

DBMS

Assignment -4 Report

TEAM ID: 8

TOPIC: OLYMPIC DATABASE

TEAM MEMBERS:

- 1)PRAJWAL KAMATH K - PES1UG19CS337
- 2)PRANAV RAJNISH - PES1UG19CS344
- 3)RAGHAVENDRA L - PES1UG19CS366

Front end

For the front end of our postgres database, we have chosen to go with a simple and robust design using HTML5, CSS and Javascript. To connect to the database, we have made use of the **node.js** library and to run a webserver and consume web requests made from our webpage, we have made use of **Express.js** framework.

Using these technologies, we can build an intuitive and responsive front end for our database.

Node.js also come pre-packaged with a postgres library, that we can make use of to easily interface with our database as well as query it.

We have to import this library using the command:

```
const { Pool } = require("pg")
```

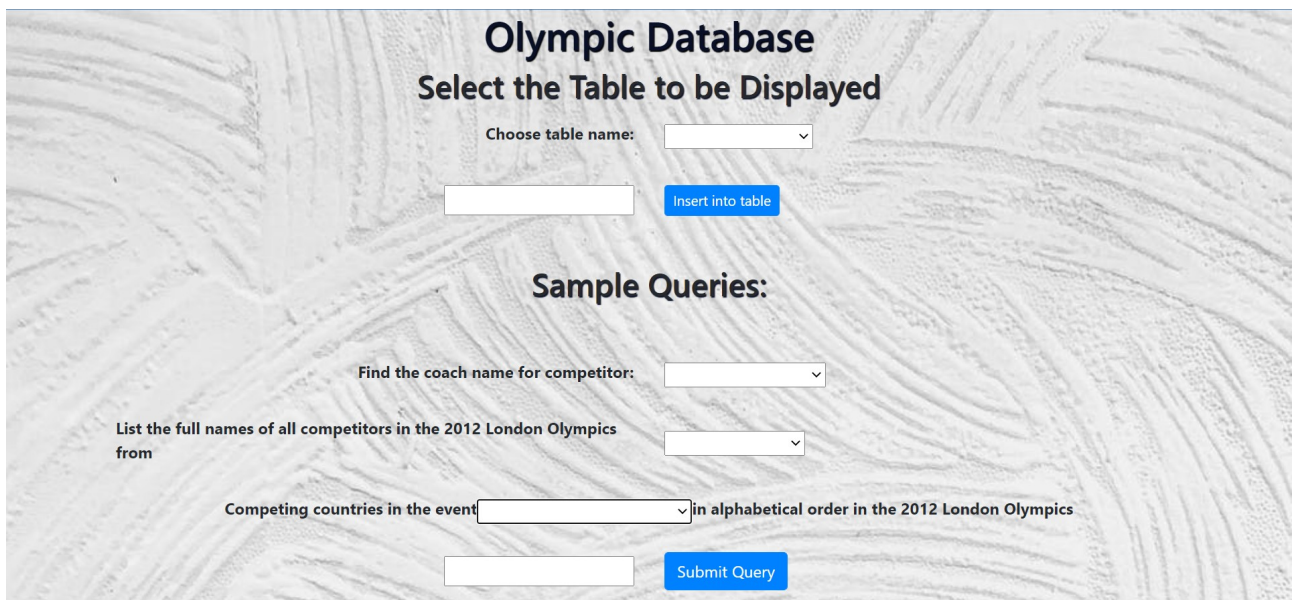
Using the pool object we can connect to our database by passing in the values for the **user, password, host, database** and **port**.

We can then pass in sql commands using the function

```
Await pool.query("[sql command]")
```

Which is an asynchronous command that may return an output in the form of an array of rows depending on the sql command.

Here is a screen shot of our front end



The screenshot shows a web application titled "Olympic Database" with the subtitle "Select the Table to be Displayed". It features a form with a dropdown menu labeled "Choose table name:" and a button labeled "Insert into table". Below this, there is a section titled "Sample Queries:" with three query examples, each with a dropdown menu and a "Submit Query" button. The queries are: "Find the coach name for competitor:", "List the full names of all competitors in the 2012 London Olympics from", and "Competing countries in the event" followed by "in alphabetical order in the 2012 London Olympics".

We have the ability to view any table in our database, insert values into the table as well as delete records in the table.

Choose table name:

country_id	country_name
1	United Kingdom
2	Greece
3	China
4	Australia
5	South Korea
6	Japan
7	Philippines
8	Africa
9	India

9, "India"

Here we chose the table named “country” to view its contents. Then we are adding a record (9, “India”) into the table. To delete any row, we simply have to click the row we want to delete. The table is automatically updated on any modifications.

