Practice questions

Due Dec 4 at 23:59 **Points** 90 **Questions** 24
Available Nov 19 at 0:00 - Dec 4 at 23:59 16 days **Time Limit** 100 Minutes
Allowed Attempts Unlimited

Instructions

The practice quiz contains only **90** points worth of questions. There are **19 MCQ questions** and **5 short answer questions**. This does NOT reflect the question type distribution in the final exam.

The practice quiz includes possible types of questions you may see in the exam. In particluar:

- Some short answer questions are based on a common scenario. For instance question 11 ~16 are based on a common scenario put in front of question 11.
- Many short answer questions have multiple parts and you need to answer all of them in a single text field. Make sure you label the answer accordingly.
- Some question may contain embedded image, make sure your browser can display them properly

It is also worth noting that the practice quiz does NOT reflect the difficult level of the final exam. It does NOT reflect the content coverage and distribution of the final exam either.

The practice quiz has a few settings different to the final exam. It is set to allow multiple attempts and to show correct response for MCQ questions. In the final exam, you are only allowed one attempt and you will not see the correct response after submission.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 4	69 minutes	38 out of 90 *
LATEST	Attempt 4	69 minutes	38 out of 90 *

Attempt 3	100 minutes	34 out of 90 *
Attempt 2	11 minutes	22 out of 90 *
Attempt 1	100 minutes	18 out of 90 *

* Some questions not yet marked

Submitted Dec 2 at 17:43

Question 1

2 / 2 pts

Which one of the following is TRUE about shard in MongoDB?

○ Each shard is a replica of the other shard in the cluster

○ Each shard holds a continuous range of shard key values of a collection

Correct!

● Each shard can have more than one chunks of a collection

○ The primary shard is responsible for write operations in the shard

Question 2

2 / 2 pts

Assuming a Bigtable tablet contains wide rows and the data about the same row are usually inserted/updated by multiple write operations. A read query looking for one such wide row would likely to assemble the result from

○ The log file and one or more SSTable files belonging to this tablet

○ The log file and memories of all tablet servers that have handled the write query of this row.

Correct!

◉ The memory of the tablet server serving this tablet and one or more SSTable files belonging to this tablet

○ The memory of the tablet server serving this tablet and the log file of this tablet

Question 3

2 / 2 pts

Master-Slave Replication is a simple scale out option. Which one of the statements is TRUE about this option?

○ The slaves should keep a consistent state with the master at all times

○ The master should not receive any read request

Correct!

◉ The master should receive all write requests

○ There should be one master and two slave nodes

Question 4

2 / 2 pts

Which one of the following is NOT TRUE about SSTable in Bigtable?

Correct!

● They are organized as table format

○ Data about one row may be stored in multiple SSTable files

○ They are immutable

○ They are created by memory flush or compaction

Question 5

2 / 2 pts

Which one of the following is NOT TRUE about MongoDB aggregation stage?

○ The output document number of $project stage is always equal to its input document number

○ The output document number of $unwind stage may be greater than, equal to or less than its input document number

Correct!

● The output document number of $lookup stage is always less than or equal to its input document number.

○ The output document number of $group stage is always less than or equal to its input document number.

Question 6

2 / 2 pts

Bigtable stores multiple versions of a column by design. This is achieved by

Correct!

◉ Adding a timestamp to each value

○ Adding a timestamp to each row

○ Adding a timestamp to each column

○ Adding a timestamp to each column family

Question 7

2 / 2 pts

Which of the following is TRUE about ROOT and METADATA tablet in Bigtable?

○ All queries to METADATA tablets should go through master; client cannot query METADATA tablet directly.

Correct!

◉

The METADATA table may split and its tablets can be managed by different tablet servers

○ The ROOT tablet is stored in Chubby to ensure strong consistency and durability

○ The ROOT tablet stores information about the tablet servers, each row represents a tablet server.

