mongoDB

Data Management

Sahar Khanlari - 2107563

Marco Natale – 1929854

A.A. 2024/2025

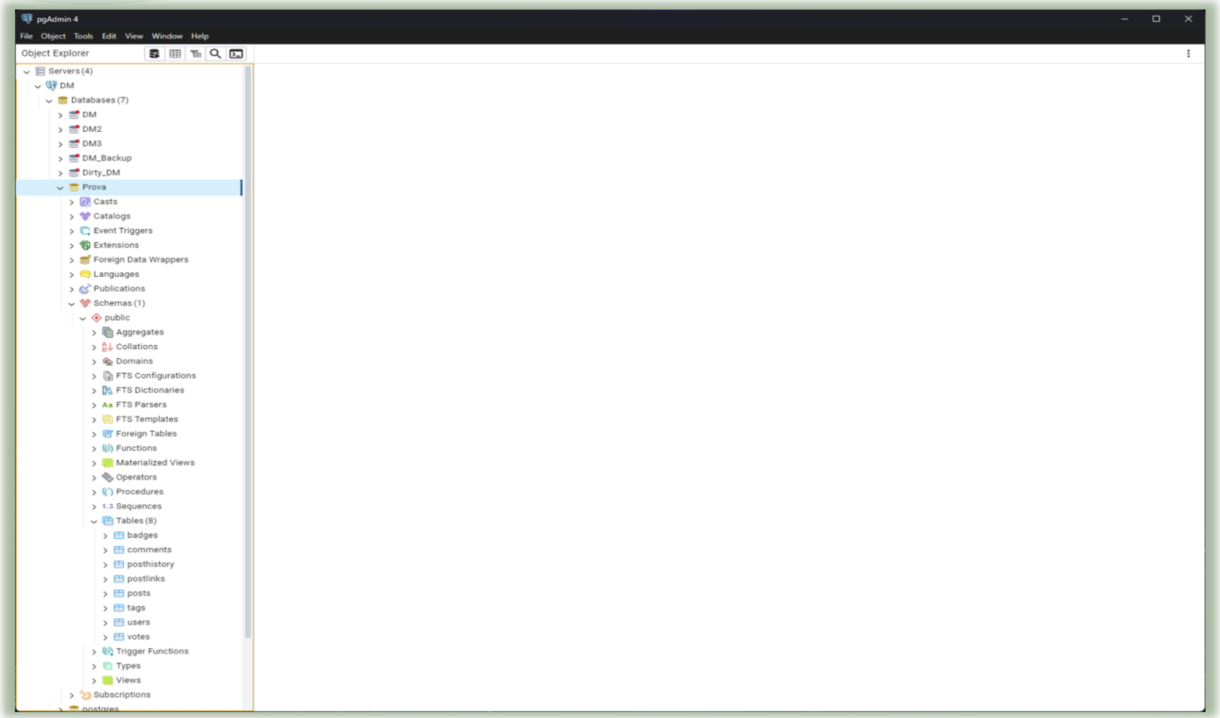
Introduction

- SQL vs NoSQL (PostgreSQL vs MongoDB)
- Goals:
  - Interface Usability
  - Flexibility
  - Scalability
  - Performance Queries and Comparison

