MONGODB ASSOCIATE DEVELOPER PRACTICE EXAM (SAMPLE)

STUDY GUIDE

SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://mongodbassociatedev.examzify.com

Table of Contents

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. What type of data operation is not performed within a transaction in MongoDB?

A. Document insertions

B. Single-document updates

C. Database queries

D. Multi-document updates

2. What does the getIndexes() command do?

A. Creates new indexes for a collection

B. Shows the indexes that are being used in a collection

C. Deletes existing indexes from a collection

D. Modifies existing indexes in a collection

3. What is a multikey index?

A. An index with a single unique field

B. An index where one of the fields contains an array

C. An index created automatically by MongoDB

D. An index that cannot be queried

4. How can multiple indexes be deleted in MongoDB?

A. By using db.collection.removeIndexes()

B. By using db.collection.dropIndex()

C. By using db.collection.dropIndexes()

D. By using db.collection.deleteIndexes()

5. Why do indexes incur a write performance cost?

A. Because they must be manually managed

B. Because they only affect read operations

C. Because the index structure must be updated on document changes

D. Because they duplicate data

6. What type of information does the replaceOne() method return?

A. A matched count, modified count, and acknowledgment document

B. A single document with the entire dataset

C. A boolean indicating success or failure

D. An array of modified documents

7. What will happen if you try to query a hidden index?

A. The query will fail

B. The query will proceed without the hidden index

C. The query will return inaccurate results

D. The hidden index will be automatically used

8. What defines single field indexes in MongoDB?

A. Indexes on multiple fields

B. Indexes on only one field

C. Indexes that are automatically created

D. Indexes that optimize aggregate functions

9. Which operation is primarily used to insert a new document if no document matches the filter criteria?

A. insertOne()

B. upsert()

C. replaceOne()

D. updateOne()

10. What arguments does the replaceOne() method accept?

A. query and update

B. filter, replacement, options

C. criteria and document

D. identifier and value

Answers

1. C
2. B
3. B
4. C
5. C
6. A
7. B
8. B
9. D
10. B

Explanations

1. What type of data operation is not performed within a transaction in MongoDB?

A. Document insertions

B. Single-document updates

C. Database queries

D. Multi-document updates

In MongoDB, a transaction is designed to provide a way to execute multiple operations in a way that ensures atomicity, consistency, isolation, and durability (ACID properties). While transactions can encompass write operations such as document insertions, updates, and deletions across multiple documents, they do not include read operations. When considering the nature of database queries, they are fundamentally read operations. A query retrieves data from the database without modifying it. Although read operations can be executed in the context of a transaction, they are not a part of the actual transaction in the sense of modification. Transactions are specifically aimed at ensuring that any changes made to the data are completed fully or not at all, which does not apply to read operations like queries. In the context of this question, it is important to recognize that while multi-document updates, document insertions, and single-document updates can be executed within a transaction, database queries are distinct from these data manipulation operations. Therefore, they are not considered part of the transactional operations, making the identification of queries as the type of operation not performed within a transaction correct.

2. What does the getIndexes() command do?

A. Creates new indexes for a collection

B. Shows the indexes that are being used in a collection

C. Deletes existing indexes from a collection

D. Modifies existing indexes in a collection

The getIndexes() command is utilized to retrieve information about the indexes defined on a specific collection within a MongoDB database. When executed, this command returns an array of documents, each representing an index and its associated properties such as the fields included in the index, the index type, and other metadata. This command is particularly useful for database administrators and developers to understand how data is indexed for query optimization. By showing the indexes that exist on a collection, it helps in assessing whether the current index structure is adequate for the queries being executed, thereby facilitating performance tuning and ensuring efficient data retrieval. Other options describe actions such as creating, deleting, or modifying indexes, which are functionalities not provided by the getIndexes() command. Instead, commands like createIndex(), dropIndex(), and reIndex() are used for those specific purposes.

