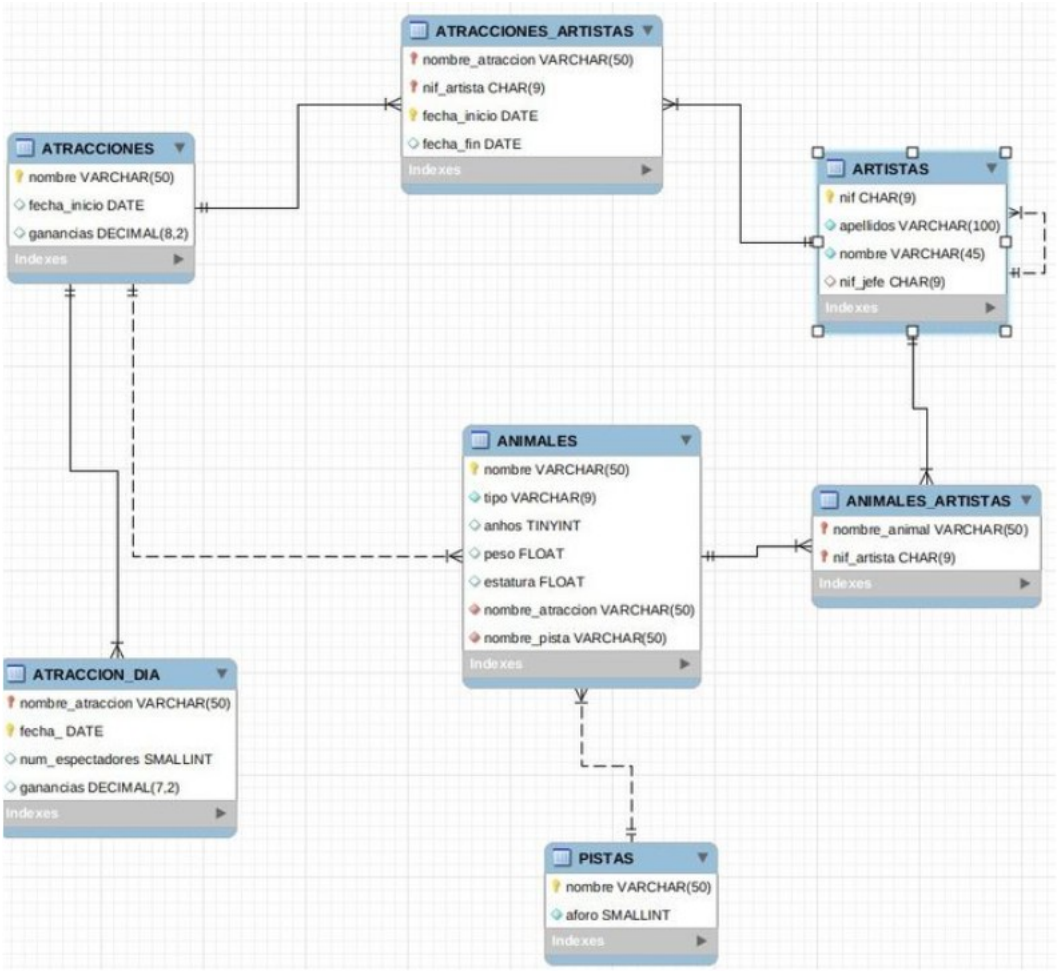
Exercises stored procedures

database circo

We are going to work with the following database called Circo:



Index

1. Exercises of stored procedures (SP)

1. Create a procedure named artistas_getList() that returns the first and last names of the artists separated by commas in the following format: lastname, firstname sorted in descending order.

2. Create a procedure named artists_getListAnimales() that returns the names of the artists along with their ID number as well as the name and weight of the animals that are attended by the artists, sorted by artist ID number and animal name.

3. Create a procedure named atracciones_getListConAntiguedad10() that returns data of attractions that started 10 years ago from the current date. You will have to make use of one of the date-time functions. Try to find out which one.

4. Create a procedure named animales_Leo_getPista() that displays the data of the ring where the animal named 'Leo' works. Use a local variable that stores the name of the ring. Then query the obtained ring using that local variable.

5. Create a procedure named atracciones_getUltima() that obtains the data of the last attraction held (table ATTRACTION_DAY), using local variables. In order to do this, store in a variable the name of the last attraction held and look for the data of that attraction. Keep in mind to limit with the keyword LIMIT the number of rows returned by a query if you are not sure that it will return a single row and you are going to store the data in a variable.

6. Create a procedure named atracciones_getArtistaUltima() that gets the data of the attraction and the artist working at the attraction whose start date has started later in time. It uses two variables: one that stores the artist's ID and another one that stores the name of the attraction.