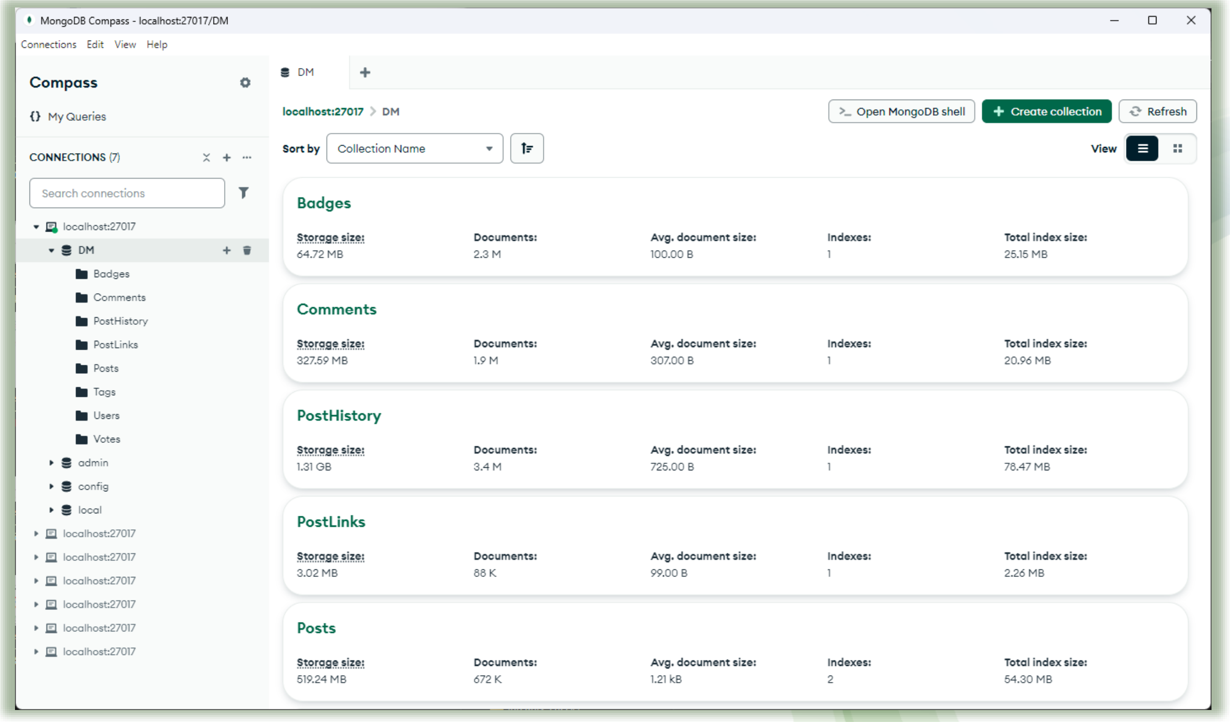
Interface Usability

Layout

pgAdmin



MongoDB Compass



Sort by: Collection Name

Badges

Storage size: 64.72 MB | Documents: 2.3 M | Avg. document size: 100.00 B | Indexes: 1 | Total index size: 25.15 MB

Comments

Storage size: 327.59 MB | Documents: 1.9 M | Avg. document size: 307.00 B | Indexes: 1 | Total index size: 20.96 MB

PostHistory

Storage size: 1.31 GB | Documents: 3.4 M | Avg. document size: 725.00 B | Indexes: 1 | Total index size: 78.47 MB

PostLinks

Storage size: 3.02 MB | Documents: 88 K | Avg. document size: 99.00 B | Indexes: 1 | Total index size: 2.26 MB

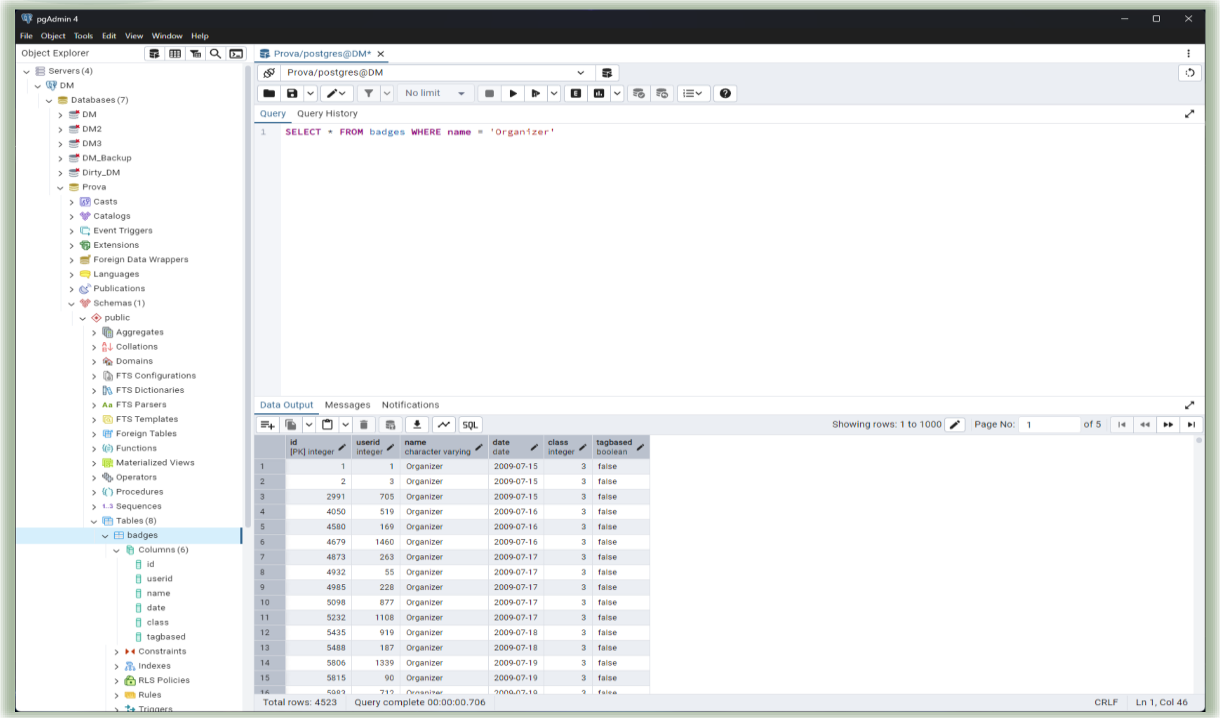
Posts

Storage size: 519.24 MB | Documents: 672 K | Avg. document size: 1.21 kB | Indexes: 2 | Total index size: 54.30 MB

