Database Modelling

COMP1204 - Coursework 2

ISAAC KLUGMAN

Student ID: 30979536

Username: ik1g19

The Relational Model

EX1

The relations and their relevant data types:
-$dateRep$ - The day, month and year in date format of when the record was collected. TEXT Data type.
-day - The number of the day of the month of which the record was collected. INTEGER Data type.
-$month$ - The number of the month of which the record was collected. INTEGER Data type.
-$year$ - The year of which the record was collected. INTEGER Data type.
-$cases$ - The number of recorded cases on that day. INTEGER Data type.
-$deaths$ - The number of deaths on that day. INTEGER Data type.
-$countriesAndTerritories$ - The name of the country or territory. TEXT Data type.
-$geoId$ - The geographical identifier for that country or territory. TEXT Data type.
-$countryTerritoryCode$ - The code for that country or territory. TEXT Data type.
-$popData2018$ - The population of the country or territory as of 2018. INTEGER Data type.
-$continentExp$ - The name of the country or territory's continent. TEXT Data type.

EX2

Functional dependencies in the Data Set:
-$dateRep \rightarrow day, month, year$
-$countriesAndTerritories \rightarrow geoId, countryterritoryCode, continentExp, popData2018$
-$geoId \rightarrow countriesAndTerritories, countryterritoryCode, continentExp, popData2018$

Individually, day, $month$, $year$, $cases$, $deaths$, $countryTerritoryCode$ and $continentExp$ do not functionally determine anything.
In this given data set, $popData2018$ functionally determines $countriesAndTerritories$, $geoId$, $continentExp$ and $countryTerritoryCode$. However it is possible that two countries could of had the same population in 2018, if a country was added to this data set with an identical population in 2018 then $popData2018$ would no longer functionally determine any other attributes.
$countryTerritoryCode$ would ordinarily determine $countriesAndTerritories$, $geoId$, $continentExp$ and $popData2018$. However in the given data set, there are some instances where $countryTerritoryCode$ is blank, meaning that the code can no longer uniquely determine any other attributes.

EX3

Potential Candidate keys:
-$dateRep, geoId$
-$dateRep, countriesAndTerritories$
-$dateRep, countryterritoryCode$

EX4

$dateRep$ and $geoId$ make a suitable composite primary key.
$dateRep$ can uniquely identify a record for each country and $geoId$ is a short string. I am not using $countriesAndTerritories$ as the strings are longer and $countryterritoryCode$ is not present for all countries.

Normalisation

EX5

Existing partial key dependencies:
-$day, month, year \rightarrow dateRep$
-$countriesAndTerritories, countryterritoryCode, continentExp \rightarrow geoId$

To solve this we can divide the original relation into three relations:
-$dateRep, day, month, year$
-$geoId, countriesAndTerritories, countryterritoryCode, continentExp, popData2018$
-$dateRep, geoId, cases, deaths$

EX6

The three new relations:

Date			
dateRep	day	month	year
DATE	INTEGER	INTEGER	INTEGER

Country				
geoId	countriesAndTerritories	countryterritoryCode	continentExp	popData2018
TEXT	TEXT	TEXT	TEXT	INTEGER

Virus Info			
dateRep	geoId	cases	deaths
TEXT	TEXT	INTEGER	INTEGER

The primary keys for their respective tables are:
-Date - $dateRep$
-Country - $geoId$
-Virus - $dateRep, geoId$

The foreign keys for their respective tables are:
-Date - None
-Country - None
-Virus - $dateRep$ and $geoId$

I established any partial key dependencies, day, $month$ and $year$ were dependent on $dateRep$ and not $geoId$, so they were only partially dependent on the key.
$countriesAndTerritories$, $countryTerritoryCode$ and $continentExp$ were dependent on $geoId$ and not $dateRep$, so they were only partially dependent on the key. I then divided the attributes so that they all depended on the entirety of their respective keys.

EX7

As the relations currently exist, there are no transitive dependencies.
$countriesAndTerritories$ and $countryterritoryCode$ would be a transitive dependency as $geoId \rightarrow countriesAndTerritories$ and $countriesAndTerritories \rightarrow countryterritoryCode$.
But this is not a valid transitive dependency as $countriesAndTerritories \rightarrow geoId$, and for a transitive dependency $A \rightarrow B \rightarrow C$, there cannot exist $B \rightarrow A$.

