# COMP1204: Data Management
# Coursework Two

Jincheng Guo
jg16n22

May 2023

# 1 The Relational Model

## 1.1 EX1

**Relation:**
dataset(dateRep, day, month, year, cases, deaths, countriesAndTerritories, geoId, countryterritoryCode, popData2020, continentExp)

**Attributes:**

1. dateRep: TEXT
2. day: INTEGER
3. month: INTEGER
4. year: INTEGER
5. cases: INTEGER
6. deaths: INTEGER
7. countriesAndTerritories: TEXT
8. geoId: TEXT
9. countryterritoryCode: TEXT
10. popData2020: INTEGER
11. continentExp: TEXT

## 1.2 EX2

**Minimal set of FDs:**

1. dateRep → day
2. dateRep → month
3. dateRep → year
4. (day, month, year) → dateRep
5. geoId → countriesAndTerritories
6. geoId → countryterritoryCode
7. countriesAndTerritories → geoId
8. countryterritoryCode → geoId
9. geoId → continentExp

10. geoId → popData2020
11. popData2020 → geoId
12. (dateRep, geoId) → deaths
13. (dateRep, geoId) → cases

## 1.3 EX3

**Potential candidate keys:**

1. dateRep, geoId
2. dateRep, countriesAndTerritories
3. dateRep, countryterritoryCode

## 1.4 EX4

**Suitable primary key:**
dateRep, geoId

**Reasons:**

1. geoId is shorter and looks much more simple
2. easy to identify

# 2 Normalisation

## 2.1 EX5

**Partial-key dependencies:**

1. dateRep → day
2. dateRep → month
3. dateRep → year
4. geoId → countriesAndTerritories
5. geoId → countryterritoryCode
6. geoId → continentExp
7. geoId → popData2020

**Decomposition:**

1. dateRep → (day, month, year)
2. geoId → (countriesAndTerritories, continentExp, popData2020)
3. (dateRep, geoId) → (deaths, cases)

## 2.2 EX6

**New relations:**

1. date(dateRep, day, month, year)
2. country(geoId, countriesAndTerritories, countryterritoryCode, popData2020, continentExp)
3. caseAndDeath(dateRep, geoId, cases, deaths)

**Primary keys:**

1. date - dateRep
2. country - geoId
3. caseAndDeath - (dateRap, geoId)

## 2.3 EX7

There is no transitive dependencies

## 2.4 EX8

Same as in the 2NF

### 2.5 EX9

It is BCNF
dateRep → ...
geoId → ...
(dateRep, geoId) → ...
Every determinant is a candidate key

# 3 Modelling

## 3.1 EX10

```
sqlite3 coronavirus.db
.open coronavirus.db // open the database without table
```

than open coronavirus.db in DataDrip, import dataset.csv and back to cmd

```
sqlite3
.open coronavirus.db
.output dataset.sql // creat dataset.sql
.dump dataset > dataset.sql // dump the dataset table to dataset.sql
```

## 3.2 EX11

```
create table caseAndDeath
(
    dateRep TEXT not null,
    geoId   TEXT not null,
    deaths  integer,
    cases   integer,
    constraint caseAndDeath_pk
        primary key (dateRep, geoId)
);

create table country
(
    geoId                   TEXT    not null
        constraint country_pk
            primary key,
    countriesAndTerritories TEXT    not null,
    countryterritoryCode    TEXT    not null,
    popData2020             INTEGER not null,
    continentExp            TEXT    not null
);

create table date
(
    dateRep TEXT    not null
        constraint date_pk
            primary key,
    day     integer not null,
    month   integer not null,
    year    integer not null
);
```

### 3.3 EX12

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

INSERT INTO Country (geoId, countryterritoryCode, continentExp,
countriesAndTerritories, popData2020)
SELECT DISTINCT geoId, countryterritoryCode, continentExp,
countriesAndTerritories, popData2020
FROM dataset;

INSERT INTO caseAndDeath (dateRep, geoId, cases, deaths)
SELECT DISTINCT dateRep, geoId, cases, deaths
FROM dataset;
```

### 3.4 EX13

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

The commands could be able to ran.

# 4 Querying

## 4.1 EX14

```
ALTER TABLE caseAndDeath ADD totalDeaths integer;
UPDATE caseAndDeath
SET totalDeaths = (SELECT SUM(deaths) FROM caseAndDeath);

ALTER TABLE caseAndDeath ADD totalCases integer;
UPDATE caseAndDeath
SET totalCases = (SELECT SUM(cases) FROM caseAndDeath);
```

## 4.2 EX15

```
SELECT caseAndDeath.dateRep, cases
FROM caseAndDeath
INNER JOIN Date
ON Date.dateRep = caseAndDeath.dateRep
WHERE caseAndDeath.geoId = 'UK'
ORDER BY year,month,day ASC;
```

## 4.3 EX16

```
SELECT country.countriesAndTerritories AS country,
caseAndDeath.dateRep AS date,
caseAndDeath.cases,
caseAndDeath.deaths
FROM caseAndDeath
JOIN date
ON date.dateRep = caseAndDeath.dateRep
JOIN country
ON caseAndDeath.geoId = country.geoId
GROUP BY country, date
ORDER BY caseAndDeath.geoId ASC, year, month, day ASC;
```

## 4.4 EX17

```
SELECT country.countriesAndTerritories AS country,
    round(sum(caseAndDeath.cases)* 1.0 / country.popData2020 * 100, 2)
    AS casesPercentage,
    round(sum(caseAndDeath.deaths)* 1.0 / country.popData2020 * 100, 2)
    AS deathsPercentage
FROM caseAndDeath
JOIN country ON caseAndDeath.geoId = country.geoId
GROUP BY countriesAndTerritories, popData2020
```

### 4.5 EX18

```
SELECT
    country.countriesAndTerritories AS country,
    round((SUM(caseAndDeath.deaths) * 1.0 / SUM(caseAndDeath.cases)) * 100, 2) A
FROM caseAndDeath
JOIN country ON caseAndDeath.geoId = country.geoId
GROUP BY country.countriesAndTerritories
ORDER BY deathRate DESC
LIMIT 10;
```

### 4.6 EX19

```
SELECT caseAndDeath.dateRep AS date,
    SUM(cases) OVER (PARTITION BY geoid ORDER BY caseAndDeath.dateRep ASC)
    AS cases1,
    SUM(deaths) OVER (PARTITION BY geoid ORDER BY caseAndDeath.dateRep ASC)
    AS deaths1
FROM caseAndDeath
JOIN date
ON date.dateRep = caseAndDeath.dateRep
WHERE geoId = 'UK'
ORDER BY year, month, day ASC;
```