

---

# Migration from Relational databases to NoSQL databases

Dhirubhai Ambani Institute of  
Information and Communication Technology  
11-07-2022



## Summer Research Internship

**Prepared by :**

Aksh Patel: 201901005

Nilay Shah: 201901026

Devesh Modi: 201901173

**Mentor :** Prof PM Jat

---

## Table of Contents

<b>Objective .....</b>	<b>3</b>
<b>Problem statement .....</b>	<b>3</b>
Input .....	3
Output.....	4
<b>Framework Description .....</b>	<b>4</b>
<b>Algorithm.....</b>	<b>5</b>
<b>Implementation of Framework .....</b>	<b>6</b>
<b>Database connections .....</b>	<b>6</b>
<b>Schema extraction .....</b>	<b>7</b>
Fetch Primary Key .....	7
Fetch Foreign Key .....	7
<b>Embedding and linking .....</b>	<b>8</b>
<b>Migrate.....</b>	<b>8</b>
<b>Use cases .....</b>	<b>9</b>
<b>Test Schema 1: E-Commerce.....</b>	<b>9</b>
<b>Test Schema 2: Basketball Training Management Center.....</b>	<b>12</b>
<b>Test Schema 3: Entertainment Booking System.....</b>	<b>15</b>
<b>Test Schema 4: Flight tracking.....</b>	<b>17</b>
<b>Conclusion .....</b>	<b>22</b>
<b>Future Works.....</b>	<b>23</b>
<b>Appendix .....</b>	<b>24</b>
<b>References .....</b>	<b>25</b>

## **Objective**

Derive a framework to map a relational database schema to No-SQL database and implement it.

## **Problem statement**

As now the demand for No-SQL databases is increasing and there is a need for software which automatically converts relational databases to No-SQL databases. Also, the algorithm which is used behind the scene and is efficient for schema conversion so that querying No-SQL databases is faster.

Here we are choosing PostgreSQL as a relational database and MongoDB as No-SQL database.

## **Input**

The needs to give the following inputs:

1. Database name
2. Schema Name
3. MongoDB connection String
4. MongoDB Database Name
5. Access path text file

## Output

- Data migrated from pgAdmin to MongoDB Atlas database.

## Framework Description

This proposes a framework which implements an algorithm.

potential collections: tables which appear first in the access path

- Create a new collection for all the **\*potential collections**.
- For every potential collection
  - If its foreign keys tables are already a collection then link
  - else embed.
- For every non-potential collection,
  - If fk
    - i. Not referred
      1. One fk
        - a. One way embedding
      2. Two fks
        - a. Two way embedding
    - ii. Referred
      1. Referring table is a collection
        - a. Referring table embeds it.
      2. Else
        - a. Referring table is embedded in it
        - b. If referred(fk) table is a collection
          - i. Embed this table in it
        - c. Else
          - i. New collection
          - ii. Embeds/ refers the referred table (so that we can at least independently query this table)
    - Ref and no fk => embeds into parent

- Notref and no fk => new collection
- If there is any ordering attribute in the access path, then create a secondary index for that attribute.

## Algorithm

1. Create potential collections (collections which are listed first in the access path eg: In a/b/c => a is potential collection)
2. We represent access paths in the form of a graph by establishing edge in parent child relationship in access path.
3. Then we run a dfs on the graph taking following points into consideration:
  - If the neighbor node of currentTable is a collection
    - => link curTable and nxtTable.
  - Else
    - => embed nxtTable into curTable
4. We represent the sql schema in the form of a graph using foreign key as an edge between tables.
5. Then we run a dfs on the graph taking following points into consideration:
  - If currTable is not referred:
    - If curTable has 0 FK =>
      - Make curTable a collection.
    - Else if curTable has 1 FK =>
      - If the neighbor node of currentTable is a collection =>
        - link curTable and nxtTable.
      - Else =>
        - embed curTable into nxtTable
    - Else =>
      - If the neighbor node of currentTable is a collection =>
        - link curTable and nxtTable.
      - Else =>
        - embed nxtTable into curTable

- Else
  - If the neighbor node of currentTable is a collection =>
    - link curTable and nxtTable.
  - Else =>
    - embed nxtTable into curTable

## Implementation of Framework

### Tools:

- Frontend: PyWebio (Python)
- Backend: Python
- Databases: PGAdmin 4 (PostgreSQL), MongoDB (No-SQL)
- Libraries: pywebio, pandas, psycopg2

## Database connections

### ***Connecting PostgreSQL***

```
pgsqldb =  
psycopg2.connect(database=dbName,user="postgres",password="admin")
```

### ***Connecting MongoDB Atals using mongodb connection string***

```
myClient =  
pymongo.MongoClient(mongodb_host,tls=True, tlsAllowInvalidCertificates=True)
```

## Schema extraction

### Fetch Primary Key

```
cursor.execute("""SELECT distinct c.column_name, c.data_type
FROM information_schema.table_constraints tc
JOIN information_schema.constraint_column_usage AS ccu USING
(constraint_schema, constraint_name)
JOIN information_schema.columns AS c ON c.table_schema =
tc.constraint_schema
AND tc.table_name = c.table_name AND ccu.column_name = c.column_name
WHERE constraint_type = 'PRIMARY KEY' and tc.table_name = '{}{}';
""".format(tableName))
```

### Fetch Foreign Key

```
cursor.execute("""select *
from (select distinct
(select r.relname from pg_class r where r.oid = c.conrelid) as table,
(select array_agg(attname) from pg_attribute
where attrelid = c.conrelid and ARRAY[attnum] <@ c.conkey) as col,
(select r.relname from pg_class r where r.oid = c.confrelid) as ftable
from pg_constraint c
where c.confrelid in (select oid from pg_class where relname = '{}{}')) as imp
where imp.table in (select table_name from
information_schema.tables where table_schema = '{}{}')
```

```
and imp.ftable in (select table_name from information_schema.tables
where table_schema = '{}') ;
"".format(tableName, schema, schema))
```

## Embedding and linking

This uses a matrix stating whether there should be embedding or linking between two tables. This function iterates through the tuples and finds the tuples which should be embedded in the which tuple of another table based on the primary key.

