Cw2 Data Management

Theodoulos Hadjilambrou Student ID: 34200134

April 2024

1 The Relational Model

1.1 EX1:

Express the relation directly represented in the dataset file. Assign relevant SQLite data types to each column.

The relational schema for the dataset is defined as follows:

COVID_Data(dateRep TEXT, day INTEGER, month INTEGER, year INTEGER,
cases INTEGER, deaths INTEGER, countriesAndTerritories TEXT,
geoId TEXT, countryterritoryCode TEXT, popData2020 INTEGER,
continentExp INTEGER)

Table 1: COVID_Data Schema

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

1.2 EX2:

List the minimal set of Functional Dependencies (FDs)

Based on the attributes of the COVID_Data table and the given assumptions, the minimal set of Functional Dependencies are as follows:

Table 2: List of Functional Dependencies

Dependency	**Explanation**
countryterritoryCode ↔ countriesAndTerritories	Bijective relationship
countryterritoryCode ↔ geoId	Bijective relationship
countryterritoryCode → popData2020	Unique per country
countryterritoryCode → continentExp	Unique per country
dateRep → day	Date decomposition
dateRep → month	Date decomposition
dateRep → year	Date decomposition
countriesAndTerritories + dateRep → cases	Event-based reporting
countriesAndTerritories + dateRep → deaths	Event-based reporting

Assumptions:

- Each country or territory is uniquely identified by the countryterritoryCode, which is a stable and unique identifier.
- The geoId and countriesAndTerritories are functionally dependent on the countryterritoryCode and do not add additional information.
- The popData2020 and continentExp are attributes that are also dependent on the countryterritoryCode.
- The dateRep is a unique identifier for each date and is used to determine the individual components of the date: day, month, and year.
- For each countriesAndTerritories and dateRep combination, the cases and deaths are recorded, indicating a unique count for each day.

1.3 EX3:

From your minimal set of functional dependencies, list the potential candidate keys

Given the functional dependencies established in the previous section, the potential candidate key for the COVID_Data table is:

- [countryTerritoryCode, dateRep]
- [geoID, dateRep]
- [countriesAndTerritories, dateRep]

1.4 EX4:

Identify a suitable primary key, and justify your decision.

Primary Key:

- [countryTerritoryCode, dateRep]

The [countryTerritoryCode, dateRep] combination is selected as the primary key due to its standardization, efficiency, and reliability. countryTerritoryCode is likely a standardized code that ensures consistency and unambiguity across different datasets, while dateRep allows us to capture the temporal aspect of the data.

2 Normalisation

2.1 EX5:

List any partial-key dependencies in the relation as it stands and any resulting additional relations you should create as part of the decomposition.

The following partial-key dependencies have been identified in the current relation:

- The attributes popData2020 and continentExp are dependent on countryTerritoryCode, which is part of the composite key [countryTerritoryCode, dateRep], but do not depend on dateRep, indicating partial dependency.
- The attributes day, month, and year are solely dependent on dateRep and do not depend on countryTerritoryCode, indicating another set of partial dependencies.

These dependencies imply that the current relation is not in 2nd Normal Form due to the existence of attributes that do not depend on the entire composite key.

2.2 EX6:

Decompose the relation to achieve 2nd Normal Form and list the new relations with their fields, types, and keys.

Table 3: Decomposed Relations to Achieve 2nd Normal Form

Relation Name	**Attributes**	**Key**
Country Info Attributes	countryTerritoryCode, popData2020, continentExp, geoID, countriesAndTerritories	countryTerritoryCode
Date Attributes	dateRep, day, month, year	dateRep
Case Data	countryTerritoryCode, dateRep, cases, deaths	[countryTerritoryCode, dateRep]

The process of decomposition involved creating separate tables where each non-key attribute is fully functionally dependent on a primary key. The Country Attributes Relation stores static information related to the country identified by countryTerritoryCode. The Date Attributes Relation maintains components of the date that can be derived from dateRep. This normalization eliminates the partial dependencies and ensures that the relations conform to 2nd Normal Form.

2.3 EX7

List transitive dependencies.

- According to my FDs:
  - countryTerritoryCode has a bijective relationship with both countriesAndTerritories and geoId, so they determine each other mutually and do not form a transitive dependency.
  - popData2020 and continentExp are directly dependent on countryTerritoryCode, which is a key attribute, not a non-key attribute.
  - dateRep directly determines day, month, and year, with no intermediate non-key attribute so not a transitive dependency.
  - The dependencies of cases and deaths on the combination of countryTerritoryCode + dateRep do not introduce any non-key to non-key dependencies.

2.4 EX8:

Convert your relations into 3rd Normal Form using your answers to the above.

In summary, all the non-key attributes are dependent on key attributes, and there are no chains of dependencies where a non-key attribute determines another non-key attribute that determines a key attribute. Therefore, the relations, as defined by these FDs, do not exhibit transitive dependencies and are in 3NF.

2.5 EX9:

Finally, convert your relations into Boyce-Codd Normal Form. Justify and explain how your relations are in BCNF.

To be in Boyce-Codd Normal Form, a relation must satisfy the following condition: for every one of its non-trivial functional dependencies $X \rightarrow A$, X must be a superkey.

