CS 1555/2055 – Database Management Systems (Spring 2024)
Dept. of Computer Science, University of Pittsburgh

Project: OlympicDB
— Phase 2 —

Release Date: Mar. 9, 2024 Due: 8:00 PM, Apr. 03, 2024

Purpose of the project

The primary goal of this project is to realize a database system that supports the operations of OlympicDB, a database system for managing the teams, team members, awards, etc. of the Olympic Games. This project has you follow the typical development cycle of a database application (i.e., conceptual design, logical design, and implementation of an application).

The secondary goal is to learn how to work as a member of a team which designs and develops a relatively large, real database application.

Project objectives

The OlympicDB application will be developed in three phases: (1) The OlympicDB database design; (2) The OlympicDB data manipulation; (3) The OlympicDB analytical queries; and (4) The OlympicDB Java App.

You will implement your application program using PostgreSQL, Java and JDBC. The assignment focuses on the database component and not on the user interface (UI). Hence, NO HTML or other graphical user interface is required for this project and carry no bonus points.

It is expected that all members of a team will be involved in all aspects of the project development and contribute equally. Division of labor to data engineering (db component) and software engineering (java component) is not acceptable since each member of the team will be evaluated on both components.

The OlympicDB Schema

In this phase, you are first required to define the relational schema in PostgreSQL of the OlympicDB database as described below (based on a modified version of Phase 1). The key modifications are (1) the additions of two tables (PLACEMENT and MEDAL) in order to track the results, and (2) all IDs have been simplified to a be sequence numbers.

Relations

- OLYMPIAD (olympiad_num, city, country, opening_date, closing_date, website)
- COUNTRY (country_code, country_name)
- SPORT (sport_id, sport_name, description, team_size, date_added)
- PARTICIPANT (participant_id, account, first, middle, last, birth_country, dob, gender)
- ACCOUNT (account_id, username, passkey, role, last_login)
- TEAM_MEMBERS (team, participant)
- TEAM (team_id, olympiad, sport, coach, country, gender, eligible)

- EVENT (event_id, venue, olympiad, sport, gender, date)
- VENUE (venue_name, capacity)
- MEDAL (medal_id, type, points)
- PLACEMENT (event, team, medal, position)

Attribute Datatypes

- VARCHAR(30): venue_name, olympiad_num, city, website, country_name, username, passkey, first, middle, last, sport_name, description
- INTEGER: capacity, account_id, participant_id, coach, team_id, team, event_id, even, medal_id, sport_id, sport, team_size, points, position
- TIMESTAMP: opening_date, closing_date, dob, date, date_added, last_login
- CHAR(3): country_code, country, birth_country
- BOOLEAN: eligible
- gender for participants needs to be described as a DOMAIN using VARCHAR(1) that only contains the following values: { 'M', 'F' }
- gender for team needs to be described as a DOMAIN using VARCHAR(1) that only contains the following values: { 'M', 'F', 'X'}
- role needs to be described as a DOMAIN using VARCHAR(12) that only contains the following values: { 'Organizer', 'Participant', 'Guest' }
- medal and type needs to be described as a DOMAIN using VARCHAR(6) that only contains the following values: { 'Gold', 'Silver', 'Bronze' }

Integrity Constraints

You are expected to define all of the structural and semantic constraints on individual tables, referential triggering actions, and default values. Semantic integrity constraints involving multiple relations should be specified using triggers.

Recall that triggers can be used to make your code more efficient in addition to enforcing integrity constraints. In this second phase, **you are expected to write and submit the following three triggers** at the minimum:

1. **ASSIGN_MEDAL**
   This trigger is responsible for assigning the appropriate medal based on the position when new results are inserted or updated in the PLACEMENT: Gold for Position 1, Silver for Position 2 and Bronze for Position 3.
2. **TEAM_DISQUALIFIED**
   This trigger is responsible for updating the placements when a team is classified as ineligible. So if, for example, a team that came in second for some event was disqualified the teams who came in third or worse should have their positions updated to be one better.

3. **CHECK_TEAM_GENDER**
   This trigger is responsible for making sure that members of a team satisfy the gender constraints. For example, a male participant can only be a member of a Men's or Mixed team.

While you are only required to implement the three triggers described above for this phase, you may find it helpful to implement additional triggers to ensure the integrity of OlympicDB. For example, you could create a trigger which ensures that the teams of an event satisfy the gender constraints. For example, a Women's team can only compete in a Women's or Mixed event.

Data Manipulation Operations for OlympicDB

The second objective of this phase of the project is to familiarize yourself with all the powerful features of SQL/PL, which include functions, procedures, views, and triggers. All tasks can be implemented in database approaches using views, common table expressions, triggers, stored procedures, and functions.

Your system should implement the following operations for managing OlympicDB. Note that you should automatically generate unique IDs for any insertions, unless the task specifies that the primary key is provided.

