MongoDB

Study Points Assignment on Databases SOFT 2024

1. **Explore the data and formulate considerations about a hosting database.**

Hosting database:

The considerations for a hosting database would focus on scalability, given the global scope of the data, and flexibility, to accommodate various data types and relationships.

We've explored the dataset and identified the key collection:

Collection:

- Cities:
    - Countries
    - Emission Targets
    - Reductions

2. **Design and develop proper database structure of the requested type.**

The finale database structure illustrated with a json snippet:


```
{
  "_id": {
    "$oid": "65facf66c6ef86ec7e1e8b7f"
  },
  "city": "Brasília",
  "country": "Brazil",
  "c40": false,
  "climateRiskAssessments": [
    {
      "yearOfPublication": 2021,
      "factorsConsidered": "Assessment considers nature; Assessment considers water security; Assessment includes a high-emissions scenario (i.e., RCP 8.5)",
      "primaryAuthors": "Consultant; Relevant department within jurisdiction",
      "adaptationGoals": "Adaptation plan",
      "population": 2817068,
      "populationYear": 2022,
      "lastUpdate": "02/07/2024 04:14:16 AM"
```

```
    }
  ],
  "location": {
    "city": "(-15.794229, -47.882166)",
    "country": "(-14.235004, -51.92528)"
  },
  "reports": [
    {
      "reportingYear": 2017,
      "baselineYear": 2005,
      "baselineEmissions": 5648140,
      "percentageReductionTarget": 40,
      "targetDate": 2020,
      "totalEmissions": 7739830
    }
  ]
}
```


3. **Ingest the data into the database, include pre-processing of it, if necessary.**

We pre-processed the datasets and ingested the data into our new structured database with MongoDB.

See the repo: https://github.com/Pejomi/dbAssignment2

4. **Design and develop operations for maintenance of the database.**

The following operations have been developed for maintaining the database:

- Regular data validation and cleanup. Removing duplicate or orphaned records.
- Database backups and recovery plans. Utilizing cron jobs for backing up.
- Performance monitoring and tuning. Identifying missing indexes.
- Security audits and updates. Regularly reviewing user permissions, and applying DBMS updates.
- Documentation and change management. Implementing a database versioning tool like Flyway, and maintaining documentation.

5. **Formulate ten relevant questions for extracting information from the database, design and develop database functionality for implementing the information extraction (for the relevance consult the instructor).**

What are the top 5 cities with the highest total emissions reported in a specific year?

Which cities have set the highest percentage reduction targets for their emissions by 2030?

How do C40 cities compare to non-C40 cities in terms of average total emissions reported?

What is the progress of a specific city towards its emissions reduction target compared to its baseline emissions?

Which countries have the most cities reporting emissions data?

What is the average emissions reduction target set by cities with a baseline year before 2010?

How many cities have already met or exceeded their emissions reduction targets?

What is the trend of total emissions reported by cities over the last 5 years?

Which cities have increased their emissions from the baseline year despite having reduction targets?

Are there regional differences in emissions reduction targets?

6. **Design and implement a model for scaling the database, considering ACID and/or CAP theorem rules.**

For scaling, a schema that can handle both structured and semi-structured data is ideal. NoSQL databases e.g. MongoDB offer flexibility and scalability, suitable for varied data types and volumes.

Data can be partitioned based on geographical regions, like country, or reporting years to distribute the load evenly across servers, enhancing query performance as the database grows.

7. **Validate and test all database operations.**

The following aspects have been validated and tested:

- Inserting new documents to ensure that the data structure conforms to our schema.
- Updating documents to ensure updates correctly modify existing documents and that additions do not create duplicates.
- Retrieving documents based on different criteria, by country, city, or reportingYear, to ensure the queries return the expected documents.
- Deleting documents or fields within documents to ensure that data is correctly removed without unintended side effects.