Interface Usability

Navigation & Usability

pgAdmin

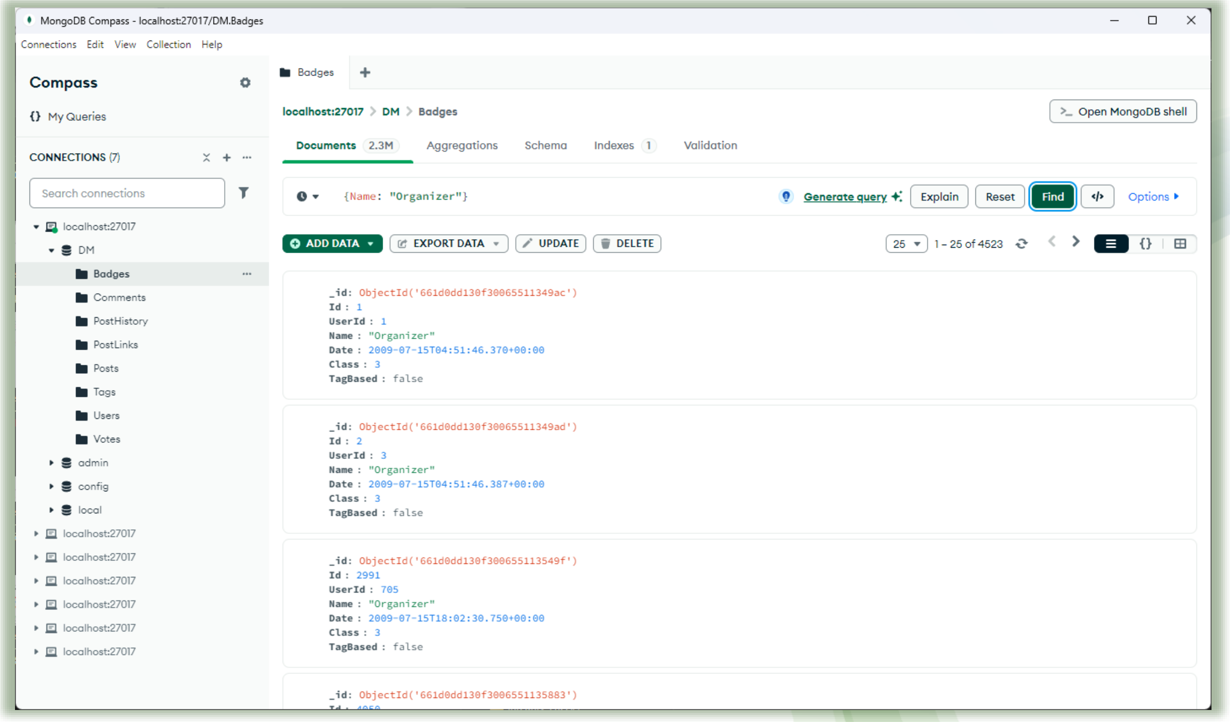


```sql
SELECT * FROM badges WHERE name = 'Organizer'
```

	id [PK] integer	userid integer	name character varying	date date	class integer	tagbased boolean
1	1	1	Organizer	2009-07-15	3	false
2	2	3	Organizer	2009-07-15	3	false
3	2991	705	Organizer	2009-07-15	3	false
4	4050	519	Organizer	2009-07-16	3	false
5	4580	169	Organizer	2009-07-16	3	false
6	4679	1460	Organizer	2009-07-16	3	false
7	4873	263	Organizer	2009-07-17	3	false
8	4932	55	Organizer	2009-07-17	3	false
9	4985	228	Organizer	2009-07-17	3	false
10	5098	877	Organizer	2009-07-17	3	false
11	5232	1108	Organizer	2009-07-17	3	false
12	5435	919	Organizer	2009-07-18	3	false
13	5488	187	Organizer	2009-07-18	3	false
14	5806	1339	Organizer	2009-07-19	3	false
15	5815	90	Organizer	2009-07-19	3	false
16	5983	712	Organizer	2009-07-19	3	false

Total rows: 4523 Query complete 00:00:00.706 CRLF Ln 1, Col 46

MongoDB Compass



```
{Name: "Organizer"}
```

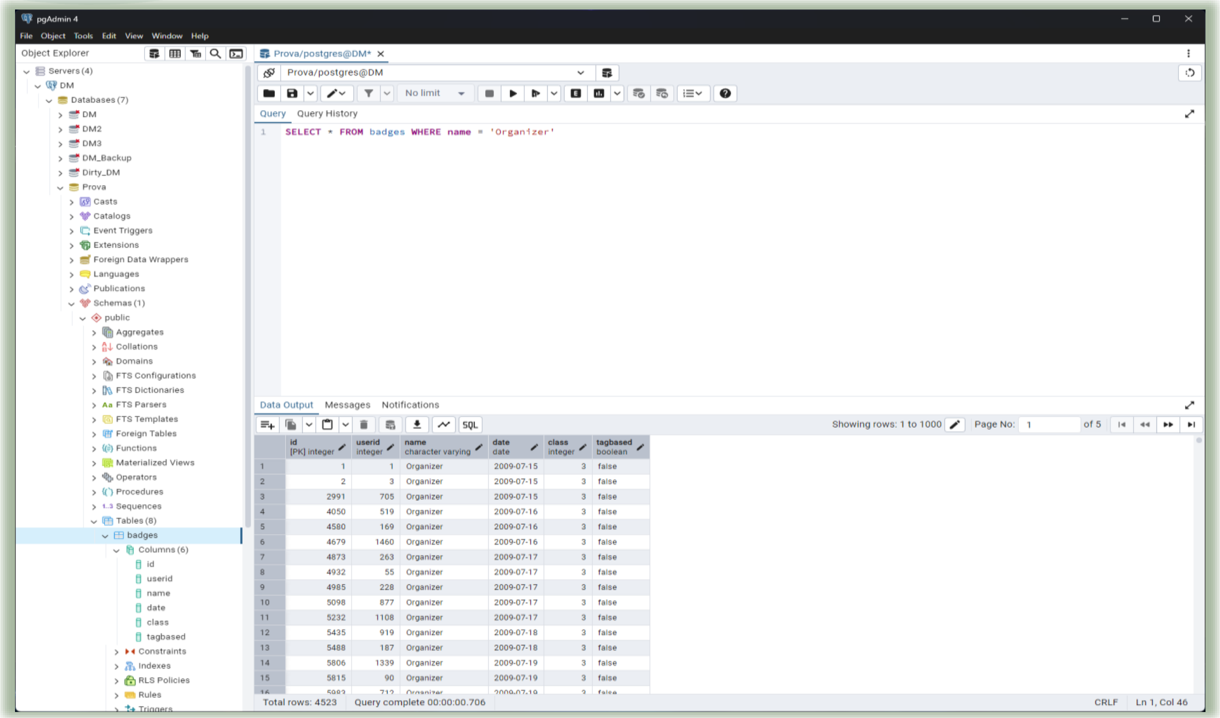
```
_id: ObjectId('661d0dd130f30065511349ac')
Id : 1
UserId : 1
Name : "Organizer"
Date : 2009-07-15T04:51:46.370+00:00
Class : 3
TagBased : false

_id: ObjectId('661d0dd130f30065511349ad')
Id : 2
UserId : 3
Name : "Organizer"
Date : 2009-07-15T04:51:46.387+00:00
Class : 3
TagBased : false

_id: ObjectId('661d0dd130f300655113549f')
Id : 2991
UserId : 705
Name : "Organizer"
Date : 2009-07-15T18:02:30.750+00:00
Class : 3
TagBased : false

_id: ObjectId('661d0dd130f3006551135883')
Id : 4050
```

