CS541 – project 2

Wen-Ling Chi

Email: chi64@purdue.edu

Part 1 – Understanding the data

1-1 Is each column present in every data entry? List which columns do not appear in every data entry.

Filename	Columns
airport.json	• city • airport_province • island • elevation
city.json	• population • elevation • country_capital
country.json	• capital • province
country-other localname.json	• othername
countrypopulations.json	• capital
economy.json	• gdp • agriculture • service • industry • inflation • unemployment
ethnicgroup.json	• country_capital • ethnic_group_percentage
language.json	• country_capital • percentage
organization.json	• city • country • province • established
politics.json	• independence • wasdependent • dependent • government
population.json	• country_province • population growth • infant mortality
province.json	• population • capital • capprov

1-2 Given the file descriptions and the knowledge graph, are there any redundant columns in the files?
List which files have redundant columns. Name the columns and provide reasons why, in your opinion, those columns are redundant?

Filename	Redundant columns / Reason
airport.json	• country_name: we have country_code as a foreign key to the country table.
city.json	• country_name: we have country as a foreign key to the country table. • country_capital: if we know the country code, then we should also know the capital.
cityothername.json	• country_area: since we know its city name, we can also know its country. Through country table, we can find the area of the country. • country_capital: since we know its city name, we can also know its country. Through country table, we can find the capital of the country.
country.json	• province: we already have "capital" as the foreign key referencing the city table, and we can find out the province the capital belongs to from "province" in the city
countrypopulations.json	• country_name: we have country_code as a foreign key to the country table. • capital: if we know the country name, then we should also know the capital.
ethnicgroup.json	• country: we have country_code as a foreign key to the country table. • country_capital: we have country_code as a foreign key to the country table.
language.json	• country_capital: we have country as a foreign key to the country table. • country_area: we have country as a foreign key to the country table.
located-on.json	• country: this table doesn't connect to country table, so this column is unnecessary. Also, if we know the city, then we should also know the country. • province_area: this is unnecessary attribute for this data.
population.json	• country_name: we have country_code as a foreign key to the country table. • country_province: if we know the country_code, we can also know its province. Only "Gaza Strip" has no province, but in the country table, it also has no province.
religion.json	• country_name: we have country_code as a foreign key to the country table. • country_population: if we know the country, then we should also know the population of the country.

1-3 What logical constraints would you set for which columns in the files? For example, latitude should always be in the range of [-90, 90], and the longitude should be in [-180, 180]. Here you don't have to set constraints for every column of every file.

Filename	Constraints
airport.json	• latitude: [-90, 90] • longitude: [-180, 180]
city.json	• latitude: [-90, 90] • longitude: [-180, 180]
country.json	• encompass_percentage: [0, 100]
economy.json	• agriculture: [0, 100] • service: [0, 100] • industry: [0, 100] • unemployment: [0, 100]
ethnicgroup.json	• ethnic_group_percentage: [0, 100]
language.json	• percentage: [0, 100]
religion.json	• percentage: [0, 100]

Part 2 – Relational database schema

2-1 Write an SQL script that contains the schema for the storage and manipulation of the data provided in the JSON files.

See ./sql_schema/project2_schema.sql for the Oracle SQL schema script.

(Use Oracle SQL Developer to run sql file**)**

2-2 Compile the entity-relationship diagram of your database.