Question 8

2 / 2 pts

Which one of the following is NOT TRUE about Bigtable Architecture

○ There is only one master server in the cluster

○ A table may be split into many tablets and managed by different tablet servers

Correct!

● The master server is responsible for all write operations

○ A tablet server may manage tablets belonging to many different tables

Question 9

2 / 2 pts

Which one of the following is NOT TRUE about Chubby service in Bigtable?

○ Chubby service ensures that there is at most one active master server at any time

○ Chubby service stores the root tablet location

○ Chubby service knows the list of tablet servers in Bigtable

Correct!

● Chubby service is contacted in every read/write operation

Question 10

2 / 2 pts

MongoDB uses GeoJSON object to store spatial data. What spatial object(s) is(are) defined by the following GeoJSON object?

```
{type: "MultiPolygon",
  coordinates : [
    [ [ [ 0, 0], [ 3, 6], [ 6, 1], [ 0, 0]],
    [ [ 2, 2], [ 3, 3], [ 4, 2], [ 2, 2]]],
    [ [ [ 0, 0], [ 0, 6], [ 6, 6], [ 6,0],[0,0]] ]
 ]
}
```

○ A polygon with one triangle exterior and two holes: a triangle and a square

○ A collection of three polygons: two triangles and one square

Correct!

● A collection of two polygons: one triangle with a triangle hole and one square

○ A polygon with one square exterior and two triangle holes

Question 11~16 is based on a MongoDB database **book_review**. The database contains two collections: **books** and **reviews.** The database is designed to keep information about books and their reviews. Each document in the **books** collection stores information about a particular book. All **book** documents include the following fields: **_id**, **parent_id**, **title**, **author**, **publisher**, **edition**. The **parent_id** field is used to associate different editions of the same book. If a book has published many editions, there will be one document for each edition, all of which will have the same **parent_id** value. The **parent_id** value will be the first edition's **_id** value.
Each document in the **reviews** collection stores a review of a particular book. A **review** document has a unique **_id**, it also stores the **_id** of the book the review is about. It may include the review text, reviewer's name, time of the review, the rating given by the reviewer and helpful vote count of this review. Note that **review_time** is of **Date** type, for simplicity, the **Date** literal is written in string format.

The following indexes have been created:

```
        db.reviews.createIndex({rating:-1, review_time:-1
, helpful_vote:-1,});
        db.reviews.createIndex({review_text: "text"})
        db.books.createIndex({parent_id:1})
       db.books.createIndex({title: 1})
```

Below are two sample documents: a **book** document and a **review** document of the book:

```
{
        _id: 5,
        parent_id: 5,
        title: “Sapiens: A Brief History of Humankind”,
        author: “Yuval Noah Harari”,
        publisher: “Harper Perennial”,
        edition: 1
}
```

```
{
        _id: 1,
        book_id: 5,
        reviewer: “Bill Gates”,
        review_text: “What’s unique about Harari’s take i
s that he focuses on the power of stories and myths to br
ing people together”,
        rating: 5,
        review_time: “2016-05-17”,
        helpful_vote: 3109
}
```

rrect Answer

Question 11

2 / 2 pts

In which one of the following queries, index will not be used in the query plan?

○ `db.books.find({parent_id: 1, editions: {$gt:1}})`

○

```
db.reviews.find({rating: {$gt: 3} , helpful_vote: {$gt:100}})
```

Correct!

●

```
db.reviews.find({review_text: {$regex: “^Sapien”} , helpful_vote: {$gt:100}})
```

○

```
db.books.find({title: {$regex: “^Sapien”},{edition:1})
```

Question 12

2 / 2 pts

Assume an early query **books.find({publisher: "ABC"}).count()** returns 5. The client then issues an update query to change the name of the publisher and to add a field to store the location of the publisher.

```
db.books.update({publisher: "ABC"}, {$set: {publisher:  " ABC Inc.", Location: "NYC"}})
```

Which of the following is NOT TRUE?