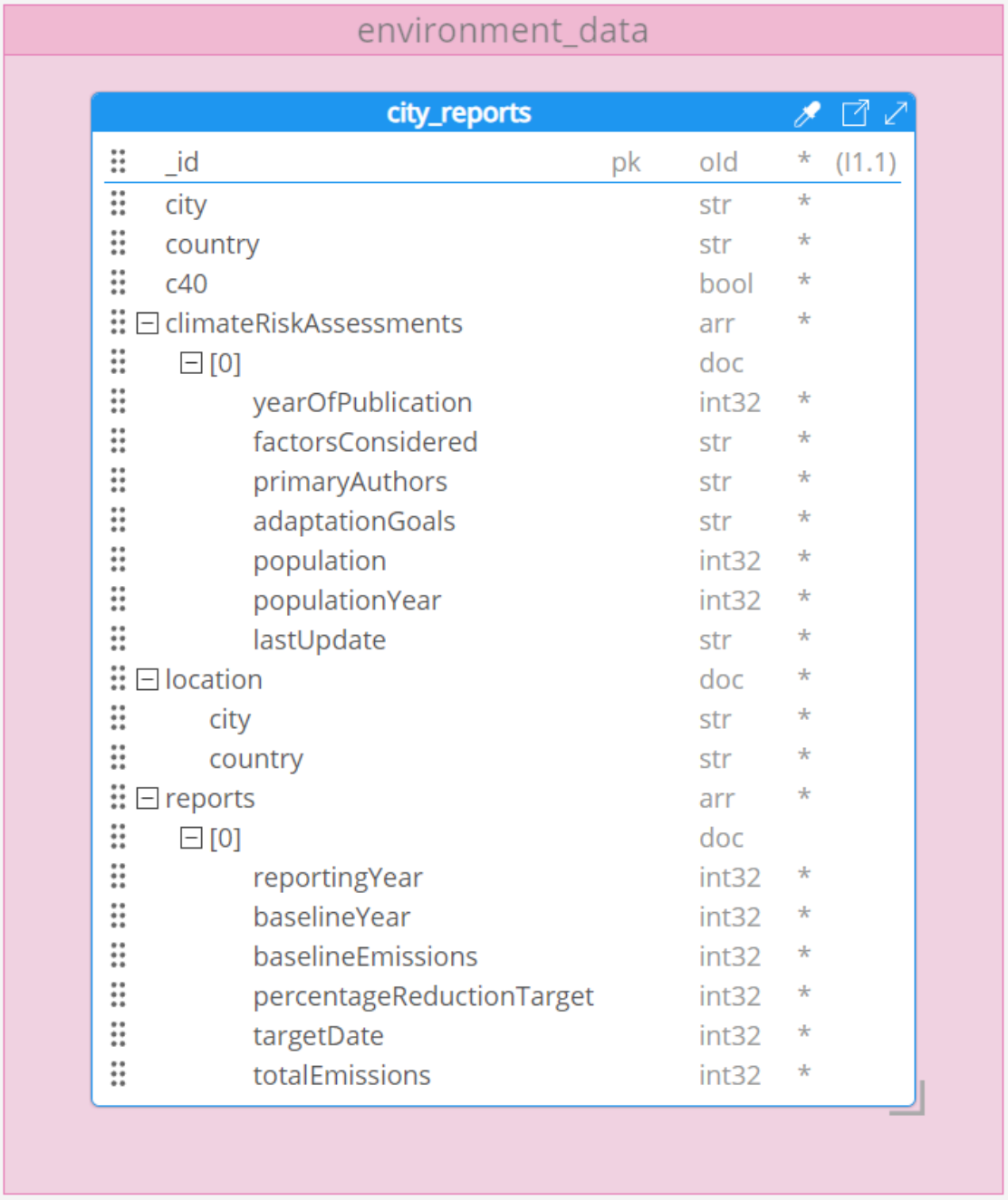
8. **Evaluate the database's performance and suggest measures for improving it.**

We considered the following measures for improving the database in the future:

- Partitioning and Sharding
- Optimize Query Statements
- Caching Frequently Accessed Data
- Scale Horizontally
- Use SSDs
- In-memory Databases

9. **Document your work, describing the product and the process. Apply graphical notation (diagrams), when it is possible.**

   We made an ER-diagram to present our work with structuring the new database using Hackolade to reverse engineer our existing schema:


environment_data
city_reports
_id pk oId * (I1.1)
city str *
country str *
c40 bool *
climateRiskAssessments arr *
[0] doc
yearOfPublication int32 *
factorsConsidered str *
primaryAuthors str *
adaptationGoals str *
population int32 *
populationYear int32 *
lastUpdate str *
location doc *
city str *
country str *
reports arr *
[0] doc
reportingYear int32 *
baselineYear int32 *
baselineEmissions int32 *
percentageReductionTarget int32 *
targetDate int32 *
totalEmissions int32 *

10. Formulate conclusions and recommendations.

We spent too much time on preprocessing the data, because it was not compatible to be ingested into the database. We tried some different strategies for handling the logic. We started with Java, but quickly realized that Python was more suitable for handling the CSV files etc.

We used some time to understand how we should construct the data for this assignment. The data structures in MongoDB are very different compared to the relational databases.

We see some advantages of using MongoDB. For example it is much easier to request a larger amount of data at once compared to relational databases where you often need to have several requests to different tables to get your actual wanted information.