UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

LAPD Crimes NoSQL

Student name: *Panagiotis Zazos. Athina Vekraki*
sdi: 7115112300009, 7115112300003

Course: *M149 Database Management Systems*
Semester: *Fall Semester 2024*

Contents

1. Introduction

The goal of this project is to design, implement, and demonstrate a NoSQL database solution to manage "Crime Data" openly published by the Los Angeles Police Department (LAPD)
This project is developed using MongoDB to efficiently store and retrieve these crime-related data. MongoDB supports up to approximately 500 MB of data on its free tier. Therefore, special attention has been given to optimizing data ignestion by excluding non-essential fields and designing indexes to support fast retrieval.
The backened API is developed using Flask and follows the Blueprint structure. PyMongo is then used to interact with the MongoDB database.

2. LAPD Crimes Data Size

LAPDcrimes

LOGICAL DATA SIZE: 298.81MB STORAGE SIZE: 99.52MB INDEX SIZE: 61.12MB TOTAL COLLECTIONS: 6 CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
areas	21	1.26KB	62B	20KB	2	40KB	20KB
crime_descriptions	158	12.22KB	80B	24KB	2	40KB	20KB
crime_reports	990293	227.85MB	242B	70.54MB	2	36.8MB	18.4MB
officers	600	65.17KB	112B	44KB	2	56KB	28KB
upvotes	593525	70.88MB	126B	28.88MB	2	24.15MB	12.08MB
weapons	79	5.7KB	74B	20KB	2	40KB	20KB

Figure 1: Analytical Storage of DB

3. Schema Design

The database consists of multiple collections to represent **crime reports**, **reference**, **officers**, and **upvotes**.

3.1. Crime Reports Collection

The `crime_reports` collection stores crime-related data, including details about the crime, the location, and the weapon used.

Field Name	Data Type	Description
`dr_no`	Integer	Unique crime report identifier (Indexed for fast lookup)
`date_rptd`	DateTime	Date when the crime was reported
`date_occ`	DateTime	Date when the crime occurred
`time_occ`	Integer	Time of occurrence (24-hour format)
`area_code`	Integer	Police area code where crime occurred
`weapon_code`	Integer	Code representing weapon used
`crm_cd_list`	Array	List of crime codes associated with the report
`location`	String	Address or location where the crime happened

Due to the size of the dataset, several fields in the original dataset were excluded to optimize storage. These fields were omitted since they weren't going to add any value to the *required queries*.

- **Victim Details**
  - **Vict Age, Vict Sex, Vict Descent**
- **Premise details**
  - **Premid Cd, Premis Desc**
- **Latitude/Longitude**
  - **LAT, LON**

3.2. Reference Collections

Certain lookup values were moved into separate reference collections to reduce redundancy.

3.2.1. Areas Collection (`areas`).

Field Name	Data Type	Description
`area_code`	Integer	Unique identifier for police area
`area_name`	String	The name of the area

- Creating this collection ensured that area names are not stored repeatedly, as they were already saved in **crime reports**.

3.2.2. Weapons Collection (`weapons`).

Field Name	Data Type	Description
`weapon_code`	Integer	Unique identifier for a weapon
`weapon_desc`	String	The name of the weapon

- Similarly to Areas, storing weapons in a different collection avoids the redundancy of the textual description, which require much space, especially when they are being duplicated.

3.2.3. Crime Descriptions Collection (`crime_descriptions`).

Field Name	Data Type	Description
`crm_cd`	Integer	Crime code
`crm_cd_desc`	String	Description of the crime

- For the exact same reason as Weapons collection, storing description (long strings) repeatedly is not efficient and costs a lot of memory.

3.3. Officers Collection

The `officers` collection tracks police officers involved in crime reports and their upvote counts.

Field Name	Data Type	Description
`badge_number`	Integer	Unique badge identifier
`name`	String	Officer's full name
`email`	String	Officer's email
`upvotes`	Integer	Number of casted upvotes

3.4. Upvotes Collection

The `upvotes` collection records the number of upvotes casted by officers.

Field Name	Data Type	Description
`dr_no`	Integer	Upvoted crime report
`badge_number`	Integer	Officer who upvoted the crime
`officer_name`	String	Officer's name
`officer_email`	String	Officer's email

4. Database Considerations

4.1. Normalization/Denormalization

While MongoDB is optimized for denormalized storage (data that are queried together should exist in the same document), we've decided to normalize some references, such as weapons and crime descriptions, as they do not change frequently and they contain textual data that would tremendously increase the document size.

4.2. Indexing

Indexes were created depending on the queries that to be answered, ultimately to optimize the query performance:

- **dr_no** (Crime Report ID)
- **date_occ** (Crime Occurrence Date)
- **area_code** (Police ID of area)
- **crm_cd_list.code** (Crime Codes from the list of crime codes)

4.3. Data Loading

To avoid exceeding memory limits, a batch preprocessing procedure for inserts has been implemented. Chunks of 10,000 documents were inserting at a time to prevent memory overflow. Moreover, indexes were only created after inserting the documents.

5. REST API Implementation

5.1. Crime Statistics

- **/reports/count_by_crime_code**
  - Returns the number of crimes grouped by crime codes for a given date range.
- **/reports/count_by_day**
  - Returns the daily count of a specific crime code within a given date range.
- **/reports/top_crimes_by_area**
  - Retrieves the most frequently occurring crimes in different police areas on a specific date.
- **/reports/least_common_crimes**
  - Identifies the least reported crimes within a specified date range.

5.2. Officer Analysis

- **/officers/top50_most_active_officers**
  - Lists the top 50 officers with the highest number of reported crimes.
- **/officers/top50_officers_by_areas**
  - Ranks officers based on the number of unique areas where they have filed crime reports.
- **/officers/voted_areas_by_officer**
  - Retrieves the areas where a given officer has received upvotes for their crime reports.

5.3. Upvote-Based Insights

- **/reports/top_50_upvoted_reports**
  - Retrieves the top 50 most upvoted crime reports for a specific date.

- **/reports/problematic_reports**
  - Identifies reports where multiple officers have reported the same crime, potentially indicating inconsistencies. **This query does not work properly due to memory constraints.**

5.4. Weapon and Crime Analysis Across Areas

- **/reports/weapons_by_crime_across_areas**
  - Lists the weapons used in crimes across different areas, filtering by crime code.

6. Conclusion

We conclude this implementation of a MongoDB-based NoSQL solution for managing LAPD crime data with a warning that we did not implement any **UPDATE** features that were asked in the assignment, such as inserting new reports and casting upvotes. The schema was carefully designed to balance query efficiency, revolving around the specific queries tasked to us.