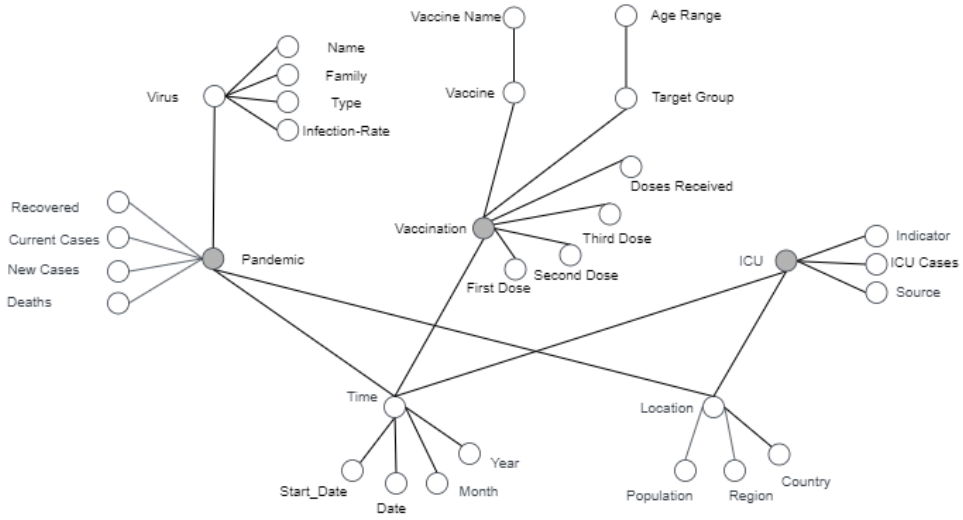
1.i - Functional dependencies:

- V-ID → Name, Family, Type, InfectionRate
- L-ID → Country, Region. Population
- T-ID → Year, Month, Date, Start-Date
- L-ID, T-ID, V-ID → Current-Cases, New-Cases, Recovered, Deaths
- Vac-ID → Name
- Trg-ID → Age-Range
- Vac-ID, T-ID, L-ID, Trg-ID → Doses-Received, First-Dose, Second-Dose, Third-Dose
- L-ID, T-ID→ Indicator, Source, ICU-Cases

1.ii – Considering the provided relational schema and extracted functional dependencies, the initial attribute tree will be as the diagram below.



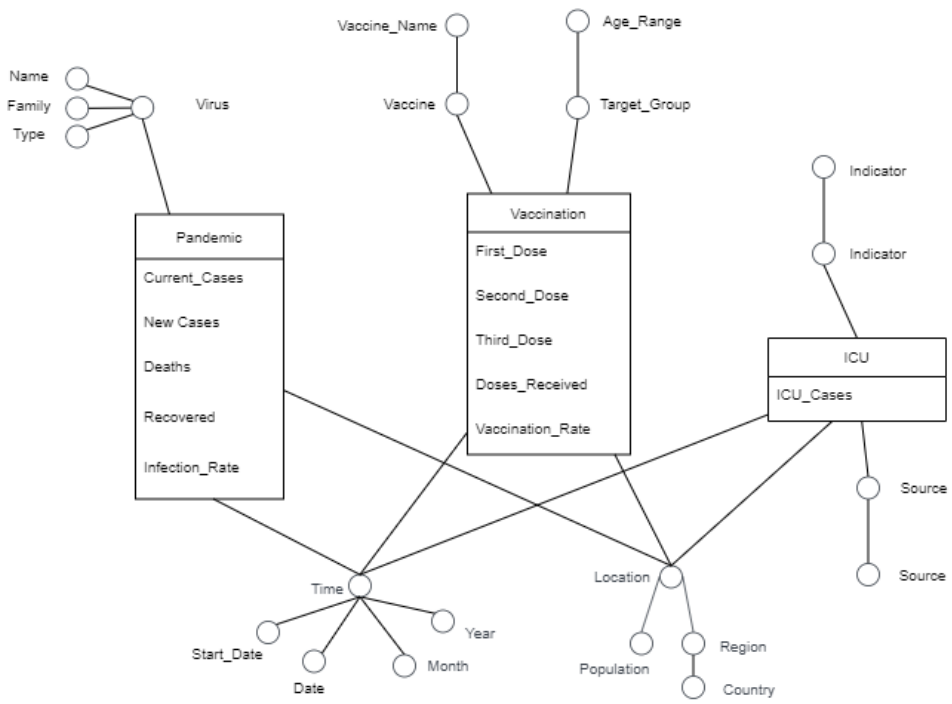
1.iii – For building a fact schema, the initial attribute tree must be pruned and grafted if necessary. Also facts, dimensions and measures must be defined. These are the changes that were applied on the tree for building the fact schema.

