

# DIT032 – DATA MANAGEMENT

## ASSIGNMENT 1

---

University of Gothenburg

Deadline: **Mo 05.02.2018** 23:55

### General Remarks

- There are in total 100 points to be reached for this assignment, and you need at least 70 points to pass the assignment. **Note that you need to pass all assignments individually.** It is not possible to "carry over" some points from this assignment to the next ones.
- Assignment solutions are to be produced in teams of two students. Discussions with other colleagues (e.g., through the forum in GUL) are encouraged, but don't share your assignment solutions, or significant parts, with your colleagues (neither through the forum nor through other means). If we find that multiple groups submitted the same assignments, we reserve the right to fail **all** involved groups.
- Remember to also hand in self- and peer assessment forms. You can use the forms available separately on GUL. Please contact us if you have any questions regarding this part. Assessment forms can either be scanned and attached to the submission (sign first!), handed in during the lecture or supervision, or brought in person to the office hour.
- We offer two teaching assistant supervision hours (Wednesday and Friday, 13:15 – 15:00), where you can talk to the teaching assistants, get feedback on your assignment work, and ask questions.
- **No** deadline extensions are given. Start early and, after finishing your assignment, upload your submission as a **single PDF file** in GUL. The submission system closes automatically, and we do not accept late submissions. If you can't make it in time, you will need to submit during the first re-submission date in May.
- You are free to draw your diagrams using any tool of your choice, but please make sure that your notation is the same as is used in the book and lecture. A notation cheat sheet is contained in the appendix. You are also free to draw the diagrams by hand and scan them, but be careful that everything is legible.
- For the last part of the assignment (installing PostgreSQL), you do not need to submit your data. This part is mainly in preparation for the second assignment, and it is sufficient to answer the question at the end (see the assignment text itself for details).
- Shortly after the deadline, we grade the assignment and send you feedback. If you fail the assignment, you have the possibility to submit an improved version of the assignment for re-grading in May or October (see the website for details). We only re-grade failed submissions. If you have detailed questions it is better to ask them in the forum or during the supervision.

## Assignment 1 Title Page

*(use this page unfilled as the title page for your submission document)*

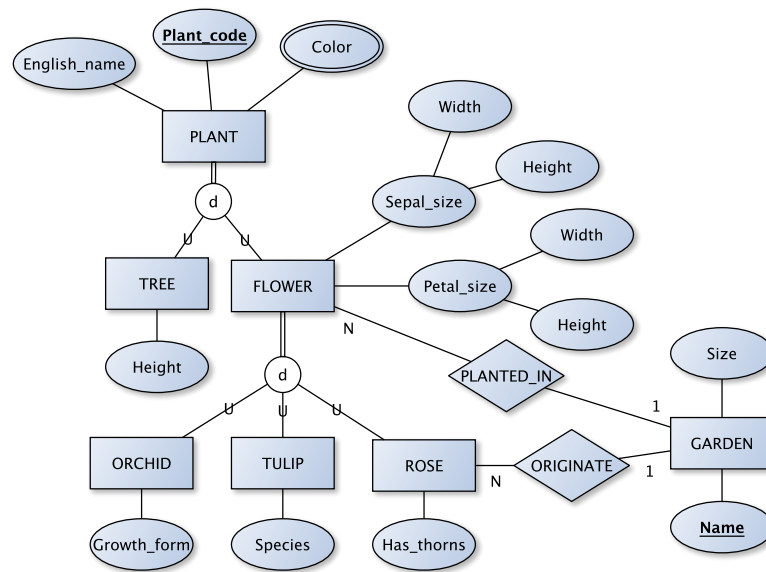
<b>Student</b>	
----------------	--

<b>Task</b>		<b>Pts.</b>	<b>Comments</b>
<b>Enhanced Entity-Relationship (EER) Model</b>	Reading EER Diagrams (max 10 pts)		
	Creating an EER Diagram (max 50 pts)		
<b>Relational Algebra (RA)</b>	Formulating RA Expressions (max. 30 pts)		
	Explaining RA Expressions (max. 5 pts)		
<b>Setup PostgreSQL</b>	(max. 5 pts)		

<b>Total</b>	
--------------	--

# 1 Enhanced Entity-Relationship (EER) Model

## 1.1 Reading EER Diagrams (10 points)

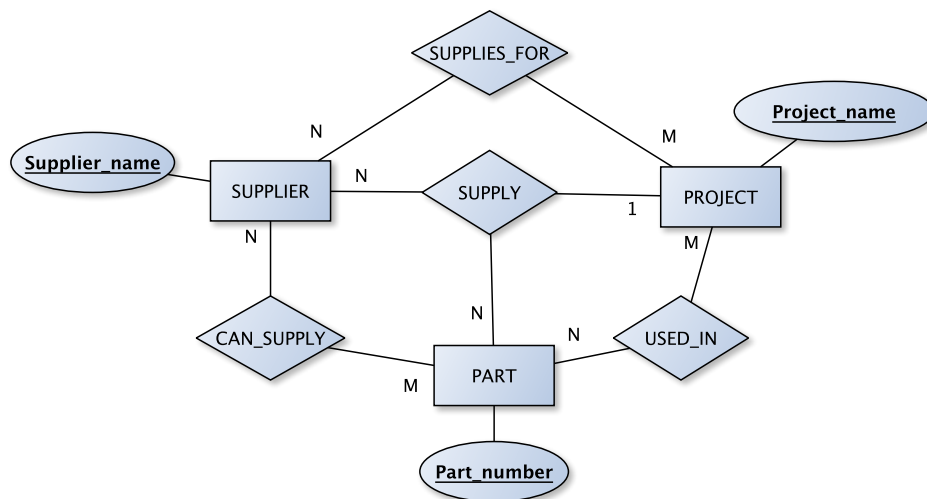


Given the EER diagram above, answer the following questions:

EER1a Can a flower have a height? Shortly explain why!

