Data Management Coursework 2

Harry Nelson
hjn2g19
30764815

May 2020

1 The Relational Model

EX1

The dataset we've been given is a relation of 11 named attributes, "dateRep" from the domain of dates, "day", "month", "year", "cases", "deaths" and "popData2018" from the domain of integers and "countriesAndTerritories", "geoId", "countryterritoryCode" and "continentExp" from the domain of strings.

$CoronavirusCasesByDay(date : dateRep, int : day, int : month, int : year, int : cases, int : deaths, String : countriesAndTerritories, String : geoId, String : countryterritoryCode, int : popData2018, String : continentExp)$

EX2

Functional dependencies from this dataset include:

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

I chose not include popData2018 as an attribute that could be functionally depended upon as despite the odds of them overlapping being very small, countries could have the same population and therefore has the potential of not uniquely identifying any other attributes. "countryterritoryCode" is also N/A for entries like "Cases on international conveyance" and so can't be used in place of geoId to obtain information, but assuming whenever N/A is used as a country code then the continent is listed as "other" then continentExp can still be functionally dependent on it. As I am basing these assuming more international conveyance type entries will be made, I also cannot use countriesAndTerritories as a dependency for anything except countryterritoryCode and continent either, as any other cruise ships from japan for example would be a duplicate entry in the current dataset.

EX3

Potential candidate keys are found by repeatedly applying the closure algorithm on the relation. Assuming the previously listed functional dependencies, we can see that possible candidate keys include:

- dateRep, geoId $\rightarrow$ all fields
- day, month, year, geoId $\rightarrow$ all fields

EX4

I believe a suitable primary key would be (dateRep, geoId) as it is the shortest of the possible candidate keys, meaning less data stored overall. There is not a unique id per entry in the dataset so with this key it is impossible to guarantee that the fields in the key are unique, due to a country logging two statistics in one day but as that doesn't occur in the dataset currently it should be safe to assume the primary key will stay valid.

2 Normalisation

EX5

Partial-key dependencies currently standing in the dataset include:

- dateRep $\rightarrow$ day, month, year
- geoId $\rightarrow$ countriesAndTerritories, countryterritoryCode, popData2018, continentExp

EX6

To decompose this relation into 2NF you would need to create relations using dateRep to get information about the date, and geoId to get information about the country, then a combination of the two to get the case statistics. This means things like day, month, popdata etc. won't be unnecessarily repeated.

To convert the relation to 2NF you would remove the partial dependencies stated before by creating the new relations. These would be:

- $DailyStats(date : dateRep, String : geoId, int : cases, int : deaths)$ This relation would contain information on the cases and deaths fields, and are tied to the (dateRep, geoId) keys.
- $CountryInfo(String : geoId, String : countriesAndTerritories, String : countryterritoryCode, int : popData2018, String : continentExp)$ This relation would contain information about the country based on the geoId key.

- $DateInfo(date : dateRep, int : day, int : month, int : year)$ This relation would contain information on the date based on the dateRep key.

I chose these because it means any data can be obtained using some combination of my original primary key (dateRep, geoId), and it prevents redundant data being listed (such as the country name and population statistics being listed in every case listing).

EX7

I can only find one transitive dependency in these new relations, which is that geoId → countryterritoryCode → continentExp, and geoId is not functionally dependent on countryterritoryCode due to my assumption N/A can occur for multiple places.

EX8

To remove this transitive dependency you can split the CountryInfo relation into two relations, CountryInfo with geoId, countriesAndTerritories, countryterritoryCode and popData2019. This means there is no redundant data stored as you can go from the countryterritoryCode to the continent. However, due to there being NULL values in the countryterritoryCode column it cannot be used as a primary key for a relation. To get around this I will introduce a surrogate key "ctc_id" which I can then use to extract the continent based on country code like before but will still return results for the null ones. The relations will then look like this:

- $CountryInfo(String : geoId, String : countriesAndTerritories, int : ctc\_id, int : popData2018)$
- $ContinentInfo(int : ctc\_id, String : countryterritoryCode, String : continentExp)$

EX9

I think every table is in BCNF as no table has more than one candidate key apart from DateInfo, which has two but only one of them is made up of more than one attribute, so all relations should satisfy the conditions for BCNF.