## Migrate

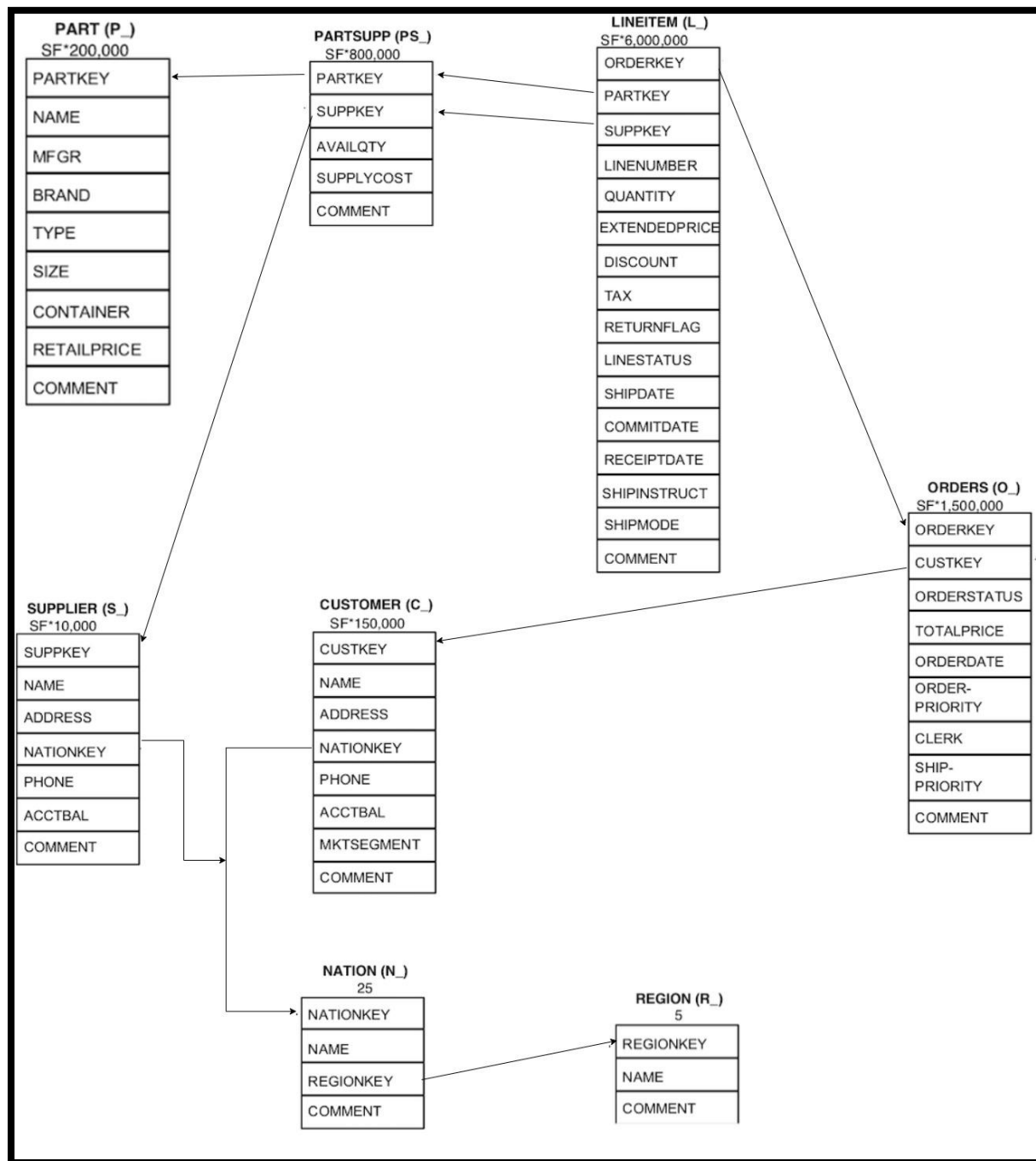
- MongoDB will create the database if it does not exist, and make a connection to it.
  - mydb = myClient[mongodb\_name]
- Create collection using following syntax
  - mycol = mydb[table\_name]
- We inserted documents in mongodb using insertmany command.
  - x = mycol.insert\_many(data) # insert the documents



## Use cases

### Test Schema 1: E-Commerce

#### Schema Diagram for Relational Databases

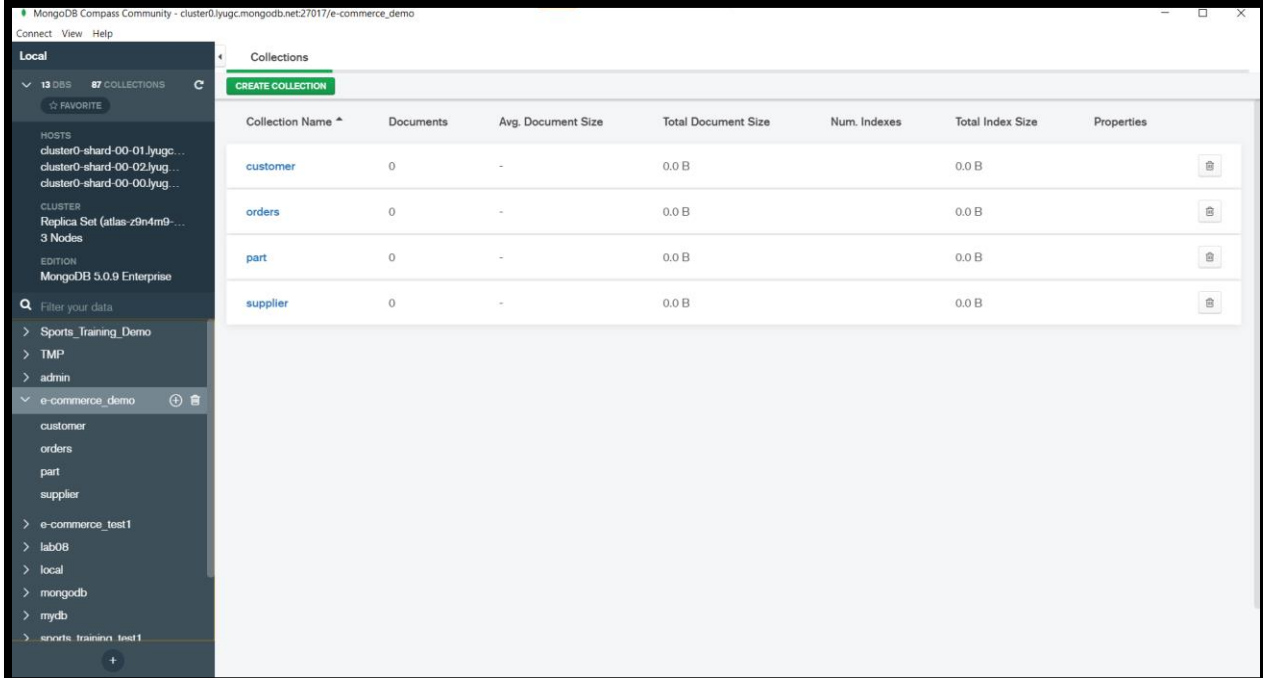


## Relational Tables Count: 8

### Access Path:

customer/orders  
orders/lineitem  
part/supplier  
supplier/part

### Migrated Result:



The screenshot shows the MongoDB Compass interface. On the left, the 'Local' sidebar displays the database 'e-commerce\_demo' with its collections: customer, orders, part, and supplier. The main panel shows the 'Collections' tab with a table listing these collections. The table has columns for Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, Total Index Size, and Properties. All collections currently have 0 documents and 0.0 B sizes.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
customer	0	-	0.0 B		0.0 B	
orders	0	-	0.0 B		0.0 B	
part	0	-	0.0 B		0.0 B	
supplier	0	-	0.0 B		0.0 B	

