NOMBRE : …………………………………………………………………………………………………

FECHA : ….…………….

EXAMEN

1. How is the data organized within MongoDB?

1. MongoDB can hold up to one database containing many collections where

documents are stored.

1. MongoDB is able to handle multiple databases holding many collections and the

documents are finally stored within them.

1. MongoDB doesn’t have collections, but databases where the documents are

persisted.

1. MongoDB stores data in tables that belong to only one database.

2. JSON is the way documents are represented in MongoDB. The mongoshell, as a javascript interpreter, uses JSON documents intensively to accomplish many different tasks such as CRUD operations, server configuration and so on. Which of the following assertions are correct (many of them might apply)?:

1. The JSON format is also used to persist data on disk, which makes more optimal

to deal with data.

1. JSON documents allow modeling our business entities easily as it enables

storing complex data types such as arrays and nested documents.

1. JSON documents don’t need to follow a schema and hence different

documents from the same collection might have different fields.

1. JSON documents cannot be bigger than 16MB.

3. Choose the correct sentence (only one applies):

1. If a document is inserted without the \_id field, the mongod process creates it

automatically with a unique ObjectId as the value.

1. If a document is inserted without the \_id field, the driver creates it

automatically with a unique ObjectId as the value.

1. Documents cannot have an \_id field.
2. None of the above is correct.

4. Consider the following sentences being executed in a mongoshell session and guess

which answer is the right one:

use mongorules

db.ilikemongodb.drop()

db.ilikemongodb.insert({firstName:"Venancia" , lastName:"García" , role:"teacher" })

db.ilikemongodb.insert({firstName:"Venancia" , lastName:"García" , role:"teacher" })

db.ilikemongodb.insert({firstName:"Venancio" , lastName:"Pérez" , role:"teacher" })

1. The second insert operation will fail because MongoDB doesn’t allow having two

documents with the same fields.

1. None of the operations will fail, but the second insert will overwrite the first one

and only two documents will persist.

1. All the documents will be added, and hence the collection will have three

documents.

1. All the inserts will fail. The \_id is a mandatory field that every document must

specify, otherwise will raise an error.

5. Inserts will fail if… (only one applies):

1. There is already a document in the collection with that \_id.
2. You try to assign an array to the \_id.
3. The argument is not a well-formed document.
4. All the answers above are correct.

8. Based on the following commands in the mongoshell , which affirmations are true? (more

than one can apply):

db.ilikemongodb.insertMany( [

{ \_id:"6748FHJ" , brand:"Fiat" , model:"Uno" },

{ \_id:"6749FHJ" , brand:"Fiat" , model:"Uno" },

{ \_id:4389, companyName:"MongoDD" },

{ \_id:[1,2], CPU:"Intel" , brand:"Dell" },

{ \_id:"6750FHJ" , brand:"Fiat" , model:"Uno" }

],

{ ordered: true } )

1. The documents are sorted by \_id before insertion.
2. The third and four inserts will fail because their \_id data types differ from the \_id

data types in the other operations.

1. The fourth insert will fail because arrays are not allowed in the \_id field.
2. The last insert won’t be executed because of the option { ordered: true }
3. prevent it due to a previous error.

9. When it comes to document deletion, how many commands are available in MongoDB?

1. MongoDB doesn’t allow deleting documents because the NoSQL paradigm is

meant to prevent data loss.

1. Documents in MongoDB can only be deleted one by one with the deleteOneByOne

command to avoid "f at finger " errors.

1. Documents can be deleted in MongoDB by using deleteOne and deleteMany

according to how many documents need to be removed.

1. The deleteOne command is used to remove documents from a Stand-alone

deployment while the deleteMany helps to delete documents in Replica sets and

Sharded Clusters.

10. Given the following query, what’s the expected output?

db.movies.find( { "year" : 1989, "title" : "Batman" } )

1. Return all the documents having the field year equals 1989 and the field

title equals Batman.

1. Display a list of all the collections within the database movies containing

documents where the field year is equal to 1989 and the field title is equal to

Batman.

1. Only pull out all the documents where the field year is equal to 1989 or the field

title is equal to Batman.

1. None of the above is valid as MongoDB always negate query’s predicates and

bring back documents fulfilling the opposite condition.