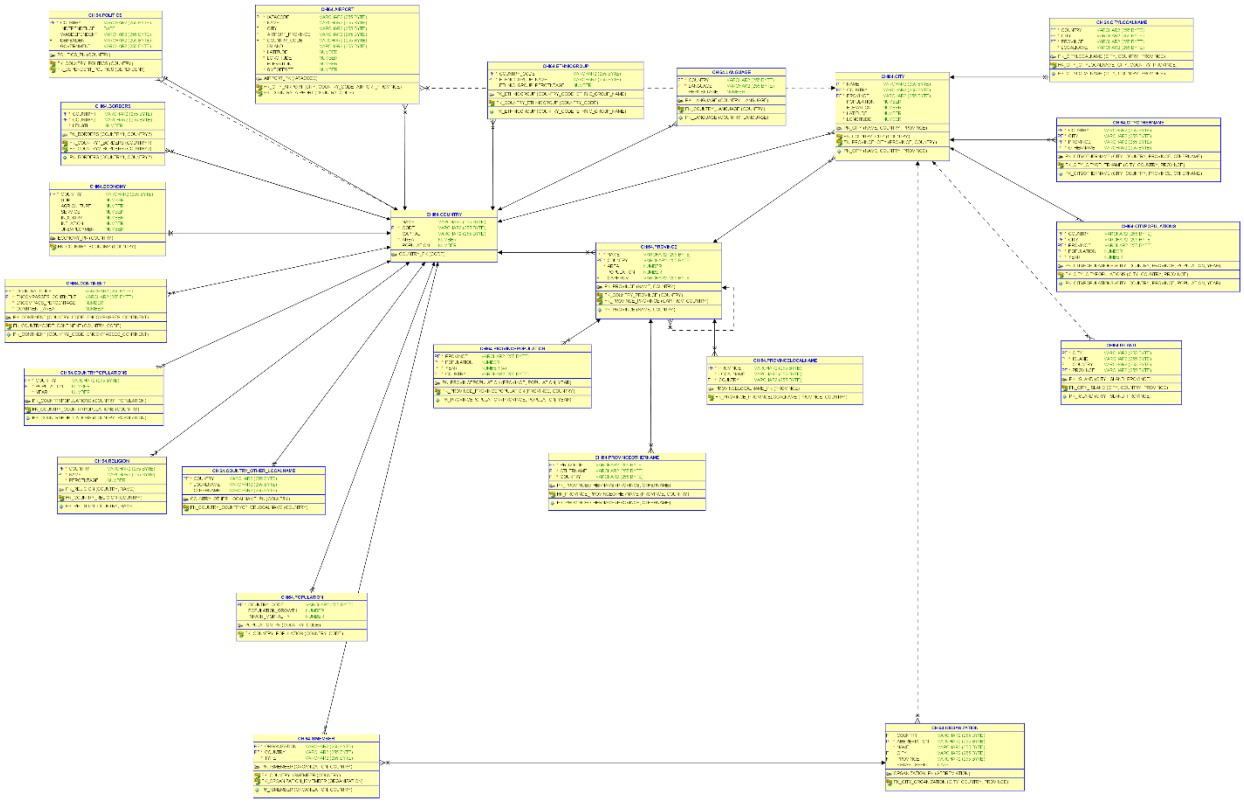

Fig. 1 - The entity-relationship diagram of my database

Part 3 – Inserting the data to Oracle DBMS

Environment

This part will use python3 and Oracle SQL Developer.

Execution Steps

1. Execute python file to create insert values for Oracle database. The output will be in ./sql_schema/project2_data.sql
   See ./sql_schema/createInsertValue_sql.py

   Executive Command:
   python sql_schema/createInsertValue_sql.py

2. Execute sql file to insert values into Oracle SQL database.
   See ./sql_schema/project2_data.sql
   (Use Oracle SQL Developer to run sql file)
   (It is possible to take a long to insert values into database)

Part 4 - Using MongoDB

Environment

This part will use python3 and MongoDB(3.6).
The reason for using MongoDB (3.6) is that we need to use a tool - mongoimport.

After downloading MongoDB(3.6), put ~/MongoDB/Server/3.6/bin into environment variables.

Execution Steps

1. Create MongoDB schema
   Execute python file to create MongoDB schema. It will generate 23 collections in MongoDB.
   See ./mongodb_schema/mongodb_schema.py

   Executive Command:
   python mongodb_schema/mongodb_schema.py

2. Create json files for the data for MongoDB
   Execute python file to create json files for inserting values in MongoDB. The output json files will be in ./mongodb_data directory.
   See ./mongodb_schema/insertValue_mongodb.py

   Executive Command:
   python mongodb_schema/insertValue_mongodb.py

3. Insert data into MongoDB
   Click bat file to insert data into MongoDB.
   See ./mongodb_schema/import_data_mongodb.bat

   Executive command in bat file (Samples)

```
mongoimport --db project2 --collection ethnicgroup .\mongodb_data\ethnicgroup.json --jsonArray
mongoimport --db project2 --collection language .\mongodb_data\language.json --jsonArray
mongoimport --db project2 --collection politics .\mongodb_data\politics.json --jsonArray
mongoimport --db project2 --collection population .\mongodb_data\population.json --jsonArray
mongoimport --db project2 --collection religion .\mongodb_data\religion.json --jsonArray
mongoimport --db project2 --collection borders .\mongodb_data\borders.json --jsonArray
mongoimport --db project2 --collection province .\mongodb_data\province.json --jsonArray
mongoimport --db project2 --collection city .\mongodb_data\city.json --jsonArray
```

Fig. 2 – samples in import_data_mongodb.bat

Part 5 – Queries

Environment

This part will use python3 and Oracle SQL Developer.

5-1

Implement the following queries in both Oracle and MongoDB. For each of the following queries, capture the execution time for both DBMS environments (question 5-2).

a

Select the name of the country that shares the longest border with one or more other countries.

• Oracle: query/oracle_query/ 5-1-a.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
python query/mongodb_query/ mongodb_query.py 1

b

For each country, list the city that has the maximum average population.