## MongoDB Collections Count: 4

## Logs:

```
Script started at: 2022-07-04 21:55:24.457444
Initializing database connections...
Connecting to PostgreSQL server...
Connection to Postgres db succeeded.
Connecting to MongoDB server...
Connection to MongoDB Server succeeded.
Database connections initialized successfully.
.....Running Algo.....
embed region in nation
embed partsupp in lineitem
embed nation in supplier
embed nation in customer
embed lineitem in orders
-----Algo Ends-----
embedding region in nation
embedding partsupp in lineitem
embedding nation in supplier
embedding nation in customer
embedding lineitem in orders
linking each other : part <-> supplier via partsupp
linking orders in customer
Processing table: orders...
Processing table: orders completed. 15000 documents inserted.
Processing table: supplier...
Processing table: supplier completed. 100 documents inserted.
Processing table: part...
Processing table: part completed. 2000 documents inserted.
Processing table: customer...
Processing table: customer completed. 1500 documents inserted.

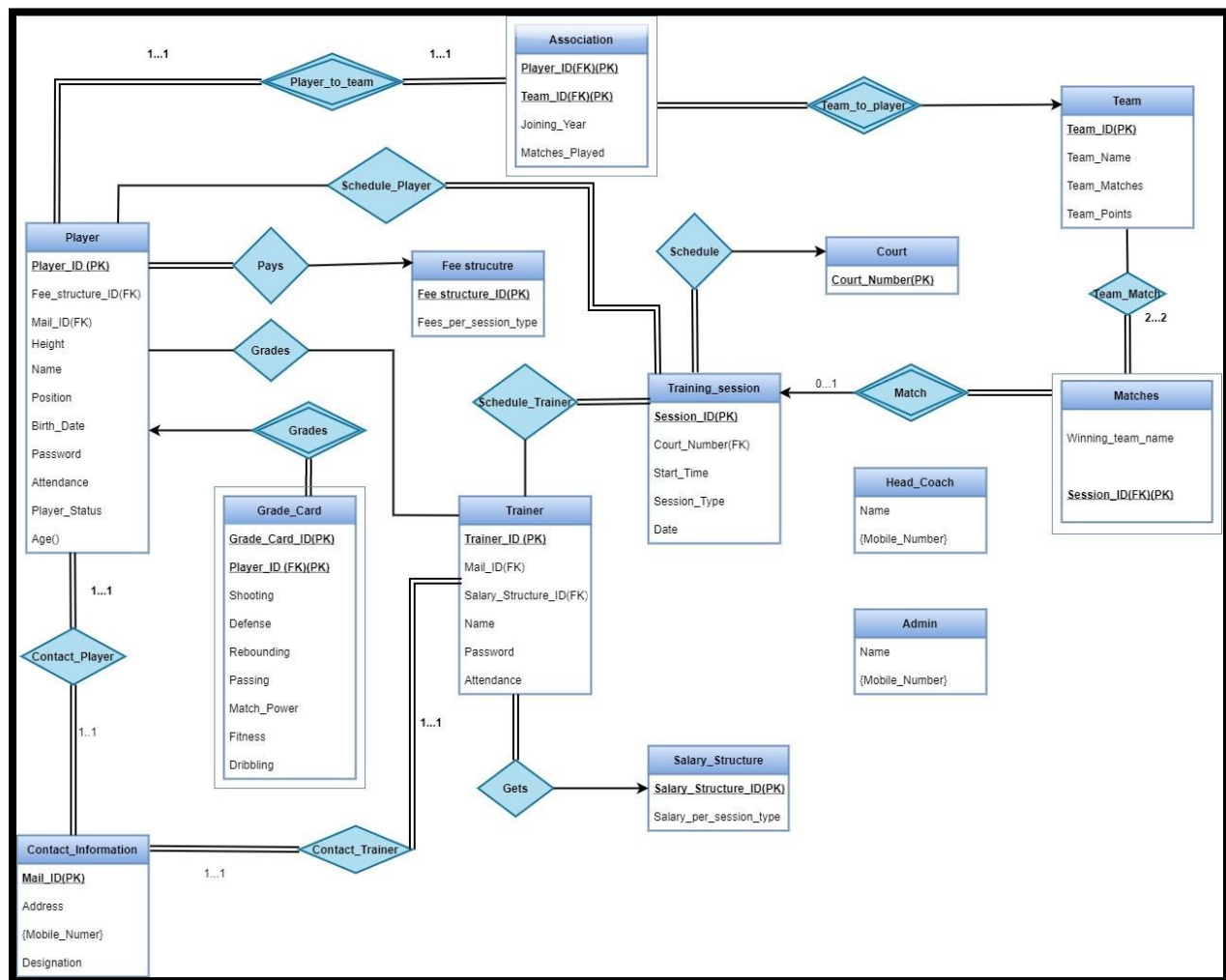
Migration completed.
4 of 4 tables migrated successfully.
Script completed at: 2022-07-04 22:09:29.227986
Total execution time: 0:14:04.770542
□
```

## Observation:

The relational schema is denormalized with regards to some relations and access paths and we get 4 collections from 8 tables and other tables are either migrated with embedding or referencing relationships in MongoDB.