○

A concurrent query **db.books.find({Location: "NYC"})** may return 1~5 documents with the following fields values: **{publisher: "ABC Inc", Location: "NYC"}**

○

A concurrent query **db.books.find({publisher: "ABC Inc."}).count()** may return any number between 0 and 5.

○

A concurrent query **db.books.find({publisher: "ABC"}).count()** may return any number between 0 and 5

Correct!

●

A concurrent query **db.books.find({Location: "NYC"})** may return 1 ~5 documents with the following fields values: **{publisher: "ABC", Location: "NYC"}**

Question 13

2 / 2 pts

Which one of the following queries cannot use index for sorting?

○

```
db.reviews.find({rating: 4).sort({review_time: 1})
```

○

```
db.reviews.find({rating: {$gte:4}, , helpful_vote: {$gte:100}}).sort({rating:-1})
```

○

```
db.reviews.find({rating: {$gte:4}}).sort({rating:1,review_time:1})
```

Correct!

●

```
db.reviews.find({rating: 4, helpful_vote: {$gte:100}}).sort({helpful_vote:-1})
```

Question 14

2 / 2 pts

What does the following query return?

```
db.reviews.find ({book_id: 3}, {review_text:1, rating:1}).sort({review_time: -1}).limit(1)
```

○ The latest review of a book with text equals 1, rating equals 1 and id equals 3

○ The latest review text with rating equals 1 of a book with id equals 3

Correct!

● The latest review text and rating of a book with id equals 3

○ The query does not return anything because the syntax is wrong

Question 15~16 are related with the following aggregation:

```
db.books.aggregate([
      {$match:{parent_id:1}},
      {$lookup:{
                  from: "reviews",
                  localField: _id,
                  foreignField: "book_id",
                  as: "reviews"
       }},
      {$project:{reviews:1, review_count:{$size: "$review
s"}}},
      {$match:{review_count:{$gte:1}}},
      {$unwind: "$reviews"},
      {$group:{_id:null, total_reviews:{$push:"$reviews"}
}}
])
```

Question 15

2 / 2 pts

Which fields are included in the output document of the **$project** stage of the aggregation?

Correct!

● _id, reviews, review_count

○ book_id, reviews, review_count

○ reviews, review_count

○ parent_id, title, author, publisher, edition, reviews, review_count

Question 16

2 / 2 pts

Which stage of the following aggregation would not change the structure of input documents?

○ The $lookup stage

Correct! ● The $match stage

○ The $project stage

○ The $group stage

Question 17-19 are related with the following scenario:

A **restaurants** collection containing the following 4 documents:

```
{_id: 1, name: "Central Perk Cafe", violations: 3}
{_id: 2, name: "Rock A Feller Bar and Grill", violations: 2}
{_id: 3, name: "Empire State Sub", violations: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

The collection is stored in a replica set with three members. All members have the same copy of the data at the beginning of the scenario. The following write query is sent to this collection:

```
db.restaurant.updateMany(
      { violations: { $gt: 4 } },
      { $set: { Review : true , reviewer: "R. Coltrane"} }
 )
```

The write was completed in primary at t_0. It was completed in secondary 1 at t_2 and completed in secondary 2 at t_3; The primary receives acknowledgement from secondary 1 in t_4; the secondary 1 receives notification from primary to update its write concern majority copy at t_5; the secondary 2 receives notification from primary to update its write concern majority copy at t_6. Note that subscript of time indicates order. For instance, t_0 is the first time and t_2 is before t_3.

The following concurrent read query also sent to the same collection. There is no other concurrent write.

