DataCourseWork2

Abdullah Haitham Maghrabifetaih
35160055

April 2024

1 The Relational Model

1.1 EX1

Table 1: Relational model attributes

Column 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

- Date attributes day, month, year are always non-null for accurate date tracking.
- Population data popData2020 isn't used as a key due to potential overlap across countries.
- Each unique identifier, including geoId and countryTerritoryCode, corresponds to one specific geographical area.
- Cases and deaths are recorded as numerical values, or may not be reported null.
- The countriesAndTerritories field consistently lists countries, excluding territories.
- Some countries span multiple continents, which is represented by the continentExp attribute. below are the FDs based of the Assumptions

- geoId → countriesAndTerritories
- geoId → countryTerritoryCode
- geoId → popData2020
- geoId → continentExp
- countryTerritoryCode → countriesAndTerritories
- countryTerritoryCode → geoId
- countryTerritoryCode → popData2020
- countryTerritoryCode → continentExp
- {dateRep, geoId} → cases
- {dateRep, geoId} → deaths
- {dateRep, countryTerritoryCode} → cases
- {dateRep, countryTerritoryCode} → deaths

1.3 EX3

Following is a list of possible candidate keys for the dataset based on functional dependencies:

Table 2: Candidate Keys

Candidate Key	**Justification**
geoId	Uniquely determines multiple attributes
countryTerritoryCode	Interchangeable with geoId, determines the same attributes
dateRep, geoId	Unique daily record for cases and deaths
dateRep, countryTerritoryCode	Equivalent to dateRep, geoId for daily records

1.4 EX4

The primary goal in choosing our dataset's key was to effectively differentiate each record. Emphasis was placed on system compatibility and ease of use. We selected `dateRep` as a key attribute because it consolidates date information into one field, simplifying data management and reducing complexity with external systems. This uniformity is crucial for ensuring consistent data integration across different systems. For geographical identification, `geoId` was chosen over `countriesAndTerritories` or `countryTerritoryCode` due to its short, widely recognized format. This choice helps reduce data processing errors and enhances integration with geographic information systems.

The combination of `dateRep` and `geoId` as our primary key effectively tracks pandemic events, ensuring each entry is uniquely identified by its date and location, thus maximizing data retrieval and accuracy.

Table 3: Chosen Primary Key for the Dataset

Primary Key Configuration
$dateRep, geoId$

2 Normalisation

2.1 EX5

Partial-Key Dependencies

- `dateRep` → `day, month, year`
- `geoId` → `countryterritoryCode,countriesAndterritories, popData2020, continentExp`

Decomposition into Relations

Decomposing the original relation into the following smaller relations is our suggested way to attain Second Normal Form (2NF) and get rid of partial-key dependencies:

Relation 1: Date Information

Primary Key: `dateRep`
Attributes: Includes `day`, `month`, `year`
The elements of the reporting date are stored in this relation, with each component defined solely by dataRep and

Relation 2: Country Information

Primary Key: `geoId`
Attributes: Includes `countriesAndTerritories`, `countryTerritoryCode`, `popData2020`, `continentExp`
This connection records information about the geographical identity of the data, whereby every attribute depends entirely on geoId in order to function.

Core Relation: Event Data

Composite Primary Key: (`dateRep`, `geoId`)
Attributes: Includes `cases`, `deaths`
By connecting comprehensive cases and death reports with the reporting date and the geographic identifier, this key connection preserves crucial event data.

2.2 EX6

Date Information:

- **Primary Key:** `dateRep`
- **Attributes:** `day`, `month`, `year` (INTEGER)

Country Information:

- **Primary Key:** `geoId`
- **Attributes:** `countriesAndTerritories`, `countryTerritoryCode` (TEXT); `popData2020` (INTEGER), `continentExp` (TEXT)

Core Event Data:

- **Composite Primary Key:** (`dateRep`, `geoId`)
- **Attributes:** `cases`, `deaths` (INTEGER)

These relations ensure no partial dependencies, maintaining data integrity and query efficiency.

2.3 EX7

In our newly formed relations aiming for Second Normal Form, we check for transitive dependencies, where an attribute indirectly relies on the primary key through another attribute.

Analysis:

- **Date Information:** No transitive dependencies exist. All attributes (`day`, `month`, `year`) directly rely on the primary key `dateRep`.
- **Country Information:** No transitive dependencies are observed. Attributes (`countriesAndTerritories`, `countryTerritoryCode`, `popData2020`, `continentExp`) all directly depend on `geoId`.
- **Core Event Data:** Similarly, no transitive dependencies are found. Both `cases` and `deaths` directly depend on the composite primary key (`dateRep`, `geoId`).

2.4 EX8

Checking for transitive dependencies:

Date Information: All attributes directly depend on the primary key `dateRep`, so no transitive dependencies exist.

Country Information: Similarly, all attributes directly depend on the primary key `geoId`, avoiding transitive dependencies.

Core Event Data: Both `cases` and `deaths` directly depend on the composite primary key (`dateRep`, `geoId`), eliminating transitive dependencies.
Since all relations are already in 3NF, no further decomposition is needed, ensuring data integrity and minimizing redundancy.