EX8

The relations in 3NF:

Date			
dateRep	day	month	year
DATE	INTEGER	INTEGER	INTEGER

Country				
geoId	countriesAndTerritories	countryterritoryCode	continentExp	popData2018
TEXT	TEXT	TEXT	TEXT	INTEGER

Virus Info			
dateRep	geoId	cases	deaths
TEXT	TEXT	INTEGER	INTEGER

Since there are no transitive dependencies, the relations are the same as they were in 2NF.

The primary keys for their respective tables are:
-Date - $dateRep$
-Country - $geoId$
-Virus - $dateRep$, $geoId$

The foreign keys for their respective tables are:
-Date - None
-Country - None
-Virus - $dateRep$ and $geoId$

EX9

The relation is in Boyce-Codd Normal Form (BCNF), BCNF requires for every all dependencies $A \rightarrow B$, A must be a superkey.
For example this is true in Country, $countriesAndTerritories$ determines other attributes, though it is a superkey so it is allowed in BCNF.

Modelling

EX11

The indexes I have created in their respective tables are:
-Date - Index $dateIndex$ created on $dateRep$
-Virus - Index $dateLocation$ created on $dateRep$ and $geoId$
-Country - Index $locationName$ created on $countriesAndTerritories$ and index $locationId$ created on $geoId$
I chose $dateRep$ as the index for Date as anyone querying the Date table is likely to be doing so using a specific date.
I chose $dateRep$ and $geoId$ to be the indexes for Virus as case and death information is likely to be queried with a time and place.
I chose $countriesAndTerritories$ as an index for Country as country information is likely to be queried with the country name.
I also chose $geoId$ as an index for Country as country information is also likely to be queried using the countries ID.

Querying

EX14

```
SELECT sum(cases) AS "total cases",
       sum(deaths) AS "total deaths"
FROM Virus;
```

EX15

```
SELECT virus.dateRep AS date,
       virus.cases AS "number of cases"
FROM Virus INNER JOIN date ON virus.dateRep = date.dateRep
WHERE virus.geoId = 'UK'
ORDER BY date.year, date.month, date.day ASC;
```

EX16

```sql
SELECT country.continentExp AS continent,
       virus.dateRep AS date,
       sum(virus.cases) AS "number of cases",
       sum(virus.deaths) AS "number of deaths"
FROM Virus
INNER JOIN country ON virus.geoId = country.geoId
INNER JOIN date ON virus.dateRep = date.dateRep
GROUP BY country.continentExp, virus.dateRep
ORDER BY date.year, date.month, date.day ASC;
```

EX17

```sql
SELECT country.countriesAndTerritories AS country,
       round((sum(virus.cases) * 1.0 / country.popData2018) *
          100, 2) AS "% cases of population",
       round((sum(virus.deaths) * 1.0 / country.popData2018) *
          100, 2) AS "% cases of deaths"
FROM Virus
INNER JOIN Country ON virus.geoId = country.geoId
GROUP BY country.countriesAndTerritories, country.popData2018;
```

EX18

```sql
SELECT country.countriesAndTerritories AS "country name",
       round((sum(virus.deaths) * 1.0 / sum(virus.cases)) *
          100, 2) AS "% cases of deaths"
FROM Virus
INNER JOIN Country ON virus.geoId = country.geoId
GROUP BY country.countriesAndTerritories, country.popData2018
ORDER BY "% cases of deaths" DESC
LIMIT 10;
```

EX19

```
SELECT date.dateRep,
       sum(virus.cases) OVER (
           ORDER BY date.year, date.month, date.day
           ROWS BETWEEN
               UNBOUNDED PRECEDING
               AND CURRENT ROW
           ) AS 'cumulative UK cases',
       sum(virus.deaths) OVER (
           ORDER BY date.year, date.month, date.day
           ROWS BETWEEN
               UNBOUNDED PRECEDING
               AND CURRENT ROW
           ) AS 'cumulative UK deaths'
FROM Date
INNER JOIN Virus ON date.dateRep = virus.dateRep AND virus.
   geoId = 'UK'
ORDER BY date.year, date.month, date.day ASC;
```