• Oracle: query/oracle_query/ 5-1-b.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
python query/mongodb_query/ mongodb_query.py 2

c

List the following information of the countries where agriculture contributes the most to their economy. Name of the country, its GDP, percentage contribution from agriculture, and inflation.

• Oracle: query/oracle_query/ 5-1-c.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
python query/mongodb_query/ mongodb_query.py 3

d

List the following information of the countries in the descending order of their ethnic diversity. Name of the country, number of ethnic groups, and the percentage of the major ethnicity. Ethnic diversity of a country increases as the number of ethnic groups that live in the country increases.

• Oracle: query/oracle_query/ 5-1-d.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
```
python query/mongodb_query/ mongodb_query.py 4
```

e

Find the number of countries with the number of ethnic groups equal to the number of languages used.

• Oracle: query/oracle_query/ 5-1-e.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
```
python query/mongodb_query/ mongodb_query.py 5
```

f

The GDP and IMR (infant mortality rate) together define the development of a country. Find the top 10 countries with the highest GDP (in decreasing order) and their corresponding IMR value.

• Oracle: query/oracle_query/ 5-1-f.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
```
python query/mongodb_query/ mongodb_query.py 6
```

g

Find the following information of the country with the highest religious freedom. Name of the country, the religions that are practiced in the country. Note that the larger the number of religions practiced in a country, the higher its religious freedom is.

• Oracle: query/oracle_query/ 5-1-g.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
```
python query/mongodb_query/ mongodb_query.py 7
```

h

What is the proportion of commonwealth countries in the top 100 countries in terms of GDP.

• Oracle: query/oracle_query/ 5-1-h.sql
• MongoDB: query/mongodb_query/mongodb_query.py
Executive Command:
```
python query/mongodb_query/ mongodb_query.py 8
```

i

Find the country with the largest population density in each continent. List the name of the country and its population density against the name of the continent. The population density of a country is the ratio between its total population and area.

- Oracle: query/oracle_query/ 5-1-i.sql
- MongoDB: query/mongodb_query/mongodb_query.py
  Executive Command:

```
python query/mongodb_query/ mongodb_query.py 9
```

j

Find the names of countries and their capitals where the capital has more than one airport.

- Oracle: query/oracle_query/ 5-1-j.sql
- MongoDB: query/mongodb_query/mongodb_query.py
  Executive Command:

```
python query/mongodb_query/ mongodb_query.py 10
```

5-2

Create a table to summarize the execution time in the two DBMS for each of the queries.

Query Number	Execution time	
	Oracle (ms)	MongoDB (ms)
5-1-a	259	97
5-1-b	650	416
5-1-c	99	45
5-1-d	316	81
5-1-e	109	359
5-1-f	95	42
5-1-g	54	35
5-1-h	71	509
5-1-i	114	106
5-1-j	98	188

5-3

Comment on the execution times you have obtained. Give educated reasons why one DBMS has performed better than the other?

In the above form, most of MongoDB's execution times are better than Oracle because MongoDB has less constraints. Thus, compare to Oracle SQL, we need less checks when we execute the query in MongoDB.

For the few cases, MongoDB's execution times are worse than Oracle. From my observation, MongoDB performs worse than Oracle SQL if we need to use multiple collections/tables. It is possible that the Oracle SQL Database will cache and create automatically indexing internally.

5-4

Suggest one performance improvement technique that you can use on each DBMS to improve the performance. Implement your technique on each DBMS and report your results in a table similar to questions 5-2.

- Oracle

Use the "CREATE INMEMORY JOIN GROUP" statement to create a join group, which is an object that specifies frequently joined columns from the same table or different tables. When you create a join group, Oracle Database stores special metadata for the columns in the global dictionary, which enables the database to optimize join queries for the columns.

Because code in country table and country in city table are frequently used by the queries, so I use the above method to create a join group.

See `query/oracle_query/optimized.sql`

- MongoDB

Create Indexes to Support Queries. If a query searches multiple fields, create a compound index. Scanning an index is much faster than scanning a collection.

See the function - optimizedDB(db) in `query/mongodb_query/mongodb_query.py`

- Execution time

Query Number	Execution time	
	Oracle (ms)	MongoDB (ms)
5-1-a	181	58
5-1-b	101	61
5-1-c	91	36
5-1-d	83	79
5-1-e	72	69
5-1-f	61	35
5-1-g	38	28
5-1-h	54	32
5-1-i	91	70
5-1-j	39	55

5-5

Comment on the execution times you have obtained after implementing your performance improvement technique. How much improvement do you observe compared to your results in 5-2?

- Oracle

All queries reduce their execution time. Because almost every query uses country table - code or city table - country. The higher the frequency they use, the more the execution time is reduced.

- MongoDB

All queries reduce their execution time. If the query needs to scan different collections, the execution time will be reduced even more. Because I store the fields in the index, scanning the index can reduce the execution time.