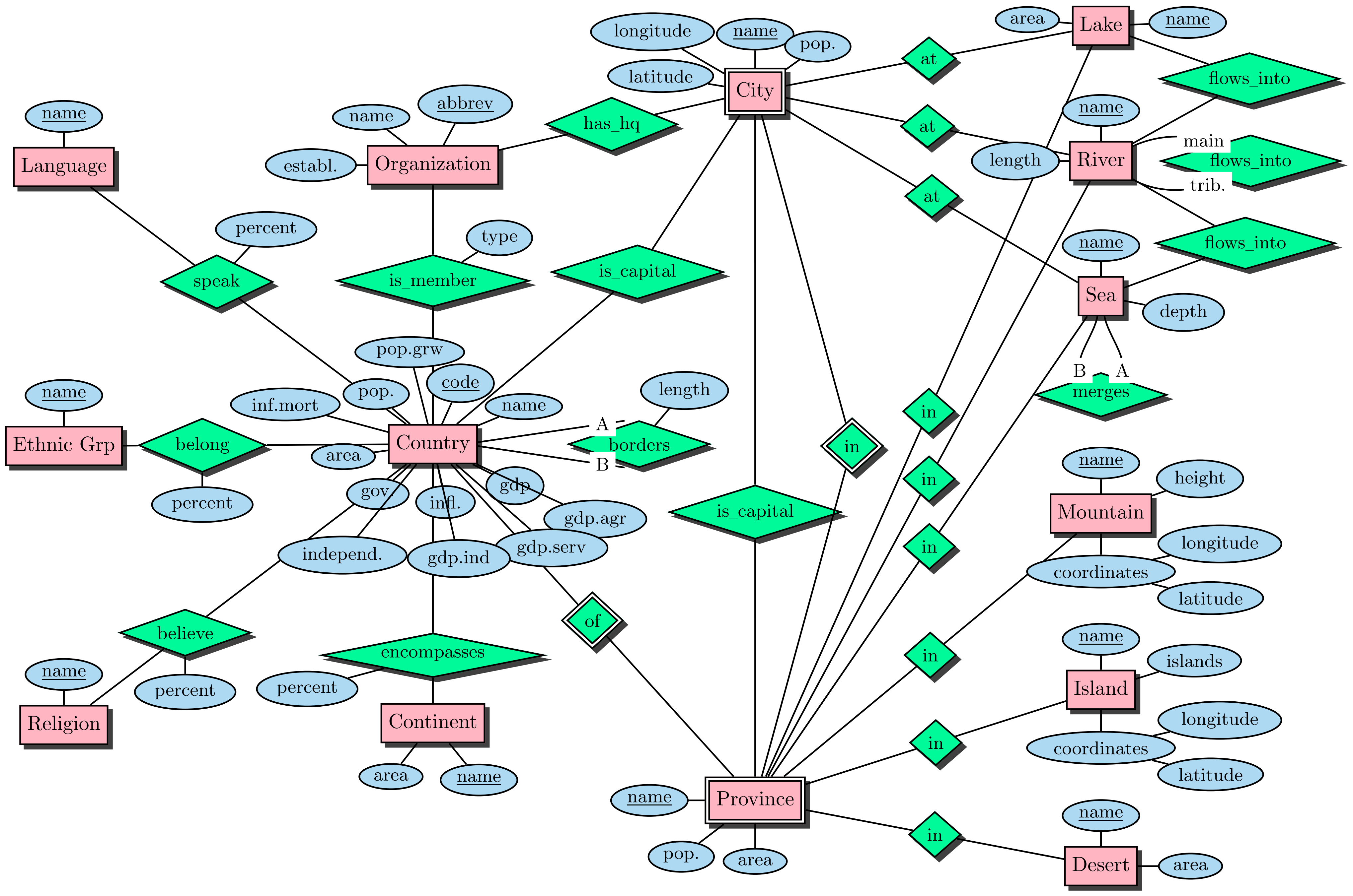
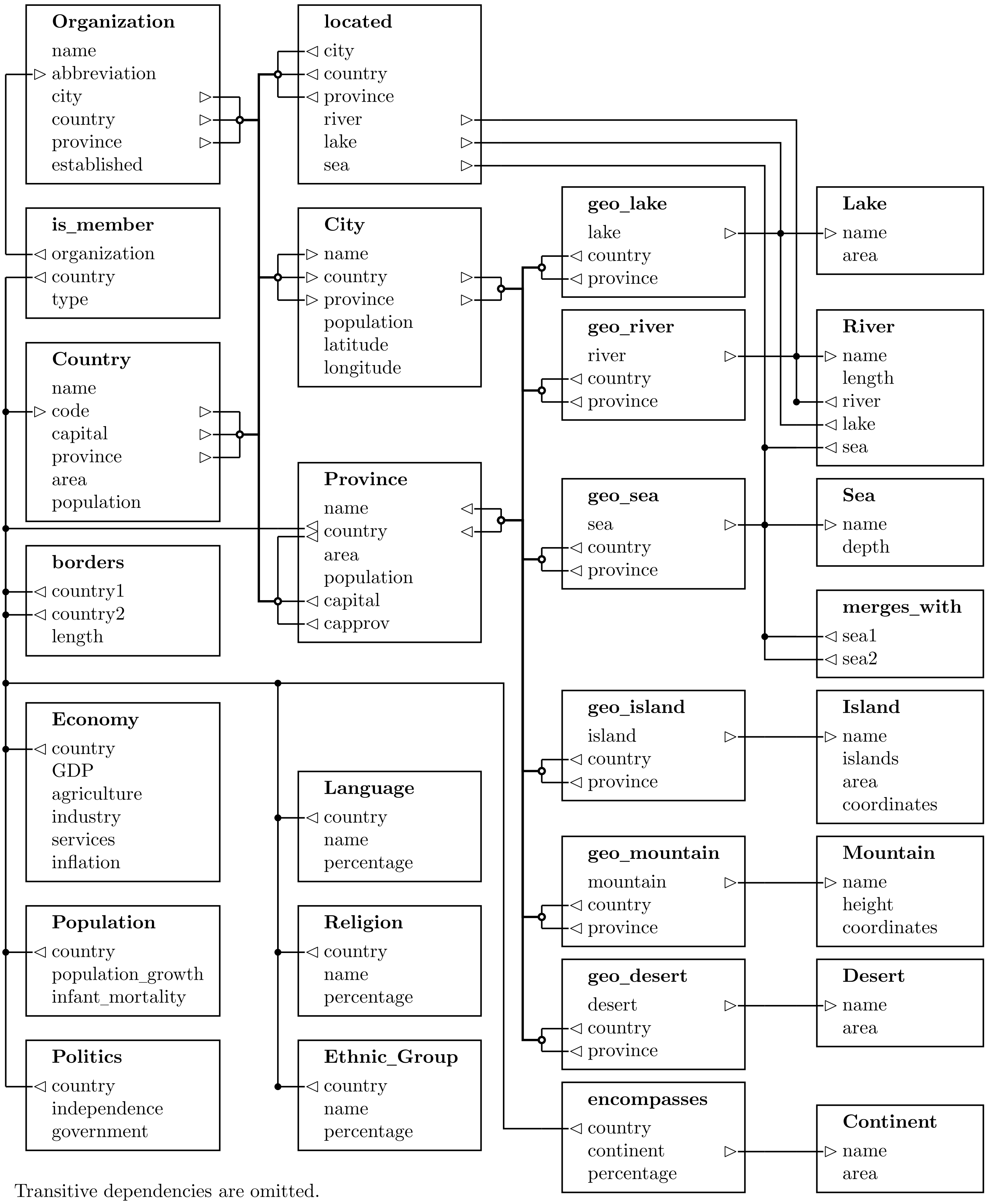
Entity Relationship Diagram of MONDIAL Database