3 Modelling

EX10

I followed the instructions on the coursework page to import the csv using ".mode csv" and ".import dataset.csv" then dumped using ".out dataset.sql" and ".dump".

EX11

When porting the dataset to an sqlite database I decided to index the DailyStats table, as it is the table containing my original primary key and so assigns an index to every entry of the original dataset. When importing the csv it reads the empty values as " rather than null, so at the end of the dumped dataset table it replaces instances of " with null.

EX12

When adding the surrogate key described in EX8 (ctc_id) I used the rowid of the original dataset table as the unique id in order to be able to insert the correct id into the CountryInfo table afterwards, as with countryterritoryCode being null sometimes there was no other way to obtain the correct id from continent without using another field like geoId and then removing it afterwards.

EX13

It did in fact work on a clean database when trying to run my dumped sql files, so I carried on to the next section.

4 Querying

EX14: The worldwide total number of cases and deaths

```
SELECT SUM(cases), SUM(deaths) FROM DailyStats;
```

EX15: The number of cases and the date, by increasing date order, for the United Kingdom

```
SELECT DailyStats.daterep, cases
FROM DailyStats
INNER JOIN DateInfo ON DailyStats.dateRep = DateInfo.DateRep
WHERE geoId = 'UK'
ORDER BY year ASC, month ASC, day ASC;u
```

EX16: The number of cases, deaths and the date, by increasing data order, for each continent

```
SELECT continentExp, DailyStats.dateRep, SUM(cases), SUM(deaths)
FROM DailyStats
INNER JOIN DateInfo ON DailyStats.dateRep = DateInfo.dateRep
INNER JOIN CountryInfo ON DailyStats.geoId = CountryInfo.geoId
INNER JOIN ContinentInfo ON CountryInfo.ctc_id = ContinentInfo.ctc_id
GROUP BY continentExp, DailyStats.dateRep
ORDER BY continentExp, year ASC, month ASC, day ASC;
```

EX17: The number of cases and deaths as a percentage of the population, for each country

```
SELECT
ROUND(((100.0 * SUM(cases))/popData2018), 2) || '%' as percentCase,
ROUND(((100.0 * SUM(deaths))/popData2018), 2) || '%' as percentDeath,
    countriesAndTerritories FROM DailyStats
INNER JOIN CountryInfo ON DailyStats.geoId = CountryInfo.geoId
GROUP BY DailyStats.geoId ORDER BY countriesAndTerritories;
```

EX18: A descending list of the the top 10 countries, by percentage deaths out of cases

```
SELECT countriesAndTerritories,
       ROUND(100.0 * SUM(deaths)/SUM(cases), 2) || '%' as percentDied
FROM DailyStats
INNER JOIN CountryInfo ON DailyStats.geoId = CountryInfo.geoId
GROUP BY countriesAndTerritories
ORDER BY ROUND(100.0 * SUM(deaths)/SUM(cases), 10) DESC
LIMIT 0, 10;
```

EX19: The date against a cumulative running total of the number of deaths by day and cases by day for the united kingdom

This was done before I heard you were allowed to update sqlite but it works.

```
SELECT a.dateRep, SUM(b.cases), SUM(b.deaths) FROM
    (SELECT dateRep, STRFTIME(
                        SUBSTR(daterep, 7, 4)|| '-' ||
                        SUBSTR(daterep, 4, 2) || '-' ||
                        SUBSTR(daterep, 1, 2)
                    ) AS date
    FROM DateInfo ORDER BY date ASC) as a,
    (SELECT cases, deaths, STRFTIME(
                        SUBSTR(daterep, 7, 4)|| '-' ||
                        SUBSTR(daterep, 4, 2) || '-' ||
                        SUBSTR(daterep, 1, 2)
                    ) AS date
    FROM DailyStats WHERE geoId = 'UK' ORDER BY date ASC) as b
WHERE a.date >= b.date
GROUP BY a.date;
```