



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Group Project: Part 2 (MongoDB)

MCSD 1123/SCSP5023

(BIG DATA MANAGEMENT)

Section 1

Dr Haizan Binti Mohammed Radzi

Video Link: <https://youtu.be/Mt5t0qrvBQI>

Ahmad Khaidir Amir bin Rodzman (A18CS0014)

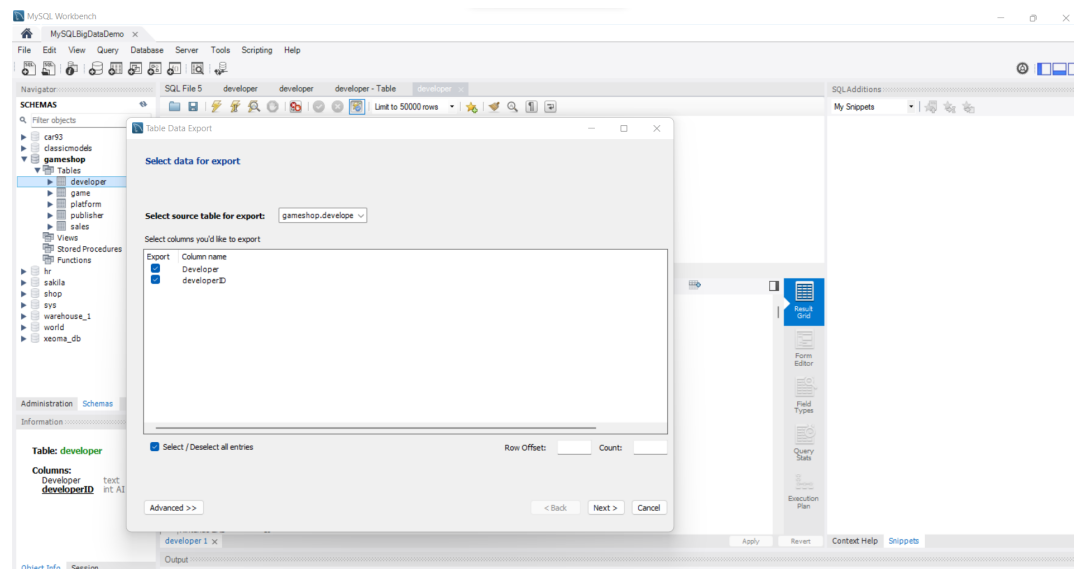
Putera Muhammad Syabil Bin Sarianto (A18CS0235)

LI YEXIN(MCS211046)

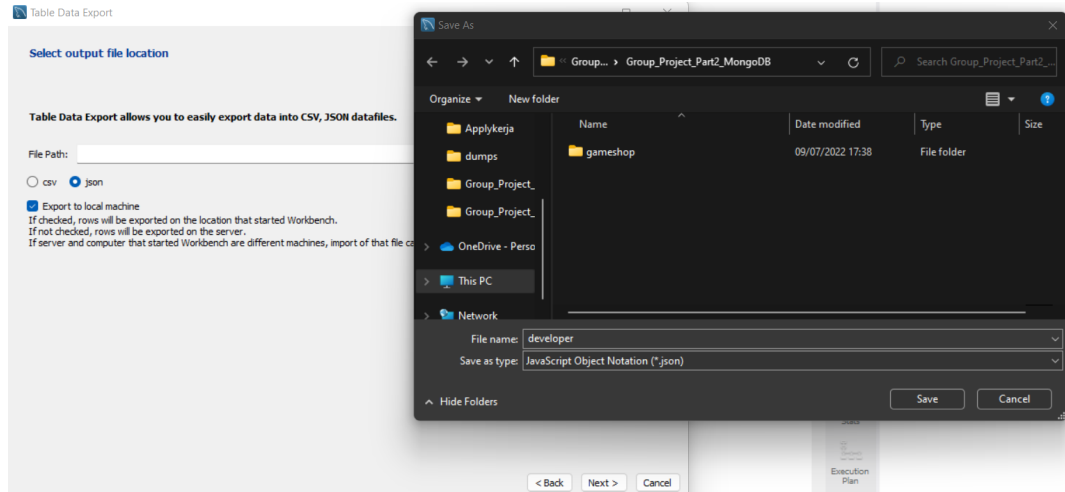
1. How do We convert the table into a document and why do We need it to appear as in the single documents (embedded form) of reference documents? (10 marks)