3. What is a multikey index?

A. An index with a single unique field

B. An index where one of the fields contains an array

C. An index created automatically by MongoDB

D. An index that cannot be queried

A multikey index is specifically designed to handle situations where one of the indexed fields contains an array. In MongoDB, when a field within a document is an array, that field cannot be indexed in the traditional manner since there are multiple values that need to be accounted for. A multikey index allows MongoDB to create an index that includes each element of the array as an individual index key. This type of indexing is particularly useful because it enables efficient queries on documents that contain array fields. For example, if you have a collection of documents where one of the fields is an array of tags, a multikey index on that field allows for quick searches based on the tags present in any of the documents. The other options do not capture the essence of what a multikey index is. An index with a single unique field describes a unique index rather than a multikey. An automatically created index refers to default behaviors in MongoDB, which doesn't specifically imply it being a multikey index. Moreover, an index that cannot be queried would not serve its purpose since the primary function of any index is to facilitate efficient data retrieval.

4. How can multiple indexes be deleted in MongoDB?

A. By using db.collection.removeIndexes()

B. By using db.collection.dropIndex()

C. By using db.collection.dropIndexes()

D. By using db.collection.deleteIndexes()

The command to delete multiple indexes in MongoDB is achieved through the use of `db.collection.dropIndexes()`. This method is specifically designed for dropping all indexes on a collection, which includes the default index on the `_id` field along with any other user-defined indexes. When you invoke `dropIndexes()`, it efficiently removes multiple indexes in one call without the need to specify each index individually, streamlining the process. This is particularly useful when you want to clean up a collection by removing unused or unnecessary indexes, thus potentially improving performance and storage efficiency. The other commands mentioned are either designed for specific actions or do not exist as valid MongoDB commands. For instance, `removeIndexes()` and `deleteIndexes()` are not valid MongoDB commands, and `dropIndex()` is intended for removing a single index at a time, which would not be suitable when you need to delete multiple indexes. Thus, `db.collection.dropIndexes()` is the appropriate choice for this action.

5. Why do indexes incur a write performance cost?

A. Because they must be manually managed

B. Because they only affect read operations

C. Because the index structure must be updated on document changes

D. Because they duplicate data

Indexes incur a write performance cost primarily because the index structure must be updated whenever a document is added, modified, or removed from the database. When a write operation occurs, the database must not only update the document itself but also ensure that the corresponding index entries reflect this change. This additional overhead can slow down write performance, particularly in systems with many indexes or frequent write operations. Each index serves as a separate data structure that optimizes read operations by allowing quick lookups, but maintaining that structure requires resources and time. Therefore, every write to the underlying data necessitates a corresponding update to any indexes that depend on that data. This is why options such as those suggesting manual management or that indexes affect only read operations do not accurately reflect the true nature of the performance costs associated with writing operations. The notion of data duplication can also mislead; while indexes do involve additional storage, the performance implications are chiefly about the maintenance of index structures during writes.

6. What type of information does the replaceOne() method return?

A. A matched count, modified count, and acknowledgment document

B. A single document with the entire dataset

C. A boolean indicating success or failure

D. An array of modified documents

The replaceOne() method in MongoDB is designed to replace a single document that matches a specified filter with a new document. When this method is executed, it returns a result that includes several important pieces of information: the matched count, the modified count, and an acknowledgment document. The matched count indicates how many documents matched the filter criteria specified; it helps you understand if the operation found a document that was eligible for replacement. The modified count reflects whether a document was actually modified by the operation, which is particularly useful when the new document is identical to the existing one, as in that case, the modified count would be zero. Lastly, the acknowledgment document provides a confirmation that the operation was completed successfully. This behavior contrasts with other methods and types of returns in MongoDB. For example, a boolean indicating success or failure would not provide sufficient detail regarding how many documents were affected, nor would it offer insight into the nature of the operation's success. Similarly, returning a single document with the entire dataset or an array of modified documents does not accurately reflect the intention and result of a replaceOne() operation. This method's return structure is specifically designed to give developers comprehensive feedback about the execution of the replace operation.

