COMP1204 Coursework 2 Report

Alessandro Nerla — ID: 31243118 — an1g19@soton.ac.uk

May 20, 2020

1 The Relational Model

Exercise 1

The dataset can be represented by the following relation:

Dataset(dateRep : date, day : int, month : int, year : int, cases : int, deaths : int, countriesAndTerritories : text, geoId : text, countryterritoryCode : text, popData2018 : int, continentExp : text)

Exercise 2

In the relation *Dataset* there are the following functional dependencies:

- $\{dateRep\} \rightarrow \{day,\ month,\ year\}$
- $\{day,\ month,\ year\} \rightarrow \{dateRep\}$
- $\{countriesAndTerritories\} \rightarrow \{geoId,\ countryterritoryCode,\ popData2018,\ continentExp\}$
- $\{geoId\} \rightarrow \{countriesAndTerritories,\ countryterritoryCode,\ popData2018,\ continentExp\}$
- $\{countryterritoryCode\} \rightarrow \{popData2018\}$
- $\{dateRep,\ countriesAndTerritories\} \rightarrow \{cases,\ deaths\}$
- $\{dateRep,\ geoId\} \rightarrow \{cases,\ deaths\}$
- $\{day,\ month,\ year,\ countriesAndTerritories\} \rightarrow \{cases,\ deaths\}$
- $\{day,\ month,\ year,\ geoId\} \rightarrow \{cases,\ deaths\}$

It can be noted that *countriesAndTerritories* is the name of the country, *geoId* seems to be the ISO 3166-1 Alpha-2 country code and *countryterritoryCode* seems to be the ISO 3166-1 Alpha-3 country code (except in the case of the Diamond Princess cruise ship). This would mean that *countryterritoryCode* uniquely determines *geoId* but, with the current data, this is not the case as, for some countries, the ISO 3166-1 Alpha-3 country code is not given. This can be seen only in some overseas territories (e.g. Falkland Islands) and disputed territories (Western Sahara). Due to it not being always provided, it will not be considered a determinant for *geoId* but, if the dataset were to be updated to resolve this issue, the database schema may need to be modified.
Based on the current dataset we can also conclude that $popData2018$ is a determinant for *continentExp* but, based on outside domain considerations, it has not been included as there could possibly be 2 countries from different continents with the same population which may be included in the dataset in the future.

Exercise 3

In the relation *Dataset* the candidate keys are the following:

- $\{dateRep,\ countriesAndTerritories\}$
- $\{dateRep,\ geoId\}$
- $\{day,\ month,\ year,\ countriesAndTerritories\}$
- $\{day,\ month,\ year,\ geoId\}$

Exercise 4

A suitable primary key would be: $(dateRep,\ geoId)$.
I believe it's the optimal primary key as, while *dateRep* conveys the same information as *day*, *month* and *year*, *dateRep* is one single attribute.
I also prefer to use *geoId* compared to *countriesAndTerritories* as its usual values are shorter, thus simplyfing queries and reducing possible errors due to misspellings.

2 Normalisation

Exercise 5

There is the following partial dependency:

- $\{geoId\} \rightarrow \{countryterritoryCode,\ popData2018,\ continentExp\}$

It has to be noted that while the following dependencies are not partial dependencies, as *day*, *month*, *year* and *countriesAndTerritories* are prime attributes, a case can be made for their decomposition into other relations:

- $\{geoId\} \rightarrow \{countriesAndTerritories\}$
- $\{dateRep\} \rightarrow \{day,\ month,\ year\}$

To eliminate the partial dependency we can create 1 new relation:

- GeoData(**geoId** : text, countryterritoryCode : text, popData2018 : bigInt, continentExp : text)

Exercise 6

To put the schema in 2^{nd} Normal Form we need to:

1. Find partial dependencies
2. Eliminate them by moving the fields that partially depend on the key to a new relation
3. Remove those fields from the original relation, instead referencing them by the key of the new relation

Thus, the schema in 2^{nd} Normal Form is as follows:

- CaseData(**dateRep** : date, day : int, month : int, year : int, ***geoId*** : text, countriesAndTerritories : text, cases : int, deaths : int)
- GeoData(**geoId** : text, countryterritoryCode : text, popData2018 : bigInt, continentExp : text)

Exercise 7

There is the following transitive dependency:

- $\{countryterritoryCode\} \rightarrow \{popData2018\}$

To eliminate the transitive dependency we can create 1 new relation:

- CountryPop(**countryterritoryCode** : text, popData2018 : bigInt)

It has to noted that the stated dependency is a transitive dependency only because countryterritoryCode has NULL values which prevent it from being a determinant for $countriesAndTerritories$ and $geoId$ (see Exercise 2).

Exercise 8

To put the schema in 3^{rd} Normal Form we need to:

1. Find transitive dependencies ($A \rightarrow B$ & $B \rightarrow C$ BUT NOT $B \rightarrow A$)
2. Eliminate them by moving the fields that transitively depend on the key to a new relation
3. Remove those fields from the original relation, instead referencing them by the key of the new relation

Thus, the schema in 3^{rd} Normal Form is as follows:

- CaseData(**dateRep** : date, day : int, month : int, year : int, ***geoId*** : text, countriesAndTerritories : text, cases : int, deaths : int)
- GeoData(**geoId** : text, *countryterritoryCode* : text, continentExp : text)
- CountryPop(**countryterritoryCode** : text, popData2018 : bigInt)

Exercise 9

The relation is not in Boyce-Codd Normal Form as we still have the following non-trivial functional dependencies, where the determinant is not a superkey of the table:

- $\{geoId\} \rightarrow \{countriesAndTerritories\}$
- $\{countriesAndTerritories\} \rightarrow \{geoId\}$
- $\{day, month, year\} \rightarrow \{dateRep\}$
- $\{dateRep\} \rightarrow \{day, month, year\}$

Further considerations

It has been previously stated that *countryterritoryCode* is the ISO-3166 Alpha-3 code, which is equivalent to *geoId*, which is the ISO-3166 Alpha-2 code. In the current dataset the Alpha-3 code (countryterritoryCode) has been omitted but, as the dataset may be updated in the future to include this code for the couple of records where it is missing, it would be opportune to consider this fact. It would also be opportune to consider separating *dateRep* and *day*, *month*, *year* to normalize the database to BCNF.
I also have to add a date field formatted as ”YYYY-MM-DD” to be able to easily and correctly order by date, as the ”DD/MM/YYYY” format of dateRep is not correctly ordered by SQLite queries. Thus, to resolve the date issue and to prevent the need to reformat the schema upon addition of new data, I will implement the following database schema:

- Dates(**dateFormatted** : date, dateRep : date, day : int, month : int, year : int)
- CaseData(***dateFormatted*** : date, ***geoId*** : text, cases : int, deaths : int)
- GeoData(**geoId** : text, popData2018 : bigInt, continentExp : text)
- CountryID(***geoId*** : text, countriesAndTerritories : text, countryterritoryCode : text)

3 Modelling

Exercise 10

dataset.sql has been included in the *.tar.gz* archive

Exercise 11

dataset2.sql and *ex11.sql* have been included in the *.tar.gz* archive

Exercise 12

dataset3.sql and *ex12.sql* have been included in the *.tar.gz* archive

Exercise 13

dataset.sql, *ex11.sql* and *ex12.sql* have been tested

4 Querying

Exercise 14

```sql
SELECT
   SUM(cases) AS "Total Cases",
   SUM(deaths) AS "Total Deaths"
FROM
   CaseData;
```

Exercise 15

```sql
SELECT
   dateFormatted AS "Date",
   cases AS "New Cases in UK"
FROM
   CaseData
WHERE
   geoId = "UK"
ORDER BY
   "Date" ASC;
```

Exercise 16

```sql
SELECT
   continentExp AS "Continent",
   dateFormatted AS "Date",
   SUM(cases) AS "NEW Cases",
   SUM(deaths) AS "NEW Deaths"
FROM
   CaseData
   INNER JOIN
      GeoData
      ON CaseData.geoId = GeoData.geoId
GROUP BY
   "Continent",
   "Date"
ORDER BY
   "Date" ASC;
```

Exercise 17

```sql
SELECT
   countriesAndTerritories AS "Country",
   CAST(SUM(cases) AS FLOAT) / popData2018*100.0
    AS "\ % Cases of Population",
   CAST(SUM(deaths) AS FLOAT) / popData2018*100.0
    AS "\ % Deaths of Population"
FROM
   CaseData
   INNER JOIN
      GeoData
      ON CaseData.geoId = GeoData.geoId
   INNER JOIN
      CountryID
      ON GeoData.geoId = CountryID.geoId
GROUP BY
   "Country";
```

Exercise 18

```sql
SELECT
   countriesAndTerritories AS "Country",
   CAST(SUM(deaths) AS FLOAT) / CAST(SUM(cases) AS FLOAT)*100.0
    AS "\ % Deaths"
FROM
   CaseData
   INNER JOIN
      GeoData
      ON CaseData.geoId = GeoData.geoId
   INNER JOIN
      CountryID
      ON GeoData.geoId = CountryID.geoId
GROUP BY
   "Country"
ORDER BY
   " % Deaths" DESC LIMIT 10;
```

Exercise 19

```sql
SELECT
   dateFormatted AS "Date",
   SUM(cases) OVER (ORDER BY dateFormatted)
    AS "Cumulative UK Cases",
   SUM(deaths) OVER (ORDER BY dateFormatted)
    AS "Cumulative UK Deaths"
FROM
   CaseData
WHERE
   geoId = "UK"
ORDER BY
   "Date";
```

5 Extension

Exercise 20

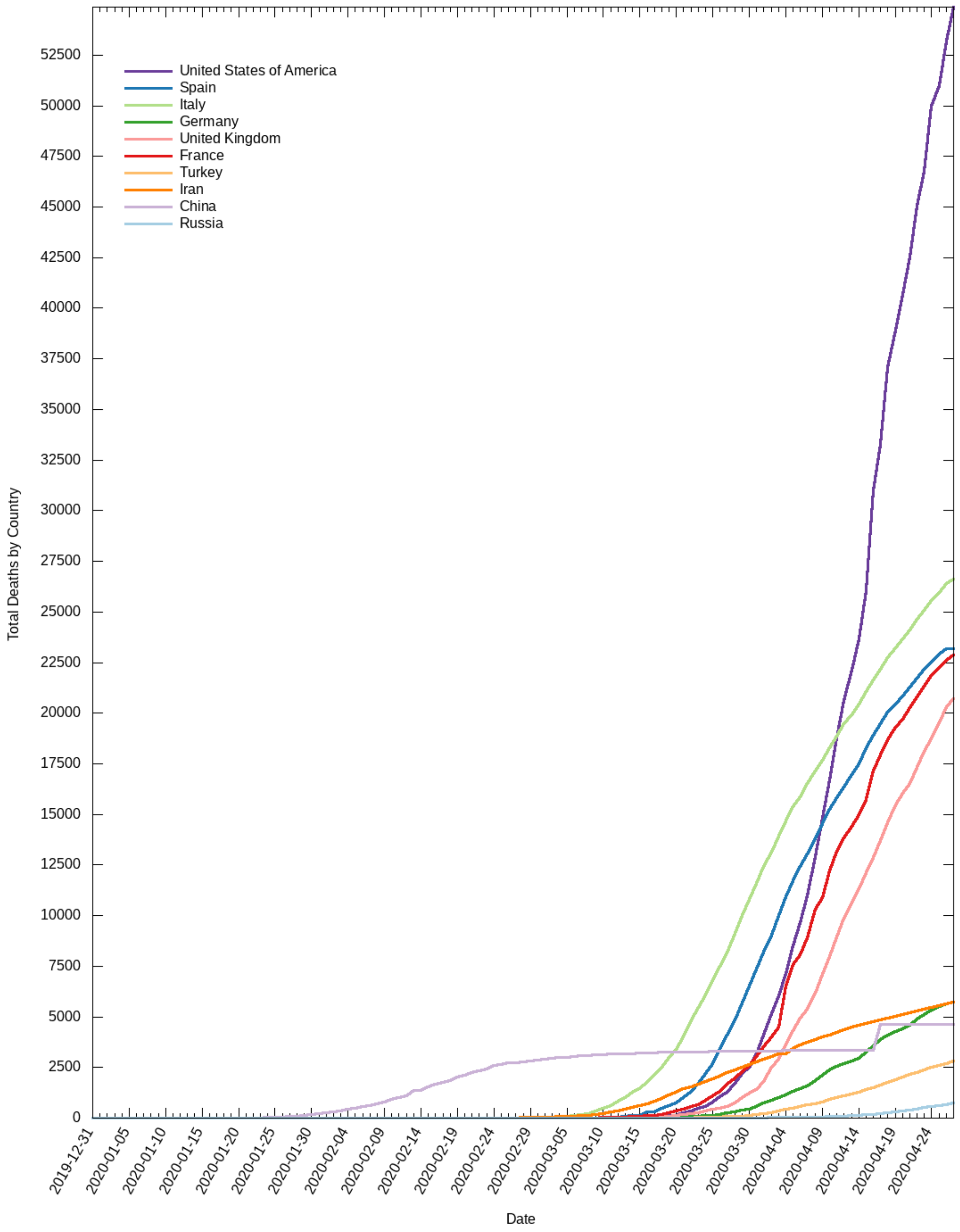

Figure 1: Cumulative Deaths by Country

```bash
#!/bin/bash
shownCountries=10;
queryResult=$(mktemp ./query-result.XXX);
countries=$(mktemp ./countries.XXX);
countryFiles=();
i=0;

sqlite3 coronavirus.db "SELECT geoId, SUM(cases) AS \"Total\" FROM CaseData GROUP BY geoId ORDER BY Total DESC LIMIT $shownCountries" > $countries;
sqlite3 coronavirus.db "SELECT T1.dateFormatted AS Date, T1.geoId AS Country, SUM(T3.deaths) AS \"Cumulative Deaths\" FROM CaseData AS T1 INNER JOIN ( SELECT geoId, SUM(cases) AS \"Total\" FROM CaseData GROUP BY geoId ORDER BY Total DESC LIMIT $shownCountries ) AS T2 ON Country = T2.geoId INNER JOIN CaseData AS T3 ON T1.dateFormatted >= T3.dateFormatted WHERE Country = T3.geoId GROUP BY Country, Date ORDER BY Country;" > $queryResult;

while IFS= read -r line
do
   grepArgument=$(echo "$line" | cut -c1-2);
   countryFile=$(mktemp ./$grepArgument.XXX);
   grep "$grepArgument" "$queryResult" > $countryFile;
   countryFiles+=($countryFile);
done < "$countries";

echo "set style line 1 lt rgb '#6a3d9a' lw 3" >> gnuplot.in;
echo "set style line 2 lt rgb '#1f78b4' lw 3" >> gnuplot.in;
echo "set style line 3 lt rgb '#b2df8a' lw 3" >> gnuplot.in;
echo "set style line 4 lt rgb '#33a02c' lw 3" >> gnuplot.in;
echo "set style line 5 lt rgb '#fb9a99' lw 3" >> gnuplot.in;
echo "set style line 6 lt rgb '#e31a1c' lw 3" >> gnuplot.in;
echo "set style line 7 lt rgb '#fdbf6f' lw 3" >> gnuplot.in;
echo "set style line 8 lt rgb '#ff7f00' lw 3" >> gnuplot.in;
echo "set style line 9 lt rgb '#cab2d6' lw 3" >> gnuplot.in;
echo "set style line 10 lt rgb '#a6cee3' lw 3" >> gnuplot.in;
echo "set term png size 1100,1500" >> gnuplot.in;
echo "set output 'graph.png'" >> gnuplot.in;
echo "set key at graph 0.35,0.95 Left reverse" >> gnuplot.in;
echo "set xlabel 'Date'" >> gnuplot.in;
echo "set ylabel 'Total Deaths by Country'" >> gnuplot.in;
echo "set ytics 0,2500,100000" >> gnuplot.in;
echo "set xtics 1577750400,432000,1977750400" >> gnuplot.in;
echo "set xtics rotate by 60 right" >> gnuplot.in;
echo "set datafile separator \"|\"" >> gnuplot.in;
echo "set xdata time" >> gnuplot.in;
echo "set format x \"%Y-%m-%d\"" >> gnuplot.in;
echo "set size ratio 1.36" >> gnuplot.in;
echo "set autoscale y" >> gnuplot.in;
echo "set autoscale x" >> gnuplot.in;
```

```
43 echo "set timefmt '%Y-%m-%d" >> gnuplot.in;
44 echo -n "plot " >> gnuplot.in;
45 for t in ${countryFiles[@]}; do
46     i=$(($i + 1));
47     countryCode=$(echo $t | cut -c3-4);
48     countryName=$(sqlite3 coronavirus.db "SELECT
       countriesAndTerritories FROM CountryID WHERE geoId = '
       $countryCode';");
49     countryName=$(echo $countryName | tr '_' ' ');
50     echo "\"$t\" using 1:3 title \"$countryName\" with lines ls $i,
       \\" >> gnuplot.in;
51 done
52 gnuplot gnuplot.in
53 
54 for t in ${countryFiles[@]}; do
55     rm -rf $t;
56 done
57 rm -rf gnuplot.in;
58 rm -rf "$queryResult";
59 rm -rf "$countries";
```

plot.sh

Explanation of plot.sh

- *Lines* 2 *to* 6: Variable declaration and temporary file creation.

  → *shownCountries* holds the number of countries to plot

  → *countryFiles* is an array that will hold the file names of the files (1 per country) containing the data to be plotted

  → *i* is a counter used to assign different styles to each line in the plot

- *Lines* 8 *to* 9: Piping query results into temporary files in preparation for further operations.

  → *countries* contains a list of the *geoId*s of the countries involved in the 2^{nd} query

  → *queryResult* contains the data to be plotted (Cumulative deaths by date limited to the top 10 countries)

- *Lines* 11 *to* 17: Separating results of the query into multiple files based on the country the data is referring to.

  For every country the program does the following:

  1. Stores the geoId into *grepArgument*
  2. Creates a temporary file named based on the geoId
  3. Stores the query results concerning that country in the temporary file
  4. Adds the temporary file to *countryFiles*, the array of files containing the data to be plotted for each country

- *Lines* 19 *to* 44: Insertion of the commands needed to setup the graph of the data in a visually pleasing and easy to read format.

- *Lines* 45 *to* 51: Insertion of the commands needed to plot the data from the query into the 'gnuplot.in' file.

  → Every country's line is plotted (line 50) with the country's name as title (obtained via the query at line 48), where underscores are replaced with spaces for aesthetic reasons (line 49)

- *Line* 52: Piping the commands stored in the file into GNUPlot.

- *Lines* 54 *to* 59: Deletion of temporary files as to clean up after creating the graph.