Mongodb is a database that sits somewhere between relational and non-relational data, embedding document content within another document. Embedding is the process of incorporating a specific type of data, such as an array containing additional data, into the document itself. For our data, we have plenty of relationships in the diagram and in MongoDB, we are allowed to store this data and relationship using embedded-document. The tables in RDBMS will become a collection of data in MongoDB and we can specify the data we want to see with linkage to another entity by using an aggregation method.

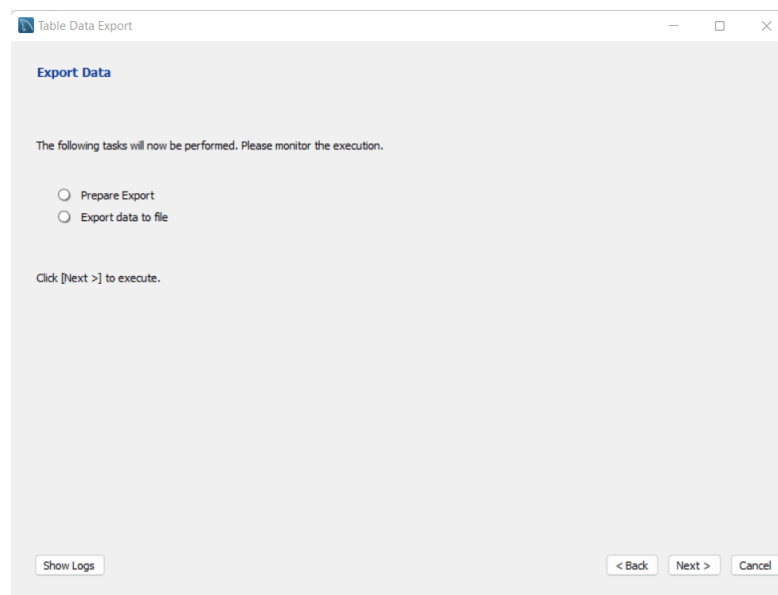
We can directly convert the sql data into a json file which is a structure language that is used by MongoDB. First, right click on the table of data that We want to convert for example the developer table. Then it will display the Table Data Export Wizard. Tick the column that We want to export out.



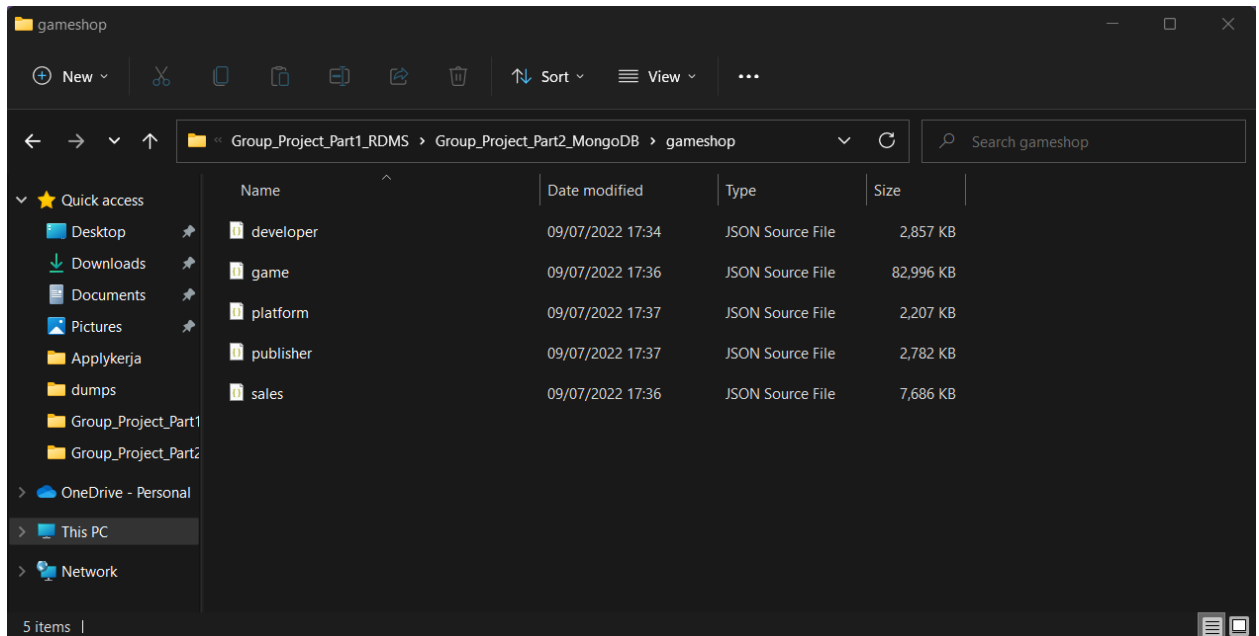
On the file path, choose the directory where We want to save the json table. Choose json below the file path and click on export to the local machine. After save then click next.