- Country becomes a child of Region.
- Source and Indicator which were child of the ICU node become a dimension because of the count of values they take and therefore are considered dimension rather than measures for the ICU Fact.
- Infection_Rate was moved to become a child of Pandemic and is considered a measure.
- New node is calculated for the fact table and named Vaccination_Rate.

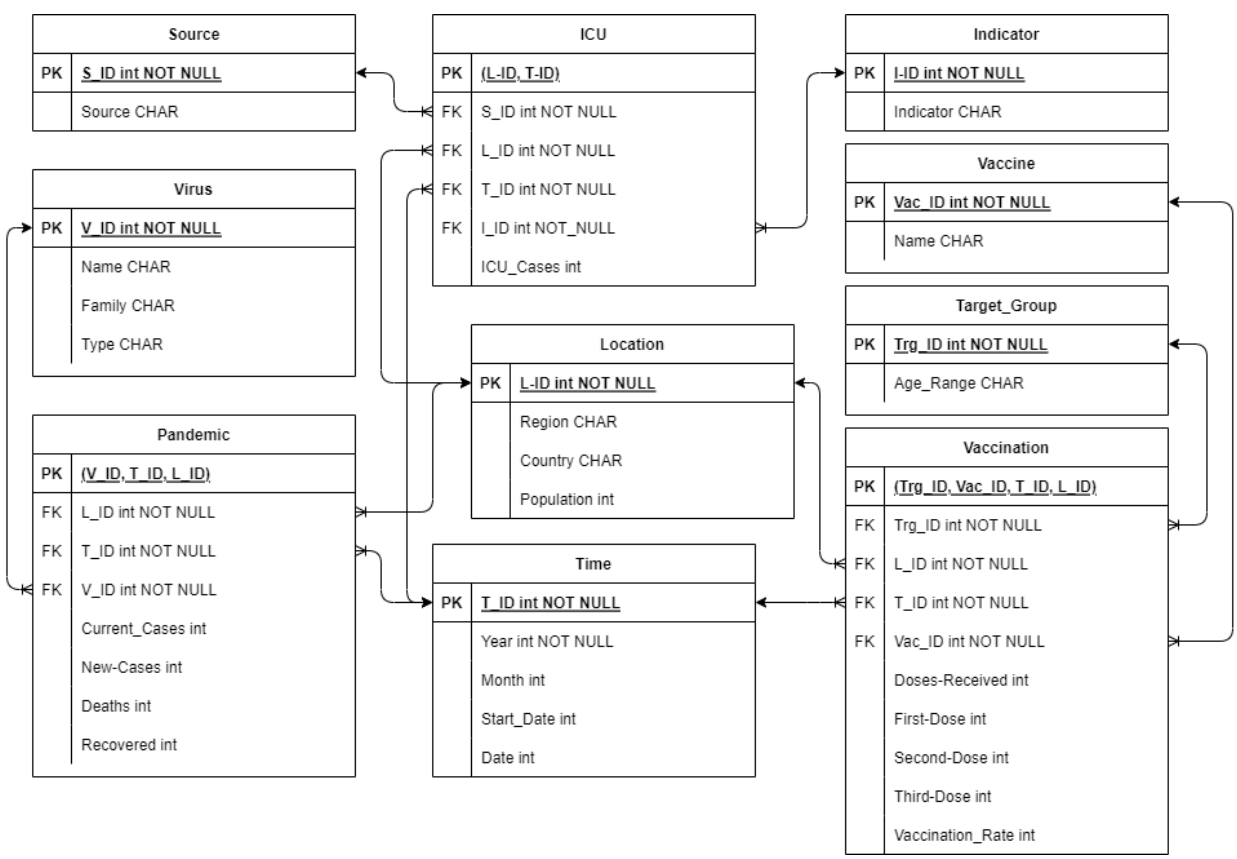
The result is a Constellation Fact Schema. Its Facts, Measures and Dimensions are defined below.

