Mid-term database

ROQUE Balthazar - 12224841

Model Explanation

The **Booking** table represents the main table for its, since it allows to gather all the information necessary for the follow-up of reservations.
A reservation contains:

- vehicle id
- driver id
- game_official's id
- the id of the start trip info
- the id of the end trip info

The **Vehicle** table is used to record all the vehicles of the ITS company as well as various information concerning the vehicles:

- a unique id per vehicle
- a registration id
- the brand
- the model
- the color
- traveler capacity
- whether or not the vehicle is available for a race
- as well as a maintenance id to save all the passages in maintenance of a vehicle

The **Maintenance** table allows to link each maintenance activity on a vehicle thanks from its id it allows you to save:

- the type of maintenance done on the vehicle
- the vehicle's odometer at the time of maintenance
- the cost of the repair
- maintenance description
- the date of maintenance

For each booking, the driver and the passenger must speak the same language.

For this I created the **Base Entity** table, this one allows to create a unique id for each object requiring a list of spoken languages (driver, official_game and country).

By grouping all its objects under the same entity, we obtain the same basis for associating a language with an object. The sqlite does not allow to save a list of elements within a record, the **Associated Lang** table is used to save the base_entity id and the iso_code of the language in pairs, which is saved in the **Language** table.

Thus the table associated_lang with a simple sql query allows from the base_entity_id to retrieve the list of code_iso of the languages spoken:

```
SELECT language_iso_code FROM associated_lang WHERE base_entity_id = base_entity_id_target
```

base_entity_id_target can be the base_entity_id of a driver, a game_official or even a country to retrieve the spoken languages of a country.

The **Driver** table is used to store the list of different drivers within the ITS company. Each driver is defined by:

- a unique id
- a first_name
- a last_name
- a clerence_level
- the id of a fatl if the driver has the certification, in order to retrieve the certification in the FATL table

- the id of an stlv if the driver has the certification, in order to retrieve the certification in the STLV table
- a base_entity_id as explained above in order to subsequently retrieve the list of languages spoken by the driver

The **FATL** table is used to store the different FATL certifications obtained by the different drivers. A fatl is defined by:

- an automatically generated unique id,
- a level
- a qualifying date,
- a driver_id to link the certification to a specific driver.

The **STLV** table identical to the FATL table but to save the STLV certifications. An STLV certification is defined by:

- an automatically generated unique id,
- a level,
- a qualifying date
- a certification body
- a driver_id to link the certification to a specific driver.

The **Game Official** table is used to save the list of game_official . A game_official is defined by:

- a unique id
- a first_name
- a last_name
- the id of the city where it comes from; this id is then used to retrieve information on the city and the country from which the official game comes from in the city and countries tables
- a role id which allows you to retrieve your role in the role table

- a base_entity_id to retrieve the list of languages spoken by the game_official in order to assign it a driver speaking the same language

The **ROLE** table is intended to store the list of the different roles that the game_official has. A role is defined by:

- an automatically generated unique id,
- the name of the role.

I decided to create this table to avoid having dozens of identical roles in the function but which could be written in different ways.

The **CITY** table is used to store the different cities from which the game_official come. A city is defined by:

- an automatically generated unique id
- a name
- a country_id in order to access information about the country in which the city is located from the Countries table.

The **Countries** table is used to store the list of the different countries from which the game_official come. A Country is defined by:

- a unique id generate automatically,
- a country name
- a base_entity_id , in order to retrieve as for drivers and game_official , the list of languages spoken in the country

The fact of linking a language list with a country makes it possible to prefill the languages spoken by a game_official by setting it by default a language spoken in its country of origin.

The **TRIP INFO** table is used to store the information used for a trip.
there are two instances of trip info in the booking . This allows you to save departure and arrival information. A trip_info is defined by:

- an automatically generated unique id
- an address
- a location_type
- a date_time
- the odometer

Script explanation

The first script block allows you to create the database if it is not created. The instance of the database is stored in the variable: my_conn

In the second block, I will initialize an array: queries , in which we find all the SQL statements necessary to create the different tables seen previously in the model if they do not already exist.

At the end of the Block, I will go through the list of queries and for each query, I will execute the table creation query and I will display the errors if there are any with the except part of the script

The next 14 blocks create records for each table.

The remaining blocks display the first 10 records in each table