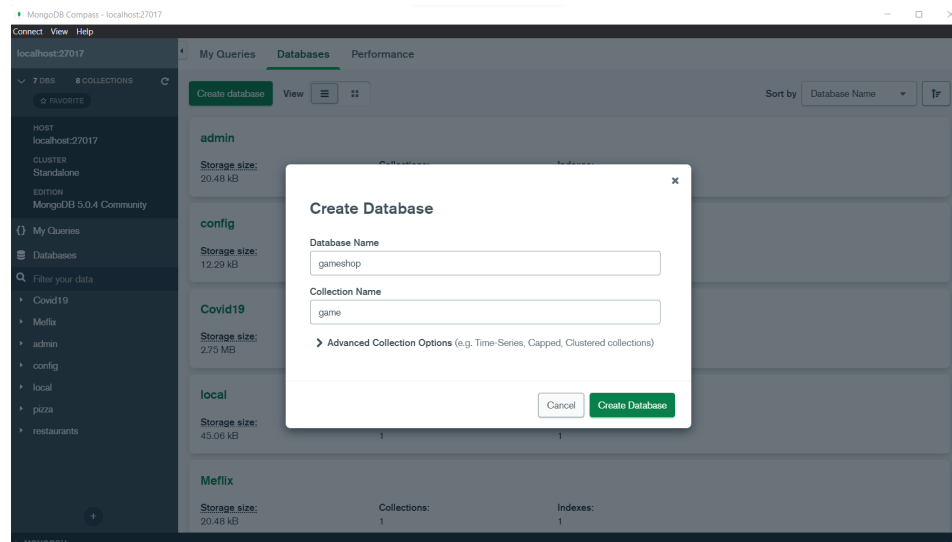
Then, just wait until all the rows and columns are exported into a json file. Repeat all the steps above for each table until all the tables are converted into json files.



After all tables are converted, We will see all the tables now are exported into a Json file like shown below.



Then go to MongoDB Compass and connect to Wer localhost 27017 on Wer device, click on create Database and name it as similar to Wer database name in mySQL Workbench. For this project, we name it gameshop. For the collection name just write any table that We want to create first. For this one we choose “game” as our first collection.



Click on the Import Data button and select file game.json file that we created before. Select input type as JSON file. Then click import. Then it will be shown as in the figure below. Need to wait until all the data was uploaded and stored inside the collection.

Import To Collection gameshop.game

Select File

game.json

Select Input File Type

JSON

CSV

Options

☐ Stop on errors

Importing documents...

