Q&A

Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt.
----- Guaranteed.

KILL EXAMS

PASS ✓

killexams.com

MongoDB

C100DEV

MongoDB Certified Developer Associate 2024

100%

BEST

100

ORDER FULL VERSION

and then calls limit.

Question: 269

In a MongoDB application where documents may contain various nested structures, which BSON type would be most suitable for storing data that includes both a list of items and metadata about those items?

A. Array
B. Object
C. String
D. Binary Data

Answer: B

Explanation: The Object BSON type is suitable for storing complex data structures that include metadata alongside other data types, allowing for a structured representation of nested information.

Question: 270

In a scenario where you manage "Products," "Orders," and "Customers," which of the following data modeling choices is likely to create an anti-pattern by introducing redundancy and complicating the update process for product information?

A. Embedding product details within each order document
B. Storing orders and customers as separate collections with references to products
C. Maintaining a separate "Product" collection linked to orders through product IDs
D. Embedding customer information within order documents for quick access

Answer: A

Explanation: Embedding product details within each order document introduces redundancy, as product information may be repeated for every order. This complicates the update process and increases storage requirements, which is an anti-pattern in data modeling.

Question: 271

In the MongoDB Python driver, how would you implement an aggregation pipeline that calculates the average "price" for products grouped by "category" in the "products" collection?

A. pipeline = [{ "$group": { "_id": "$category", "averagePrice": { "$avg": "$price" } } }]
B. pipeline = [{ "group": { "category": "$category", "avgPrice": { "$avg": "$price" } } }]
C. collection.aggregate([{ "$group": { "_id": "$category", "avgPrice": { "$avg": "$price" } } }])
D. pipeline = [{ "$average": { "$group": { "_id": "$category", "price": "$price" } } }]

Answer: C

Explanation: The correct syntax for the aggregation pipeline uses $group to aggregate the results and calculate the average.

Question: 272

You need to enrich a dataset of users with their corresponding purchase history from another collection. You plan to use the $lookup stage in your aggregation pipeline. What will be the structure of the output documents after the $lookup is executed?

A. Each user document will contain an array of purchase documents that match the user ID.
B. Each purchase document will contain an array of user documents that match the purchase ID.
C. Each user document will contain a single purchase document corresponding to the user ID.
D. The output will flatten the user and purchase documents into a single document.

Answer: A

Explanation: The $lookup stage allows you to join documents from one collection into another, resulting in each user document containing an array of purchase documents that match the user ID. Option B misrepresents the direction of the join. Option C incorrectly assumes a one-to-one relationship. Option D misunderstands how MongoDB handles joined data.

Question: 273

You need to replace an entire document in the inventory collection based on its itemCode. The command you are executing is db.inventory.replaceOne({itemCode: "A123"}, {itemCode: "A123", itemName: "New Item", quantity: 50}). What will happen if the document does not exist?

A. A new document will be created with the given details.
B. The command will fail because the document must exist to be replaced.
C. The command will succeed, but no changes will be made since the document is missing.
D. The command will log a warning but will not create a new document.

Answer: A

Explanation: The replaceOne command with upsert set to true (which is implicit) will create a new document if no document matches the query. However, since upsert is not specified, it will not create a new document in this case.

Question: 274

In the context of MongoDB's aggregation framework, which of the following operations can be performed using the aggregation pipeline in the MongoDB driver?

A. Filtering documents based on specific criteria.
B. Grouping documents by a specific field and performing calculations.
C. Sorting the results of a query based on specified fields.
D. All of the above.

Answer: D

Explanation: The aggregation pipeline in MongoDB allows for filtering, grouping, and sorting of documents, making it a powerful tool for data transformation and analysis.

Question: 275

You need to delete a document from the users collection where the username is "john_doe". The command you intend to use is db.users.deleteOne({username: "john_doe"}). What happens if multiple documents match this criteria?

A. All documents with the username "john_doe" will be deleted.
B. Only the first document matching the criteria will be deleted.
C. The command will fail since multiple matches exist.
D. No documents will be deleted, and an error will occur.

Answer: B

Explanation: The deleteOne command removes only the first document that matches the specified filter. Even if multiple documents match, only one will be deleted.

Question: 276

You have a requirement to insert a document into the users collection with a unique identifier. The command you execute is db.users.insertOne({userId: "user001", name: "John Doe"}). If this command is repeated without removing the existing document, which outcome will occur?

A. The command will succeed, and the existing document will be duplicated.
B. The command will fail due to a unique constraint violation on userId.
C. The existing document will be updated with the new name.
D. The command will throw an error indicating a missing required field.

Answer: B

Explanation: If userId is a unique field, attempting to insert a document with the same userId will result in an error due to the unique constraint violation, preventing the insertion.

Question: 277

In the MongoDB Go driver, what is the correct syntax for finding a single document in the "employees" collection where the "employeeId" is 12345?

A. collection.FindOne(context.TODO(), bson.M{"employeeId": 12345})
B. collection.FindOne(context.TODO(), bson.D{{"employeeId", 12345}})
C. collection.FindOne(bson.M{"employeeId": 12345})
D. collection.Find(bson.M{"employeeId": 12345}).Limit(1)

Answer: B

Explanation: The FindOne method takes a filter as a parameter, and using bson.D is a common way to construct the filter in the Go driver.

Question: 278

You have a collection called transactions with fields userId, transactionType, and createdAt. A query is scanning through the collection to find all transactions of a certain type and then sorts them by createdAt. What index should you create to enhance performance?