```
db.restaurant.find({violations: { $gt: 4 }})
```

Question 17

2 / 2 pts

Assume the read preference is set to *secondary* and the read concern is set to *majority*. Secondary 2 receives the read quest between t_4 and t_5. What would be the results of the query?

Correct!

●
```
{_id: 3, name: "Empire State Sub", violations: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

○
```
{_id: 3, name: "Empire State Sub", violations: 5, Review: true, Reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8, Review: true, Reviewer: "R. Coltrane"}
```

○
```
{_id: 3, name: "Empire State Sub", violation: 5, Review: true, reviewer: "R. Coltrane" }
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8}
```

○
```
{_id: 3, name: "Empire State Sub", violations: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8, Review: true, Reviewer: "R. Coltrane"}
```

Question 18

2 / 2 pts

Assume the read preference is set to *primary*; and the read concern is set to *local*. Which of the followings could NOT be the results of the read query?

○

```
{_id: 3, name: "Empire State Sub", violation: 5, Review :
true, Reviewer: "R. Coltrane" }
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8 , Rev
iew : true, Reviewer: "R. Coltrane"}
```

○

```
{_id: 3, name: "Empire State Sub", violation: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8 , Rev
iew : true, Reviewer: "R. Coltrane"}
```

Correct!

●

```
{_id: 4, name: "Pizza Rat's Pizzaria", violation : 8 , Re
view : true, Reviewer: "R. Coltrane" }
```

○

```
{_id: 3, name: "Empire State Sub", violations: 5, Review:
true, Reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

Question 19

2 / 2 pts

Now assume the read preference is set to *primary*; and the read concern is set to *majority*. The primary receives the read request between t_4 and t_5, what could be the results of the read query?

Correct!

●

```
{_id : 3, name : "Empire State Sub", violation : 5, Revie
w : true, reviewer: "R. Coltrane" }
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 , R
eview : true, reviewer: "R. Coltrane"}
```

○

```
{_id: 3, name : "Empire State Sub", violation : 5}
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 }
```

○

```
{_id: 3, name : "Empire State Sub", violation : 5}
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 , R
eview : true, reviewer: "R. Coltrane"}
```

○