7. What will happen if you try to query a hidden index?

A. The query will fail

B. The query will proceed without the hidden index

C. The query will return inaccurate results

D. The hidden index will be automatically used

When querying a hidden index in MongoDB, the query will proceed without utilizing the hidden index. Hidden indexes are designed to remain in the database for various purposes, such as maintaining historical data or testing query performance, but they are not considered during query execution unless explicitly referenced. This allows developers to retain the index and potentially analyze its effectiveness without impacting the performance of ongoing queries. In terms of query execution, when a hidden index is present, it does not obstruct or disrupt the process, allowing MongoDB to either use an alternate index that is not hidden or perform a collection scan to retrieve the data, depending on the nature of the query and the available indexes. This characteristic demonstrates how hidden indexes provide flexibility and do not interfere with the performance of database operations when not needed.

8. What defines single field indexes in MongoDB?

A. Indexes on multiple fields

B. Indexes on only one field

C. Indexes that are automatically created

D. Indexes that optimize aggregate functions

Single field indexes in MongoDB are specifically designed to optimize queries on a single field within a document. When a single field index is created, it allows MongoDB to quickly locate documents based on the value of that field. This is especially beneficial for queries that involve filter criteria based on a specific field, as it reduces the amount of data that needs to be scanned and improves query performance. The efficiency of single field indexes arises from their structure, which is optimized for quick lookups. By focusing on one field, MongoDB can maintain a straightforward index structure that speeds up the retrieval process significantly compared to full collection scans. These indexes also support various types of queries for the indexed field, including equality matches, range queries, and sorting, thereby enhancing the overall database performance. In contrast, indexes on multiple fields involve combinations of values from two or more fields and are designed for query patterns that filter based on those combined values. Automatic index creation refers to indexes that MongoDB may create on the default `_id` field for uniqueness and quick lookups. Indexes that optimize aggregate functions pertain to more complex operations rather than the simple retrieval of documents based on a single field. Therefore, the definition of single field indexes is distinct and emphasizes their focus on a single

9. Which operation is primarily used to insert a new document if no document matches the filter criteria?

A. insertOne()

B. upsert()

C. replaceOne()

D. updateOne()

The operation that is primarily designed to insert a new document if no document matches the filter criteria is the upsert operation. An upsert operation is a combination of an update and an insert; it updates a document if a match is found, and if no match is found based on the filter criteria, it inserts a new document with the specified values. Using updateOne with the upsert option set to true allows you to achieve this functionality. When invoking the updateOne method, setting the upsert parameter to true instructs MongoDB to create a new document if the filter criteria don't match any existing documents. Thus, the upsert capability is crucial for situations where you want to ensure that a document is created in the event that it does not already exist in the database. The other operations mentioned, such as insertOne, replaceOne, and updateOne without the upsert option, do not directly provide this insert-if-not-found functionality. For instance, insertOne only adds a new document, while replaceOne and updateOne are focused on updating existing documents without an inherent mechanism to check the presence of a matched document and insert a new one if needed, unless specifically configured to do so with upsert. Therefore, the upsert operation is essential for

10. What arguments does the replaceOne() method accept?

A. query and update

B. filter, replacement, options

C. criteria and document

D. identifier and value

The replaceOne() method is specifically designed to replace a single document within a MongoDB collection that matches a specified filter. It accepts the following arguments: a filter, a replacement document, and optional parameters. The filter is used to locate the document that needs to be replaced, while the replacement document contains the new data that will completely overwrite the existing document. The optional parameters can include settings like whether to upsert the document (insert it if it does not exist) or modifiers for the operation. This method is crucial for performing bulk updates or for ensuring that an exact match of criteria leads to the intended replacement in the database. By utilizing this method correctly, developers can ensure data integrity and achieve the intended effect of document replacement efficiently.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://mongodbassociatedev.examzify.com

We wish you the very best on your exam journey. You've got this!