Database Systems – CS54100 – Project 2

Submission by – Rishabh Chaddha
Email – rchaddha@purdue.edu
PUID - 0033750099

Following files and folders are attached

Name	Description
dataset (folder)	Contains raw json data files
Insert_Scripts (folder)	Constains Insert scripts generated by parser
Project2SchemaDesign (folder)	Created on saving ER diagram
Project2SchemaDesign.dmd	ER diagram file
DDLScript.sql	CREATE statements and constraints (PRIMARY KEY, FOREIGN KEY and CHECK constraints)
Disable Foreign Constraints.sql	Script to disable foreign keys
Enable Foreign Constraints.sql	Script to enable foreign key constraints
MongoParser.py	Parser for Mongo DB
Parser.py	Parser for generating insert statements
NULLValChecker.ipynb	Jupyter notebook for checking NULL values in RAW json files

Additionally there are 2 separate folders for parsers. So, parsers are present outside (in the main folder), and in separate folders (OracleSQLParser and MongoDBParser) too. Meaning there are 2 copies of the same parser in the submission

ANSWER 1

1-1) All columns values are not present in all data entries. The following is the list of columns having empty values for some data entries.

This is derived through a python script in a Jupyter notebook (NULLValChecker.ipynb) that runs on these files and checks the empty values. The python script is attached with the submission.

Columns with empty values

JSON Name	Column Name
airport.json	airport_province City Island elevation
city.json	population country_capital elevation
country-other-localname.json	othername
country.json	province capital
countrypopulations.json	capital
economy.json	agriculture inflation gdp industry unemployment service
ethnicgroup.json	ethnic_group_percentage country_capital
language.json	country_capital percentage
organization.json	country province city established
politics.json	wasdependent independence dependent government
population.json	infant_mortality country_province population_growth
province.json	population capprov capital

1-2) Yes, there are some redundant columns. The following are the redundant columns along with the reason.

Redundant Column List

JSON File	Redundant Columns	Reason
airport.json	Country_name	This value can be derived from the country.json data.
City.json	Country_name	This value can be derived from the country.json data.
	Country_capital	This value can be derived from the country.json data.
	Population	This value can be derived from the citypopulations.json data.
Cityothername.json	Country_area	This value can be derived from the country.json data.
	Country_capital	This value can be derived from the country.json data.
Country.json	Population	This value can be derived from the countrypopulations.json data.
Country_populations.json	Country_name	This value can be derived from the country.json data.
Ethnicgroup.json	Country (name, not code)	This value can be derived from the country.json data.
	Country_capital	This value can be derived from the country.json data.
Language.json	Country_capital	This value can be derived from the country.json data.
	Country_area	This value can be derived from the country.json data.
Located-on.json	Province_area	This value can be derived from the province.json data.
Population.json	Country_name	This value can be derived from the country.json data.
	Country_province	This value can be derived from the country.json data.
Province.json	Population	This value can be derived from the provincepopulations.json data.
Religion.json	Country_name, country_population	These value can be derived from the country.json data.

The abovementioned fields are redundant as they are already present in a main master json file and need not be repeated in every JSON file.

1-3) Following are the constraints on columns (This is true for whichever JSON file these columns appear i n)

Constraints

Column	Logical Constraint
IATA Code	Uppercase
Latitude	-90 to +90
Longitude	-180 to +180
Gmt offset	-12 to + 14 No decimal
Country Code	Only Uppercase
Length (border or country perimeter)	Only positive values
All area fields	Only positive values
year	Only positive values
Population	Only Positive and no decimal values
GDP	Only Positive
Agriculture	Range in 0-100
Service	Range in 0-100
Industry	Range in 0-100
Ethnic group percentage	Range in 0-100
Any other percentage field	0-100

Please note – The applied check constraints are present in the attached DDLScript.sql file

ANSWER 2

2-1) Please find the SQL script named DDLScript.sql attached with the submission. This script contains the DDL commands for schema creation. The script has CREATE TABLE, PRIMARY Key constraints and CHECK constraints together as a group and foreign key constraints as a group in the end.