Stop

43000 / ~228014.73997123906

CANCEL

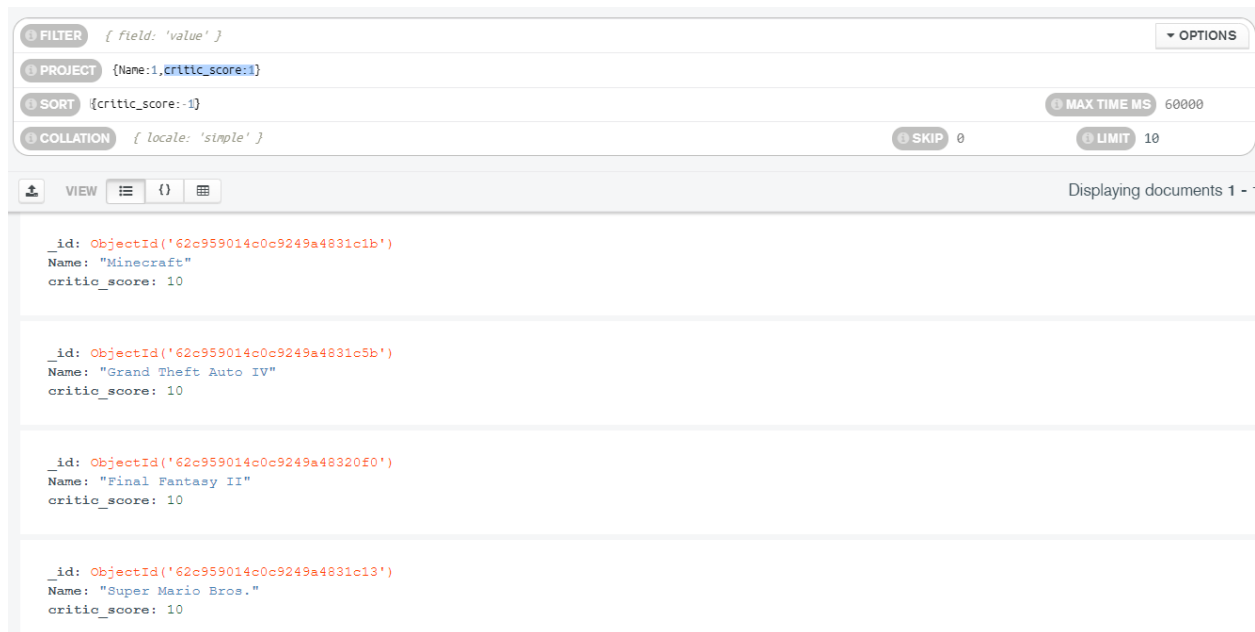
IMPORTING...

Repeat all the steps above for each collection until all the collections of data are uploaded into MongoDB database as shown below.

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
developer	4.10 kB	56 K	67.00 B	1	4.10 kB
game	4.10 kB	400 K	207.00 B	1	4.10 kB
platform	4.10 kB	56 K	55.00 B	1	4.10 kB
publisher	4.10 kB	56 K	66.00 B	1	4.10 kB
sales	4.10 kB	53 K	152.00 B	1	4.10 kB

2) Perform similar queries to the one We made in SQL. The queries must include aggregate queries, lookup query, index query. Save the queries and include them in Wer report. Export output (or partial of output) of query in csv, excel or json file.

a) Top 10 highest critic score games.



The screenshot shows a MongoDB query interface with the following settings:

- FILTER:** { field: 'value' }
- PROJECT:** { Name: 1, critic_score: 1 }
- SORT:** { critic_score: -1 }
- MAX TIME MS:** 60000
- COLLATION:** { locale: 'simple' }
- SKIP:** 0
- LIMIT:** 10

The results display shows the following documents:

```
{ "_id": ObjectId('62c959014c0c9249a4831c1b'), "Name": "Minecraft", "critic_score": 10 }
{ "_id": ObjectId('62c959014c0c9249a4831c5b'), "Name": "Grand Theft Auto IV", "critic_score": 10 }
{ "_id": ObjectId('62c959014c0c9249a48320f0'), "Name": "Final Fantasy II", "critic_score": 10 }
{ "_id": ObjectId('62c959014c0c9249a4831c13'), "Name": "Super Mario Bros.", "critic_score": 10 }
```

Query:

```
filter={}
```

```
project={
```

```
  'Name': 1,
```

```
  'critic_score': 1
```

```
}
```

```
sort=list({
```

```
  'critic_score': -1
```

```
}).items())
```

```
limit=10
```

```
result = client['gameshop']['game'].find(
```

```

filter=filter,
projection=project,
sort=sort,
limit=limit
)

```

As We can see in the chart, we looked up the top 10 highest critic score games by name and score.

