



M149 – Database Management Systems

NOSQL LA CRIME API

Stefanos Giorkas |
7115112300005

University of Athens | Dept. of
Informatics & Telecommunications



Table of Contents

1. Introduction	2
2. Database Schema Design and Justification	3
📌 Collections Overview	3
📁 Crimes Collection	3
📁 Victims Collection	4
📁 Weapons Collection	4
📁 Upvotes Collection	5
3. Query Optimization & Indexing	6
Crimes Collection Indexes	6
Upvotes Collection Indexes	6
4. Sample Responses for Each Query	7
◆ Query 1: Find the total number of reports per crime code in a given time range.	7
◆ Query 2: Find the total number of reports per day for a specific “Crm Cd” and time range.	7
◆ Query 3: Find the three most common crimes per area for a given date.	8
◆ Query 4: Find the two least common crimes in a given time range.	9
◆ Query 5: Find the types of weapons used for the same “Crm Cd” across different areas.	9
◆ Query 6: Find the top 50 most upvoted reports for a given date.	10
◆ Query 7: Find the 50 most active officers by number of upvotes given.	11
◆ Query 8: Find the 50 officers who have upvoted reports across the most different areas.	12
◆ Query 9: Find reports that have received upvotes from the same email for more than one badge number.	12
◆ Query 10: Find all areas where a given name has upvoted a report.	12
5. Conclusion	14

1. Introduction

This report outlines the design and implementation of a NoSQL database system using MongoDB and FastAPI to store and manage crime reports from the Los Angeles Police Department (LAPD). The system is designed to efficiently handle large volumes of crime data, allowing police officers to query, update, and vote on reports.

A key feature of this implementation is the upvote system, which enables officers to highlight important crime reports while ensuring data integrity by preventing duplicate votes. The database schema is structured with separate collections for crimes, victims, weapons, and upvotes, improving data organization and retrieval performance.

The system was developed to meet the project's requirements, ensuring efficient queries, optimized indexing, and scalability. It supports real-time updates, allowing crime reports and upvotes to be added dynamically. This document provides an overview of the database schema, design decisions, implemented queries, and the performance optimizations applied to enhance efficiency.

2. Database Schema Design and Justification

The database schema was designed based on the principle of normalization and efficiency, ensuring query optimization and logical separation of concerns.

Collections Overview


The system consists of the following four primary collections:

1. **crimes** → Stores crime reports
2. **victims** → Stores information about victims
3. **weapons** → Stores weapon usage information
4. **upvotes** → Stores upvotes by police officers

Crimes Collection


```
{  
  "_id": ObjectId("..."),  
  "DR_NO": 190326475,  
  "date_reported": "03/01/2020 12:00:00 AM",  
  "date_occurred": "03/01/2020 12:00:00 AM",  
  "time_occurred": 2130,  
  "area": { "id": 7, "name": "Wilshire", "reporting_district": 784 },  
  "crime_code": 510,  
  "crime_description": "VEHICLE - STOLEN",  
  "status": { "code": "AA", "description": "Adult Arrest" },  
  "location": { "address": "1900 S LONGWOOD AV", "coordinates": [-118.3506, 34.0375] }  
}
```

NOSQL LA CRIME API

 **Justification:** Crime reports are the core data of the system and are indexed on ``crime_code`` and ``date_occurred`` for fast retrieval.

Victims Collection


```
{
  "_id": ObjectId("..."),
  "crime_id": ObjectId("..."),
  "age": 34,
  "sex": "M",
  "descent": "H"
}
```

 **Justification:** Victims are stored separately to reduce redundancy and allow better privacy handling.

Weapons Collection


```
{
  "_id": ObjectId("..."),
  "crime_id": ObjectId("..."),
  "weapon_code": 400,
  "weapon_description": "STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)"
}
```

NOSQL LA CRIME API

 **Justification:** Since not all crimes involve weapons, we store weapon data separately, optimizing storage.

Upvotes Collection

```
{
  "_id": ObjectId("..."),
  "crime_id": ObjectId("..."),
  "officer": {
    "badge_number": "123456",
    "name": "John Doe",
    "email": "jdoe@lapd.gov"
  },
  "upvote_date": "2025-02-11"
}
```

 **Justification:** This structure ensures police officers can only vote once per crime report and enables tracking of officer engagement.

3. Query Optimization & Indexing


To enhance query performance, the following indexes were implemented:

Crimes Collection Indexes

```
db.crimes.createIndex({ "crime_code": 1 });  
db.crimes.createIndex({ "date_occurred": 1 });  
db.crimes.createIndex({ "area.name": 1 });
```

Upvotes Collection Indexes

```
db.upvotes.createIndex({ "crime_id": 1 });  
db.upvotes.createIndex({ "officer.email": 1, "officer.badge_number": 1 }, { unique: true });
```

 **Impact:** These indexes optimize retrieval operations by reducing search time in large datasets.

4. Sample Responses for Each Query

Below is a list of queries as specified in the project description, where sample responses should be manually inserted.

◆ Query 1: Find the total number of reports per crime code in a given time range.