Table	Primary Key	Foreign Key
Airport	Iata code	Country code (for country) Country code, city, province (for city)
Borders	Country1, country2	Country1 (for country) Country2 (for country)
City,localname,othername	Name, country, province	Countrycode,Province(for province)
citypopulations	City,country,province,year	City, province, country (for city)
Country, country other name, country local name	Country code	Code, capital, province (for city)
Countrycontinent	Continent Name, country	Continent name(for continent) Country (for country)
Continent	Continent Name	No foreign key
countrypopulations	Countrycode,year	Code(for country)
Economy	countrycode	Countrycode(for country)
EthnicGroup	Countrycode,ethnic group name	Countrycode(for country)

ismember	Organization,country	Country(for country)
		Organization(for organization)

Language	Countrycode, language	Country(for country)
Located on (not very sure about the foreign key)	Country, city, province, island	Country,city,province(for city)
Organization	abbreviation	Countrycode, city, province(for city)

Politics	Countrycode	Wasdependent(for country)
		Dependent(for country)

population	Countrycode	Countrycode(for country)
Province, provincelocalname, provinceothername	Country,province	Countrycode (for country)
Province capital	Country,province	Capprov,countrycode(for province) Capital,province,countrycode(for city)
Province population	Province,countrycode,year	Province,countrycode,year(for province)
Religion	Country, name (religion name)	Country(for country)

2-2) A .dmd file named Project2SchemaDesign.dmd and a schema folder named Project2SchemaDesign are attached with the submission. These depict the ER diagram.

Answer 3

3-1) The parser which is implemented is named as **Parser.py**

This parser takes 4 command line arguments –

1) JSON file name from which data needs to be picked up
2) SQL Table Name in which data needs to be inserted
3) Column names of SQL table (comma separated)
4) Fields names in JSON corresponding to the SQL table (comma separated)

On execution, the file will create a <table_name>.sql (table name being the name of the table provided for insert). This file will contain the required insert statements.

Following points are worth noting –

1. The directory in which this parser will be present should have 2 folders –
   i) A folder named 'dataset' which will contain the JSON files
   ii) A folder named 'Insert_Scripts' which will store the .sql files output by the parser.
2. To run the parser,
   i) keep json files in the dataset folder and create a folder named Insert_scripts.
   ii) Run the python file in command line in the following way
   python Parser.py json_name table_name table_column_names JSON_key_names

As an example, to generate insert scripts for religion table, the following should be the command in the command line

```
python Parser.py religion.json religion country_code,name,percentage country,name,percentage
```

here religion.json is the file containing raw religion data
religion is the table name
country_code,name,percentage are comma separated values for columns of the religion table
country,name,percentage are comma separated json keys which have data corresponding to table columns.

Please note – the order of table columns and json keys should be same

The output file from the parser will give insert statements in the following form –

```
INSERT INTO religion (country_code,name,percentage) VALUES
('CGO','Muslim','2.1');
```

(INSERT INTO <TABLE-NAME> (<TABLE_COLUMNS>) VALUES (<Corresponding values from JSON>);)

After compiling the SQL file, user has to do a commit himself (commit is not present in the output file)

Working of the parser – Parser contains one function which extracts values for the json keys provided in the command line and prints these values in an output .sql file. Basically, it is printing the insert statements in the file and the changing (dynamic) part of the statement (table name, json values) are being printed at appropriate parts of insert statements.

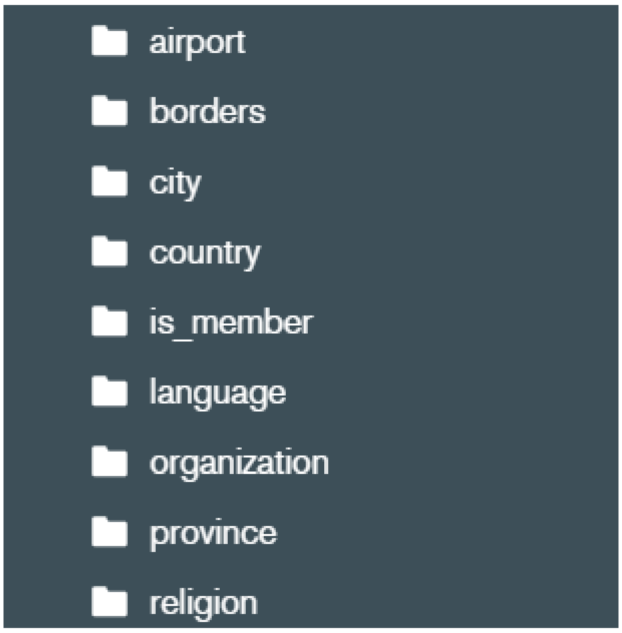
ANSWER 4

4-1) In mongo DB, the database need not be normalized. Taking advantage of this thing, there are a few changes done in the model from those done from SQL.

