Fifth Assignment

Performance summary

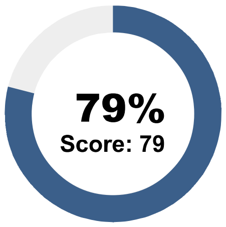
Maximum number of attempts	Unlimited
Number of attempts	1
Your score	79
Status	**Not available**

Results

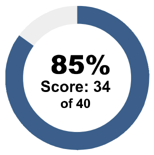
Course	**Big Data**
Test	**5_Assignment_BD**

This are your test results

Duration	7h 40m 45s
Answered	7 of 7 questions (100%)
Your score	79 of 100 points (79%)



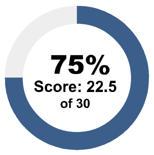
Knowledge Questions 3



go to section ›

✗ ✗ ✓

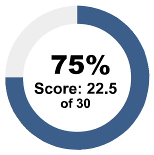
Practice Questions 2



go to section ›

✗ ✓

Programming Questions 2



go to section ›

✓ ✗

Knowledge Questions 34 of 40 points (85%)

✓ Efficiency aspects

Status	Answered		
Your score	3 / 6		50%

Response

Which of the following efficiency aspects are correct about Spark?

True	**False**	
✓	☐	Code generation for expression evaluation
✓	☐	Each task requires 2 types of memory: execution memory and storage memory
☐	✓	Express execution and storage memory as two different regions
☐	✓	Use of cache-aware compuation

▸ Solution

✓ NoSQL Databases

Status	Answered		
Your score	6 / 9		67%

Response

What are the charactersitics of NoSQL databases?

Please note:

- **Maximum Overall Score -> 9 points**
- **Minimum Overall Score -> 0 points**
- **Incorrect Answer -> -1 points**
- **Unanswered -> 0 points**

☐ Able to process both structured and semi-structured data

✓ Can run well on clusters

☐ Enforce consistency

☐ Use SQL for access

✓ Easily and cheaply scalable

▸ Solution

Relational vs non-relational database

Status	Answered	
Your score	25 / 25	100%

Response

Explain the main differences between relational and non-relational databases using real-world examples.

1. Format

Relational: Data is stored in a structured format in the form of tables having rows and columns. Tables having relationships with each other are connected via foreign and primary keys and there is one master table having all keys. For example, A customer's data of a bank is stored in different tables so the customer's master details are in one table, loan and account details are in some other table.

Non-Relational: It is mostly used for storing semi-structured data, it uses key-value pairs which allows storing related items as one document in the same table. For example, A bank customer's whole data will be stored in a single row as a document in the table.

2. Usability

Relational: It can be used where data accuracy and safety is the highest priority and response time has lower priority since data is stored carefully and is being sorted, unstructured and repetitive information is removed and data can be accessed easily. for example, a bank customer's data we want to maintain the integrity and can not afford duplicacy or irrelevant/wrong records.

Non-Relational: It can be used where we want faster response time for example, on websites like eBay. On eBay, every day millions or billions of people just want to look at the items(read-only) and not actually want to place a bid(read-write) so developers are more interested in better response time rather than the balance between read and write operations.

3. Scalability

Relational: These are vertically scalable, which means we can increase the load on a single server by increasing RAM, SSD, or CPU. It is easier to implement but over time, it becomes expensive as the requirement increases. For example, vertical scaling is like retiring your Toyota and buying a Ferrari when you need more horsepower.

Non-Relational: On the other hand, non-relational databases are horizontally scalable which means you can handle more traffic by adding more servers in your NoSQL database. For example, you can think of it as several vehicles you can drive all at once, the combined horsepower is the horsepower you need.

4. Property followed :

Relational: It follows ACID(Atomicity, Consistency, Isolation, Durability) property. For example, a banking system uses a relational database and we have to send money from one account to another so it follows ACID property in the following way:
Atomicity: During the transaction, two operations will occur, deposit and withdrawal, if the deposit fails, withdrawal won't happen.
Consistency: For each transaction, it validates if check numbers are unique.
Isolation: Customer looking at the balance must be isolated from current transaction going and only after current transaction gets committed, the balance should be updated to the customer when he looks at the balance again.
Durability: System crash or another type of failure must not lose results of current transaction or contents of the database, logs of all the transcations should be maintained.

