EV4. Script Building and NoSQL Databases

Assessable Task 4

Database Management (DBM)
NSM/ASIR

Pau Miñana
2023-2024

To Consider

This activity is optional but evaluative: it may affect your evaluation grade.

> 🚫 If you look for solutions on the internet or ask the ChatGPT oracle, you will only be fooling yourself. Remember that ChatGPT is not infallible or omnipotent. It is a great tool for speeding up your work once you have mastered a subject, but using it as a shortcut when you are acquiring basic skills and knowledge is seriously detrimental to your learning.
>
> Try to solve the activities using the resources we have seen and the extended documentation you will find in the Virtual Classroom.

INDEX

1. Instructions and rules

1.1. Description

You are required to write the necessary scripts to perform the operations described in the statement.

1.2. Delivery deadline and rates

- EVALUATION ratio: 30% of the total grade is for the assessable tasks.
- ACTIVITY ratio: 50% of the assessable tasks grade (there are two per evaluation).
- DEADLINE: *23:59 on Sunday 28th April 2024* (3 WEEKS).

1.3. Grading

Submission is not compulsory and there is no minimum mark. It will be graded from 0 to 10 according to the rating section provided in this document.

1.4. Resources

You should study and consult all the resources given on the virtual classroom, paying particular attention to the non-assessable tasks and all the extra material.

1.5. Plagiarism

You should avoid other students copying your work and take care to prevent this situation.

This is an individual assignment. If authorship is suspected, an oral interview will be required.

1.6. Submission instructions

The assignment must be submitted as a single PDF containing the exercises solved according to the template.

ANY OTHER FORMAT WILL NOT BE ACCEPTED

1.7. Solutions and grading results

You will receive the grade broken down by section, and the overall score, along with any comments that provide suggestions on how you could improve.

2. Procedure to follow

Part 1

1. Download the the script from the Virtual Classroom (ASSESSABLE 3).
2. The physical schema for the relational database is available in this document, although you can generate it with MySQL Workbench by following the steps in this video: https://www.youtube.com/watch?v=8M1eGDkWffk
3. Remember that you need to work with this database, regardless of the diagram you presented in the previous assessable activities.
4. Recommendations:
    1. Design first the procedures and functions and, if needed, create new data to check them.
    2. Use MySQL syntax.
    3. Use ALIAS where appropriate, case sensitivity, spaces and line breaks to make scripts as readable as possible.
    4. Avoid user variables inside the procedures and functions. Use local variables and parameters when needed.
    5. Use the appropriate data types for the variables and parameters.
    6. You can use this lead for error handling.