## Test Schema 2: Basketball Training Management Center

### Schema Diagram for Relational Databases:

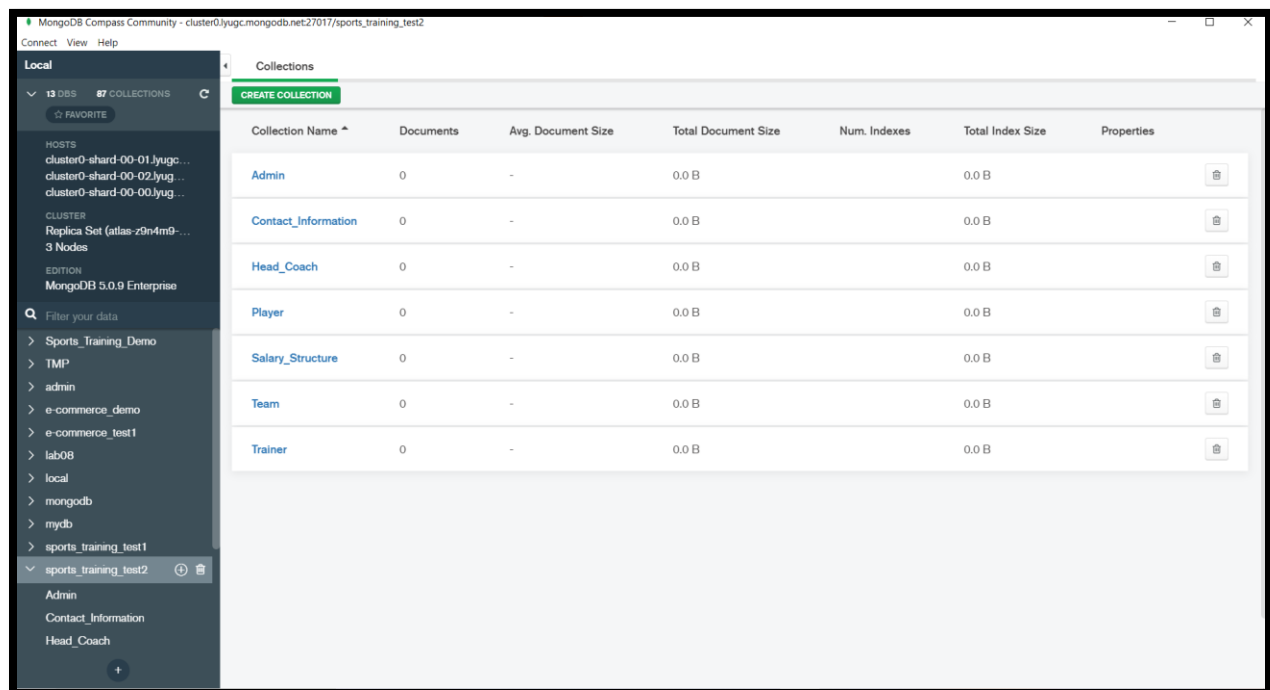


Relational Tables Count: 17

## Access Path:

Admin  
Player/Contact Information  
Contact\_Information/Player  
Trainer/Grades  
Salary\_Structure/Trainer  
Team

## Migrated Result:



The screenshot shows the MongoDB Compass interface. On the left, a sidebar lists databases and collections. The 'collections' tab is active, displaying a table of collections. The table has columns for Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, Total Index Size, and Properties. The collections listed are Admin, Contact\_Information, Head\_Coach, Player, Salary\_Structure, Team, and Trainer, all with 0 documents and 0.0 B sizes.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Admin	0	-	0.0 B		0.0 B	
Contact_Information	0	-	0.0 B		0.0 B	
Head_Coach	0	-	0.0 B		0.0 B	
Player	0	-	0.0 B		0.0 B	
Salary_Structure	0	-	0.0 B		0.0 B	
Team	0	-	0.0 B		0.0 B	
Trainer	0	-	0.0 B		0.0 B	

**MongoDB Collections Count: 7**

## Logs:

```
Script started at: 2022-07-04 22:32:45.027352
Connecting to PostgreSQL server...
Connection to Postgres db succeeded.
Connecting to MongoDB server...
Connection to MongoDB Server succeeded.
Database connections initialized successfully.
-----Algo Ends-----
embedding Court in Training_Session
embedding Fee_Structure in Player
embedding Grade_Card in Player
embedding Grades in Trainer
embedding Training_Session in Matches
embedding Training_Session in Schedule_Trainer
embedding Training_Session in Schedule_Player
embedding Matches in Team_Match
linking Player in Contact_Information
linking Trainer in Salary_Structure
Processing table: Admin...
Processing table: Admin completed. 7 documents inserted.
Processing table: Player...
Processing table: Player completed. 96 documents inserted.
Processing table: Team...
Processing table: Team completed. 8 documents inserted.
Processing table: Salary_Structure...
Processing table: Salary_Structure completed. 4 documents inserted.
Processing table: Contact_Information...
Processing table: Contact_Information completed. 135 documents inserted.
Processing table: Trainer...
Processing table: Trainer completed. 10 documents inserted.
Processing table: Head_Coach...
Processing table: Head_Coach completed. 5 documents inserted.

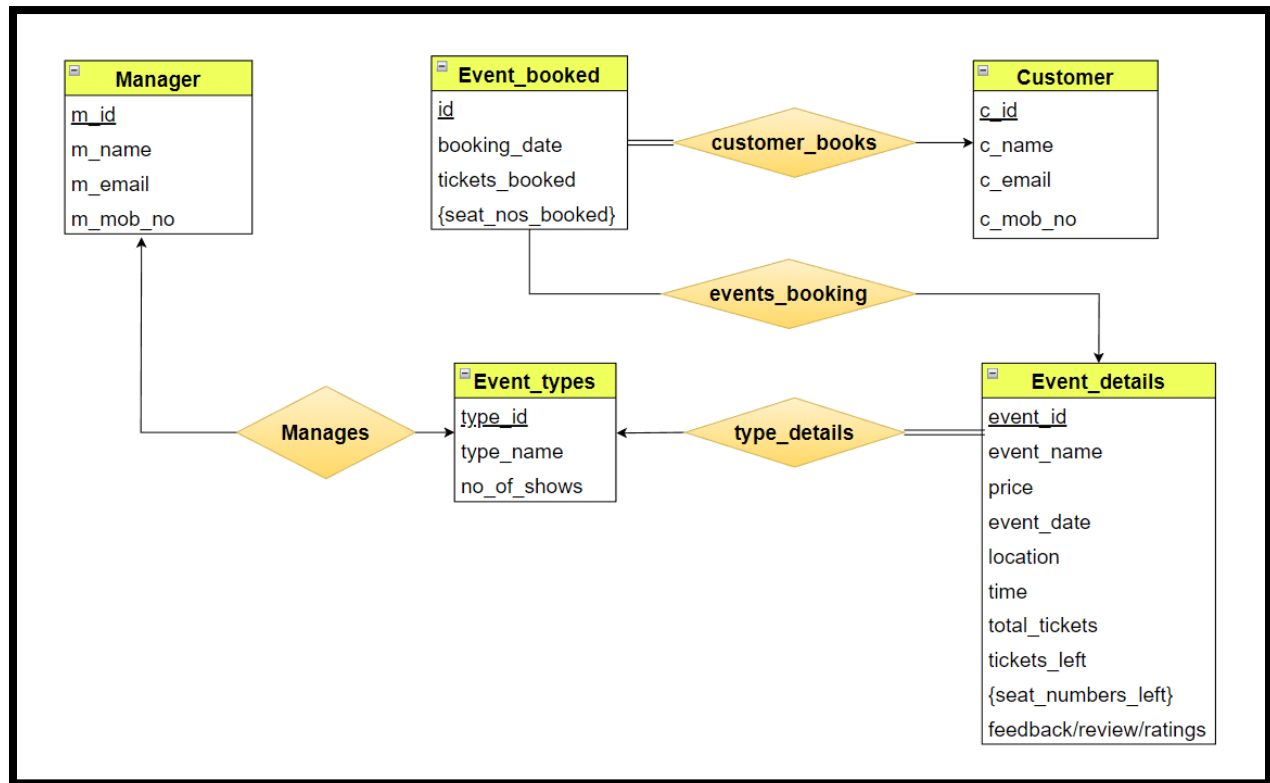
Migration completed.
7 of 7 tables migrated successfully.
Script completed at: 2022-07-04 22:32:47.309955
Total execution time: 0:00:02.282603
```

## Observation:

The relational schema is denormalized with regards to some relations and access path and we get 7 collections from 17 tables and other tables are either migrated with embedding or referencing relationships in MongoDB.