Interface Usability

Navigation & Usability

pgAdmin

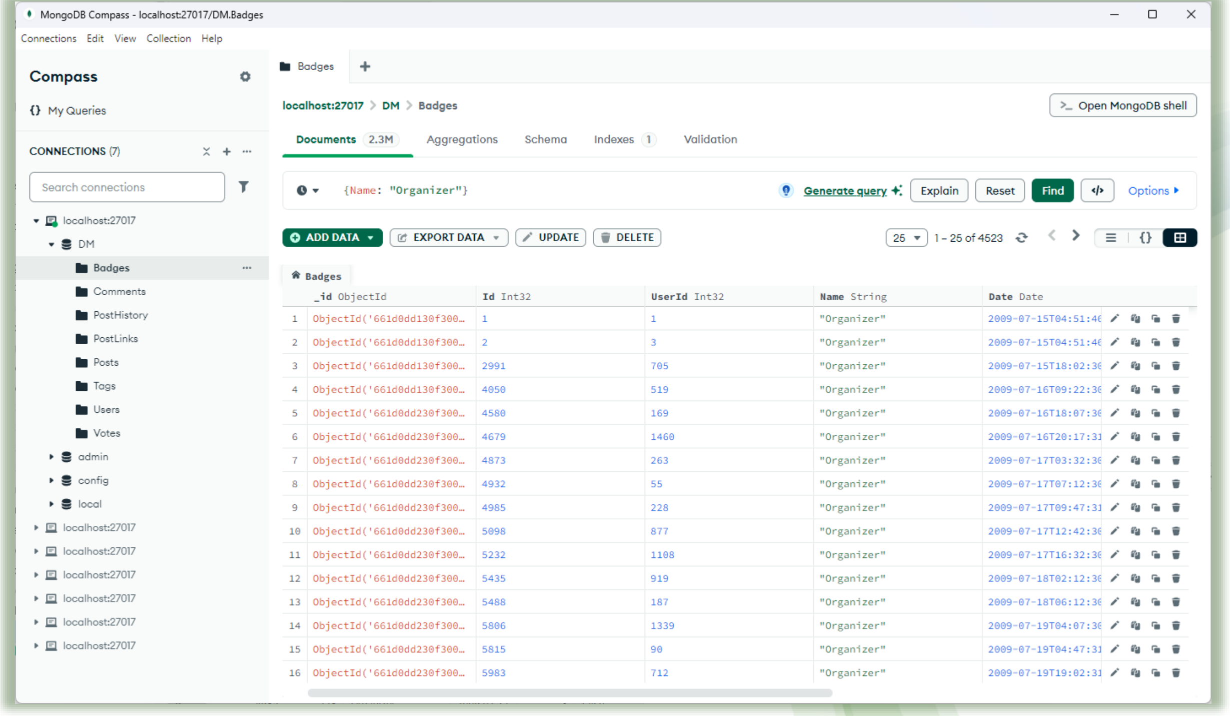
MongoDB Compass



```
SELECT * FROM badges WHERE name = 'Organizer'
```

	id [PK] integer	userid integer	name character varying	date date	class integer	tagbased boolean
1	1	1	Organizer	2009-07-15	3	false
2	2	3	Organizer	2009-07-15	3	false
3	2991	705	Organizer	2009-07-15	3	false
4	4050	519	Organizer	2009-07-16	3	false
5	4580	169	Organizer	2009-07-16	3	false
6	4679	1460	Organizer	2009-07-16	3	false
7	4873	263	Organizer	2009-07-17	3	false
8	4932	55	Organizer	2009-07-17	3	false
9	4985	228	Organizer	2009-07-17	3	false
10	5098	877	Organizer	2009-07-17	3	false
11	5232	1108	Organizer	2009-07-17	3	false
12	5435	919	Organizer	2009-07-18	3	false
13	5488	187	Organizer	2009-07-18	3	false
14	5806	1339	Organizer	2009-07-19	3	false
15	5815	90	Organizer	2009-07-19	3	false
16	5983	712	Organizer	2009-07-19	3	false