```
{_id: 3, name : "Empire State Sub", violation : 5, Review
: true, reviewer: "R. Coltrane" }
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8}
```

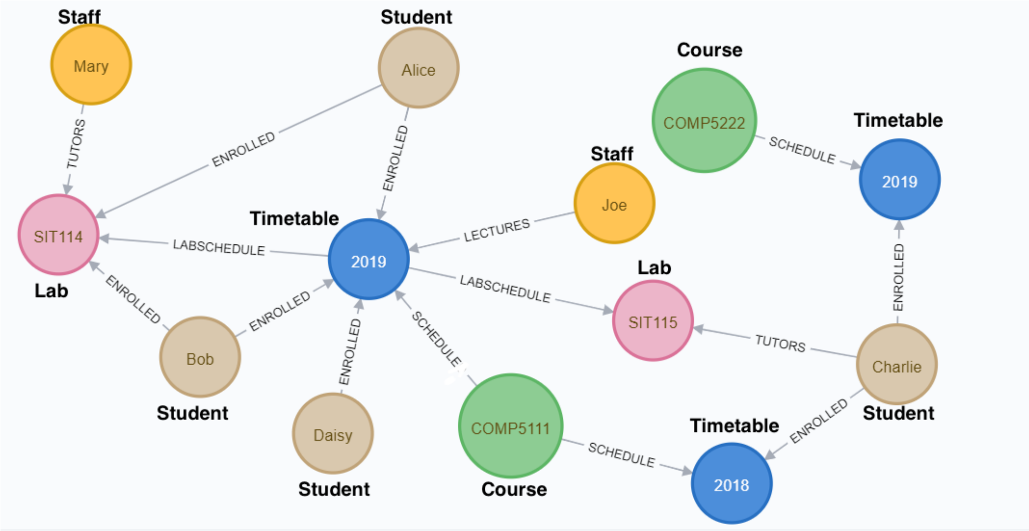
nanswered

Question 20

Not yet marked / 25 pts

All parts of this question refer to university enrolment data modelled as Neo4j graph. The graph contains five node labels: **Course**, **Timetable**, **Lab**, **Staff** and **Student**. The **Course** node

Course, **Timetable**, **Lab**, **Staff** and **Student**. The **Course** node captures basic information about a course. It has two properties: code and title. A course can be offered per year or per semester. Each offering is modelled as a **Timetable** node. The **Timetable** node has four properties: year, semester, venue and time. The **SCHEDULE** relationship between **Course** and **Timetable** captures the course offering information. Most courses have labs, this is modelled as **Lab** node. The **Lab** node has three properties: code and location and time. The **LABSCHEDULE** relationship is used to indicate which course offering the lab belongs to. The **Staff** node models academic staff in the system. It has two properties: id and name. Staff can be assigned to course offerings as tutor or lecturer. This is modelled as **LECTURES** or **TUTORS** relationship respectively. The **LECTURES** relationship is between **Staff** and **Timetable** node; while the **TUTORS** relationship is between **Staff** and **Lab** node. The **Student** node models enrolled student in the system. It has two properties: sid and name. A student can enrol in multiple course offerings. This is modelled as **ENROLLED** relationship. A student can obtain a score from each course offering. The score is modelled as the property of the **ENROLLED** relationship. Each student, when enrolled in a course is also assigned in one of the labs. This is also modelled as **ENROLLED** relationship. A student can work as tutors in a lab, this is modelled as **TUTORS** relationship. Below is a sample graph showing a few nodes and their relationships.



1. [**3 points**] Write a query to find the average passing mark of each course offered in 2018. Here passing mark means a mark that is 50 or above.

2. [**6 points**] The problem domain has many constraints for nodes and their relationships. Maintaining such constraint is largely the responsibility of developers. One option is to write queries to periodically check if there is any violation. This part asks you to develop queries to test the following constraints.

a) A student cannot be the tutor of any course he/she is also currently enrolled in as a student.

b) A student cannot enrol in more than one lab of the same course.

3. [**3 points**] We want to use the data to find candidate tutors of a given course. A candidate tutor is a student achieved HD (85 or above) in previous offerings of the same course. Now write a query to find candidate tutors for 2019 COMP5222 offerings.

4. [**4 points**] Assuming no node property index has been set. Describe the execution plan of the following query

