Answer the following questions

Chapter 6

﻿1. Define a document with respect to document databases.  
**Answer:** Documents in document databases are composed of a set of attribute tags and values. Developers can make up their own set of attribute tags; they are not constrained to a predefined set of tags for specifying structure.

2. Name two types of formats for storing data in a document database.  
**Answer:** JSON and XML.

3. List at least three syntax rules for JSON objects.  
**Answer:**

Data is organized in key-value pairs, similar to key-value databases.

Documents consist of name-value pairs separated by commas.

Documents start with a { and end with a }.

Names are strings, such as "customer\_id" and "address".

Values can be numbers, strings, Booleans (true or false), arrays, objects, or the null value.

The values of arrays are listed within square brackets, such as [ and ].

The values of objects are listed as key-value pairs within curly brackets, such as {and}.

4. Create a sample document for a small appliance with the following attributes:

appliance ID, name, description, height, width, length, and shipping weight. Use the

JSON format.  
**Answer:**   
{ "appliance ID": 12345,

"name": "Mixer A",

"description": "Small 4 whisker",

"height": "30 cm",

"width": "30 cm",

"length": "15 cm",

"shipping weight": "3.2 kg" }

5. Why are highly abstract entities often avoided when modeling document collections?  
**Answer:** Highly abstract entities can lead to document collections with many subtypes. These subtypes will need type indicators to support the frequent filtering required when different document types are in the same collection. Large collections can lead to inefficient retrieval operations.

6. When is it reasonable to use highly abstract entities?

**Answer:** Abstract entities should be used when many of the queries used against a collection apply to all or many subtypes, for example, in a products document collection. Also, if there is a potential for the number of subtypes to grow into the tens or hundreds, it could become difficult to manage collections for all of those subtypes.

7. Using the db.books collection described in this chapter, write a command to insert a

book to the collection. Use MongoDB syntax.

**Answer:** db.books.insert( {"title":"Dám nghĩ lớn", "author": "David J. Schwartz"} )

8. Using the db.books collection described in this chapter, write a command to remove

books by Isaac Asimov. Use MongoDB syntax.

**Answer:** db.books.remove("author": "Hien"})

9. Using the db.books collection described in this chapter, write a command to retrieve

all books with quantity greater than or equal to 20. Use MongoDB syntax.

**Answer:** db.books.find( {"quantity" : {"$gte" : 20 }})

10. Which query operator is used to search for values in a single key?

**Answer:** The $in operator is used to search for a value in a single key.

Chapter 7

1. ﻿Describe how documents are analogous to rows in relational databases.

SQL: INSERT INTO sample (a,b,c) VALUES (1,2,3) 🡺 a = 1, b = 2, c = 3

INSERT INTO sample (a,b,c,d) VALUES (4,5,6,7) 🡺 error

🡺 Field: description VARCHAR(50) 🡺 8, “camnh”, false

🡺 Field: age INT => “camnh” 🡺 err

Document: insertOne({a: 1, b: 2, c:3}) 🡺 ok

insertOne({a: 1, z: 5, c: 7, i:8, m: 9}) 🡺 ok

🡺 Field: age (4,5,6, “camnh” ,false, [], {}) 🡺 ok

**Answer:** Documents are ordered sets of key-value pairs. Keys are used to reference particular values and are analogous to column names in relational tables. Values in a document database are analogous to values stored in a row of a relational database table.

2. Describe how collections are analogous to tables in relational databases.

**Answer:** Collections are sets of documents; tables are sets of rows. Both documents and rows have unique identifiers and may have other attributes as well.

3. Define a schema.

**Answer:** A schema is a formal specification of a database structure.

4. Why are document databases considered schemaless? (schema)

**Answer:** Document databases do not require data modelers to formally specify the structure of documents.

5. Why are document databases considered polymorphic?

**Answer:** A document database is polymorphic because the documents that exist in collections can have many different forms.

6. How does vertical partitioning differ from horizontal partitioning, or sharding?

**Answer:** Vertical partitioning is a technique for improving database performance by separating columns of a relational table into multiple separate tables. This technique is particularly useful when you have some columns that are frequently accessed and others that are not.

Horizontal partitioning is the process of dividing a database by documents in a document database or by rows in a relational database. These parts of the database, known as shards, are stored on separate servers.

7. What is a shard key?

**Answer:** A shard key is one or more keys or fields that exist in all documents in a collection that is used to separate documents into different partitions.

8. What is the purpose of the partitioning algorithm in sharding?

**Answer:** The partitioning algorithm determines how to distribute documents over shards. Common techniques include range, hash, and list partitioning.

9. What is normalization?

**Answer:** Database normalization is the process of organizing data into tables in such a way as to reduce the potential for data anomalies. An anomaly is an inconsistency in the data. Normalization reduces the amount of redundant data in the database.

10. Why would you want to denormalize collections in a document database?

**Answer:** Denormalization is used to improve performance over normalized versions of databases.