b) Which game and publisher produce the games that have the highest user score?

```

_id: ObjectId('62c959014c0c9249a4831c59')
gameID: 72
Name: "The Legend of Zelda: Breath of the Wild"
Genre: "Action-Adventure"
year: 2017
url: "http://www.vgchartz.com/game/118753/the-legend-of-zelda-breath-of-the-..."
image_url: "/games/boxart/full_4436858AmericaFrontccc.png"
_rank: 72
critic_score: 9.9
user_score: 10
platformID: 72
publisherID: 72
developerID: 72
Publisher: Array
  0: Object
    _id: ObjectId('62c959744c0c9249a48aebc5')
    Publisher: "EA Sports"
    publisherID: 72

```

```

1
2 {
3   from: "publisher",
4   localField: "publisherID",
5   foreignField: "publisherID",
6   as: "Publisher"
7 }
8

```

```

1
2 {
3   user_score: -1
4 }

```

```

1
2 1

```

Query:

\$lookup

```
{
```

```

    from: "publisher",
    localField: "publisherID",
    foreignField: "publisherID",
    as: "Publisher"
  }
}
$sort
{
  user_score: -1
}
$limit
1

```

From this query, we can see the Legend of Zelda: Breath of the Wild that published by the EA Sports is the game that have the highest user_score.

c) Which platform has the top 20 highest critic scores?

gameshop.game 400.3k 1
DOCUMENTS INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Pipeline ▼ \$lookup \$sort \$limit Edit Explain Export Run More Options ▸

Showing 1 – 20 [count results](#) VIEW ≡ {}

```

{
  "_id": ObjectId("62c959014c0c9249a4831c7c"),
  "gameID": 107,
  "Name": "Super Mario Kart",
  "Genre": "Racing",
  "year": 1992,
  "url": "http://www.vgchartz.com/game/4435/super-mario-kart/?region=All",
  "image_url": "/games/boxart/full_3980311AmericaFrontccc.jpg",
  "_rank": 107,
  "critic_score": 10,
  "user_score": 0,
  "platformID": 107,
  "publisherID": 107,
  "developerID": 107,
  "Platform": Array
    0: Object
      "_id": ObjectId("62c959634c0c9249a48a11f6")
      "Platform": "PS4"
      "platformID": 107

```




Query:

\$lookup

{

 from: "platform",

 localField: "platformID",

 foreignField: "platformID",

 as: "Platform"

}

\$sort

{

 critic_score: -1

}

\$limit

20

d) Which publisher and game has the top 10 highest global sales?

```
_id: ObjectId('62c959854c0c9249a48bc64b')
Total_Shipped: 0
Global_Sales: 20.32
NA_Sales: 6.37
PAL_Sales: 9.85
JP_Sales: 0.99
Other_Sales: 3.12
publisherID: 220
gameID: 220
> Publisher: Array
> Game: Array
```

```
_id: ObjectId('62c959854c0c9249a48bc64c')
Total_Shipped: 0
Global_Sales: 19.39
NA_Sales: 6.06
PAL_Sales: 9.71
JP_Sales: 0.6
Other_Sales: 3.02
publisherID: 221
gameID: 221
> Publisher: Array
> Game: Array
```

\$lookup

```
1
2 ▾ {
3   from: "publisher",
4   localField: "publisherID",
5   foreignField: "publisherID",
6   as: "Publisher"
7 }
```

\$lookup

```
1
2 ▾ {
3   from: "game",
4   localField: "gameID",
5   foreignField: "gameID",
6   as: "Game"
7 }
```

\$sort

```
1
2 ▾ {
3   Global_Sales: -1
4 }
```

\$limit

```
1
2 10
```

Query:

\$lookup

```
{
  from: "publisher",
  localField: "publisherID",
  foreignField: "publisherID",
  as: "Publisher"
}
```

\$lookup

```
{
  from: "game",
  localField: "gameID",
  foreignField: "gameID",
```

```

    as: "Game"
  }
}
$sort
{
  Global_Sales: -1
}
$limit
10

```

e) Which top 5 genre contribute the most sales In Japan.