Non-Relational: It uses CAP(Consistency, Availability and Partition Tolerance) theorem.Lets try to understand it with the help of a real life example. Suppose you are a customer of a very popular mobile operator in your city, it follows CAP Theorem:
Consistency : Suppose you want to update your address registered with your number, you call the customer service and they updates the address . After cutting the call, you realize you gave them wrong address, so you call them again but this a different customer care operator picks the call. However, they also are able to process your request and even knew about your previous conversation. So even the different customer care was able to retrieve same information, it is called consistency.
Availability: The customer is able to get any information about his account, usage and balance by just connecting with customer care anytime, it is called availability.
Partition Tolerance: You want change your current plan and therefore you connect to operator, they tell you they were unable to update your address due to some issues so information lying with operator might not be up to date, therefore they can not process your request, this is called partition tolerance. It is basically communication break between nodes and distributed system.

678 words

▸ Solution

‹ go back to overview

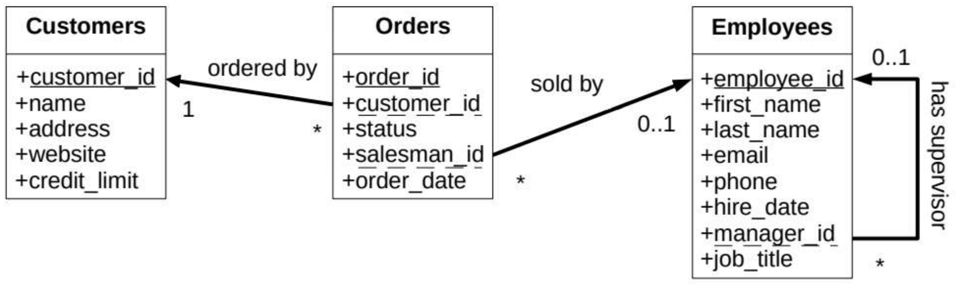
Practice Questions 22.5 of 30 points (75%)

☑ JSON

Status	Answered		
Your score	7.5 / 15		50%

Response

Which of the following JSON formats better represent the given schema:



True	False	
☐	✓	(see first code block)
✓	☐	(see second code block)

```
{
  orders: [
     id:"5",
     status: "delivered",
     order_date: "10/01/2021 20:21",
     ordered_by: {
     id: "971",
     name: "Keely O'Ryan",
     address: "Universitaetsstrasse 1"
     },
     sold_by: {
          id: "891",
          first_name: "Zeyd",
          last_name: "Boukhers",
          email: "boukhers@uni-koblenz.de",
          job_title: "University professor"
     }
 ]
}
```

```
{
 order: {
  id:"5",
  status: "delivered",
  order_date: "10/01/2021 20:21",
  ordered_by: {
  id: "971",
  name: "Keely O'Ryan",
  address: "Universitaetsstrasse 1"
  },
  sold_by: {
     id: "891",
     first_name: "Zeyd",
     last_name: "Boukhers",
     email: "boukhers@uni-koblenz.de",
     job_title: "University professor"
     }
  }
}
```