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'\nUnexpected parameter\n*****\n====> Error message.\n*****\n';
```

Part 2

1. Create scripts for MongoDB to solve the tasks.
2. Recommendations:
    1. You can use as many javascript variables as you consider necessary to make the solution more readable.
    2. Make sure your script returns only the requested records and fields.
    3. Make sure your script is in the indicated format, as structured and readable as possible.
    4. Use uppercase/lowercase and tabs/spaces to make the code as readable as possible.

3. Rating

Part 1

SECTION	RATING
Script 1	1
Script 2	2
Script 3	2
Script 4	2

GRADING CRITERIA

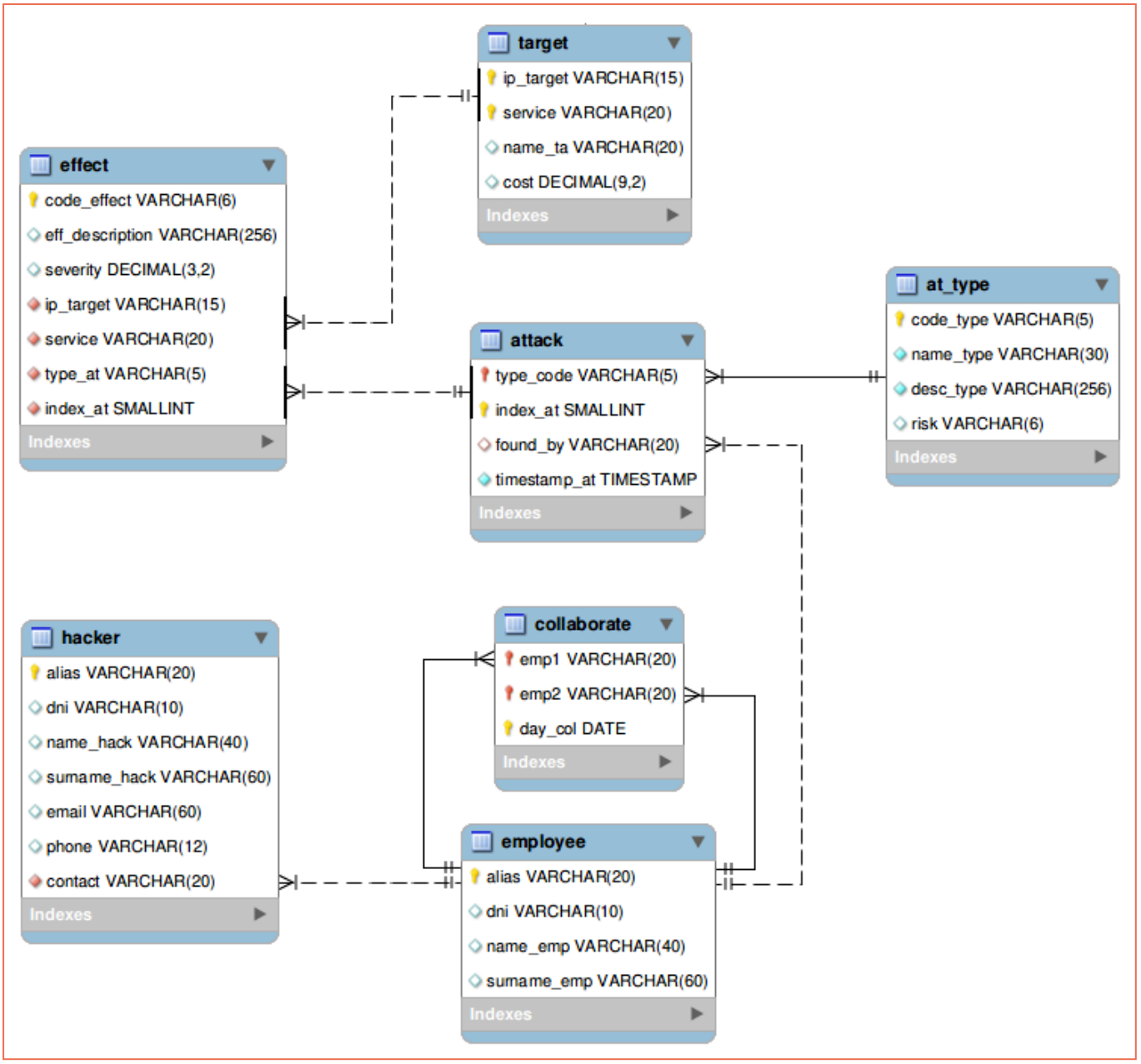
- Script is correct and does what is asked.
- Uses ALIAS where appropriate to make the queries and responses easier and clearer.
- Makes appropriate use of case, spaces and line breaks to make the scripts as readable as possible.
- Uses the appropriate data types for the variables and parameters.
- Uses error messages to avoid unauthorized operations and to report errors in the parameters.
- Checks that the programs work correctly with usage tests.

Part 2

SECTION	GRADING CRITERIA	RATING
Script 1	Deletion and creation of database. Collection creation. Data insertions.	1.5
Script 2	Correct data update. Displays the correct data.	1.5

4. Statement. Cyber Security Division (ASSESSABLE)

4.1. Part 1. Relational Database Scripts.



4.1.1. Script 1

Create a function that returns the number of times a target has been attacked. The function must receive the target identifier and return the number of attacks, or raise an error if the target does not exist.

4.1.2. Script 2

Create a function that receives a timestamp and returns only the date part of it. If the timestamp is not valid the function must raise an error.

Create a procedure that displays the attacks that have been found by a given employee on the same day that they worked in collaboration with another employee. The procedure must receive an employee *alias* and display the attack identifier, the *timestamp_at* of the attack and the *alias* of the employee who worked with them that da

4.1.3. Script 3

Create a procedure that receives a *type_code* and displays a report about the attacks of that type. For each attack, the report must display a table with the *index_at*, *timestamp_at*, *found_by* (1 row) and a second table with the *eff_description*, *severity* and target identifier for each effect of the attack.

You can use a loop to display that information for all the attacks of that type. If the *type_code* is not valid, the procedure must raise an error.

4.1.4. Script 4

Create a procedure to insert a new attack. The procedure must receive a *type_code* and insert the new attack with the current timestamp and the *index_at* field being the next number for that type of attack. The founder of the attack must be the user that is executing the procedure, without host part. If the user is not an employee, then *found_by* will be empty. Make the necessary arrangements to check the procedure works correctly.

If the *type_code* is not valid, the procedure must raise an error. The procedure must "return" 1 if the attack has been inserted by an employee and 0 if it was not.

4.2. Part 2. NoSQL Database Scripts.

We are going to model the data of the attacks and their effects. For this, we need a database in MongoDB called *cyber_security_dep*, with a collection called *attacks* that includes the following fields with the following values:

type	found_by	effects	timestamp
DDoS	MrSmith	ARRAY	2016-01-15T12:34:56
SQLi	Lock	ARRAY	
SQLi		ARRAY	2016-07-31 19:20:21

The *effects* field is an array of documents with the following data:

attack	code	ip	severity	cost
First				
	RFD789	52.58.78.16	0.7	1000
	HGF321	99.86.230.121	0.3	500
Second				
	CVB654	52.58.78.16	0.65	800
	ZQW123	13.249.134.92	0.5	300
	LKJ963	13.249.134.92	0.5	100
Third				
	WSX258	104.18.26.25	0.6	600
	EDC369	104.18.26.25	0.75	800

Attack (first,second and Third) is not a field for the database, just a reference for you to know to which attack the effects belong to.

4.2.1. Script 1. Create database and insert data

Create the script (in javascript) necessary to:

1. Delete the database.
2. Create the database.
3. Create the indicated collection.
4. Insert all the data, taking into account that fields without data should not appear in that element of the collection and the data type should be the most appropriate.

4.2.2. Script 2. Update the information and list data.

Create the script in javascript necessary to:

1. Update the information of the attacks that have the *type* "DDos" and change the *found_by* field to "Neo".
2. List ALL the fields of ALL the records, ordered by *found_by*.
3. List the *type* and *effects* of all the attacks found by "Neo" or without a *found_by* field.
4. List the *type*, *found_by*, *timestamp* and *ip* of the attacks that have any effect with a *cost* greater than 700.