COMP1204: Data Management
Coursework Two: COVID-19 Cases Database

Damien Ta
34830294

May 7, 2024

1 The Relational Model

1.1 EX1

Relation	Type
dateRep - day	One-One
dateRep - month	One-One
dateRep - year	One-One
dateRep - cases	Many-Many
dateRep - deaths	Many-Many
dateRep - countriesAndTerritories	Many-Many
dateRep - geoId	Many-Many
dateRep - countryterritoryCode	Many-Many
dateRep - popData2020	Many-One
dateRep - continentExp	Many-One
day - month	Many-Many
day - year	Many-Many
day - cases	Many-Many
day - deaths	Many-Many
day - countriesAndTerritories	Many-Many
day - geoId	Many-Many
day - countryterritoryCode	Many-Many
day - popData2020	Many-Many
day - continentExp	Many-One
month - year	Many-Many
month - cases	Many-Many
month - deaths	Many-Many
month - countriesAndTerritories	Many-Many
month - geoId	Many-Many
month - countryterritoryCode	Many-Many
month - popData2020	Many-Many
month - continentExp	Many-One
year - cases	Many-Many

Relation	Type
year - deaths	Many-Many
year - countriesAndTerritories	Many-Many
year - geoId	Many-Many
year - countryterritoryCode	Many-Many
year - popData2020	Many-Many
year - continentExp	Many-One
cases - deaths	Many-Many
cases - countriesAndTerritories	Many-Many
cases - geoId	Many-Many
cases - countryterritoryCode	Many-Many
cases - popData2020	Many-Many
cases - continentExp	Many-One
deaths - countriesAndTerritories	Many-Many
deaths - geoId	Many-Many
deaths - countryterritoryCode	Many-Many
deaths - popData2020	Many-Many
deaths - continentExp	Many-One
countriesAndTerritories - geoId	One-One
countriesAndTerritories - countryterritoryCode	One-One
countriesAndTerritories - popData2020	One-One
countriesAndTerritories - continentExp	Many-One
geoId - countryterritoryCode	One-One
geoId - popData2020	One-One
geoId - continentExp	Many-One
countryterritoryCode - popData2020	One-One
countryterritoryCode - continentExp	Many-One
popData2020 - continentExp	Many-One

Attribute	Data Type
dateRep	TEXT
day	INTEGER
month	INTEGER
year	INTEGER
cases	INTEGER
deaths	INTEGER
countriesAndTerritories	TEXT
geoId	TEXT
countryterritoryCode	TEXT
popData2020	INTEGER
continentExp	TEXT

Table 1: Data Types of each attribute

1.2 EX2

List of assumptions for the Functional Dependencies:

- countriesAndTerritories, countryterritoryCode and geoId are interchangable as they are all very similar
- The only continent is Europe
- Assume all populations are the same, can't add different populations to a country that already has a set population
- Negative numbers will be included and not seen as an outlier
- Assume there are no typos or errors

Functional Dependency
dateRep -> day
dateRep -> month
dateRep -> year
day, month, year -> dateRep
countriesAndTerritories -> geoId
countriesAndTerritories -> countryterritoryCode
countriesAndTerritories -> popData2020
countriesAndTerritories -> continentExp
dateRep, countriesAndTerritories -> cases
dateRep, countriesAndTerritories -> deaths

Table 2: Minimal set of Functional Dependencies

1.3 EX3

As there are no redundant attributes, the potential candidate keys list is small. We could've included candidate keys like "dateRep, geoId" or "dateRep, countryterritoryCode" but they may cause redundancy.

Potential Candidate Keys
dateRep, countriesAndTerritories
day, month, year, countriesAndTerritories

Table 3: List of potential candidate keys

1.4 EX4

The primary key I selected is dateRep and countriesAndTerritories. I chose them because dateRep includes 3 attributes day, month, year which was our other candidate key. This makes the primary key more simple as it is a single attribute

Selected Primary Key
dateRep, countriesAndTerritories

Table 4: List of potential candidate keys

2 Normalisation

2.1 EX5

The partial dependencies of the table are:

- dateRep -> day, month, year
- countriesAndTerritories -> geoId, countryterritoryCode, popData2020, continentExp

These are partial dependencies because firstly; day, month, year do not depend on countriesAndTerritories meaning these attributes don't fully depend on the primary key. Secondly, geoId, countryterritoryCode, popData2020, continentExp do not rely on dateRep which is a part of the primary key meaning they don't fully depend on the primary key.

2.2 EX6

Using decomposition I broke down the dataset into new relations.

I created a new relation for the day, month and year attributes. I kept dateRep as my primary key in my original dataset and created a new table with dateRep as my primary key and the decomposed attributes day, month, year.

Date table: dateRep -> day, month, year

I also decomposed my other partial dependency countriesAndTerritories -> geoId, countryterritoryCode, popData2020, continentExp. I created a new table with countriesAndTerritories being primary keys in both my original table and my new table.

Country table: countriesAndTerritories -> geoId, countryterritoryCode, popData2020, continentExp

The original table is then left with just the primary keys and cases and deaths.

Dataset: dateRep, countriesAndTerritories -> cases, deaths

2.3 EX7

Transitive dependencies are when non-key attributes depend on other non-key attributes. In my tables there are no transitive dependencies. This is because all of my attributes depend on my primary keys such as day, month, year all depend on dateRep in the Date table.

2.4 EX8

Since there are no transitive dependencies and my table is already in second normal form, no further changes are needed to achieve third normal form, this is because it already meets the requirements for third normal form.

2.5 EX9

To be in Boyce-Codd normal form, every determinant must be a candidate key. In my tables there are 2 determinants, dateRep and countriesAndTerritories in the Date table and the Country table respectively. dateRep determines day, month and year. Whereas, countriesAndTerritories determine geoId, countryterritoryCode, popData2020 and continentExp. Both of these determinants are candidate keys meaning we are in Boyce-Codd normal form.

3 Modelling

3.1 EX10

Steps to creating an SQL version of our dataset:

1. Open terminal where SQLite is installed
2. 'cd' into the correct folder where you want your SQL file to be
3. Launch SQLite and create/open a database by typing the 'sqlite3 dataset.db' command
4. You can then create a table labelled 'dataset' using the attributes we used previously. You should also make sure that after each attribute, you define them as NOT NULL so there are no null values in your table.
5. Use the command '.mode csv' to set the SQLite output mode to CSV meaning it treats data as if it is in CSV format
6. Use the command '.import dataset.csv dataset' to import data from the csv file into the table named dataset
7. After importing you can output the imported data to a file using the command '.output dataset.sql' which creates a new SQL file with the values retrieved from the CSV file.
8. Finally, you can dump the table using the command '.dump dataset'

3.2 EX11

For EX11, I created the two tables that I decomposed in EX6 and put them into SQL full normalised representation. This included my Date table which consisted of dateRep, day, month and year and my other table Country which consisted of countriesAndTerritories, countryterritoryCode, geoId, popData2020 and continentExp.

```
CREATE TABLE Date (
    dateRep TEXT PRIMARY KEY NOT NULL,
    day INTEGER NOT NULL,
    month INTEGER NOT NULL,
    year INTEGER NOT NULL
);

CREATE TABLE Country (
    countriesAndTerritories TEXT PRIMARY KEY NOT NULL,
    geoId TEXT NOT NULL,
    countryterritoryCode TEXT NOT NULL,
    popData2020 INTEGER NOT NULL,
    continentExp TEXT NOT NULL
);
```

3.3 EX12

For EX12, I populated the two new tables I created using the INSERT INTO statement. This allowed me to retrieve columns from the dataset table and put it into my new tables.

```
INSERT INTO Date (dateRep, day, month, year)
SELECT DISTINCT dateRep, day, month, year
FROM dataset;

INSERT INTO Country(countriesAndTerritories, geoId, countryterritoryCode, popData2020,
    continentExp)
SELECT DISTINCT countriesAndTerritories, geoId, countryterritoryCode, popData2020,
    continentExp
FROM dataset;
```

3.4 EX13

After running the commands as seen below, I was able to populate a new database with the existing data from the exercises I had just completed.

```
sqlite3 coronavirus.db < dataset.sql
sqlite3 coronavirus.db < ex11.sql
sqlite3 coronavirus.db < ex12.sql
```

4 Querying

4.1 EX14

In order to find the worldwide total number of cases and deaths, I used the SQL query below. What this does is it calculates the sum of the cases and deaths columns from the dataset table and sets the sums as columns total_cases and total_deaths.

```
SELECT SUM(cases) AS total_cases, SUM(deaths) AS total_deaths
FROM dataset
```

4.2 EX15

To find the number of cases by date, in increasing date order, for the United Kingdom, I selected dateRep and cases from the dataset table. I then specified all values must have countriesAndTerritories = "United_Kingdom", I then ordered by year, month, day so it starts from years 2019-2022 and months January to December.

```
SELECT dateRep, cases
FROM dataset
WHERE countriesAndTerritories='United_Kingdom'
ORDER BY year, month, day ASC;
```

4.3 EX16

To find the number of cases and deaths by date, in increasing date order, for each country. I selected dateRep, countriesAndTerritories, cases and deaths from the dataset table and organised my SQL statement by the years, months and days in ascending order.

```
SELECT countriesAndTerritories, dateRep, cases, deaths
FROM dataset
ORDER BY year, month, day;
```

4.4 EX17

To find The total number of cases and deaths as a percentage (to 2DP) of the population, for each country. I began by selecting countriesAndTerritories from my Country table and also selected 2 other statements I created, this included percentage number of cases and percentage number of deaths. For percentage number of cases I took the SUM of cases of a particular country, CAST it as a real so the sum would be a decimal, divided the sum by the popData2020 and multiplied by 100 to make it a percentage. ROUND "100, 2" would round the number to 2 decimal places and I set the statement name AS cases_percentage.

```
SELECT Country.countriesAndTerritories,
ROUND(CAST(SUM(cases)as real) / Country.popData2020 * 100, 2) AS cases_percentage,
ROUND(CAST(SUM(deaths)as real) / Country.popData2020 * 100, 2) AS deaths_percentage
FROM dataset
JOIN Country ON dataset.popData2020 = Country.popData2020
GROUP BY Country.countriesAndTerritories
```

4.5 EX18

To find the descending list of the the top 10 countries by name, by percentage (to 2DP) total deaths out of total cases in that country, I selected the countriesAndTerritories column and created a new column that was made from the SUM of deaths divided by the SUM of cases. ROUND "100, 2" rounds the answer to 2 dp, CAST sets the sums to real so they are in a decimal form when the division occurs. I set the name to this new column as deathsToCasesPercentage. I then ordered the results into descending order using DESC and limited the number of results to 10 using LIMIT 10

```
SELECT Country.countriesAndTerritories,
ROUND(CAST(SUM(deaths)as real) / CAST(SUM(cases)as real) * 100, 2) AS
    deathsToCasesPercentage
FROM dataset
JOIN Country ON dataset.popData2020 = Country.popData2020
GROUP BY Country.countriesAndTerritories
ORDER BY deathsToCasesPercentage DESC LIMIT 10;
```

4.6 EX19

To find the date against a cumulative running total of the number of deaths by day and cases by day for the united kingdom, I first selected the dateRep, then I added two new columns called cumulative_cases and cumulative_deaths. These added cumulatively added the number of cases and the number of deaths ordered from the years 2019-2022, months January-December and then days. I selected it so that only results from the 'United_Kingdom' would show up, this info came from the table dataset.

```
SELECT dateRep,
SUM(cases) OVER (ORDER BY year, month, day) as cumulative_cases,
SUM(deaths) OVER (ORDER BY year, month, day) as cumulative_deaths
FROM dataset
WHERE countriesAndTerritories = 'United_Kingdom'
ORDER BY year, month, day
```