In our case, all the functional dependencies identified:

- countryTerritoryCode $\rightarrow$ popData2020, continentExp
- dateRep $\rightarrow$ day, month, year
- countryTerritoryCode, dateRep $\rightarrow$ cases, deaths

have superkeys as their determinants. The determinant for each functional dependency is either a candidate key or a part of a candidate key:

- The attribute countryTerritoryCode is a candidate key in the Country Information relation, uniquely identifying each record.
- The attribute dateRep is a candidate key in the Date Information relation, uniquely identifying each record.
- The composite key $[countryTerritoryCode, dateRep]$ is a superkey in the Case Data relation, uniquely identifying each record.

Since there are no attributes determined by anything less than a candidate key, our relations conform to BCNF. There are no partial key dependencies or transitive dependencies, and all determinants are candidate keys.

Therefore, our data is in Boyce-Codd Normal Form.

3 Modelling

3.1 EX10:

Import,Export,Link

To prepare the data for analysis, the following steps were performed:

1. **SQLite Database Creation:**

```
sqlite3 coronavirus.db
```

2. **CSV Data Import:**

```
.mode csv
.import '/path/to/dataset.csv' dataset
```

3. **Data Export:**

```
.output dataset.sql
.dump dataset
```

4. **DataGrip Connection:** DataGrip was opened and configured to connect to the `coronavirus.db` database. The database was successfully added, allowing for further data manipulation and querying within DataGrip's GUI.

3.2 EX11:

```
CREATE TABLE CountryInfo (
  countryterritoryCode TEXT PRIMARY KEY,
  countriesAndTerritories TEXT,
  popData2020 INTEGER,
  continentExp TEXT
);
CREATE TABLE DateInfo (
  dateRep TEXT PRIMARY KEY,
  day INTEGER,
  month INTEGER,
  year INTEGER
);
CREATE TABLE CaseData (
  countryterritoryCode TEXT ,
  dateRep TEXT,
  cases INTEGER,
  deaths INTEGER,
  PRIMARY KEY
(countryterritoryCode,dateRep)
);
```

3.3 EX12:

```
INSERT INTO CountryInfo (countryterritoryCode, countriesAndTerritories, popData2020, continentExp)

SELECT DISTINCT countryterritoryCode,countriesAndTerritories,popData2020, continentExp)
FROM dataset;

INSERT INTO CaseData (countryterritoryCode, dateRep, cases, deaths)
SELECT DISTINCT countryterritoryCode, dateRep, cases, deaths
FROM dataset;

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

3.4 EX13:

The following commands all work normally when run:

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

4 Querying

4.1 EX14:

```
SELECT SUM(cases) AS TotalCases, SUM(deaths) AS TotalDeaths FROM CaseData;
```

4.2 EX15:

```
SELECT
  dateRep,
  SUM(cases) as TotalCases
FROM
  CaseData
WHERE
  countryterritoryCode = 'GBR'
GROUP BY
  dateRep
ORDER BY
  strftime('%Y-%m-%d', SUBSTR(dateRep, 7, 4) || '-' ||SUBSTR(dateRep,4,2) || '-' ||
  SUBSTR(dateRep, 1, 2));
```

4.3 EX16:

```
SELECT
  CountryInfo.countriesAndTerritories,
  CaseData.dateRep,
  CaseData.cases,
  CaseData.deaths
FROM
  CaseData
JOIN
  CountryInfo ON CaseData.countryterritoryCode = CountryInfo.countryterritoryCode
ORDER BY
  CountryInfo.countriesAndTerritories,strftime('%Y-%m-%d',SUBSTR(CaseData.dateRep, 7, 4)
  || '-' || SUBSTR(CaseData.dateRep, 4, 2)
  || '-' || SUBSTR(CaseData.dateRep, 1, 2));
```

4.4 EX17:

```
SELECT CountryInfo.countriesAndTerritories,
       ROUND((SUM(CaseData.cases) * 100.0) / CountryInfo.popData2020, 2)  AS CasesPercentage,
       ROUND((SUM(CaseData.deaths) * 100.0) / CountryInfo.popData2020, 2) AS DeathsPercentage

FROM CaseData
         JOIN
     CountryInfo ON CaseData.countryterritoryCode = CountryInfo.countryterritoryCode
GROUP BY
    CountryInfo.countriesAndTerritories
```

4.5 EX18:

```
SELECT
  CountryInfo.countriesAndTerritories,
  ROUND((SUM(CaseData.deaths) * 100.0 / SUM(CaseData.cases)), 2) AS DeathPercentage
FROM
  CaseData
JOIN
  CountryInfo ON CaseData.countryterritoryCode = CountryInfo.countryterritoryCode
GROUP BY
  CountryInfo.countriesAndTerritories
ORDER BY
  DeathPercentage DESC
LIMIT
  10;
```

4.6 EX19:

```
SELECT
  dateRep,
  SUM(cases) OVER (ORDER BY strftime('%Y-%m-%d', SUBSTR(dateRep, 7, 4) || '-' || SUBSTR(dateRep, 4, 2)
  || '-' || SUBSTR(dateRep, 1, 2))) AS CumulativeCases,
  SUM(deaths) OVER (ORDER BY strftime('%Y-%m-%d', SUBSTR(dateRep, 7, 4)|| '-' || SUBSTR(dateRep, 4, 2)
  SUBSTR(dateRep, 1, 2))) AS CumulativeDeaths
FROM
  CaseData
WHERE
  countryterritoryCode = 'GBR'
ORDER BY
  strftime('%Y-%m-%d', SUBSTR(dateRep, 7, 4) || '-' ||
  SUBSTR(dateRep, 4, 2) || '-' || SUBSTR(dateRep, 1, 2));
```