Referential Dependencies of Mondial Database



Transitive dependencies are omitted.

Relational Schema Description of Mondial Database

Types

GeoCoord: geographic position.

Latitude:	geographic latitude
Longitude:	geographic longitude

Tables

Country: the countries of the world with some data.

Name:	the country name
Code:	the internet country code (two letters)
Capital:	the name of the capital
Province:	the province where the capital belongs to
Population:	the population number
Area:	the total area

Province: information about administrative divisions.

Name:	the name of the administrative division
Country:	the country code where it belongs to
Population:	the population of the province
Area:	the total area of the province
Capital:	the name of the capital
CapProv:	the name of the province where the captital belongs to

⇒ Note that *capprov* is not necessarily equal to *name*. For example, the municipality of *Bogota (Columbia)* is a province of its own, and *Bogota* is the capital of the surrounding province *Cudinamarca*.

City: information about cities.

Name:	the name of the city
Country:	the country code where it belongs to
Province:	the name of the province where it belongs to
Population:	population of the city
Latitude:	geographic latitude
Longitude:	geographic longitude

Continent: information about continents.

Name:	name of the continent
Area:	total area of the continent

encompasses: information to which continents a country belongs.

Country:	the country code
Continent:	the continent name
Percentage:	percentage, how much of the area of a country belongs to the continent

borders: informations about neighboring countries.

Country1:	a country code
Country2:	a country code
Length:	length of the border between country1 and country2

⇒ Note that in this relation, for every pair of neighboring countries (A, B), only one tuple is given – thus, the relation is *not* symmetric.

Organization: information about political and economical organizations.

Abbreviation:	the abbreviation of the organization
Name:	the full name of the organization
Established:	date of establishment
City:	the city where it is seated
Province:	the province of its seat
Country:	the country code of its seat

is_member: memberships in political and economical organizations.

Organization:	the abbreviation of the organization
Country:	the code of the member country
Type:	the type of membership

Economy: economical information about the countries.

Country:	the country code
GDP:	gross domestic product (in million dollar)
Agriculture:	percentage of agricultural sector of the GDP
Industry:	percentage of industrial sector of the GDP
Services:	percentage of service sector of the GDP
Inflation:	inflation rate (percentage, per annum)

Population: information about the population of the countries.

Country:	the country code
Population_Growth:	population growth rate (percentage, per annum)
Infant_Mortality:	infant mortality (per thousand)

Politics: political information about the countries.

Country:	the country code
Independence:	date of independence
Government:	type of government

Language: information about the languages spoken in a country.

Country:	the country code
Name:	name of the language
Percentage:	percentage of the language in this country

Religion: information about the religions in a country.

Country:	the country code
Name:	name of the religion
Percentage:	percentage of the religion in this country

Ethnic_Group: information about the ethnic groups in a country.

Country:	the country code
Name:	name of the ethnic group
Percentage:	percentage of the ethnic group in this country

located: information about cities located at rivers, lakes, and seas.

City:	the name of the city
Province:	the province where the city belongs to
Country:	the country code where the city belongs to
River:	the river where it is located at
Lake:	the lake where it is located at
Sea:	the sea where it is located at

$\Rightarrow$ Note that for a given city, there can be several lakes/seas/rivers where it is located at.

River: information about rivers.

Name:	the name of the river
River:	the river where it flows to
Lake:	the lake where it flows to
Sea:	the sea where it flows to
Length:	the length of the river

Mountain: information about mountains.

Name:	the name of the mountain
Height:	the height of the mountain
Coordinates:	its geographical coordinates as (longitude, latitude)

Lake: information about lakes.

Name:	the name of the lake
Area:	the total area of the lake

Sea: information about seas.

Name:	the name of the sea
Depth:	the maximal depth of the sea

Island: information about islands.

Name:	the name of the island
Islands:	the group of the islands where it belongs to
Area:	the total area of the island
Coordinates:	its geographical coordinates as (longitude, latitude)

Desert: information about deserts.

Name:	the name of the desert
Area:	the total area of the desert

geo_river: geographical information about rivers.

River:	the name of the river
Country:	the country code where it is located
Province:	the province of this country

geo_mountain: geographical information about mountains.

Mountain:	the name of the mountain
Country:	the country code where it is located
Province:	the province of this country

geo_lake: geographical information about lakes.

Lake:	the name of the lake
Country:	the country code where it is located
Province:	the province of this country

geo_sea: geographical information about seas.

Sea:	the name of the sea
Country:	the country code where it is located
Province:	the province of this country

geo_island: geographical information about islands.

Island:	the name of the island
Country:	the country code where it is located
Province:	the province of this country

geo_desert: geographical information about deserts.

Desert:	the name of the desert
Country:	the country code where it is located
Province:	the province of this country

merges_with: information about neighboring seas.