We also have a Sample Queries section where we have a few of the queries used earlier in the assignment. We have given the user to modify these queries by changing specific parameters like which competitor to find the coach for etc.

Sample Queries:

coach_first_name	coach_last_name
Anne	Keel

Find the coach name for competitor:

List the full names of all competitors in the 2012 London Olympics from

Competing countries in the event in alphabetical order in the 2012 London Olympics

In the next screen shot we are querying the database to find the competing countries in various sports in the 2012 London Olympics, sorted in ascending order.

Sample Queries:

country_name
Australia
Greece
Philippines

Find the coach name for competitor:

List the full names of all competitors in the 2012 London Olympics from

Competing countries in the event in alphabetical order in the 2012 London Olympics

We also have provided a custom query box, where the user may enter any sql command they want and the result will be displayed on our webpage.

Sample Queries:

country_id	country_name
9	India

Find the coach name for competitor:

List the full names of all competitors in the 2012 London Olympics from

Competing countries in the event in alphabetical order in the 2012 London Olympics

Here we are querying the database to find all records in country table where country_id is equal to 9.

SCHEMA AND CONSTRAINT CHANGES

1) For the Event Table we want to make a change to the schema by adding a column for world record, which has the information about the world record time or score for the concerned event.

If the attribute doesn't apply to the event, or we don't have the data, we leave it as null.

Select * from event;

You are now connected to database "olympic_database" as user "postgres".

event_id	olympiad_id	category_id	event_name	gender	team	location	start_date	end_date
1	1	1	Men Basketball	M	T	Burnham Park	2012-03-08	2012-03-10
2	1	6	Men 1500 Freestyle event	M	F	Pool Park	2012-02-10	2012-02-12
3	5	1	Women Basketball	F	T	Shrine Court	1998-04-12	1998-04-14
4	4	5	Men Welter Weight Boxing	M	F	Douglas Stadium	1999-05-14	1999-05-16
5	3	5	Men Heavy Weight Boxing	M	F	Pilar Park	2008-06-16	2008-06-18
6	1	1	Women Basketball	F	T	London Bridge Court	2012-08-18	2012-08-20
7	3	6	Men 1500 Freestyle event	M	F	Xiang Pool	2008-09-20	2008-09-22
8	6	2	Men 100 meter Archery	M	F	Sahara Desert	2010-10-22	2010-10-24
9	6	2	Women 100 meter Archery	F	F	Sahara Desert	2010-11-24	2010-11-26

(9 rows)

Event Table before Modifying

```
ALTER TABLE event ADD COLUMN WORLD_RECORD VARCHAR(20);
UPDATE event SET WORLD_RECORD='14:31:04' WHERE event_id=2 or
event_id=7;
UPDATE event SET WORLD_RECORD='1440' WHERE event_id=8;
UPDATE event SET WORLD_RECORD='1340' WHERE event_id=9;
select * from event;
```

ALTER TABLE
UPDATE 2
UPDATE 1
UPDATE 1

event_id	olympiad_id	category_id	event_name	gender	team	location	start_date	end_date	world_record
1	1	1	Men Basketball	M	T	Burnham Park	2012-03-08	2012-03-10	
3	5	1	Women Basketball	F	T	Shrine Court	1998-04-12	1998-04-14	
4	4	5	Men Welter Weight Boxing	M	F	Douglas Stadium	1999-05-14	1999-05-16	
5	3	5	Men Heavy Weight Boxing	M	F	Pilar Park	2008-06-16	2008-06-18	
6	1	1	Women Basketball	F	T	London Bridge Court	2012-08-18	2012-08-20	
2	1	6	Men 1500 Freestyle event	M	F	Pool Park	2012-02-10	2012-02-12	14:31:04
7	3	6	Men 1500 Freestyle event	M	F	Xiang Pool	2008-09-20	2008-09-22	14:31:04
8	6	2	Men 100 meter Archery	M	F	Sahara Desert	2010-10-22	2010-10-24	1440
9	6	2	Women 100 meter Archery	F	F	Sahara Desert	2010-11-24	2010-11-26	1340

(9 rows)

We have successfully added World_record column in Event Table

2)For the Competitor table, we want to make a constraint change, that is to remove the not null constraint for the Country_id column.

This is to accommodate refugees and other competitors who do not belong to any country.