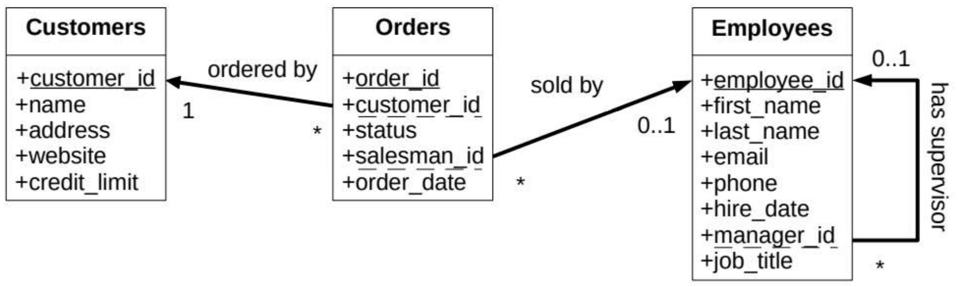
True	False	
☐	✓	{ orders: [id:"5", status: "delivered", order_date: "10/01/2021 20: sold_by: { id: "891", first_name: "Zeyd", last_name: "Boukhers", email: "boukhers@uni-koblenz.de", job_title: "University professor", ordered_by: { id: "971", name: "Keely O'Ryan", address: "Universitaetsstrasse 1" }, }] }
☐	✓	{ customer: { id: "971", name: "Keely O'Ryan", address: "Universitaetsstrasse 1" }, employee: { id: "891", first_name: "Zeyd", last_name: "Boukhers", email: "boukhers@uni-koblenz.de", job_title: "University professor" }, order: { id:"5", status: "delivered", order_date: "10/01/2021 20:21", } }

```
{
  orders: [
    id:"5",
    status: "delivered",
    order_date: "10/01/2021 20:
  sold_by: {
    id: "891",
    first_name: "Zeyd",
    last_name: "Boukhers",
    email: "boukhers@uni-koblenz.de",
    job_title: "University professor",
    ordered_by: {
        id: "971",
        name: "Keely O'Ryan",
        address: "Universitaetsstrasse 1"
    },
  }
]
}
```

```
{
  customer: {
    id: "971",
    name: "Keely O'Ryan",
    address: "Universitaetsstrasse 1"
  },
  employee: {
    id: "891",
    first_name: "Zeyd",
    last_name: "Boukhers",
    email: "boukhers@uni-koblenz.de",
    job_title: "University professor"
  },
  order: {
    id:"5",
    status: "delivered",
    order_date: "10/01/2021 20:21",
  }
}
```

▾ Solution

Which of the following JSON formats better represent the given schema:



True	False	
✓	☐	see below
✓	☐	see below

```
{
  orders: [
     id:"5",
     status: "delivered",
     order_date: "10/01/2021 20:21",
     ordered_by: {
     id: "971",
     name: "Keely O'Ryan",
     address: "Universitaetsstrasse 1"
     },
     sold_by: {
          id: "891",
          first_name: "Zeyd",
          last_name: "Boukhers",
          email: "boukhers@uni-koblenz.de",
          job_title: "University professor"
     }
  ]
}
```

```
{
 order: {
   id:"5",
   status: "delivered",
   order_date: "10/01/2021 20:21",
   ordered_by: {
   id: "971",
   name: "Keely O'Ryan",
   address: "Universitaetsstrasse 1"
   },
   sold_by: {
      id: "891",
      first_name: "Zeyd",
      last_name: "Boukhers",
      email: "boukhers@uni-koblenz.de",
      job_title: "University professor"
      }
   }
}
```

True	False	
☐	☑	see below
☐	☑	see below

```
{
  orders: [
    id:"5",
    status: "delivered",
    order_date: "10/01/2021 20:
  sold_by: {
    id: "891",
    first_name: "Zeyd",
    last_name: "Boukhers",
    email: "boukhers@uni-koblenz.de",
    job_title: "University professor",
    ordered_by: {
        id: "971",
        name: "Keely O'Ryan",
        address: "Universitaetsstrasse 1"
    },
  }
]
}
```

```
{
  customer: {
    id: "971",
    name: "Keely O'Ryan",
    address: "Universitaetsstrasse 1"
  },
  employee: {
    id: "891",
    first_name: "Zeyd",
    last_name: "Boukhers",
    email: "boukhers@uni-koblenz.de",
    job_title: "University professor"
  },
  order: {
    id:"5",
    status: "delivered",
    order_date: "10/01/2021 20:21",
  }
}
```

◉ CouchDB

Status	Answered	
Your score	15 / 15	100%

Response

Please install CouchDB in your local machine (preferrably using docker).