2. Exercises of SP with input parameters

1. Create a procedure named artistas_getAnimalesPorNif that returns the animals that an artist takes care of. It must take the artist's nif as a parameter.

2. Create a procedure named artistas_getAnimalesPorNombreApel that returns the animals that an artist takes care of. It shall take the artist's first and last name as a parameter. We assume that the first and last names form an alternative key.

3. Create a procedure named atracciones_getListConAntiguedad that returns data for attractions that started a number of years ago in relation to the current date. You will need to use time functions.

4. Create a procedure named artistas_getListMasAnimalesCuida that returns the data of the artist(s) that take care of more animals than the indicated number (parameter sent to it).
Hint: As the query can return more than one artist, we cannot use INTO.

5. Create a procedure named atracciones_getListPorFecha that returns the data of the attractions that have started from the specified date.

Hint: Remember that dates are treated as strings and note the formatting.

Add a new attraction with the current start date.

Call the procedure using the current date minus 3 days (make use of the DATE_SUB and CURDATE functions).

6. Create a procedure named pistas_add that adds a new track.

Regarding data validation, we could have ‘if conditions’ and check if the capacity is greater than zero.

You can use the ROW_COUNT() function to find out how many rows were added, deleted or modified.
Important: The parameters must have the same data type and size as the one defined at column level in the PISTAS table.

7. Create a procedure named atracciones_update that allows modifying the data of an attraction (it is not allowed to update its primary key).

Modify the start date of the attraction 'El gran felino' and set it one day after the current date.

Hint: You will have to save the earnings and the start date it has in order to send that data to the procedure.

Check that when you call the method with an attraction that doesn't exist, row_count will return 0.

8. Create a procedure named pistas_delete that deletes a ring by name. Base the deletion on the pattern name (using the like function). Delete the attraction you added in exercise 6 by sending the first letters (be careful that there is no other attraction stored with those letters at the beginning). Hint: Use the CONCAT function to help you with the LIKE operation.

3. Exercises of SP with output parameters

1. Create a procedure named pistas_getAforo where the name of a ring is passed and its capacity is returned as an output parameter.

2. Create a procedure named artistas_getNumAnimalesCuida for which the nif of an artist is passed and returns as an output parameter how many animals are being cared of.

3. Create a procedure with the name animales_getNombreAforo which receives the name of an animal and returns, using an output parameter and making use of the procedure created in exercise 1, a string with the following format: AnimalName:weight:track:capacity.

Hint: Use the CONCAT function.

4. Create a procedure named artistas_getNumAtracAnimal that receives the surname and first name of an artist and returns, using an output parameter, the number of attractions in which he works and the number of animals he takes care of (using the procedure of exercise 2) with the following format: nif: NumAtractions: NumAnimals.

Note: We assume that there are no artists with the same first and last name.

4. Exercises of SP with input and output parameters

1. Create a procedure called pistas_addAforo that receives as parameters the name of the ring and an amount that represents the increase of the capacity.

The procedure must return in the same parameter the new capacity of the ring.

2. Create a procedure with the name artistas_getNombreCompleto that receives the nif as parameter and returns in the same parameter the full name with the format: surname, name.

3. Create a procedure with the name animales_addAforo where the parameters received are the name of the animal and the increase in the capacity of the ring where the animal is working. It must make use of the stored procedure created in exercise 1 and must return using the two previous parameters, the name of the ring and its new capacity.

5. Exercises of functions

1. Create a function named utilidades_getMesEnLetra that receives a number and returns the name of the month. In case the number does not correspond to any month, it must return 'No existe'.

2. Modify a previous exercise in which we created the procedure named 'atracciones_getNumPorMes' and create a function named utilidades_getMesEnNumero where the name of a month is passed and returns the number that corresponds to that name. If the month does not exist it should return -1. Modify the procedure so that it makes use of the function.

3. Create a function named animales_getEstadoPorAnhos that returns the following string:

If type = Lion

years < 2: 'JOVEN'

years >=2 and <=5: 'MADURO'

years > 5: 'VIEJO'

Any other type:

years < 1: 'JOVEN'

years >=1 and <=3: 'MADURO'

years > 3: 'VIEJO'

Call the function to show the status by year of each of the CIRCUS animals.

4. Create a function named pistas_getDiferenciaAforo in which you pass the new capacity of a ring and it returns the difference between the new capacity and the previous capacity.

- **If difference < 100 it should return the string 'PEQUEÑA menor que 100'.**
- **If the difference is between 100 and 500 it should return the string 'REGULAR entre 100 y 500'.**
- **If the difference > 500 it should return the string 'ABISMAL mayor que 500'.**

Example: TRACK1, 150 => If the ring currently has a capacity of 100, it should return 150-100 = 50 => 'PEQUEÑA menor que 100'. If the ring does not exist, it should return NULL.

Remember to add the appropriate modifiers when creating the function.