Total rows: 4523 Query complete 00:00:00.706



```
{Name: "Organizer"}
```

Badges

	_id ObjectId	Id Int32	UserId Int32	Name String	Date Date
1	ObjectId('661d0dd130f300…	1	1	"Organizer"	2009-07-15T04:51:4
2	ObjectId('661d0dd130f300…	2	3	"Organizer"	2009-07-15T04:51:4
3	ObjectId('661d0dd130f300…	2991	705	"Organizer"	2009-07-15T18:02:3
4	ObjectId('661d0dd130f300…	4050	519	"Organizer"	2009-07-16T09:22:3
5	ObjectId('661d0dd230f300…	4580	169	"Organizer"	2009-07-16T18:07:3
6	ObjectId('661d0dd230f300…	4679	1460	"Organizer"	2009-07-16T20:17:31
7	ObjectId('661d0dd230f300…	4873	263	"Organizer"	2009-07-17T03:32:3
8	ObjectId('661d0dd230f300…	4932	55	"Organizer"	2009-07-17T07:12:3
9	ObjectId('661d0dd230f300…	4985	228	"Organizer"	2009-07-17T09:47:31
10	ObjectId('661d0dd230f300…	5098	877	"Organizer"	2009-07-17T12:42:3
11	ObjectId('661d0dd230f300…	5232	1108	"Organizer"	2009-07-17T16:32:3
12	ObjectId('661d0dd230f300…	5435	919	"Organizer"	2009-07-18T02:12:3
13	ObjectId('661d0dd230f300…	5488	187	"Organizer"	2009-07-18T06:12:3
14	ObjectId('661d0dd230f300…	5806	1339	"Organizer"	2009-07-19T04:07:3
15	ObjectId('661d0dd230f300…	5815	90	"Organizer"	2009-07-19T04:47:31
16	ObjectId('661d0dd230f300…	5983	712	"Organizer"	2009-07-19T19:02:31

Interface Usability

Functionality

➢ Differences

Query Interface:

- pgAdmin: Uses a simple text editor for writing SQL queries.
- MongoDB Compass: Provides a visual aggregation pipeline builder for NoSQL queries.

Predefined Queries:

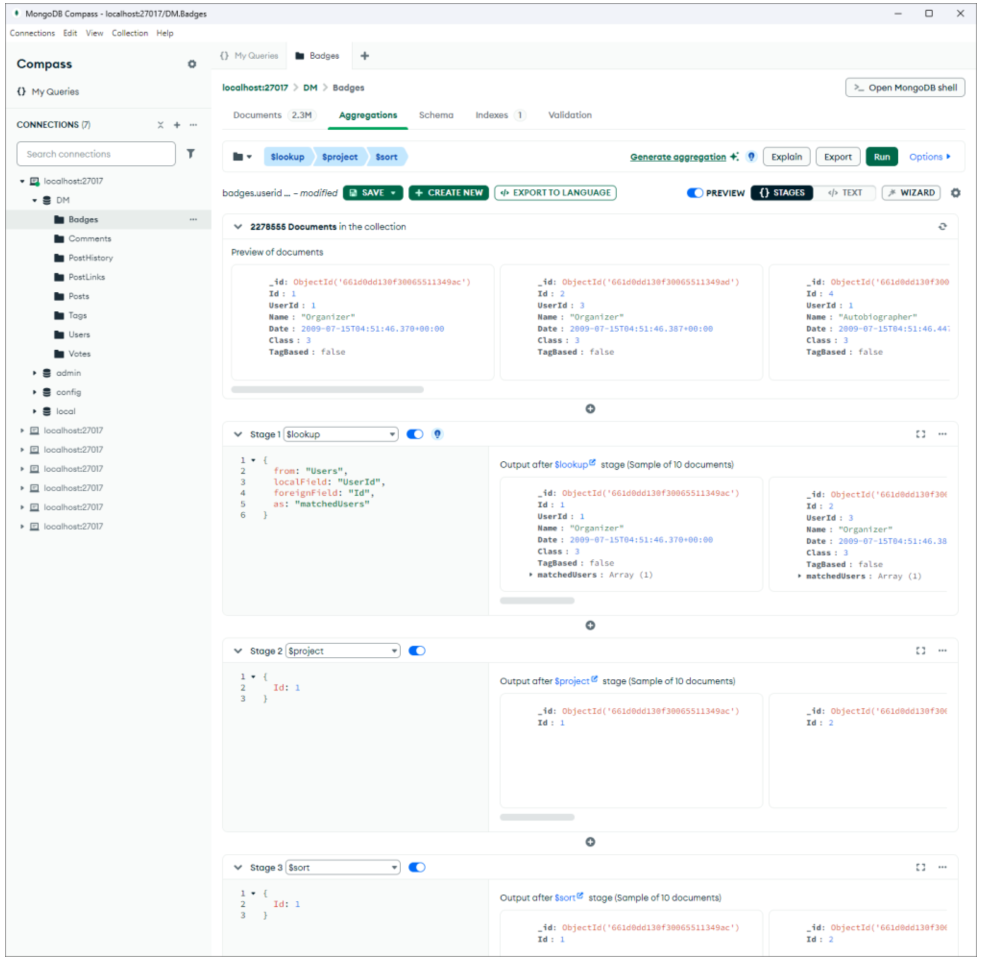
- pgAdmin: Offers predefined queries (e.g., view data, CRUD scripts) via right-click.
- MongoDB Compass: No predefined SQL-like queries, relies on document-based query building.

