F1 GrandPrix case study

Grandprix is a database which holds data about Formula 1 racing. Such data may be found as open datasets available through Internet (e.g., Kaggle). We will assume that such a dataset has been downloaded and prepared for further analysis. Specifically, the F1.csv file attached contains indicative data about the qualifying positions of the first 10 drivers in two GPs - the Buhrain grand prix and the Saoudi Arabian grand prix.

Assignment 1 (Functional dependencies and joinless decomposition)

The first assignment entails two parts. In the first part you should use PostgreSQL to define a universal relation to hold data in the file F1.csv. Then, using the COPY command upload the data in the csv file. Extend your database by introducing similar data for two different 2022 grand prix of your choice. You can find raw data in the official site of F1 by selecting a grand prix and locating the Qualifying results. Given your data set, investigate possible candidate keys for the table you have created to host the data. You are specifically required to investigate if the pair (GRANDPRIXID, BASELINEPOSITION) can be a candidate key and summarize your response in a short document. In the second part you should assess possible relational decompositions. Specifically, in your effort to assess the possibility of the pair (GRANDPRIXID, BASELINEPOSITION) to be a candidate key for the single-relation schema, you may have also identified additional attribute combinations as being candidate keys. In this part of the assignment you are asked to critically assess the sufficiency of the single-relation schema and devise a suitable decomposition in which all relations are in 3NF or BCNF. You need to justify your solution accordingly. Once you are happy with the designated decomposition you should implement it in PostgreSQL using a suitable dataset subset.

Assignment 2 (New data types)

Assignment 2 requires PostgreSQL so that you can experiment with new data types and the capabilities offered. You will begin with the Postgresql arrays (https://www.postgresql.org/docs/current/arrays.html) and the way in which they can be used to in a relational context. The specific task you will have to assess is described as follows. Imagine that your GRANDPRIX data base needs to be extended so that it includes the lineup of the drivers and their cars in each grand prix. As lineup you will consider the starting positions of the drivers before the actual race. This is obviously different from the final result after the race which is coded in your csv data sets so far. You will then use Postgres array to create an appropriate table depicting the lineup as ardency lists (i.e., who follows / is followed by a driver in the lineup). Once the table is created you may insert a suitable data set for one grandprix and answer queries such as:

- Find the drivers who started the race behind the driver with a code of your choice in the grandprix of your choice

- Find the driver who started the race one position behind the driver with a code of your choice in the grandprix of your choice
- Find the driver who started the race two positions behind the driver with a code of your choice in the grandprix of your choice
- Find the driver who started the race in the up front position in the grandprix of your choice

Assignment 3 (JSON/XML and specialization hierarchies)

Assignment 3 is your first attempt at considering the use of multiple different data types to address the requirements of a problem. As the issues raised are slightly more challenging the time scale in which you are required to complete the assignment is in the range of a few weeks rather than a few days. Specifically, you will continue working on the GRANDPRIX database but this time you will be exposed to additional data types such as JSON, complex user-defined data types as well as inheritance.

Your development team has set new requirements which need to be addressed in your relational design. Firstly, you are required to extend your database to record periodic measurements (i.e., every 5 minutes starting at a designated time) of the outside temperature in the place where the race takes place. For this purpose, the organizers rely on several identical measurement kits spread in three different locations in the circuit of each race. Each measurement taken by these kits records the timestamp of the measurement, the variable measured and the actual value. As for the variables, these include temperature, precipitation, humidity and wind. An indicative measurement could therefore be as follows:

Wednesday 10:00 AM, Mostly sunny Temperature: 13C, Precipitation: 0%, Humidity: 44%, Wind: 37 km/h

The second requirement is that our database should also record live data for each car during the race. Such data are collected by sensors installed throughout the car. Examples of such sensors include temperature sensors of various sorts (i.e., engine and airbox temperature sensors, non-contact temperature sensors that measure friction between parts with infrared energy, etc.), accelerometers which measure g-forces created during turns or braking, pressure sensors measuring hydraulic systems, dual-axis sensors measuring braking and steering, etc. You should select at least five different types of sensors (of your choice) and for each sensor you should define an appropriate/suitable data type to hold measurements during a lap/race of a grand prix. Please note that you should include sensors whose measurements are recorded (i) continuously (by time stamp) during the lap (ii) periodically e.g., every 3' secs for a total of 5 times per lap and (iii) conditionally upon human action by the driver. The latter three categories of sensor types should be used to establish an inheritance hierarchy for thew five sensors to be selected.

To address the above requirements, you need to study the problem at hand, investigate the appropriate data type(s) to be used for each challenge and populate your database with suitable data. Please also consider that some of the data can be hardcoded (by hand) while other may be

obtained as public data from open data sets. Your solution should include enum datatypes, arrays, user-defined data types, JSON and XML datatypes as you consider appropriate and fit.

Assignment 4 (Graphs in Postgres and recursive SQL)

Exercise 4 has the dual objective of firstly familiarizing students with graph development and secondly to allow the analysis of alternative ways of representing graphs using relational technology. Progressively, this analysis will allow the understanding and practice of some advanced PostgreSQL techniques such as the recursive function for traversal and transitive closure of graphs. Students are still working on the current implementation of the GRANDPRIX database where the following will need to be implemented:

4A (Creating a property graph and implementing it in PostgreSQL): You will start by developing (initially on paper or using a tool) a property graph (see theory) that will capture: (a) Relationships between drivers and manufacturers (i.e., Luis Hamilton is a member of the Ferrari team) and (b) the driver making the fastest lap in a grand prix (i.e., Luis Hamilton made the fastest lap at 2025 Bahrain grand prix). You should aim to include sufficient data to make querying the graph meaningful. Having designed the property graph, you should then represent the property graph using relational technology. It is at your discretion to choose a technique from those you have learned so far, i.e. the use of normalized relations, unnormalized relations, adjacency lists, etc., while you should study the way in which the alternative nodes and edges that will exist in your graph will be supported (see PostgreSQL inheritance mechanism).

4B (Trajectory queries with recursion): Having represented the property graph with relations, you should become familiar with modern graph traversal techniques using recursive SQL calls. Specifically, in the current version of the Grandprix database that you have implemented, you should answer the following queries:

- For each / one node calculate the next ones (in a direction of your choice)
- For each / one node calculate the previous ones (in a direction of your choice)
- For each / one node calculate the next ones regardless of direction
- For each / one node calculate the previous ones regardless of direction
- Calculate all possible paths connecting the listed developers in the property graph you implemented

Assignment 5 (No SQL)

The final assignment is to select a NoSQL system of your choice to revisit (a relevant subset of) the GRANDPRIX database. The idea is to gain hands-on experience (only) with a NoSQL system rather than to create yet another (brand new or comparable) implementation with the PostgreSQL version. To this end, I kinldly ask each student to notify me (by email at da@hmu.gr) as to the NoSQL system to be used and a tentative deadline for presenting the work in class.