## Test Schema 3: Entertainment Booking System

### ER Diagram for Relational Databases:

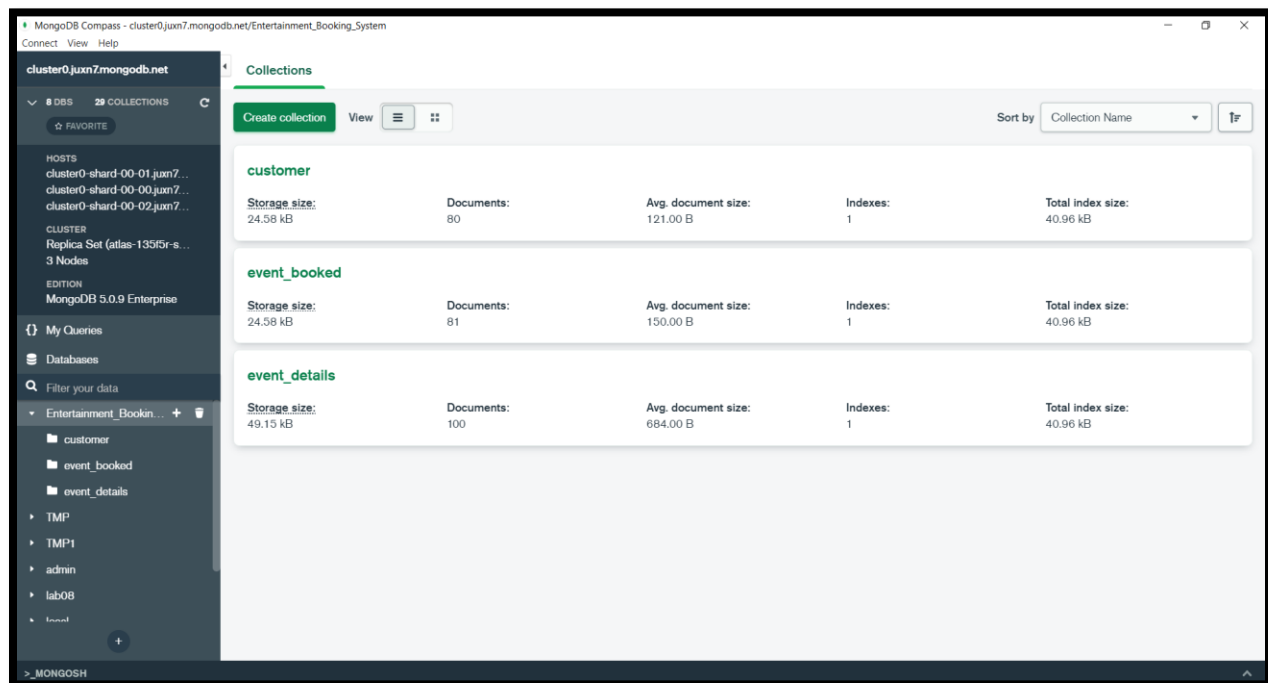


**Relational Tables Count:** 7 = 5+2 (from ER conversion to relational)

### Access Path:

event\_details/event\_type/manager  
event\_booked  
customer/event\_booked

## Migrated Result:



The screenshot shows the MongoDB Compass interface. On the left sidebar, the 'Entertainment\_Booking\_System' database is selected, showing a tree view of collections: customer, event\_booked, and event\_details. The main panel displays the 'Collections' tab with a table of collection statistics.

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
customer	24.58 kB	80	121.00 B	1	40.96 kB
event_booked	24.58 kB	81	150.00 B	1	40.96 kB
event_details	49.15 kB	100	684.00 B	1	40.96 kB

## MongoDB Collections Count: 3

## Logs:

```
PS C:\Users\Aksh Patel\Desktop\SRI\Migration_Project> python -u "c:\Users\Aksh Patel\Desktop\SRI\Migration_Project\home.py"
Script started at: 2022-07-04 22:32:15.076640
Initializing database connections...
Connecting to PostgreSQL server...
Connection to Postgres db succeeded.
Connecting to MongoDB server...
Connection to MongoDB Server succeeded.
Database connections initialized successfully.
embedding manager in event_type
embedding seats_left in event_details
embedding seat_booked in event_booked
embedding event_type in event_details
linking event_booked in customer
Processing table: customer...
Processing table: customer completed. 80 documents inserted.
Processing table: event_booked...
Processing table: event_booked completed. 81 documents inserted.
Processing table: event_details...
Processing table: event_details completed. 100 documents inserted.

Migration completed.
3 of 3 tables migrated successfully.
Script completed at: 2022-07-04 22:32:16.523706
Total execution time: 0:00:01.447066
```

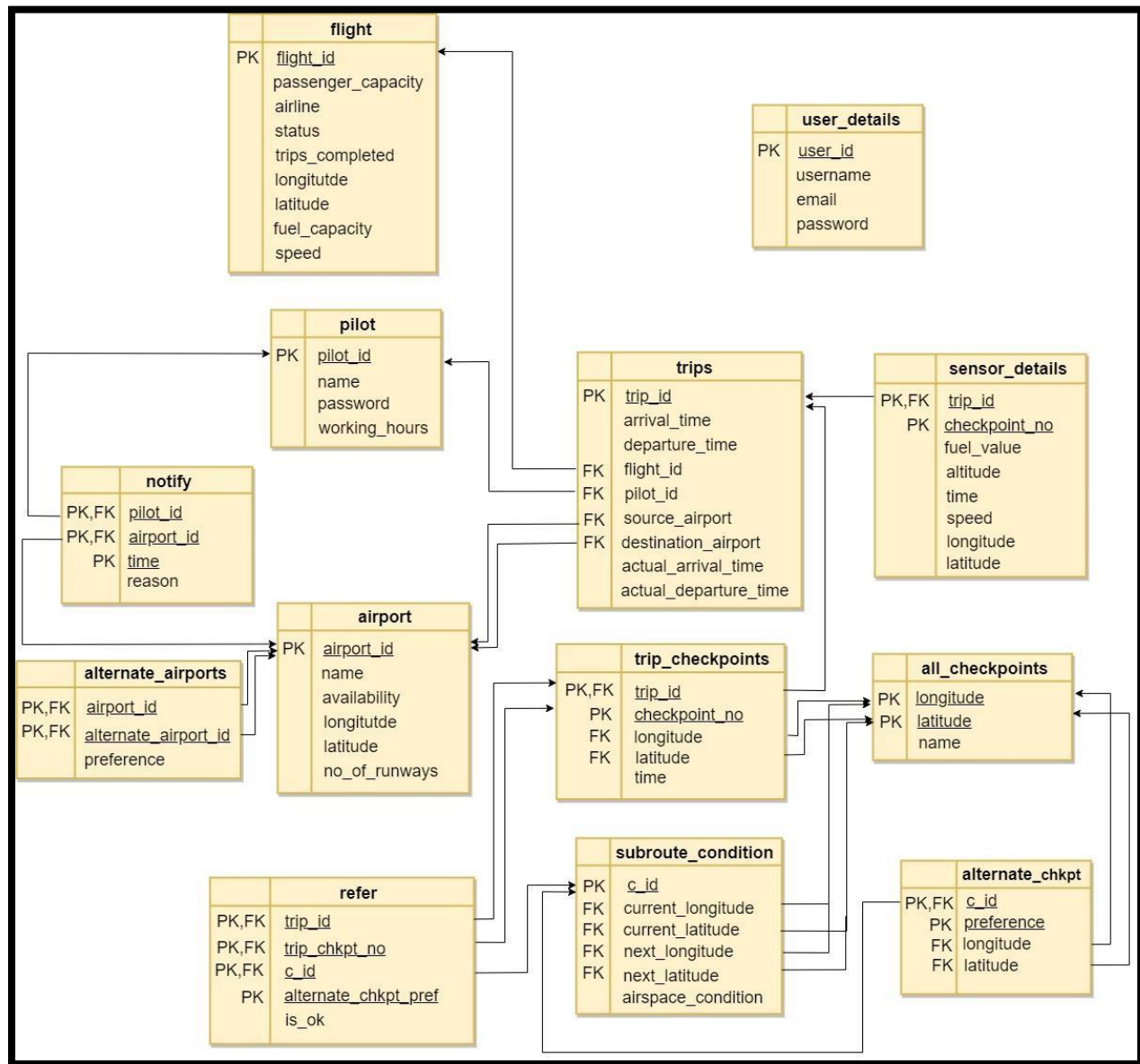
## Observation:

The relational schema is denormalized with regards to some relations and access paths and we get 3 collections from 7 tables and other tables are either migrated with embedding or referencing relationships in MongoDB.



## Test Schema 4: Flight tracking

### Schema Diagram for Relational Databases

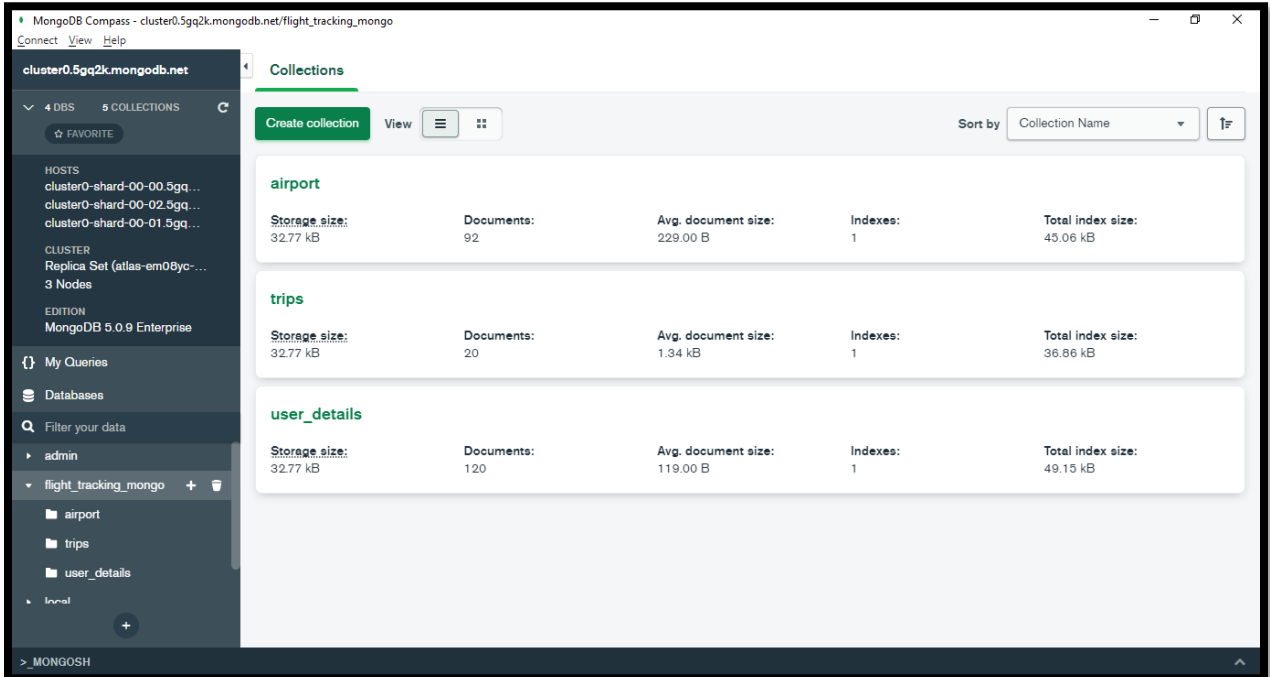


Relational Tables Count: 13

## Access Path:

airport/alternate\_airports  
trips/trip\_checkpoints  
user\_details

## Migrated Result:



Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
airport	32.77 kB	92	229.00 B	1	45.06 kB
trips	32.77 kB	20	1.34 kB	1	36.86 kB
user_details	32.77 kB	120	119.00 B	1	49.15 kB

MongoDB Collections Count: 3

## Logs:

```
Database connections initialized successfully.
.....Running Algo.....
embed pilot in trips
embed pilot in notify
embed flight in trips
embed sensor_details in trips
embed alternate_airports in airport
embed all_checkpoints in trip_checkpoints
embed all_checkpoints in subroute_condition
embed all_checkpoints in alternate_chkpt
embed trip_checkpoints in trips
embed trip_checkpoints in refer
embed subroute_condition in alternate_chkpt
embed subroute_condition in refer
-----Algo Ends-----
embedding pilot in trips
embedding pilot in notify
embedding flight in trips
embedding sensor_details in trips
embedding alternate_airports in airport
embedding all_checkpoints in trip_checkpoints
embedding all_checkpoints in subroute_condition
embedding all_checkpoints in alternate_chkpt
embedding trip_checkpoints in trips
embedding trip_checkpoints in refer
embedding subroute_condition in alternate_chkpt
embedding subroute_condition in refer
Processing table: airport...
Processing table: airport completed. 92 documents inserted.
Processing table: trips...
Processing table: trips completed. 20 documents inserted.
Processing table: user_details...
Processing table: user_details completed. 120 documents inserted.

Migration completed.
3 of 3 tables migrated successfully.
Script completed at: 2022-07-04 22:57:52.420280
```