gameshop.sales 52.6k 1
DOCUMENTS INDEXES

Documents **Aggregations** Schema Explain Plan Indexes Validation

Pipeline ▼ \$lookup \$lookup \$sort \$limit Edit Explain Export Run More Options

Showing 1 – 5 count results < > VIEW ≡ { }

```

{
  "_id": ObjectId("62c959854c0c9249a48bc573"),
  "Total_Shipped": 5.5,
  "Global_Sales": 0,
  "NA_Sales": 0,
  "PAL_Sales": 0,
  "JP_Sales": 0,
  "Other_Sales": 0,
  "publisherID": 4,
  "gameID": 4,
  "Publisher": Array,
  "Game": Array,
  "0": {
    "_id": ObjectId("62c959014c0c9249a4831c15"),
    "gameID": 4,
    "Name": "PlayerUnknown's Battlegrounds",
    "Genre": "Shooter",
    "year": 2017,
    "url": "http://www.vgchartz.com/game/215988/playerunknowns-battlegrounds/?regi...",
    "image_url": "/games/boxart/full_8052843AmericaFrontccc.jpg",
    "_rank": 4,
    "critic_score": 0,
    "user_score": 0,
    "platformID": 4,
    "publisherID": 4,
    "developerID": 4
  }
}

```

⋮ ▼ \$lookup ⬢

```

1  {
2  {
3    from: "publisher",
4    localField: "publisherID",
5    foreignField: "publisherID",
6    as: "Publisher"
7  }

```

⋮ ▼ \$lookup ⬢

```

1  {
2  {
3    from: "game",
4    localField: "gameID",
5    foreignField: "gameID",
6    as: "Game"
7  }

```

⋮ ▼ \$sort ⬢ 🗑 +

```

1  {
2  {
3    JP_Sales: -1
4  }

```

⋮ ▼ \$limit ⬢

```

1  5
2

```

Query:

```

$lookup
{

```

```
    from: "publisher",
    localField: "publisherID",
    foreignField: "publisherID",
    as: "Publisher"
  }
  $lookup
  {
    from: "game",
    localField: "gameID",
    foreignField: "gameID",
    as: "Game"
  }
  $sort
  {
    JP_Saless: -1
  }
  $limit
  5
```

3) Having these TWO types of data model, which one We think is more applicable to the case study. Discuss Wer justification.

From two data model implemented in this dataset, NoSQL is better than RDMS. The main advantage that we realised when using the NoSQL is that the query speed is reliably faster than RDBMS. This is because NoSQL databases are de-normalized, there is no need to worry about data duplication, and all the information needed for a particular query is usually already stored together -- no connection required. This can make lookups easier, especially when dealing with large amounts of data. This also means that NoSQL can be very fast for simple queries. It goes without saying that SQL databases can also return very fast queries. They also support highly complex queries of structured data. However, as SQL databases grow and complex connection requirements increase, query speeds can quickly decline. RDBMS on the other hand, when applying a dataset that is so large will require some more time display the result. In our case, the dataset contain more than 50,000 data and we realised that when we query to the database made from our end, the machine require some times to display.

Not only that, we can start a NoSQL database without spending time defining its structure, and We can easily add data types and fields without downtime. And when we look into our question one, we did not include the entity relationship design for NoSQL as it does not require to do so. Only importing from RDBMS to NoSQL. This promotes agility because NoSQL database schemas are not limited by rows and columns and do not have to be defined beforehand. Instead, they are dynamic and capable of handling all types of data, including structured, semi-structured, unstructured, and polymorphic data.

Having NoSQL that are able to support semi and unstructured data type, makes it a great option for our case because some of the records in the dataset does not have a value or NaN. Compared to RDBMS, it will display this type of data. Thus when using NoSQL, this information will be cut off from the dataset and only display the attribute that has value. Removing this kind of data will allow us to save computational resources as it only takes data that has the value.

All of this makes NoSQL a great fit for modern agile development teams. Developers can start building databases without spending time and effort on upfront planning. Allow them to be modified quickly as requirements change and new data types are added. The flexibility and adaptability of NoSQL databases make them ideal for organizations that have multiple data types and want to constantly add new features and functionality.