➢ Similarities

- Both provide a GUI interface for database management.
- Both allow import/export of data.
- Both include a CLI window for command-line interaction.

Interface Usability

Functionality



2278555 Documents in the collection

Preview of documents

```
_id: ObjectId('661d0dd130f30065511349ac')
Id : 1
UserId : 1
Name : "Organizer"
Date : 2009-07-15T04:51:46.370+00:00
Class : 3
TagBased : false
```

```
_id: ObjectId('661d0dd130f30065511349ad')
Id : 2
UserId : 3
Name : "Organizer"
Date : 2009-07-15T04:51:46.387+00:00
Class : 3
TagBased : false
```

```
_id: ObjectId('661d0dd130f300
Id : 4
UserId : 1
Name : "Autobiographer"
Date : 2009-07-15T04:51:46.44
Class : 3
TagBased : false
```

Stage 1 $lookup

```
{
  from: "Users",
  localField: "UserId",
  foreignField: "Id",
  as: "matchedUsers"
}
```

Output after $lookup stage (Sample of 10 documents)

```
_id: ObjectId('661d0dd130f30065511349ac')
Id : 1
UserId : 1
Name : "Organizer"
Date : 2009-07-15T04:51:46.370+00:00
Class : 3
TagBased : false
matchedUsers : Array (1)
```

```
_id: ObjectId('661d0dd130f30
Id : 2
UserId : 3
Name : "Organizer"
Date : 2009-07-15T04:51:46.38
Class : 3
TagBased : false
matchedUsers : Array (1)
```

Stage 2 $project

```
{
  Id: 1
}
```

Output after $project stage (Sample of 10 documents)

```
_id: ObjectId('661d0dd130f30065511349ac')
Id : 1
```

```
_id: ObjectId('661d0dd130f30
Id : 2
```

Stage 3 $sort

```
{
  Id: 1
}
```

Output after $sort stage (Sample of 10 documents)

```
_id: ObjectId('661d0dd130f30065511349ac')
Id : 1
```

```
_id: ObjectId('661d0dd130f30
Id : 2
```