Instructions for installing docker can be found here: https://docs.docker.com/get-docker/

Here and **here** you can find additional informations on how to run CouchDB using docker.

Let's assume we have a database with the name "movies" and we want to insert the following document:

```
{
  id: "1",
  name: "Spider-Man 3",
  year: 2007
}
```

Which of the following is correct to insert the document:

- curl -X GET http://127.0.0.1:5984/movies/"1" -d '{"Name":"Spider-Man 3", "year":"2007" }'
- curl -X PUT http://127.0.0.1:5984/movies/ --insert '{"id": "1", "name":"Spider-Man 3", "year":"2007" }'
- [✓ curl -X PUT http://127.0.0.1:5984/movies/"1" -d '{"Name":"Spider-Man 3", "year":"2007" }']

Solution

Please install CouchDB in your local machine (preferrably using docker).

Instructions for installing docker can be found here: https://docs.docker.com/get-docker/

Here and **here** you can find additional informations on how to run CouchDB using docker.

Let's assume we have a database with the name "movies" and we want to insert the following document:

```
{
  id: "1",
  name: "Spider-Man 3",
  year: 2007
}
```

Which of the following is correct to insert the document:

- curl -X GET http://127.0.0.1:5984/movies/"1" -d '{"Name":"Spider-Man 3", "year":"2007" }'
- curl -X PUT http://127.0.0.1:5984/movies/ --insert '{"id": "1", "name":"Spider-Man 3", "year":"2007" }'
- [✓ curl -X PUT http://127.0.0.1:5984/movies/"1" -d '{"Name":"Spider-Man 3", "year":"2007" }']

go back to overview

Programming Questions 22.5 of 30 points (75%)

CouchDB Query

Status	Answered	
Your score	15 / 15	100%

Response

Assume we have a movies database, each movie contains the id, title, and year.

Which of the following requests is correct for obtaining movies that are released after 2010?

[✓]
```
POST /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "selector": {
        "year": {"$gt": 2010}
    },
    "fields": ["_id", "year", "title"],
    "sort": [{"year": "asc"}]
}
```

[ ]
```
GET /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "select": {
        "year": {"$gt": 2010}
    },
    "fields": ["_id", "year", "title"],
    "sort": [{"year": "asc"}]
}
```

[ ]
```
POST /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "fields": ["_id", "_rev", "year", "title"],
    "filter": [{"year": "$gt 2010"}
    "sort": [{"year": "asc"}],
}
```

Solution

Assume we have a movies database, each movie contains the id, title, and year.

Which of the following requests is correct for obtaining movies that are released after 2010?

●
```
POST /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "selector": {
        "year": {"$gt": 2010}
    },
    "fields": ["_id", "year", "title"],
    "sort": [{"year": "asc"}]
}
```

○
```
GET /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "select": {
        "year": {"$gt": 2010}
    },
    "fields": ["_id", "year", "title"],
    "sort": [{"year": "asc"}]
}
```

○
```
POST /movies/_find HTTP/1.1
Accept: application/json
Content-Type: application/json
Content-Length: 168
Host: localhost:5984

{
    "fields": ["_id", "_rev", "year", "title"],
    "filter": [{"year": "$gt 2010"}
    "sort": [{"year": "asc"}],
}
```

✓ CouchDB

Status	Answered
Your score	7.5 / 15 — 50%

Response

What is correct about CouchDB?

True	False	
✓	☐	CouchDB supports master-master setups with automatic conflict detection
✓	☐	Using CouchDB's incremental replication you can distribute your data efficiently
✓	☐	CouchDB uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.
☐	✓	CouchDB follows the working of a client/server architecture

Solution

What is correct about CouchDB?

True	False	
✓	☐	CouchDB supports master-master setups with automatic conflict detection
✓	☐	Using CouchDB's incremental replication you can distribute your data efficiently
☐	✓	CouchDB uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.
☐	✓	CouchDB follows the working of a client/server architecture

Test period was over at **5/19/2021, 8:00 PM**.