A. { transactionType: 1, createdAt: 1 }
B. { createdAt: 1, userId: 1 }
C. { userId: 1, transactionType: -1 }
D. { transactionType: -1, createdAt: -1 }

Answer: A

Explanation: An index on { transactionType: 1, createdAt: 1 } allows efficient filtering on transactionType while providing sorted results by createdAt, thus avoiding a collection scan and optimizing query execution time.

Question: 279

In a MongoDB collection where some documents include nested arrays, which query operator would be most effective in retrieving documents based on a specific condition related to the elements of those nested arrays?

A. $unwind
B. $or
C. $not

D. $where

Answer: A

Explanation: The $unwind operator is specifically designed to deconstruct an array field from the input documents to output a document for each element, making it effective for querying nested arrays based on specific conditions.

Question: 280

When utilizing the MongoDB C# driver, which of the following methods would you employ to bulk insert multiple documents efficiently, taking advantage of the driver's capabilities?

A. InsertManyAsync()
B. BulkWrite()
C. InsertAll()
D. AddRange()

Answer: B

Explanation: The BulkWrite() method is designed for efficiently performing bulk operations, including inserts, updates, and deletes, in a single call, which improves performance.

Question: 281

When querying a MongoDB collection where documents may contain an array of sub-documents, which of the following methods or operators would be most effective for retrieving documents based on a condition applied to an element within the array?

A. $exists

B. $elemMatch
C. $type
D. $size

Answer: B

Explanation: The $elemMatch operator allows for precise querying of documents by applying conditions to elements within an array. This is particularly effective when dealing with complex data structures that include arrays of sub-documents.

Question: 282

You have a collection named orders that contains documents with fields customerId, amount, and status. You execute the following query: db.orders.find({ status: 'completed' }).sort({ amount: -1 }).limit(5). Given that amount values are non-unique, what will be the expected output format when you retrieve the documents?

A. An array of the top 5 completed orders with the highest amounts, sorted in descending order by amount.
B. An array of all completed orders regardless of amount, sorted in ascending order.
C. A single document representing the highest completed order only.
D. An empty array if there are no completed orders.

Answer: A

Explanation: The query filters for completed orders, sorts them by amount in descending order, and limits the results to 5 documents, thus returning the top 5 completed orders based on amount.

Question: 283

In a complex aggregation pipeline, you observe that certain stages are significantly slower than others. If you find that a stage is not utilizing an index, which of the following options would be the best initial step to investigate and potentially resolve this performance bottleneck?

A. Increase the size of the aggregation pipeline
B. Analyze the query with the explain() method to check index usage
C. Rewrite the aggregation pipeline to simplify its stages
D. Increase the server's hardware resources

Answer: B

Explanation: Using the explain() method provides insights into how the aggregation stages are executed and whether indexes are being utilized. This information is crucial for identifying potential issues and optimizing performance.

Question: 284

In a music library application with "Artists," "Albums," and "Tracks," where each artist can produce multiple albums and each album can contain multiple tracks, which of the following data modeling approaches would likely lead to redundancy and inefficiencies in retrieving album and track information?

A. Embedding track details within album documents
B. Storing artists and albums in separate collections linked by artist IDs
C. Keeping all entities in a single collection for ease of access
D. Maintaining a separate collection for tracks linked to albums through IDs

Answer: C

Explanation: Keeping all entities in a single collection for ease of access can

lead to redundancy and inefficiencies in retrieving album and track information. This anti-pattern complicates data retrieval and can hinder the performance of the application.

KILLEXAMS.COM

Killexams.com is an online platform that offers a wide range of services related to certification exam preparation. The platform provides actual questions, exam dumps, and practice tests to help individuals prepare for various certification exams with confidence. Here are some key features and services offered by Killexams.com:
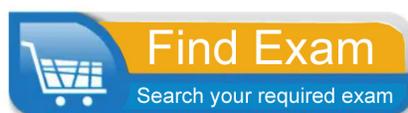
Actual Exam Questions: Killexams.com provides actual exam questions that are experienced in test centers. These questions are updated regularly to ensure they are up-to-date and relevant to the latest exam syllabus. By studying these actual questions, candidates can familiarize themselves with the content and format of the real exam.

Exam Dumps: Killexams.com offers exam dumps in PDF format. These dumps contain a comprehensive collection of questions and answers that cover the exam topics. By using these dumps, candidates can enhance their knowledge and improve their chances of success in the certification exam.

Practice Tests: Killexams.com provides practice tests through their desktop VCE exam simulator and online test engine. These practice tests simulate the real exam environment and help candidates assess their readiness for the actual exam. The practice tests cover a wide range of questions and enable candidates to identify their strengths and weaknesses.

Guaranteed Success: Killexams.com offers a success guarantee with their exam dumps. They claim that by using their materials, candidates will pass their exams on the first attempt or they will refund the purchase price. This guarantee provides assurance and confidence to individuals preparing for certification exams.

Updated Content: Killexams.com regularly updates its question bank and exam dumps to ensure that they are current and reflect the latest changes in the exam syllabus. This helps candidates stay up-to-date with the exam content and increases their chances of success.

Technical Support: Killexams.com provides free 24x7 technical support to assist candidates with any queries or issues they may encounter while using their services. Their certified experts are available to provide guidance and help candidates throughout their exam preparation journey.