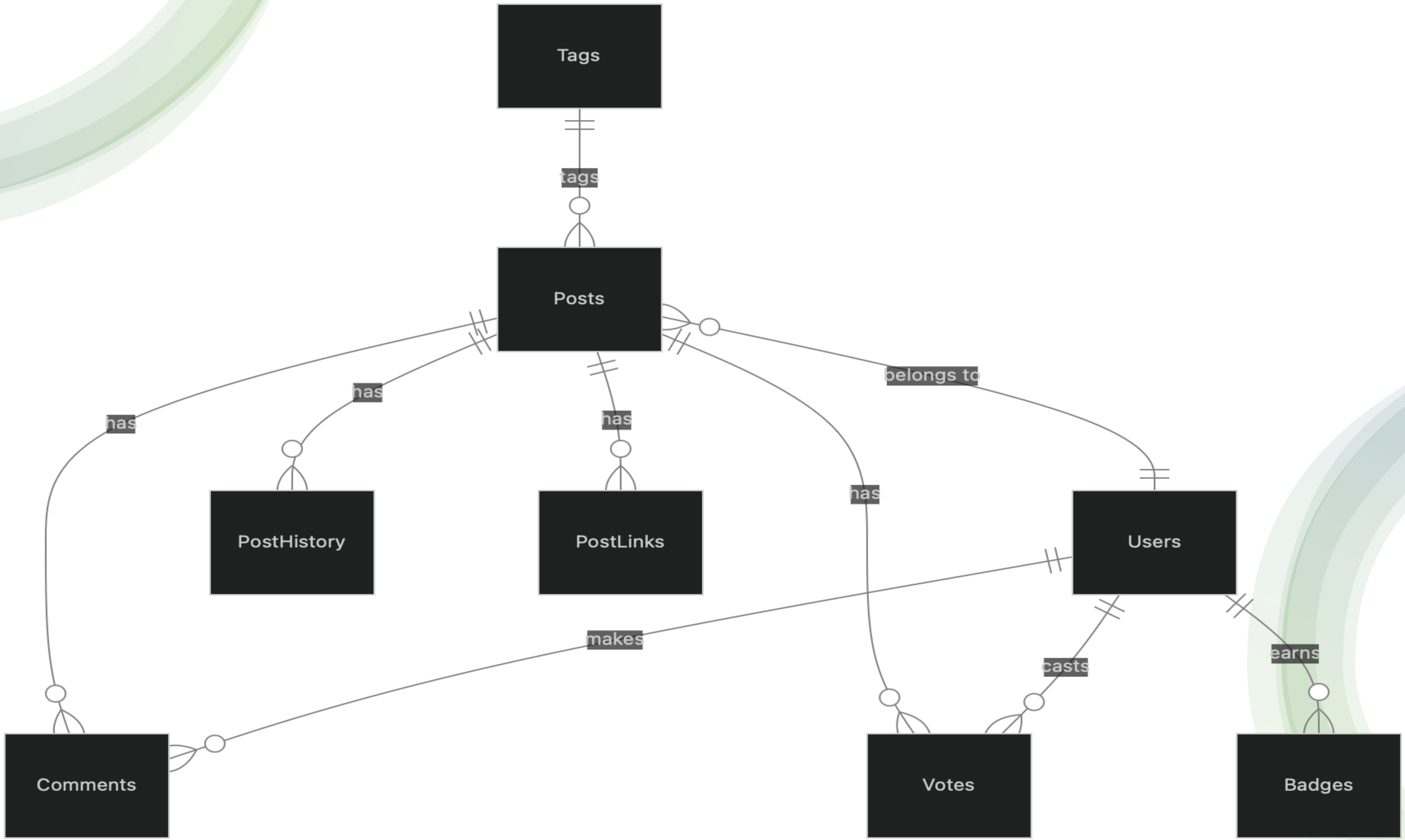
Flexibility

Feature	SQL (Relational Databases)	MongoDB (NoSQL)
Schema	Strict, predefined schema	Schema-less (Flexible documents)
Structure	Tables, rows, and columns	Documents (JSON/BSON format)
Data Relationships	Normalized, uses joins to relate tables	Denormalized, embeds related data
Schema Changes	Requires migrations, downtime	Dynamic, fields can be added/removed
Use Case	Best for structured & transactional data	Best for unstructured, evolving data

Scalability

Feature	SQL (Relational Databases)	MongoDB (NoSQL)
Scaling Strategy	Vertical Scaling (Scale-Up)	Horizontal Scaling (Scale-Out)
Data Distribution	Difficult to shard, manual partitioning	Built-in sharding for automatic distribution
Read/Write Performance	Efficient for transactions, joins slow large queries	Faster reads, distributed writes
Replication	Master-slave or cluster-based replication	Replica sets with automatic failover
High Availability	Requires complex setup	Native support for replication & redundancy
Global Distribution	Requires additional tools	Geo-distributed databases supported

The Dataset


Tags
tags
Posts
has
has
has
has
belongs to
PostHistory
PostLinks
Users
makes
casts
earns
Comments
Votes
Badges

Importing the tables

- **PostgreSQL**:
  - Badges: 12,78 sec
  - Comments: 16,87 sec
  - PostHistory: 62 sec
  - PostLinks: 0,72 sec
  - Posts: 27,38 sec
  - Tags: 0,11 sec
  - Users: 14,72 sec
  - Votes: 23,34 sec

- **MongoDB** doesn’t measure time

Performance

Integrity Checks

Data Cleaning

Foreign Keys (only for SQL)

Different Query Types

Key Observation

Checking if PostLinks references Posts

- **Inefficient query**:

```
SELECT pl.id
FROM postlinks pl
WHERE pl.postid NOT IN (SELECT p.id FROM posts p);
```

Execution Time: 2 hours 37 minutes

- **Efficient query**:

```
SELECT pl.id
FROM postlinks pl
EXCEPT
SELECT pl.id
FROM postlinks pl JOIN posts p ON p.id = pl.postid
ORDER BY id;
```

Execution Time: 258 milliseconds

Key Observation #2

- MongoDB requires some indexing for very large data
- The queries couldn’t complete/took very long without them

- The following indexes have been created:
  - Posts(Id)
  - Users(Id)

Integrity Checks

Among 8 queries	PostgreSQL	MongoDB
BEST **PostLinks → Posts**	258 ms	1.825 sec
AVERAGE	4.212 sec	33.466 sec (~7x)
WORST **Votes → Posts**	7.914 sec	76.631 sec

Data Cleaning

Among 8 queries	PostgreSQL	MongoDB
BEST **PostLinks → Posts**	379 ms	1.460 sec
AVERAGE	9.644 sec	34.108 sec (~2.5x)
WORST	29.958 sec PostHistory → Posts	87.51 sec Votes → Posts

- Votes table is **too large** for MongoDB’s $lookup.
- PostgreSQL’ s join is optimized on large tables.

Foreign Keys

➢ **Posts** references:

- Users(Id) → OwnerUserId
- Posts(Id) → ParentId & AcceptedAnswerId

➢ **Comments** references:

- Posts(Id) → PostId

➢ **Votes** references:

- Users(Id) → UserId
- Posts(Id) → PostId

➢ **Badges** references:

- Users(Id) → UserId

➢ **PostHistory** references:

- Users(Id) → UserId
- Posts(Id) → PostId

➢ **PostLinks** references:

- Posts(Id) → PostId & RelatedPostId

Performance Queries Overview

	Read	Write	Joins	Aggregation	Text Search	Nested Updates	Complex Joins	Pagination	Deletion	Count
SQL	0.779	0.453	14.673	2.236	1.286	3.259	8.998	0.583	>8h	0.884
MongoDB	6.867	0.003	88.112	2.518	2.3296	1.7278	>6h	2.470	10.869	1.144

Simple Read and Write

```
SELECT *
FROM Posts
WHERE CreationDate BETWEEN '2009-01-01' AND '2009-12-31';
```