Select * from competitor;

competitor_id	country_id	firstname	lastname	gender	date_of_birth
1	4	Mercedes	Foster	M	1988-12-18
2	4	Samatha	Gardner	F	1984-07-27
3	1	Riley	George	M	1952-08-19
4	1	Clarence	Rodriguez	F	1986-12-23
5	7	James	Pinkard	M	1985-04-08
6	7	Marcia	Hardwick	F	1980-07-30
7	2	James	Noonan	M	1987-09-25
8	2	Patricia	Kowell	F	1986-06-16

(8 rows)

Competitor Table before Modifying

**ALTER TABLE competitor ALTER COLUMN country_id drop NOT NULL;
INSERT INTO COMPETITOR VALUES (9,NULL, 'Harry', 'Stephan', 'M', '10-DEC-1988');**

Select * from competitor;

ALTER TABLE
INSERT 0 1

competitor_id	country_id	firstname	lastname	gender	date_of_birth
1	4	Mercedes	Foster	M	1988-12-18
2	4	Samatha	Gardner	F	1984-07-27
3	1	Riley	George	M	1952-08-19
4	1	Clarence	Rodriguez	F	1986-12-23
5	7	James	Pinkard	M	1985-04-08
6	7	Marcia	Hardwick	F	1980-07-30
7	2	James	Noonan	M	1987-09-25
8	2	Patricia	Kowell	F	1986-06-16
9		Harry	Stephan	M	1988-12-10

(9 rows)

We have successfully inserted a row with no country_id

3) For the Coaches and Competitor_event table, we want to make a constraint change, that is to drop the foreign key constraint ON DELETE RESTRICT and add the foreign key ON DELETE CASCADE

This is to accommodate athletes who are banned due to malpractice or due to injury

```
ALTER TABLE COACHES DROP CONSTRAINT coaches_fk2;
ALTER TABLE COACHES add constraint coaches_fk2 FOREIGN KEY
(competitor_id) REFERENCES competitor(competitor_id) ON DELETE
CASCADE;
ALTER TABLE COMPETITOR_EVENT DROP CONSTRAINT
athlete_event_fk1;
ALTER TABLE COMPETITOR_EVENT add constraint athlete_event_fk1
FOREIGN KEY (competitor_id) REFERENCES competitor(competitor_id) ON
DELETE CASCADE;
SELECT * FROM COMPETITOR;
```

```
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
```

competitor_id	country_id	firstname	lastname	gender	date_of_birth
1	4	Mercedes	Foster	M	1988-12-18
2	4	Samatha	Gardner	F	1984-07-27
3	1	Riley	George	M	1952-08-19
4	1	Clarence	Rodriguez	F	1986-12-23
5	7	James	Pinkard	M	1985-04-08
6	7	Marcia	Hardwick	F	1980-07-30
7	2	James	Noonan	M	1987-09-25
8	2	Patricia	Kowell	F	1986-06-16
9		Harry	Stephan	M	1988-12-10

(9 rows)

Altering the tables and competitor table before modifying

```
SELECT * FROM COACHES;
```

coach_id	competitor_id
2	3
1	7
6	1
5	2
8	6
3	5
7	4
4	8

(8 rows)

Coach Table before Modifying

SELECT * FROM COMPETITOR_EVENT;

competitor_event_id	competitor_id	event_id	medal_id
1	1	1	2
2	7	6	1
3	8	1	4
4	5	5	1
5	2	6	3
6	6	6	2
7	1	2	1
8	8	4	2
9	5	7	2
10	2	7	1

(10 rows)

Competitor event table before modifying

DELETE FROM COMPETITOR WHERE COMPETITOR_ID=7;
SELECT * FROM COMPETITOR;

DELETE 1

competitor_id	country_id	firstname	lastname	gender	date_of_birth
1	4	Mercedes	Foster	M	1988-12-18
2	4	Samatha	Gardner	F	1984-07-27
3	1	Riley	George	M	1952-08-19
4	1	Clarence	Rodriguez	F	1986-12-23
5	7	James	Pinkard	M	1985-04-08
6	7	Marcia	Hardwick	F	1980-07-30
8	2	Patricia	Kowell	F	1986-06-16
9		Harry	Stephan	M	1988-12-10

(8 rows)

Competitor Table after deleting the Competitor with Competitor_id=7

SELECT * FROM COACHES;

coach_id	competitor_id
2	3
6	1
5	2
8	6
3	5
7	4
4	8

(7 rows)

We observe the competitor with competitor_id=7 has been deleted from the Coaches Table

SELECT * FROM COMPETITOR_EVENT;

competitor_event_id	competitor_id	event_id	medal_id
1	1	1	2
3	8	1	4
4	5	5	1
5	2	6	3
6	6	6	2
7	1	2	1
8	8	4	2
9	5	7	2
10	2	7	1

(9 rows)

We observe the competitor with competitor_id=7 has been deleted from the Competitor_Event Table

4)For the Medal table, we want to make a constraint change, that is to add Unique Constraint to Medal_Name.