Request:

GET /api/crimes/count-by-code?start_date=01/01/2023&end_date=01/01/2024

Response:

```
[
  {
    "crime_code": 354,
    "count": 396
  },
  {
    "crime_code": 510,
    "count": 58
  },
  ...
  {
    "crime_code": 840,
    "count": 1
  }
]
```

◆ Query 2: Find the total number of reports per day for a specific “Crm Cd” and time range.

Request:

GET /api/crimes/daily-count?crime_code=510&start_date=01/01/2023&end_date=01/01/2024

Response:

```
[
  {
```


NOSQL LA CRIME API

```
"date": "01/01/2023",
"report_count": 58
},
{
  "date": "01/02/2023",
  "report_count": 63
},
...
{
  "date": "01/01/2024",
  "report_count": 52
}
]
```

◆ Query 3: Find the three most common crimes per area for a given date.

Request:

GET /api/crimes/most-common?date=01/01/2023

Response:

```
[
  {
    "_id": "77th Street",
    "area": "77th Street",
    "crimes": [
      {
        "crime": "THEFT OF IDENTITY",
        "count": 37
      },
      {
        "crime": "ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT",
        "count": 9
      },
      {
        "crime": "VEHICLE - STOLEN",
        "count": 6
      }
    ]
  },
]
```

NOSQL LA CRIME API

```
...  
]
```

◆ Query 4: Find the two least common crimes in a given time range.

Request:

GET /api/crimes/least-common?start_date=01/01/2023&end_date=01/01/2024

Response:

```
[  
  {  
    "crime": "LEWD CONDUCT",  
    "count": 1  
  },  
  {  
    "crime": "FALSE IMPRISONMENT",  
    "count": 1  
  }  
]
```

◆ Query 5: Find the types of weapons used for the same “Crm Cd” across different areas.

Request:

GET /api/crimes/weapons-used?crime_code=510

Response:

```
[  
  {  
    "_id": "Pacific",  
    "weapons": [  
      {  
        "weapon": "UNKNOWN FIREARM",  
        "count": 3  
      },  
      {  
        "weapon": "MACE/PEPPER SPRAY",  
        "count": 1  
      },  
    ]  
  }  
]
```

NOSQL LA CRIME API

```
{
  "weapon": "SEMI-AUTOMATIC PISTOL",
  "count": 1
},
{
  "weapon": "SCREWDRIVER",
  "count": 1
},
{
  "weapon": "STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)",
  "count": 1
}
],
"area": "Pacific"
},
...
]
```

◆ Query 6: Find the top 50 most upvoted reports for a given date.

Request:

GET /api/crimes/top-upvoted?date=01/01/2023

Response:

```
[
  {
    "DR_NO": 232004216,
    "date_occurred": "01/01/2023 12:00:00 AM",
    "area": {
      "name": "Olympic"
    },
    "crime_description": "BURGLARY",
    "upvote_count": 0
  },
  {
    "DR_NO": 230413259,
    "date_occurred": "01/01/2023 12:00:00 AM",
    "area": {
```

NOSQL LA CRIME API

```
    "name": "Hollenbeck"
  },
  "crime_description": "THEFT OF IDENTITY",
  "upvote_count": 0
},
...
]
```

◆ Query 7: Find the 50 most active officers by number of upvotes given.

Request:

GET /api/officers/top-active

Response:

```
[
  {
    "name": "John Doe",
    "email": "jdoe@lapd.gov",
    "upvote_count": 1,
    "badge_number": "123456"
  },
  {
    "name": "Stefanos G",
    "email": "stefanos@email.com",
    "upvote_count": 1,
    "badge_number": "22222"
  },
  {
    "name": "22222",
    "email": "stefanos@email.com",
    "upvote_count": 1,
    "badge_number": "Stefanos G"
  }
]
```

NOSQL LA CRIME API

◆ Query 8: Find the 50 officers who have upvoted reports across the most different areas.

Request:

GET /api/officers/top-by-area

Response:

```
[
  {
    "_id": {
      "badge_number": "123456",
      "name": "John Doe",
      "email": "jdoe@lapd.gov"
    },
    "badge_number": "123456",
    "name": "John Doe",
    "email": "jdoe@lapd.gov",
    "unique_area_count": 1
  },
  ...
]
```

◆ Query 9: Find reports that have received upvotes from the same email for more than one badge number.

Request:

GET /api/upvotes/multiple-badge

Response:

```
{
  "message": "No duplicate badge upvotes found"
}
```

◆ Query 10: Find all areas where a given name has upvoted a report.

Request:

GET /api/officers/upvoted-areas?officer_name=John Doe

NOSQL LA CRIME API

Response:

```
[  
  {  
    "area": "Wilshire"  
  }  
]
```

5. Conclusion

This implementation successfully meets all project requirements, providing a high-performance and scalable NoSQL database for LAPD crime data management. The structured schema ensures efficient data organization and retrieval, while indexing and query optimization enable fast filtering by crime type, date, and location.

The upvote system allows officers to highlight important reports while maintaining data integrity by preventing duplicate votes. The system supports real-time updates, ensuring continuous data accuracy.

By leveraging MongoDB's aggregation pipelines and indexing, queries remain efficient even with large datasets. The use of Docker simplifies deployment and scalability. Overall, this project delivers a robust and well-optimized solution, with potential for further enhancements like geospatial queries and real-time analytics.