- Facts: Pandemic, ICU and Vaccination
- Dimensions: Time, Location, Source, Indicator, Vaccine, Target_Group, Virus

- Measures: Current Cases, New Cases, Deaths, Recovered, ICU Cases, Doses Received, First Dose, Second Dose, Third Dose



2 – The fact schema turns into the logical model below.



3 – For creating the warehouse schema following DDL statements will be used.

Location

```
CREATE TABLE IF NOT EXISTS Location (
L_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Region TEXT,
Country TEXT,
Population INTEGER
);
```

Time

```
CREATE TABLE IF NOT EXISTS Time (
T_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Year INTEGER,
Month INTEGER,
Date INTEGER,
Start_Date INTEGER
);
```

Vaccine

```
CREATE TABLE IF NOT EXISTS Vaccine (
Vac_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Name TEXT
);
```

Source

```
CREATE TABLE IF NOT EXISTS Source (
S_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Source TEXT
);
```

Indicator

```
CREATE TABLE IF NOT EXISTS Indicator (
I_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Indicator TEXT
);
```

Taget_Group

```
CREATE TABLE IF NOT EXISTS Target_Group (
Trg_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Age_Range TEXT
);
```

Virus

```
CREATE TABLE IF NOT EXISTS Virus (
V_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
Name TEXT,
Family TEXT,
Type TEXT
);
```

ICU

```
CREATE TABLE IF NOT EXISTS ICU (
L_ID integer  NOT NULL,
T_ID integer NOT NULL,
I_ID int NOT NULL,
S_ID int NOT NULL ,
IcuCases int,
PRIMARY KEY(L_ID, T_ID, I_ID, S_ID),
FOREIGN KEY (L_ID) REFERENCES Location (L_ID),
FOREIGN KEY (T_ID) REFERENCES Time (T_ID),
FOREIGN KEY (S_ID) REFERENCES Source (S_ID),
FOREIGN KEY (I_ID) REFERENCES Indicator (I_ID)
);
```

Vaccination

```
CREATE TABLE IF NOT EXISTS Vaccination (
L_ID integer  NOT NULL,
T_ID integer NOT NULL,
Vac_ID integer NOT NULL,
Trg_ID integer NOT NULL,
DosesReceived int,
FirstDose int,
SecondDose int,
ThirdDose int,
Vaccination_Rate int,
PRIMARY KEY(L_ID, T_ID, Vac_ID, Trg_ID),
FOREIGN KEY (L_ID) REFERENCES Location (L_ID),
FOREIGN KEY (T_ID) REFERENCES Time (T_ID),
FOREIGN KEY (Vac_ID) REFERENCES Vaccine (Vac_ID),
FOREIGN KEY (Trg_ID) REFERENCES Target_Group (Trg_ID)
);
```

Pandemic

```
CREATE TABLE IF NOT EXISTS Pandemic (
L_ID integer  NOT NULL,
T_ID integer NOT NULL,
V_ID integer NOT NULL,
CurrentCases int,
NewCases int,
Deaths int,
Recovered int,
PRIMARY KEY(L_ID, T_ID, V_ID),
FOREIGN KEY (L_ID) REFERENCES Location (L_ID),
FOREIGN KEY (T_ID) REFERENCES Time (T_ID),
FOREIGN KEY (V_ID) REFERENCES Virus (V_ID)
);
```

4 - The data warehouse design comprises three fact tables - ICU, Vaccination, and Pandemic. To efficiently answer queries that seek information by location and time, we can group the data based on the smallest categories specified in the queries. For instance, we can group the data by region in the location dimension and by month in the time dimension. Additionally, to satisfy the queries, some fact tables may require further summation and grouping, which can be achieved through materialized views.