The following is the model design –

1. Redundant fields are removed from the tables as it was done for SQL.
2. Following references are added –
   - i) Airport data is referenced from city collection so that we get to know which airports are in which cities.
   - ii) Data related to ethnic group, language, membership in organization and religion is referenced from country collection.
   - iii) Organizational membership data is referenced from organization collection.
3. Following embeddings are added –
   - i) City local name, city other name, city population and island (located-on) data is embedded to the city collection.
   - ii) Country other name and country local name data is embedded to country collection.
   - iii) Politics, population, economy and continent data is embedded to country collection.
   - iv) Province other, local names and population data is embedded in the province table.

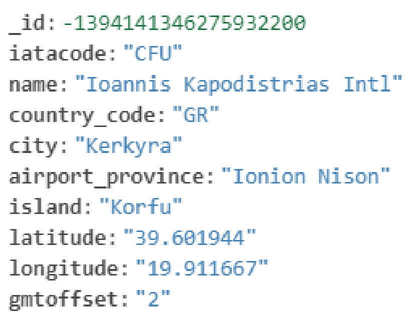
Following are the resultant collections



airport
borders
city
country
is_member
language
organization
province
religion

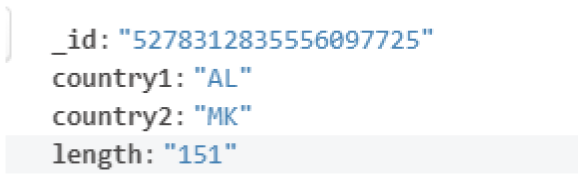
Following are sample BSON structures of these collections –

1. Airport –



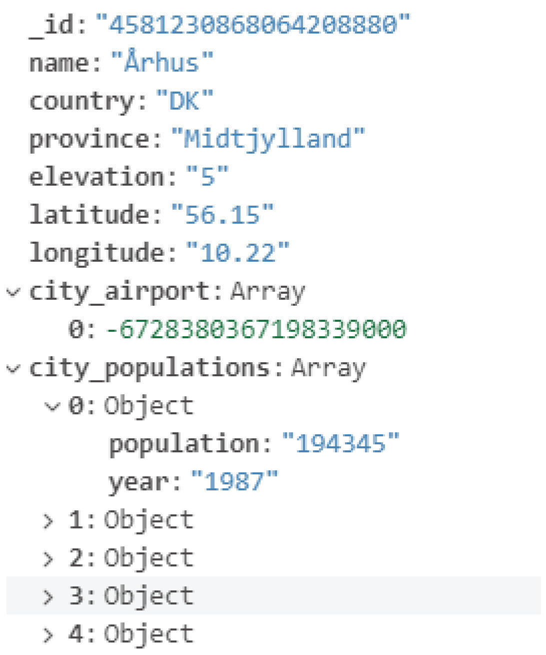
```
_id: -1394141346275932200
iatacode: "CFU"
name: "Ioannis Kapodistrias Intl"
country_code: "GR"
city: "Kerkyra"
airport_province: "Ionion Nison"
island: "Korfu"
latitude: "39.601944"
longitude: "19.911667"
gmtoffset: "2"
```

2. Borders



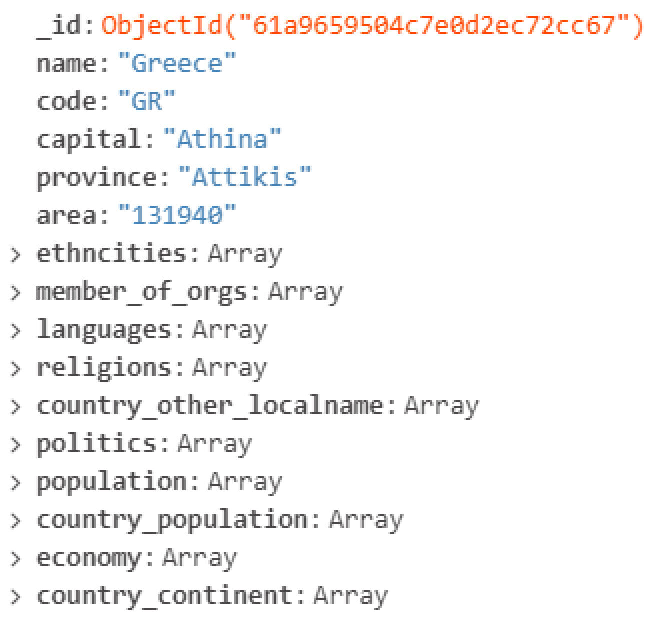
```
_id: "5278312835556097725"
country1: "AL"
country2: "MK"
length: "151"
```

3. City



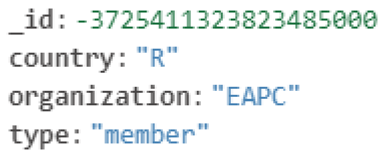
```
  _id: "4581230868064208880"
  name: "Århus"
  country: "DK"
  province: "Midtjylland"
  elevation: "5"
  latitude: "56.15"
  longitude: "10.22"
⌄ city_airport: Array
     0: -6728380367198339000
⌄ city_populations: Array
   ⌄ 0: Object
        population: "194345"
        year: "1987"
   › 1: Object
   › 2: Object
   › 3: Object
   › 4: Object
```

4. Country



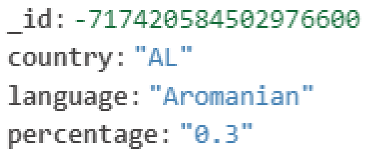
```
_id: ObjectId("61a9659504c7e0d2ec72cc67")
name: "Greece"
code: "GR"
capital: "Athina"
province: "Attikis"
area: "131940"
> ethncities: Array
> member_of_orgs: Array
> languages: Array
> religions: Array
> country_other_localname: Array
> politics: Array
> population: Array
> country_population: Array
> economy: Array
> country_continent: Array
```

5. Is_member



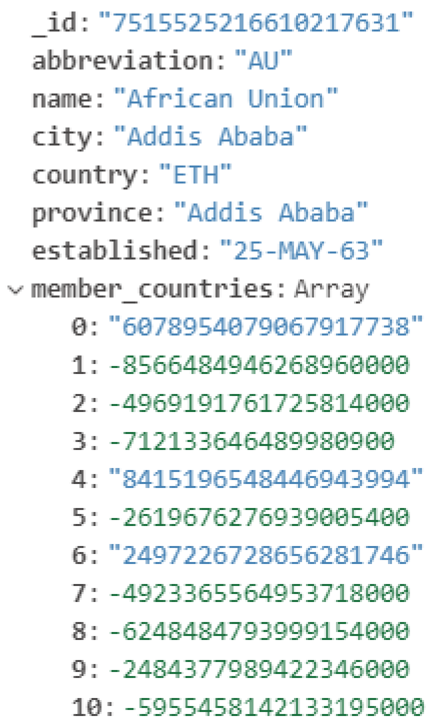
```
_id: -3725411323823485000
country: "R"
organization: "EAPC"
type: "member"
```

6. Language



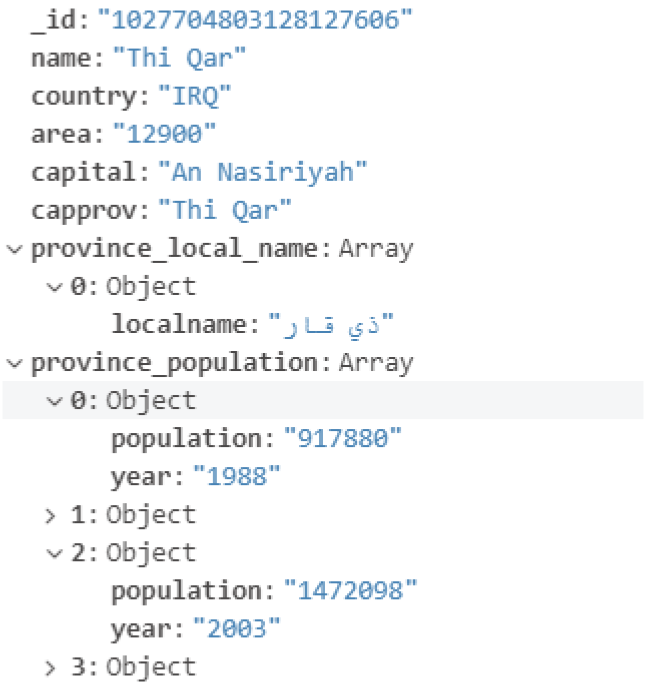
```
_id: -7174205845029766000
country: "AL"
language: "Aromanian"
percentage: "0.3"
```

7. Organization



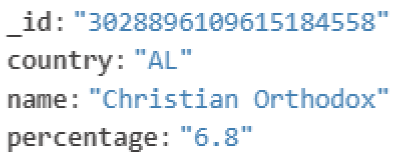
```
_id: "7515525216610217631"
abbreviation: "AU"
name: "African Union"
city: "Addis Ababa"
country: "ETH"
province: "Addis Ababa"
established: "25-MAY-63"
member_countries: Array
   0: "6078954079067917738"
   1: -8566484946268960000
   2: -4969191761725814000
   3: -712133646489980900
   4: "8415196548446943994"
   5: -2619676276939005400
   6: "2497226728656281746"
   7: -4923365564953718000
   8: -6248484793999154000
   9: -2484377989422346000
   10: -5955458142133195000
```

8. Province –



```
_id: "1027704803128127606"
name: "Thi Qar"
country: "IRQ"
area: "12900"
capital: "An Nasiriyah"
capprov: "Thi Qar"
province_local_name: Array
   0: Object
      localname: "ذي قار"
province_population: Array
   0: Object
      population: "917880"
      year: "1988"
   1: Object
   2: Object
      population: "1472098"
      year: "2003"
   3: Object
```

9. Religion –



```
_id: "3028896109615184558"
country: "AL"
name: "Christian Orthodox"
percentage: "6.8"
```

4-2) Parser for Mongo DB –
Name of the Parser file is **MongoParser.py** –

This is a bit more complicated than SQL parser.
It is a python file and it also takes command line arguments. And it gives JSON documents that can be directly uploaded to MONGO DB database.
It proceeds in the following steps –

1. Remove redundant columns.
2. Assign _id (unique id value) – This is provided through the HASH value of provided primary keys
3. Form References
4. Form Embeddings

Now, in this case, the first command line argument always instructs which step to carry out.

Following are the examples of how to execute –
Step 1 –
Write the following in the command line –
python MongoParser.py remove_redundant_keys city.json city name,country,province,elevation,latitude,longitude

For the first step, the input json is picked up from the dataset folder

(this is a single line with space separated values)
MongoParser.py is the file name of the parser file
remove_redundant_keys is the instruction to remove redundant keys
city.json is the raw input json file
city is the name of output json to be generated
name,country,province,elevation,latitude,longitude – these are the columns to be retained in the output JSON

The output json is saved to Collection_JSONS folder (This should be already created before running the parser)

Step 2 –
Write the following in the command line –
python MongoParser.py assign_id city name,country,province
assign_id is the instruction to the parser to assign _id value to the city collection
city collection is the output of step 1 (and is already present in Collection_JSONS folder)
name,country,province – these specify unique valued column combination of JSON, the value for which needs to be provided to calculate a unique hash value for each record

Step 3 –
Write the following in the command line for creating references –
python MongoParser.py create_reference country.json ethnic_group.json code country_code ethnicities
create_reference instructs parser to start creating a reference
country.json is the name of the parent file

ehnic_group.json is the name of the child file whose references will be created in the parent

code – this is the primary key value or the unique key of the parent table which needs to be compared with analogous key of child table to create reference

country_code – this is type of foreign key. This is the key of the child table which needs to be compared with corresponding unique key of parent table to create reference

ethnicities – this will be the name of the key which will contain object ids of referenced fields

Step 4 –

Write the following in command line to embed jsons –

python MongoParser.py embed_json city.json city_local_name.json name,country,province city,country,province localname

embed_json is the instruction to follow the embed logic

city.json is the parent json file

city_local_name.json is the child json file whose values will be embedded

name,country,province – these are unique values for parent json

city,country,province – these are values in child json corresponding to parent values

The final outputs will be stored to the Collection_JSONS folder

Answer 5

5-1) The queries are present in the attached ‘SQL Queries.sql’ file (along with part numbers in comments).

5-2) Following are the execution times

Query	Oracle Time
a	0.054
b	0.065
c	0.054
d	0.201
e	0.048
f	0.037
g	0.042
h	0.046
i	0.055
j	0.044

5-3) Mongo DB performs better because many join operations are not required in mongo DB. Mongo DB has lesser joins because data can be embedded and referenced, because of which many tables which were actually different in Oracle database are combined to a single table in Mongo DB.

5-4) On both DBMS, indexes can be created on the database tables. As an example, an index can be created on columns city, province and country in all tables. Another index can be created on GDP of economy table as gdp is a field which is frequently referred to.

The execution time for SQL db after creation of indexes is as follows –

Query	Oracle Time
a	0.050
b	0.064
c	0.054
d	0.178
e	0.048
f	0.032 – becomes better
g	0.042
h	0.037 – becomes better
i	0.055
j	0.039 – becomes better

5-5) Execution time becomes a bit better (not too better) on oracle DB after creation of indexes. In mongo DB, the execution time remains the same.