This is done so that no two medals have same name

ALTER TABLE MEDAL ADD CONSTRAINT UK UNIQUE(MEDAL_NAME);
INSERT INTO MEDAL VALUES(5,'Gold');

```
ALTER TABLE
psql:additional_queries.sql:31: ERROR:  duplicate key value violates unique constraint "uk"
DETAIL:  Key (medal_name)=(Gold) already exists.
```

After adding the constraint we try to insert a value with medal_name="Gold"
Since Gold already exists in table it raises error .

5)For the Event table, we want to add a Check constraint on Event start_date and end_date.

This is done to ensure that start_date is less than end_date

ALTER TABLE EVENT ADD CONSTRAINT CK
CHECK(START_DATE<=END_DATE);
INSERT INTO EVENT VALUES(10, 6, 2, 'Women 100 meter Archery', 'F', 'F',
'Sahara Desert', '24-NOV-2010', '23-NOV-2010');

```
ALTER TABLE
psql:additional_queries.sql:35: ERROR:  new row for relation "event" violates check constraint "ck"
DETAIL:  Failing row contains (10, 6, 2, Women 100 meter Archery, F, F, Sahara Desert, 2010-11-24, 2010-11-23, null).
```

After adding the check constraint ,we try to insert a value in which start_date is greater than end_date.This raises a error since check constraint is not satisfied.

Migrating databases:

Migration from one RDBMS to another RDBMS is much easier than migrating from an sql database to a non-sql database.

In our case suppose we need to migrate our database from postgres to MySQL we have 3rd party tools like

.pg2mysql converter

.Layer7 Portal Migration Utility

which are an online tools to convert/migrate existing PostgreSQL databases into MySQL..Simply dumping from Postgres and importing to MySQL does not work because of differences in syntax and data types.

Some of the general steps are:

1)Dumping the database, we can either dump the entire database or we can do it table wise.

The command to dumb the database is:

```
pg_dump olympic_database> /tmp/dump.sql
```

2)Converting syntax and data types.

3)Hosting the new file.

4)Importing it into the database.

No-Sql Databases:

There are four main types of no-sql databases namely:

1) Document databases:

A document database stores data in JSON, BSON , or XML documents. It allows the storage of data in an object oriented way and allows the user flexibility to easily change the design of the database as the application develops.

2) Key-Value:

Every data element in the database is stored as a key value pair consisting of an key and a value. A key-value store is like a relational database with only two columns: the key and the value. They are used usually for small scale application and have difficulties scaling up to larger datasets and complex relationships.

3) Column-Oriented Databases:

A column-oriented database is organized as a set of columns rather than rows. Columnar databases can quickly aggregate the value of a given column. Use cases include highly analytical and complex query needs.

4) Graph databases:

A graph database gives equal importance to the relationships between the data points and the data itself. They are suitable and preferred when working with highly connected data with complex relationships. Use cases include social networks.

Since our database is not highly connected, nor is it used for highly analytical usage, we would prefer to switch to a document oriented database like MongoDB. This allows us to think in terms of objects and entities and gives us the flexibility to modify the database design down the road if we chose to.

Let us take the example of MongoDB, to illustrate the process of migrating databases from sql to no-sql structure.

The process often involves more than just extracting and migrating data.

We also need to examine the details of the applications that access your database. For example, if you're using an ORM(Object-relational mapper) that does not support both relational and document databases, we need to find a new library that can connect to MongoDB.

To migrate data, we will need to use another tool to extract it from PostgreSQL and then import it to MongoDB using the **mongoimport** tool.

Before migrating, we might have to fundamentally restructure our data to fit a more document-oriented database instead of our original relational model. This is quite a huge task and we need to carefully look at all our relations and how best to represent them.

LINKS REFERRED:

- 1) [https://www.mongodb.com/compare/mongodb-postgresql/dsl-migrating-postgres to-mongodb](https://www.mongodb.com/compare/mongodb-postgresql/dsl-migrating-postgres-to-mongodb)
- 2) <https://www.cockroachlabs.com/docs/stable/migrate-from-postgres.html>
- 3) <http://www.lightbox.ca/pg2mysql.php>

Contributions

- 1)Prajwal Kamath K- Front End,Schema and Constraint Changes
Hours Spent-4
- 2)Pranav Rajnish- Frontend,Backend,Database Connection
Hours Spent-5
- 3)Raghavendra L -Front End,Migrating Database
Hours Spent-4