```
MATCH (c:Course)-[]-(t:Timetable{year:2019})-[]-(:Lab)-[r:TUTORS]
-(tutor)
RETURN c.title, labels(tutor), count(tutor)
```

Part 5 - 6 refer to the following nodes/relationships and their respective IDs. Assume the ID value indicates the creation order; smaller value means early creation. For instance, relationship **s** with id **0** is created before the relationship **ls** with id **1**.

Node or Relationship	**ID**
`(c1 :Course {code: “COMP5111”, title: “C1” })`	0
`(t1 :Timetable {year:2019, semester:2, venue:LT110, time:“ Tue18”})`	1

`(l1 :Lab{code:“T20A”; location:“SIT114”})`	10
`(st1 :Student{sid:1234; name: “Alice”})`	20
`(c1)-[s :SCHEDULE]->(t1)`	0
`(t1)-[ls :LABSCHEDULE]->(l1)`	1
`(st1)-[e1 :ENROLLED]->(t1)`	2
`(st1)-[e2 :ENROLLED]->(l1)`	3

5. [**6 points**] Write down the content of the following byte ranges in the relationship record at byte offset 34:

- byte 1~4
- byte 5~8
- byte 13~16
- byte 17~20
- byte 21~24
- type 25~28

6. [**3 points**] Which node has its record at byte offset 340? Which records(s) are included in this node’s doubly linked list of relationship records?

Your Answer:

nanswered

Question 21

Not yet marked / 15 pts

All parts of this question are based on a Dynamo cluster with five nodes: n_0, n_1, n_2, n_3 and n_4. Their corresponding tokens are shown on the following left hand side table. The ring space for consistency hashing is between 0~99. The cluster has a replication factor 3. The preference list contains 4 nodes. The consistency configuration (N, R, W) of the system has the value (3,2,2). One of the tables stored in this cluster contains information about faculties in a university. The faculty name is used as key. Sample keys and their corresponding hash values in the ring space are given in the right hand side table.

Node	**Token**
n_0	5, 50
n_1	20, 85
n_2	35, 60
n_3	75
n_4	95

Key	**Hash value**
Arts	31
Business	93

Education	29
Engineering	13
Law	71
Medicine	47
Science	53

1. [**4 points**] What is the preference list of key “**Science**”?
2. [**4 points**] Which node has the least number of keys? What are the keys on this node?
3. [**2 points]** Suppose all versions of the object with key “**Law**” have the same vector clock ([n_3, 10]), what do we know about the update history of this key?
4. [**5 points**] Now suppose all other nodes except $\mathbf{n_3}$ are available during the next update of key “**Law”**, what would be the vector clock of the new version? Which nodes would have the new version?

Your Answer:

nanswered

Question 22

Not yet marked / 4 pts

Nine-Intersection Model can be used to specify topological relationship of objects in 2D space. Assuming row represents object A, column represents object B. What topological relationship does the following nine-intersection model matrix represents:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

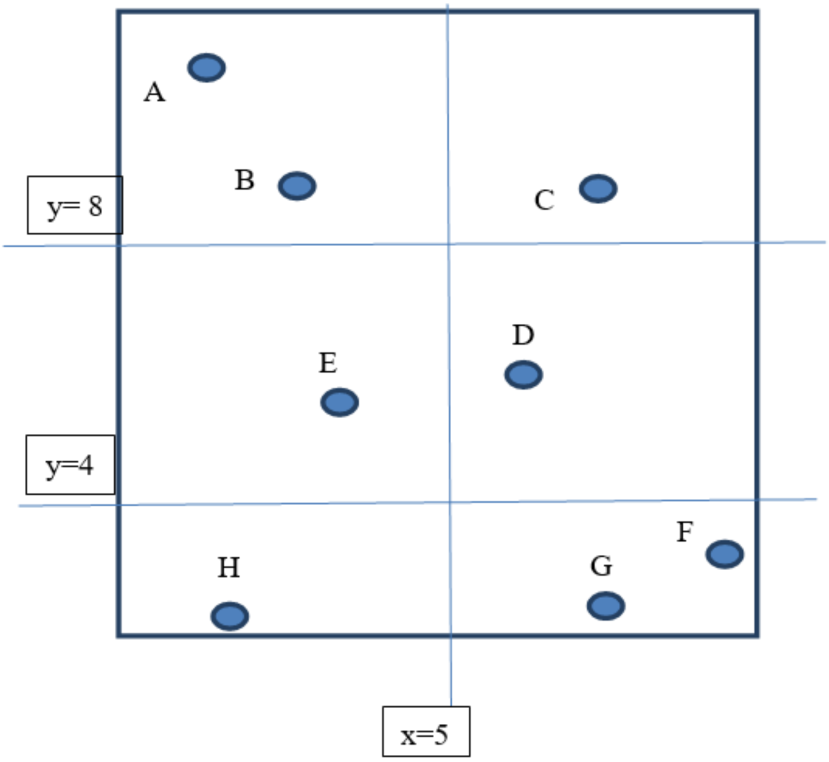
Your Answer:

nanswered

Question 23

Not yet marked / 4 pts

Assume we have a collection of 2D points and our chosen indexing method segments the underlying space as follows. Name the indexing method used and show the index structure using the sample points in the figure.



Your Answer:

nanswered

Question 24

Not yet marked / 4 pts

MBR is an important concept in spatial data model. Explain what is MBR and how it is used in spatial query.

Your Answer: