COMP1204: Data Management

Coursework Two

Zhengbin Lu
33839239

April 11, 2023

1 The Relational Model

1.1 EX1

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

(a) Relation Table 1

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

(b) Relation Table 2

Figure 1: Relations and Type

Attributes	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: Attribute Tab

1.2 EX2

The continentExp can't identify a continent uniquely if there is any other continent will be covered in the data in the future. The popData2020 can't uniquely identify a country when there is other countries with the same populatuion in the data. The date information, for instance, dateRep, day, month, and year are not null. The cases and deaths attributes can be a integer or a null value, but can't be a blank. The attribute countriesAndTerritories, geoId, countryterritoryCode, popData2020 can map to their countries as injections, thus these attributes can be determinant and they can uniquely identify their countries.

The FDs table is as follow.

Functional Dependencies
$dateRep \rightarrow day$
$dateRep \rightarrow month$
$dateRep \rightarrow year$
$countriesAndTerritories \rightarrow geoId$
$geoId \rightarrow countryterritoryCode$
$countryterritoryCode \rightarrow popData2020$
$countryterritoryCode \rightarrow continentExp$
$day, month, year \rightarrow dateRep$
$dateRep, geoId \rightarrow cases$
$dateRep, geoId \rightarrow deaths$
$geoId \rightarrow countriesAndTerritories$
$countryterritoryCode \rightarrow geoId$

Table 2: FDs Table

1.3 EX3

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

Table 3: Candidate Keys Table

1.4 EX4

The candidate key is composed by two part, the date attribute and the country attribute, so the primary key that I choose is

dateRep, geoId

dataRep contains the exact date of each case and death, and geoId contains the country detail. Compared with other candidate key, this one is more concise.

2 Normalisation

2.1 EX5

$dateRep \rightarrow continentExp, popData2020$
$geoId \rightarrow continentExp, popData2020$

continentExp and popData2020 are not key attributes, they dependent on the part of candidate key geoId or dateRep, thus these two attributes partialy dependent on the primary key.

The partial dependencies result in an addtional relation for decomposition.
$geoId, dateRep \rightarrow continentExp, popData2020$

2.2 EX6

When decomposing, I need to introduce extra two INTEGER attributes, **data_ID** and **geo_ID** as surrogate keys. Using surrogate keys can guarantee that each row in the table can be identified uniquely. Furthermore, the surrogate keys will replace original primary key and as new primary key. Thus, we have the new relations in the tables:

$data\_ID \rightarrow dateRep, day, month, year$
$geo\_ID \rightarrow geoId, countriesAndTerritories, countryterritoryCode$
$geo\_ID \rightarrow popData2020, cotinentExp$
$data\_ID, geo\_ID \rightarrow cases, deaths$

The field of these two attributes are geo_ID and data_ID, and the type of these two attributes is INTEGER, as surrogate keys in the table.

2.3 EX7

According to the definition of the transitive dependent, the primary key in relation geo_ID $->$ geoId, countriesAndTerritories, countryterritoryCode is geo_ID, the rest of attributes are the part of candidate key, and dependent on the primary key, so there is no transitive dependent. For the other relation, it is obviousely that there is no transitive dependent.

2.4 EX8

Depending on assumptions I have made and earlier processes, i'm already in the 3NF. In the relations I provided, there is no transitive dependency. Therefore there are no non-key attributes that are transitively dependent on any candidate key. For every FD $A \rightarrow b$, A is a super key, b is either a key or non-key. For the key attribute in the relation, it can only be implied by super key, for non-key attribute, there is no transitive dependency.

2.5 EX9

The relations above are already in BCNF, for each FD, the determinant is candidate key, and it can identify a unique row. There is only one relation $data\_ID \rightarrow dateRep, day, month, year$ have more than one attribute in the candidate but the keys are disjoint. Thus, the relations are already in BCNF.

3 Modelling

3.1 EX10

1. Starts terminal and input **sqlite3 coronavirus.db** to create a empty coronavirus.db
2. Inputs **.mode csv** to shift into the csv mode and use **.import dataset.csv dataset** to import the csv file into the table dataset in coronavirus.db.
3. Uses **.output dataset.sql** to set the output file.
4. Uses the **.dump** to dump the database as the dataset.sql.
5. Runs **sqlite3 coronavirus.db < dataset.sql** to populate the dataset table.

3.2 EX11

The SQL with full normalised representation is as follow. The surrogate keys that I'm introduced is ID in the table Data and CountriesAndTerritories, these keys can increase automatically and identify each row uniquely in the table.

```
PRAGMA foreign_keys = ON;

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

CREATE TABLE CountriesAndTerritories
(
    ID                      INTEGER PRIMARY KEY AUTOINCREMENT,
    geoId                   TEXT NOT NULL UNIQUE,
    countriesAndTerritories TEXT NOT NULL UNIQUE ,
    countryterritoryCode    TEXT NOT NULL UNIQUE
);

CREATE TABLE PopAndContinent
(
    geo_ID       INTEGER NOT NULL,
    popDate2020  INTEGER NOT NULL,
    continentExp TEXT    NOT NULL,
    FOREIGN KEY (geo_ID) references CountriesAndTerritories (ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE CasesAndDeaths
(
    data_ID INTEGER,
    geo_ID  INTEGER,
    cases   INTEGER,
```

```
    deaths  INTEGER,
    PRIMARY KEY (data_ID, geo_ID),
    FOREIGN KEY (data_ID) references Date (ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (geo_ID) REFERENCES CountriesAndTerritories (ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

3.3 EX12

```
INSERT INTO Date (dateRep, day, month, year)
SELECT DISTINCT dateRep, day, month, year
FROM dataset
LIMIT -10 OFFSET 1;

INSERT INTO CountriesAndTerritories (geoId, countriesAndTerritories, countryterritoryCode)
SELECT DISTINCT geoId, countriesAndTerritories, countryterritoryCode
FROM dataset
LIMIT -10 OFFSET 1;

INSERT INTO PopAndContinent (geo_ID, popDate2020, continentExp)
SELECT DISTINCT C.ID, popData2020, continentExp
FROM dataset
         JOIN CountriesAndTerritories C ON dataset.geoId = C.geoId;

INSERT INTO CasesAndDeaths (data_ID, geo_ID, cases, deaths)
SELECT DISTINCT D.ID, C.ID, cases, deaths
FROM dataset
         JOIN Date D ON dataset.dateRep = D.dateRep
         JOIN CountriesAndTerritories C ON dataset.geoId = C.geoId;
```

3.4 EX13

Successfully run the command and end up with a fully populated database.

4 Querying

4.1 EX14

```
SELECT SUM(cases) as casesSum, SUM(deaths) as deathsSum
FROM CasesAndDeaths;
```

4.2 EX15

```
SELECT D.dateRep, cases FROM  CasesAndDeaths
JOIN Date D on CasesAndDeaths.data_ID = D.ID
JOIN CountriesAndTerritories CAT on CasesAndDeaths.geo_ID = CAT.ID
WHERE geoId = 'UK'
ORDER BY year, month, day;
```

4.3 EX16

```
SELECT geoId,dateRep, cases, deaths
FROM CasesAndDeaths
         JOIN Date D on CasesAndDeaths.data_ID = D.ID
         JOIN CountriesAndTerritories CAT on CasesAndDeaths.geo_ID = CAT.ID
ORDER BY year, month, day, countriesAndTerritories;
```

4.4 EX17

```
SELECT CAT.geoId,
    (SUM(CasesAndDeaths.cases) * 100.0 / PAC.popDate2020) || '%'  as casesPercentage,
    (SUM(CasesAndDeaths.deaths) * 100.0 / PAC.popDate2020) || '%' as deathsPercentage
FROM CasesAndDeaths
         JOIN CountriesAndTerritories CAT on CasesAndDeaths.geo_ID = CAT.ID
         JOIN PopAndContinent PAC ON CAT.ID = PAC.geo_ID
GROUP BY CAT.geoId
ORDER BY CAT.geoId;
```

4.5 EX18

```
SELECT CAT.geoId,
       (SUM(CasesAndDeaths.deaths) * 100.0 / SUM(CasesAndDeaths.cases)) || '%' as
    deathsPercentage
FROM CasesAndDeaths
         JOIN CountriesAndTerritories CAT on CasesAndDeaths.geo_ID = CAT.ID
GROUP BY CAT.geoId
ORDER BY deathsPercentage DESC;
```

4.6 EX19

```
SELECT dateRep,
       SUM(deaths) OVER (ORDER BY year, month, day) AS cumulativeDeaths,
       SUM(cases) OVER (ORDER BY year, month, day)  AS cumulativeCases
FROM CasesAndDeaths
         JOIN Date D on D.ID = CasesAndDeaths.data_ID
         JOIN CountriesAndTerritories CAT on CAT.ID = CasesAndDeaths.geo_ID
WHERE geoId = 'UK';
```

5 Extension

5.1 EX20

```
#!/bin/bash

# Create an empty array to store the temporary files
declare -a data

# Create a command which will be excuted by the gnuPlot
plot_command="plot "

# Fetch the top 10 death country and iterate them, for each loop, grab the date, sum of the
    death and geoId from the table, and put them into a temp file after processing
# Append the gnuPlot command
for geoId in $(sqlite3 coronavirus.db "SELECT geoId FROM CountriesAndTerritories JOIN
    CasesAndDeaths CAD on CountriesAndTerritories.ID = CAD.geo_ID GROUP BY geoId ORDER BY SUM(
    deaths) DESC LIMIT 10;")
do
    Deaths=$(mktemp)
    sqlite3 coronavirus.db "SELECT dateRep, SUM(deaths) OVER (ORDER BY year, month, day) AS
    cumulativeDeaths, '$geoId' as country FROM CasesAndDeaths JOIN Date D ON D.ID =
    CasesAndDeaths.data_ID JOIN CountriesAndTerritories CAT ON CAT.ID = CasesAndDeaths.geo_ID
    WHERE geoId = '$geoId' ORDER BY year, month, day;" | tr '|' ' ' > "$Deaths"
    plot_command+=" '$Deaths' using 1:2 with lines title columnheader(3),"
    data+=("$Deaths") # Add the temporary file to the array, and delete later
done

# Remove the final comma from the gnuPlot
plot_command="${plot_command%,}"
```

```
21
22 # Run the gnuPlot script by using a heredoc.
23 # Set the key's position on the top right of picture, out of the graph.
24 # Set the size and the font of the picture.
25 # Set the output file
26 # Set the X,Y label and the title
27 # Set the x data type as time
28 # Set the time format which is the same format as dateRep
29 # Set the display format
30 # Set the distance between two major tick marks
31 # Rotate the x coordinates
32 gnuplot << EOF
33 set key outside top right
34 set key autotitle columnheader
35 set terminal pngcairo size 1000,750 enhanced font 'Verdana,10'
36 set output 'graph.png'
37 set xlabel 'Date'
38 set ylabel 'Deaths'
39 set title 'Cumulative Deaths'
40 set xdata time
41 set timefmt "%d/%m/%Y"
42 set format x "%Y-%m-%d"
43 set xtics 60*60*24*30*2
44 set xtics mirror rotate by -60
45 $plot_command
46 EOF
47
48 # Delete all the temporary files by using a loop
49 for data in "${data[@]}"; do
50     rm "$data"
51 done
```

5.2 The graph of the top ten countries for cumulative deaths