to execute query 3 in the section above:

```
>>> exec(open("query3.py").read())
```

6. Run the following command to fill the AthleteParticipation table with values from the AthleteParticipation.csv file. The table must not be filled at this time.

```
>>> exec(open("loadathpar.py").read())
```