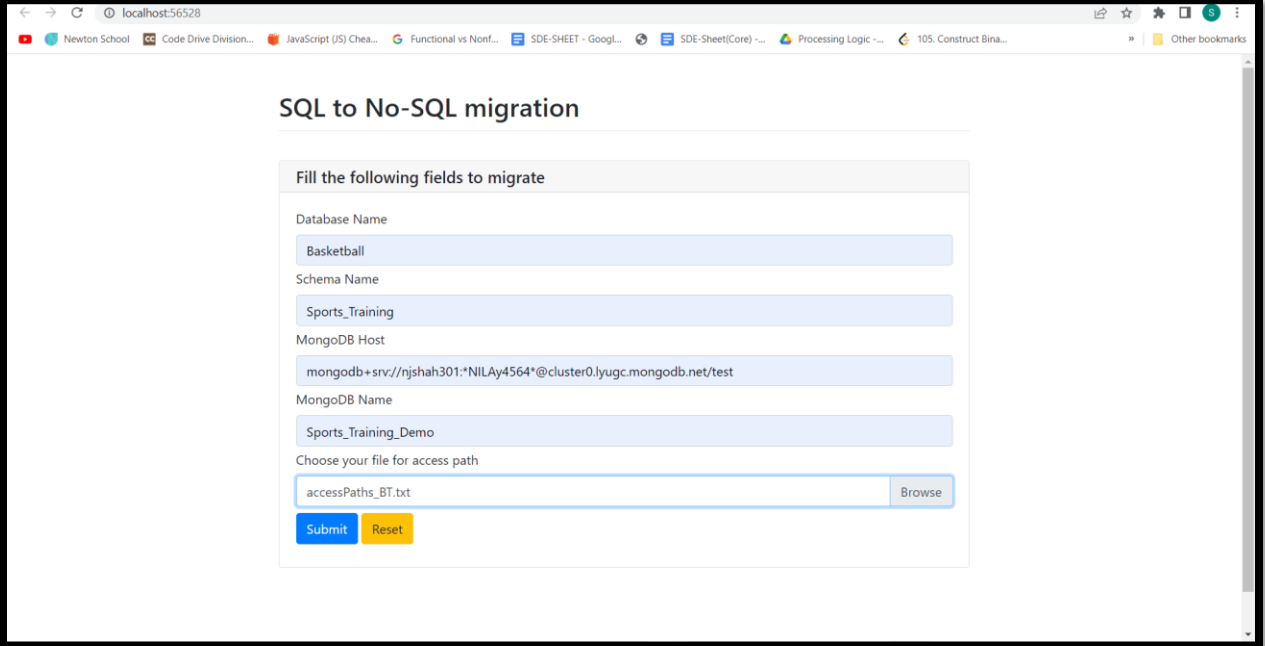
## User Interface:

### Home Page:

The screenshot shows a web browser window with the URL `localhost:56528`. The page title is "SQL to No-SQL migration". Below the title, there is a form titled "Fill the following fields to migrate". The form contains the following fields and controls:

- Database Name:
- Schema Name:
- MongoDB Host:
- MongoDB Name:
- Choose your file for access path:
- Choose file:

## Input:

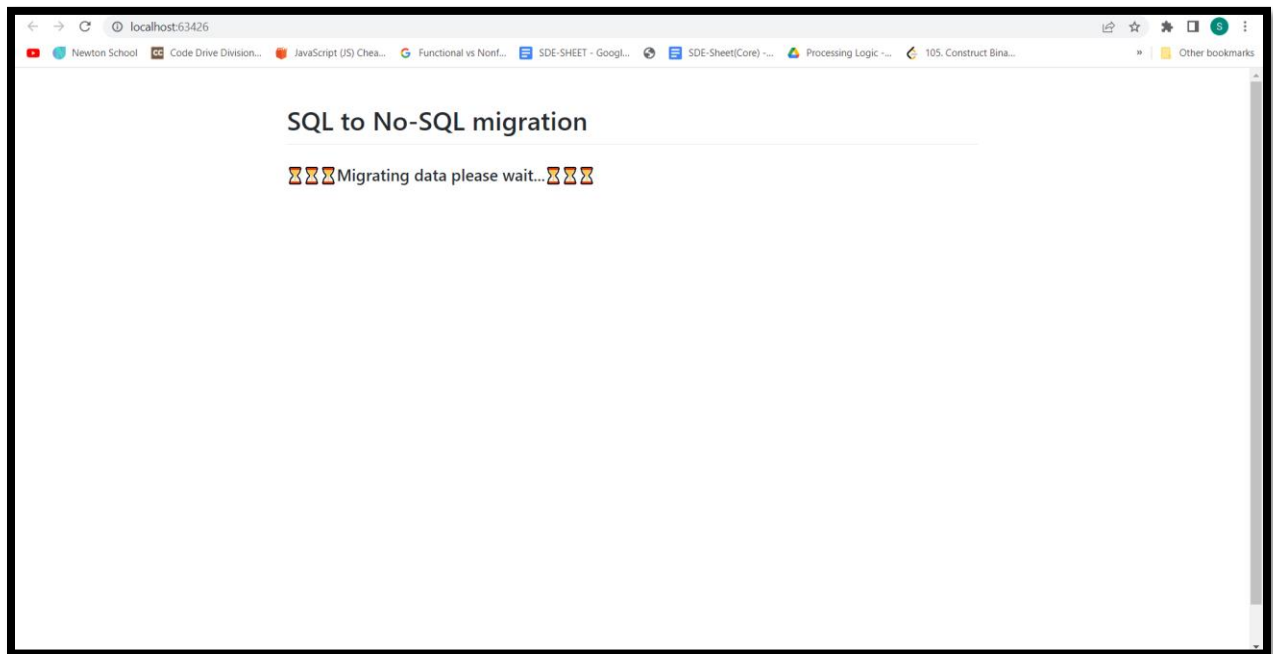


The screenshot shows a web browser window with the address bar at localhost:56528. The page title is "SQL to No-SQL migration". Below the title is a section titled "Fill the following fields to migrate". The form contains the following fields:

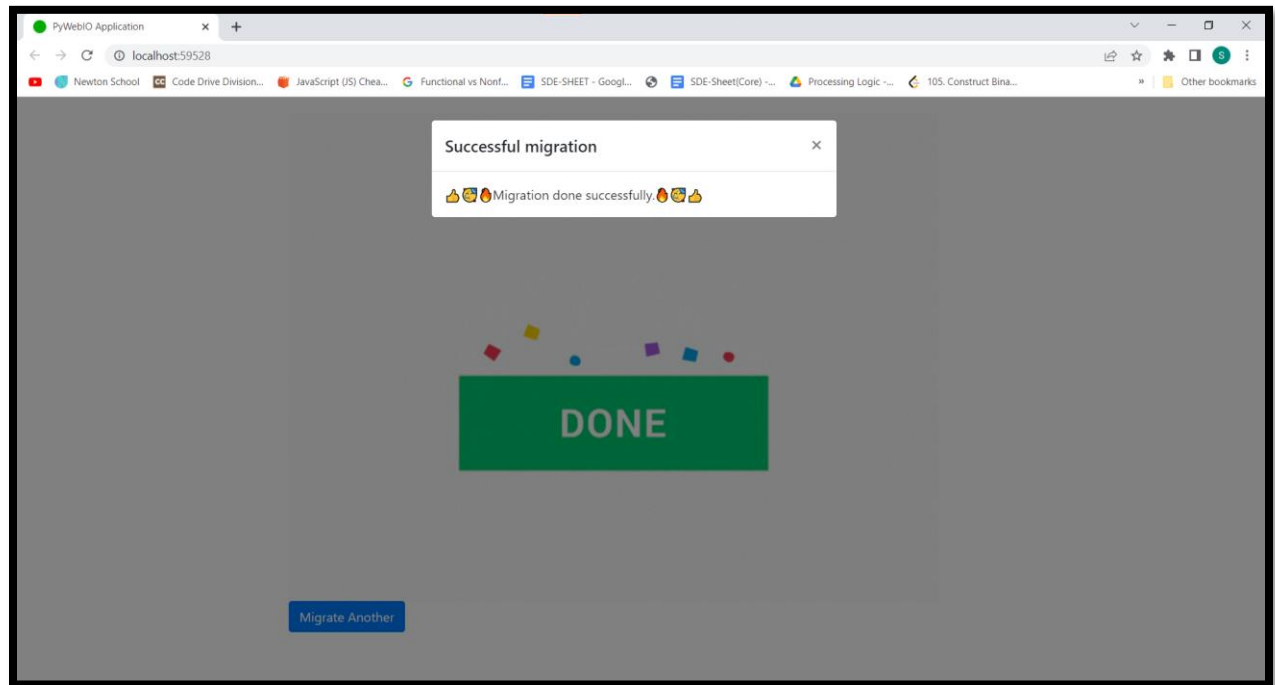
- Database Name: Basketball
- Schema Name: Sports\_Training
- MongoDB Host: mongodb+srv://njshah301:\*NILAy4564\*@cluster0.lyugc.mongodb.net/test
- MongoDB Name: Sports\_Training\_Demo
- Choose your file for access path: accessPaths\_BT.txt (with a "Browse" button)

At the bottom of the form are two buttons: "Submit" (blue) and "Reset" (yellow).

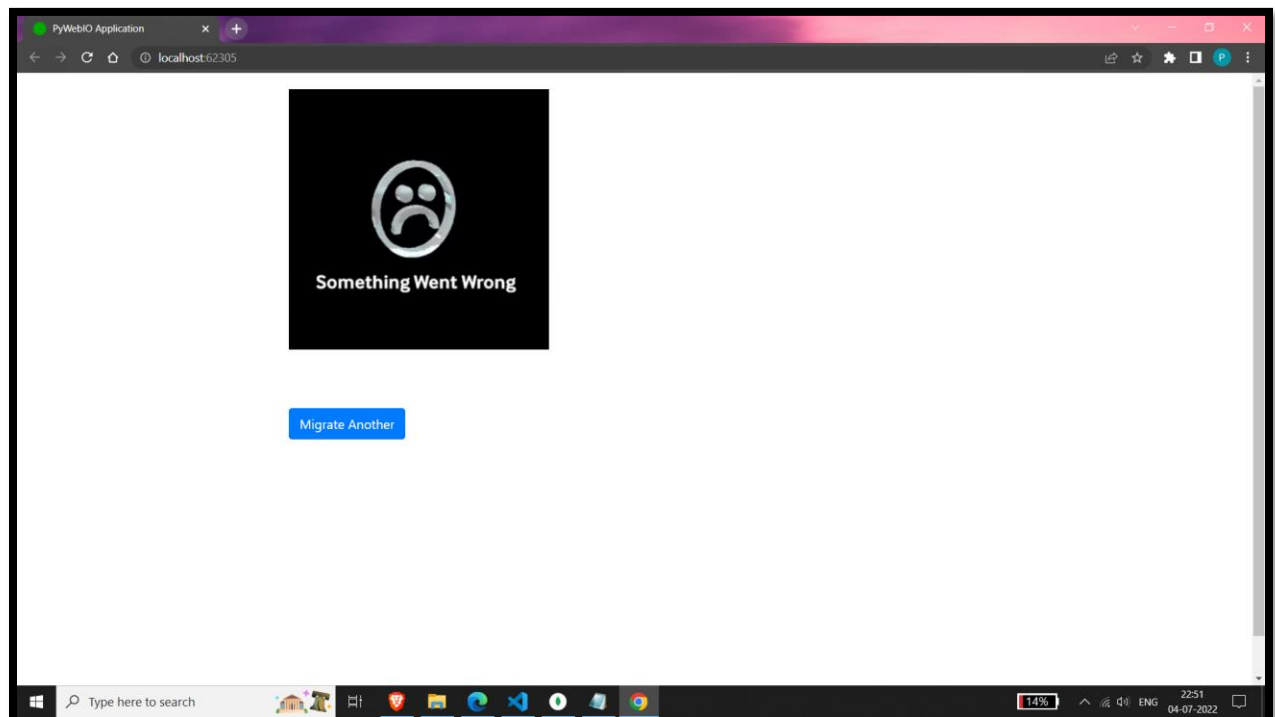
## Processing of Algorithm:



## Migration Done Successfully



## Error:



## Conclusion

This report presents a framework that implements our original algorithm of automatic mapping a PostgreSQL relational database to a MongoDB NoSQL database. Given a relational database this implementation can map it to MongoDB collection with appropriate tuples of relations embedded and referred best for querying the NoSQL database.

The algorithm uses the metadata stored in the PostgreSQL system tables. It takes into consideration the concepts from Entity-Relationship (ER) model: entity type represented by a relation in the Relational Model (RM), 1:1 and 1:M relationship type represented with Foreign Keys (FK) in the RM and N:M relationship type represented in RM with a join table that contains the Primary Keys (PK) from the original tables, each representing an FK and two 1:M relationships between the original tables and the join table.

The algorithm was presented in detail, in steps. Also, the report contains an example of automatic mapping of a PostgreSQL database to MongoDB using our algorithm. The report also presents the initial results of our algorithm that was tested on the E-Commerce Database consisting of 8 tables with many tuples, the results being encouraging.

## **Future Works**

- Experiments on complex databases (many tables and a large number of records/tables).
- Deciding creation of a secondary index based on any ordering attribute in the access path.
- Study Schema validation in MongoDB.
- Taking into consideration the number of records in the tables and the operations on the database (insert, update, delete query) in order to implement the more appropriate model of mapping to MongoDB
- Modeling tree structures with parent references.
- Extending the framework to execute mapping to MongoDB of other relational databases (Oracle, MS SQL Server and so on).
- Deployment on Azure portal via GitHub connection using App service.

## Appendix

[GitHub Repository Link](#)



## References

1. [https://www.alberton.info/postgresql\\_meta\\_info.html](https://www.alberton.info/postgresql_meta_info.html)
2. [https://www.w3schools.com/python/python\\_mongodb\\_getstarted.asp](https://www.w3schools.com/python/python_mongodb_getstarted.asp)
3. <https://pymongo.readthedocs.io/en/stable/>
4. <https://dzone.com/articles/migrate-mysql-table-data-to-mongodb-collections-us>
5. <https://stackoverflow.com/>
6. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7024609>