2.5 EX9

Already, our relationships are in BCNF. There is a primary key for each relation, and a candidate key for each determinant. Between candidate keys, there are no significant functional dependencies. Therefore, in order to accomplish BCNF, additional normalization is not required.

3 Modelling

3.1 EX10

Command 1: Open terminal.

```
sqlite3 coronavirus.db
```

Command 2: Set CSV mode.

```
.mode csv
```

Command 3: Create dataset table.

```
CREATE TABLE dataset (
    dateRep TEXT NOT NULL,
    day INTEGER NOT NULL,
    month INTEGER NOT NULL,
    year INTEGER NOT NULL,
```

```
    cases INTEGER NOT NULL,
    deaths INTEGER NOT NULL,
    countriesAndTerritories TEXT NOT NULL,
    geoId TEXT NOT NULL,
    countryTerritoryCode TEXT NOT NULL,
    popData2020 INTEGER NOT NULL,
    continentExp TEXT NOT NULL
);
```

Command 4: Import CSV data.

```
.import  dataset.csv dataset
```

Command 5: Set output file for SQL dump.

```
.output dataset.sql
```

Command 6: Dump database to file.

```
.dump
```

Command 7: Exit SQLite.

```
.exit
```

3.2 EX11

```
CREATE TABLE IF NOT EXISTS DateInformation (
    dateRep TEXT PRIMARY KEY,
    day INTEGER NOT NULL,
    month INTEGER NOT NULL,
    year INTEGER NOT NULL
);
CREATE TABLE IF NOT EXISTS CountryInformation (
    geoId TEXT PRIMARY KEY,
    countryTerritoryCode TEXT NOT NULL,
    countriesAndTerritories TEXT NOT NULL,
    popData2020 INTEGER NOT NULL,
    continentExp TEXT NOT NULL
);
CREATE TABLE IF NOT EXISTS EventData (
    dateRep TEXT,
    geoId TEXT,
    cases INTEGER NOT NULL,
    deaths INTEGER NOT NULL,
    PRIMARY KEY (dateRep, geoId),
    FOREIGN KEY (dateRep) REFERENCES DateInformation(dateRep),
    FOREIGN KEY (geoId) REFERENCES CountryInformation(geoId)
);
```

3.3 EX12

```

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

```
5
6 INSERT INTO CountryInformation (geoId, countryterritoryCode, continentExp,
      countriesAndTerritories, popData2020)
7 SELECT DISTINCT geoId, countryterritoryCode, continentExp,
    countriesAndTerritories, popData2020
8 FROM dataset;
9
10 INSERT INTO EventData (dateRep, geoId, cases, deaths)
11 SELECT DISTINCT dateRep, geoId, cases, deaths
12 FROM dataset;
```

3.4 EX13

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

The above commands have successfully worked

4 Querying

4.1 EX14

```
1 SELECT sum(cases) as totalcasese,
2        sum(deaths) as totaldeaths
3 From EventData;
```

4.2 EX15

```
1 SELECT dateRep,cases
2 FROM EventData
3 NATURAL JOIN DateInformation
4 WHERE geoId = 'UK'
5 ORDER BY year,month,day asc ;
```

4.3 EX16

```
1 SELECT CountryInformation.countriesAndTerritories,daterep,cases,deaths
2 FROM EventData
3 NATURAL JOIN CountryInformation
4 NATURAL JOIN  DateInformation
5 ORDER BY countriesAndTerritories asc;
```

4.4 EX17

```
1 SELECT
2     c.countriesAndTerritories AS CountryName,
3     ROUND(SUM(d.cases) * 100.0 / c.popData2020, 2) AS
    PercentCasesOfPopulation,
```

```
    ROUND(SUM(d.deaths) * 100.0 / c.popData2020, 2) AS
   PercentDeathsOfPopulation
FROM
    dataset d
JOIN
    CountryInformation c ON d.geoId = c.geoId
GROUP BY
    c.countriesAndTerritories, c.popData2020
ORDER BY
    c.countriesAndTerritories;
```

4.5 EX18

```
SELECT
    c.countriesAndTerritories AS CountryName,
    ROUND(SUM(d.deaths) * 100.0 / SUM(d.cases), 2) AS `Percent Deaths of
   Country Cases`
FROM
    dataset d
JOIN
    CountryInformation c ON d.geoId = c.geoId
GROUP BY
    c.countriesAndTerritories
HAVING
    SUM(d.cases) > 0
ORDER BY
    `Percent Deaths of Country Cases` DESC
LIMIT 10;
```

4.6 EX19

```
SELECT
    dateRep AS Date,
    SUM(deaths) OVER (ROWS UNBOUNDED PRECEDING) AS `Cumulative UK Deaths`,
    SUM(cases) OVER (ROWS UNBOUNDED PRECEDING) AS `Cumulative UK Cases`
FROM
    EventData
WHERE
    geoId = 'UK'
ORDER BY
    date(Date) ASC ;
```