Materialized View Pandemic:

CREATE VIEW Pandemic_MV AS
SELECT
SUM(A.NewCases) AS Cases,
SUM(A.Deaths) AS Deaths,
B.Month,
B.Year,
C.Region,
C.Country

FROM Pandemic A, Time B, Location C, Virus D
WHERE A.T_ID = B.T_ID AND A.L_ID = C.L_ID AND A.V_ID = D.V_ID
GROUP BY B.Month,C.Region;

Query History

```
CREATE VIEW Pandemic_MV AS
SELECT
    SUM(A.NewCases) AS Cases,
    SUM(A.Deaths) AS Deaths,
    B.Month,
    B.Year,
    C.Region,
    C.Country

FROM Pandemic A, Time B, Location C, Virus D
WHERE A.T_ID = B.T_ID AND A.L_ID = C.L_ID AND A.V_ID = D.V_ID
GROUP BY B.Month,C.Region;
```

Materialized View ICU:

CREATE VIEW ICU_VM AS
SELECT
SUM(A.I_ID) AS Cases,
SUM(A.Deaths) AS Deaths,
B.Month,
B.Year,
C.Region,
C.Country

FROM ICU A, Time B, Location C, Indicator I, Source S

WHERE
A.T_ID = B.T_ID AND
A.L_ID = C.L_ID AND
A.I_ID = I.I_ID AND
A.S_ID = S.S_ID

GROUP BY B.Month,C.Region, A.S_ID;

```sql
CREATE VIEW ICU_VM AS
SELECT
    SUM(A.I_ID) AS Cases,
    SUM(A.Deaths) AS Deaths,
    B.Month,
    B.Year,
    C.Region,
    C.Country

FROM ICU A, Time B, Location C, Indicator I, Source S

WHERE A.T_ID = B.T_ID AND
    A.L_ID = C.L_ID AND
    A.I_ID = I.I_ID AND
    A.S_ID = S.S_ID

GROUP BY B.Month, C.Region, A.S_ID;
```

Materialized view Vacc:

CREATE VIEW Vacc_VM AS
SELECT
B.Month,
B.Year,
C.Region,
C.Country,
V.Name,
A.VaccinationRate

FROM Vaccination A, Time B, Location C, Vaccine V, Target_Group T

WHERE
A.T_ID = B.T_ID AND
A.L_ID = C.L_ID AND
A.Vac_ID = V.Vac_ID AND
A.Trg_ID = T.Trg_ID
GROUP BY B.Month, C.Region, V.Name;

Query History

```sql
CREATE VIEW Vacc_VM AS
SELECT
B.Month,
B.Year,
C.Region,
C.Country,
V.Name,
A.VaccinationRate

FROM Vaccination A, Time B, Location C, Vaccine V, Target_Group T

WHERE
A.T_ID = B.T_ID AND
A.L_ID = C.L_ID AND
A.Vac_ID = V.Vac_ID AND
A.Trg_ID = T.Trg_ID
GROUP BY B.Month, C.Region, V.Name;
```

5 -

First Query:

SELECT Region, Month, Cases, Deaths
FROM Pandemic_VM;

SELECT Region, Month, IcuCases, Source
FROM ICU_VM;

SELECT Region, Month, Vaccine, VaccinationRate
FROM Vacc_VM;

```
Query History
SELECT Region, Month, Cases, Deaths
FROM Pandemic_VM;

SELECT Region, Month, IcuCases, Source
FROM ICU_VM;

SELECT Region, Month, Vaccine, VaccinationRate
FROM Vacc_VM;
```

Second Query:

SELECT Country, Month/3 +1 AS Quarter, Cases, Deaths
FROM Pandemic_VM
GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, IcuCases
FROM ICU_VM
GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, Vaccine, VaccinationRate
FROM Vacc_VM
GROUP BY Country, Quarter;

```
Corona_DWH
Query History
SELECT Country, Month/3 +1 AS Quarter, Cases, Deaths
FROM Pandemic_VM
GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, IcuCases
FROM ICU_VM
GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, Vaccine, VaccinationRate
FROM Vacc_VM
GROUP BY Country, Quarter;
```