Sea1:	a sea
Sea2:	a sea

Schema Definitions of Mondial Database

mondial-schema.sql:

```
CREATE OR REPLACE TYPE GeoCoord AS OBJECT (
    Latitude NUMBER,
    Longitude NUMBER
);
/

CREATE TABLE Country (
    Name VARCHAR2(40)
        CONSTRAINT Country_Name_NotNull NOT NULL
        CONSTRAINT Country_Name_Unique UNIQUE,
    Code CHAR(2)
        CONSTRAINT Country_Key PRIMARY KEY,
    Capital VARCHAR2(40),
    Province VARCHAR2(40),
    Population NUMBER
        CONSTRAINT Country_Population_Check CHECK (
             Population >= 0
        ),
    Area NUMBER
        CONSTRAINT Country_Area_Check CHECK (
             Area >= 0
        )
);

CREATE TABLE Province (
    Name VARCHAR2(40),
    Country CHAR(2),
    Population NUMBER
        CONSTRAINT Province_Population_Check CHECK (
             Population >= 0
        ),
    Area NUMBER
        CONSTRAINT Province_Area_Check CHECK (
             Area >= 0
        ),
    Capital VARCHAR2(40),
    CapProv VARCHAR2(40),
    CONSTRAINT Province_Key PRIMARY KEY (Country, Name)
);
```

```
CREATE TABLE City (
    Name VARCHAR2(40),
    Country CHAR(2),
    Province VARCHAR2(40),
    Population NUMBER
        CONSTRAINT City_Population_Check CHECK (
             Population >= 0
        ),
    Latitude NUMBER
        CONSTRAINT City_Latitude_Check CHECK (
             (Latitude >= -90) AND (Latitude <= 90)
        ),
    Longitude NUMBER
        CONSTRAINT City_Longitude_Check CHECK (
             (Longitude >= -180) AND (Longitude <= 180)
        ),
    CONSTRAINT City_Key PRIMARY KEY (Country, Province, Name)
);


CREATE TABLE Continent (
    Name VARCHAR2(20)
        CONSTRAINT Continent_Key PRIMARY KEY,
    Area NUMBER
        CONSTRAINT Continent_Area_Check CHECK (
             Area >= 0
        )
);


CREATE TABLE encompasses (
    Country CHAR(2),
    Continent VARCHAR2(20),
    Percentage NUMBER
        CONSTRAINT encompasses_Percentage_Check CHECK (
             (Percentage > 0) AND (Percentage <= 100)
        ),
    CONSTRAINT encompasses_Key PRIMARY KEY (Continent, Country)
);


CREATE TABLE borders (
    Country1 CHAR(2),
    Country2 CHAR(2),
    Length NUMBER
        CONSTRAINT borders_Length_Check CHECK (
             Length > 0
        ),
    CONSTRAINT borders_Key PRIMARY KEY (Country1, Country2)
);
```

```sql
CREATE TABLE Organization (
    Abbreviation VARCHAR2(15)
        CONSTRAINT Organization_Key PRIMARY KEY,
    Name VARCHAR2(100)
        CONSTRAINT Organization_Name_NotNull NOT NULL
        CONSTRAINT Organization_Name_Unique UNIQUE,
    Established DATE,
    City VARCHAR2(40),
    Province VARCHAR2(40),
    Country CHAR(2)
);


CREATE TABLE is_member (
    Organization VARCHAR2(15),
    Country CHAR(2),
    Type VARCHAR2(30),
    CONSTRAINT is_member_Key PRIMARY KEY (Country, Organization)
);


CREATE TABLE Economy (
    Country CHAR(2)
        CONSTRAINT Economy_Key PRIMARY KEY,
    GDP NUMBER
        CONSTRAINT Economy_GDP_Check CHECK (
             GDP >= 0
        ),
    Agriculture NUMBER,
    Industry NUMBER,
    Services NUMBER,
    Inflation NUMBER
);


CREATE TABLE Population (
    Country CHAR(2)
        CONSTRAINT Population_Key PRIMARY KEY,
    Population_Growth NUMBER,
    Infant_Mortality NUMBER
);


CREATE TABLE Politics (
    Country CHAR(2)
        CONSTRAINT Politics_Key PRIMARY KEY,
    Independence DATE,
    Government VARCHAR2(120)
);


CREATE TABLE Language (
    Country CHAR(2),
    Name VARCHAR2(50),
    Percentage NUMBER
        CONSTRAINT Language_Percentage_Check CHECK (
             (Percentage > 0) AND (Percentage <= 100)
        ),
    CONSTRAINT Language_Key PRIMARY KEY (Country, Name)
);
```

```