1. **createAccount**
   Given a username, passkey, role, date and time, add a new account to the system. The last login should be set with the creation date and time. Account IDs should be auto-generated.

2. **removeAccount**
   Given an account_id, remove the account from the system.

3. **addParticipant**
   Given an account_id, first name, middle initial / name, last name, country of birth, dob, and gender, add a new participant. Participant IDs should be auto-generated.

4. **removeParticipant**
   Given a participant_id, delete them and their account from the system.

5. **addTeamMember**
   Given a participant_id and team_id, add the participant to that team. This should activate the CHECK_TEAM_GENDER trigger.

6. **removeTeamMember**
   Given a participant_id and team_id, removes the participant from that team.

7. **registerTeam**
   Given an olympiad_id, sport_id, coach_id, country_id, and participating genders, add a new team to system. Team IDs should be auto-generated.

8. **addEvent**
   Given a venue_id, olympiad_id, sport_id, participating genders, and date, add a new event to the system.

9. **addTeamToEvent**
   Given an event_id and team_id, add a (placeholder) entry in PLACEMENT. As a default, position and metal should be set to a NULL.

10. **addEventOutcome**
    Given an event_id, team_id, and position, update the team's position in PLACEMENT.

11. **disqualifyTeam**
    Given a team_id, mark the team as disqualified by setting their placement position to -1.

12. **listVenuesInOlympiad**
    Given an olympiad_id, displays all the details of the venues.

13. **listEventsOfOlympiad**
    Given an olympiad_id, displays all the details of the events.

14. **listTeamsInEvent**
    Given an event_id, displays the teams in that event.

15. **showPlacementsInEvent**
    Given an even_id, displays the team placements in that event.

16. **listParticipantsOnTeam**
    Given a team_id, display all the information about the members of the team and its coach.

17. **listCountryPlacementsInOlympiad**
    Given a country_id and olympiad_id, display all the placements of that country in all events of that olympiad.

18. **listAthletePlacement**
    Given a participant_id, display all the placements of that participant throughout all olympiads.

Sample Data

In order to help you with debugging, we will provide sample data (via a SQL script), though you may need to generate further data to fully test your OlympicDB system.

Project Submission

Your submission should contain SQL components for the SQL DDL, triggers, and basic data manipulation operations. Specifically,

- **schema.sql** the script to create the database schema.
- **trigger.sql** the script containing definitions of the triggers
- **operations.sql** the script containing the implementation for the data manipulation operations (tasks 1-18)

The project will be collected by submitting your GitHub repository to Gradescope. Therefore, at the beginning of the project, you need to do two things:

1. Create one common private GitHub repository as a team, where all team members are contributing and use it to develop the project.
2. Give full permission of the project repository to your TAs (GitHub ID: sarrisv, GitHub ID: nixonb91).

To turn in your code, you must do three things by each deadline:

1. Make a commit to your project repository that represents what should be graded as your group's submission for that phase.
2. Push that commit to the GitHub repository that you have shared with the TAs
3. Submit your GitHub repository (including all necessary SQL files) to Gradescope under the Project Phase 2 Part X assignment link, where X is 1 or 2.

   To submit to Gradescope, you will need to:

   - Select the appropriate assignment submission link (as you've previously done with homework assignments).
   - On the "Submit Programming Assignment" window that appears, choose "GitHub." If this is your first time submitting to Gradescope via GitHub, you will be prompted to link your GitHub account and authenticate.
   - Select your team's GitHub repository in the dropdown (searching will filter the repositories listed).
   - Select the branch of your GitHub repository with the code your team wishes to submit (typically just the main branch). Then click the green upload button in the bottom left of the window.
   - After uploading your team's submission, you will be taken to the submission results page. The next step is to add each team member to the assignment to allow for a single linked submission. Note: **There should only be one submission per team, i.e., every team member does not need to submit**.
   - To link team members, click "Add Group Member" in the top right corner of this page.
   - On the new window, add all of your corresponding team members to the group, then hit the green "Save" button in the bottom left of this window.
   - If done correctly, every team member should receive a confirmation email from Gradescope.

Multiple submissions are allowed for each part for each team. The last submission before the corresponding deadline will be graded. *NO late submission is allowed.*

Grading

The project will be graded on correctness (e.g., coping with violation of integrity constraints), robustness (e.g., coping with failed transactions) and readability. You will not be graded on efficient code with respect to speed although bad programming will certainly lead to incorrect programs. Programs that fail to compile or run or connect to the database server earn **zero** and *no partial points.*

Academic Honesty

The work in this assignment is to be done *independently* by each team. Discussions with other students or teams on the project should be limited to understanding the statement of the problem. Cheating in any way, including giving your work to someone else will result in an F for the course and a report to the appropriate University authority.

Enjoy the second phase of your class project!