	Read	Write
SQL	0.779	0.453

```
INSERT INTO Posts (Id, PostTypeId, CreationDate, Score, Title, Body, Tags)
VALUES
(2, 1, NOW(), 10, 'Title 1', 'Body 1', '<mac><crash>'),
(3, 2, NOW(), 15, 'Title 2', 'Body 2', '<windows>');
```

- Declarative syntax
- Standard SQL

```
posts.find({"CreationDate":
                {"$gte": datetime(2009, 1, 1),      "$lte":
datetime(2009, 12, 31)}
                })
```

	Read	Write
MongoDB	6.867	0.003

```
posts.insert_many([
{"Id": 2, "PostTypeId": 1, "CreationDate": datetime.now(), "Score": 10,
"Title": "Title 1", "Body": "Body 1", "Tags": "<mac><crash>"},
{"Id": 3, "PostTypeId": 2, "CreationDate": datetime.now(), "Score": 15,
"Title": "Title 2", "Body": "Body 2", "Tags": "<windows>"}
])
```

- JSON-like documents
- Operators and expressions: $gt, $eq, ...

Simple Joins and Aggregations

```
SELECT p.*, u.DisplayName
FROM Posts p
JOIN Users u ON p.OwnerUserId = u.Id;
```

	Joins	Aggregation
SQL	14.673	2.236

```
SELECT Tags, COUNT(*)
FROM Posts
GROUP BY Tags;
```

- Optimized for joins
- Relationships are optimized

- Aggregation with group by

```
posts.aggregate([
{"$lookup": {"from": "Users", "localField": "OwnerUserId",
"foreignField": "Id", "as": "owner"}},
{"$unwind": "$owner"},
{"$project": {"_id": 0, "Title": 1, "OwnerDisplayName":
"$owner.DisplayName"}}
])
```

	Joins	Aggregation
MongoDB	88.112	2.518

```
posts.aggregate([
{"$unwind": "$Tags"},
{"$group": {"_id": "$Tags", "count": {"$sum": 1}}}
])
```

- No native join operation
- $lookup stage is used in aggregations

- Aggregation in stages

Text Search and Nested Updates

```
SELECT *
FROM Posts
WHERE Body LIKE '%virtual machine%';
```

	Text Search	Nested Updates
SQL	1.286	3.259

```
UPDATE Users
SET Reputation = Reputation + 10
WHERE Id IN
(SELECT OwnerUserId
FROM Posts
GROUP BY OwnerUserId
HAVING COUNT(*) > 10);
```

```
posts.find({"Body":
{"$regex": "virtual machine", "$options": "i"}
})
```

	Text Search	Nested Updates
MongoDB	2.3296	1.7278

```
owners_with_more_than_10_posts =
posts.aggregate([
{"$group": {"_id": "$OwnerUserId", "postCount": {"$sum": 1}}},
{"$match": {"postCount": {"$gt": 10}}}
])
owner_ids = [owner["_id"] for owner in
owners_with_more_than_10_posts]
users.update_many({"Id": {"$in": owner_ids}}, {"$inc": {"Reputation":
10}})
```

Complex Joins

```sql
SELECT p.Title, c.Text, u.DisplayName
FROM Posts p
JOIN Comments c ON p.Id = c.PostId
JOIN Users u ON c.UserId = u.Id;
```

	Complex Joins
SQL	8.998
MongoDB	>6h

```
posts.aggregate([
{"$lookup": {"from": "Comments", "localField": "Id", "foreignField": "PostId", "as": "comments"}},
{"$unwind": "$comments"},
{"$lookup": {"from": "Users", "localField": "comments.UserId", "foreignField": "Id", "as": "commentUser"}},
{"$unwind": "$commentUser"},
{"$project": {"_id": 0, "Title": 1, "CommentText": "$comments.Text", "CommenterName": "$commentUser.DisplayName"}}
])
```

Pagination, Deletion and Count

	Pagination	Deletion	Count
SQL	0.583	>8h	0.884
MongoDB	2.470	10.869	1.144

```
SELECT *
FROM Posts
ORDER BY CreationDate DESC LIMIT 10
OFFSET 20;
```

```
DELETE FROM Posts
WHERE Score < 5 AND Id NOT IN (SELECT
PostId FROM Comments);
```

```
SELECT COUNT(*)
FROM Users
WHERE Reputation > 100;
```

```
posts.find().sort("CreationDate", -
1).skip(20).limit(10)
```

```
post_ids_with_comments = comments.distinct("PostId")
posts.delete_many({"Score": {"$lt": 5}, "Id": {"$nin":
post_ids_with_comments}})
```

```
users.count_documents({"Reputation": {"$gt": 100}})
```

Conclusions

	Read	Write	Joins	Aggregation	Text Search	Nested Updates	Complex Joins	Pagination	Deletion	Count
SQL	0.779	0.453	14.673	2.236	1.286	3.259	8.998	0.583	>8h	0.884
MongoDB	6.867	0.003	88.112	2.518	2.3296	1.7278	>6h	2.470	10.869	1.144

- MongoDB better suited for changing data (faster write, update and delete operations)
- PostgreSQL faster in structured data

Use case of this dataset: large forum

→ Dynamic data, but frequent join requests (loading posts and comments)

→ Optimizations are put in place (e.g., on delete set null)

Thank you

Sahar Khanlari - 2107563

Marco Natale – 1929854

A.A. 2024/2025