CREATE TABLE Religion (
    Country CHAR(2),
    Name VARCHAR2(50),
    Percentage NUMBER
        CONSTRAINT Religion_Percentage_Check CHECK (
             (Percentage > 0) AND (Percentage <= 100)
        ),
    CONSTRAINT Religion_Key PRIMARY KEY (Country, Name)
);


CREATE TABLE Ethnic_Group (
    Country CHAR(2),
    Name VARCHAR2(50),
    Percentage NUMBER
        CONSTRAINT Ethnic_Group_Percentage_Check CHECK (
             (Percentage > 0) AND (Percentage <= 100)
        ),
    CONSTRAINT Ethnic_Group_Key PRIMARY KEY (Country, Name)
);


CREATE TABLE located (
    City VARCHAR2(40)
        CONSTRAINT located_City_NotNull NOT NULL,
    Province VARCHAR2(40)
        CONSTRAINT located_Province_NotNull NOT NULL,
    Country CHAR(2)
        CONSTRAINT located_Country_NotNull NOT NULL,
    River VARCHAR2(30),
    Lake VARCHAR2(30),
    Sea VARCHAR2(30)
);


CREATE TABLE River (
    Name VARCHAR2(30)
        CONSTRAINT River_Key PRIMARY KEY,
    River VARCHAR2(30),
    Lake VARCHAR2(30),
    Sea VARCHAR2(30),
    Length NUMBER
        CONSTRAINT River_Length_Check CHECK (
             Length >= 0
        )
);
```

```
CREATE TABLE Mountain (
    Name VARCHAR2(30)
        CONSTRAINT Mountain_Key PRIMARY KEY,
    Height NUMBER
        CONSTRAINT Mountain_Height_Check CHECK (
             Height >= 0
        ),
    Coordinates GeoCoord
        CONSTRAINT Mountain_Coordinates_Check CHECK (
             (Coordinates.Longitude >= -180) AND
             (Coordinates.Longitude <= 180) AND
             (Coordinates.Latitude >= -90) AND
             (Coordinates.Latitude <= 90)
        )
);


CREATE TABLE Lake (
    Name VARCHAR2(30)
        CONSTRAINT Lake_Key PRIMARY KEY,
    Area NUMBER
        CONSTRAINT Lake_Area_Check CHECK (
             Area >= 0
        )
);


CREATE TABLE Sea (
    Name VARCHAR2(30)
        CONSTRAINT Sea_Key PRIMARY KEY,
    Depth NUMBER
        CONSTRAINT Sea_Depth_Check CHECK (
             Depth >= 0
        )
);


CREATE TABLE Island (
    Name VARCHAR2(30)
        CONSTRAINT Island_Key PRIMARY KEY,
    Islands VARCHAR2(30),
    Area NUMBER
        CONSTRAINT Island_Area_Check CHECK (
             Area >= 0
        ),
    Coordinates GeoCoord
        CONSTRAINT Island_Coordinates_Check CHECK (
             (Coordinates.Longitude >= -180) AND
             (Coordinates.Longitude <= 180) AND
             (Coordinates.Latitude >= -90) AND
             (Coordinates.Latitude <= 90)
        )
);
```

```
CREATE TABLE Desert (
    Name VARCHAR2(30)
        CONSTRAINT Desert_Key PRIMARY KEY,
    Area NUMBER
        CONSTRAINT Desert_Area_Check CHECK (
             Area >= 0
        )
);

CREATE TABLE geo_river (
    River VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_river_Key PRIMARY KEY (Country, Province, River)
);

CREATE TABLE geo_mountain (
    Mountain VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_mountain_Key PRIMARY KEY (Country, Province, Mountain)
);

CREATE TABLE geo_lake (
    Lake VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_lake_Key PRIMARY KEY (Country, Province, Lake)
);

CREATE TABLE geo_sea (
    Sea VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_sea_Key PRIMARY KEY (Country, Province, Sea)
);

CREATE TABLE geo_island (
    Island VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_island_Key PRIMARY KEY (Country, Province, Island)
);

CREATE TABLE geo_desert (
    Desert VARCHAR2(30),
    Country CHAR(2),
    Province VARCHAR2(40),
    CONSTRAINT geo_desert_Key PRIMARY KEY (Country, Province, Desert)
);

CREATE TABLE merges_with (
    Sea1 VARCHAR2(30),
    Sea2 VARCHAR2(30),
    CONSTRAINT merges_with_Key PRIMARY KEY (Sea1, Sea2)
);
```