EER1b Is it mandatory for a rose to originate from a garden? Shortly explain why!

EER1c What are the attributes and relationships of a rose? List them all including identifying keys (for attributes) and the cardinalities (for relationships)!

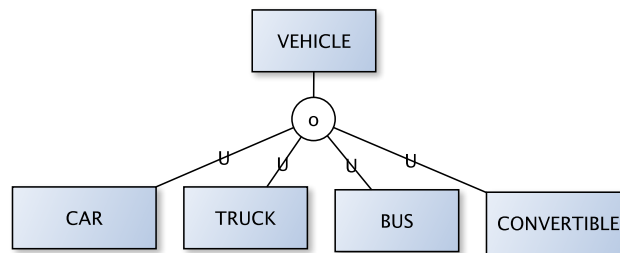


Given the EER diagram above, answer the following questions:

EER2a Can a project exist that does not use parts? Shortly explain why!

EER2b Can parts be used in a project without having a supplier? Shortly explain why!

EER2c Given the instances of a supplier *supplier1*, a part *part1*, and a project *project1*. If supplier1 "SUPPLIES\_FOR" project1 and supplier1 "CAN\_SUPPLY" part1, and part1 is "USED\_IN" project1, does supplier1 "SUPPLY" part1 for project1? Explain why!



EER3a Can a vehicle exist that is neither a car, truck, bus, nor convertible? Shortly explain why!

EER3b Can a convertible bus exist? Shortly explain why!

Refer to the schema for the Mondial-III database in the appendix of this assignment. Notice the slight difference to our standard notation used in the lecture (e.g., the partial keys underlining is not dashed, different naming conventions, relationships do not follow directional rules).

EER4a How is a city identified? List all identifying attributes, explain why, and give an (imaginary) example when such a compound is required!

EER4b How many mandatory and optional relationships does a river have? List them all by writing a sentence that follows the pattern:

A SOURCE ENTITY can|must RELATIONSHIP one|N TARGET ENTITY

Example for airport: [Airport on Island] An airport can be on one island. (Reverse example for island: [Airport on Island] An island can "accommodate" multiple airports) Make reasonable assumptions for the cardinalities. Look carefully!

## 1.2 Creating an EER Diagram (50 points)

The aim of this task is to design an EER model for a personal travel planner system similar to Google Trips.

### 1.2.1 Task

Your task is to draw an EER model that accurately reflects the requirements of the following *description* (1.2.2). Specify the entities, attributes, keys, specializations, relationships including names, cardinalities, and totality constraints to model the description. Make sure that all necessary elements are models such that the system can fulfill the described requirements. Follow the notation from the lecture, as also shown in the notation cheat sheet in the appendix of this assignment. Explicitly state any unspecified requirements and make appropriate assumptions to fill in the gaps only where necessary.

### 1.2.2 Description

Every **user** in the system must have **an email address** and no two users can have the same email address. Every user can mark favorite **hotels, restaurants, landmarks, and museums** that he or she intends to visits. It is very likely that multiple restaurants with the same **name** exist. Further, it might also happen (e.g., in a multi-storey mall) that multiple restaurants are located at the same **location**, which we store **as longitude and latitude**. However, we assume that there are no restaurants at the same location with exactly the same name. To suggest recommendable restaurants at any time, we also store a **rating** and an **opening hour**. However, a restaurant is not the only place a user can mark as a favorite: landmarks, museums, and hotels are all places (of interest), which have assigned **an opening hour and a rating and are also identified in the same way as restaurants**. Additionally, a museum stores the current **exhibition** and a restaurant stores its **price level**. For a restaurant, we also want to **be able to specify several categories such as Italian, Asian, Mexican, Take-away**. The system also needs to be able to filter hotels based on their amenities (e.g., only show hotels with a gym, swimming pool, and sauna). Hotels typically offer rooms and in our system we want to keep track of the room number and size. Users can book these rooms by making a reservation. Every reservation stores a **confirmation number, the check-in, the check-out, and the price for the booking**. To quickly find out the length of the stay, we also want to keep track of the **duration** for which the reservation has been made.

Our system has to support the handy feature of automatically detecting reservations. Therefore, we keep track of emails that a user receives. We create an **identifier** for every email and **store its date, message, and the sender**. A **reservation** can now link to every related email. Additionally, an email can also have one or multiple attached invoices. For an **invoice**, we store an **invoice id, a date, and the amount of the invoice**. An invoice may refer to a reservation. It is also possible that invoices for a single reservation are split into multiple ones.

To organize various travel destinations, a user can manually create individual trips to a certain destination with a defined start date and end date. However, oftentimes, the system recognizes reservations from emails and then automatically creates a trip where the reservation can belong to. Trips can also have a set of named day plans. A day plan stores its date and references a list of places to visit in a particular order (i.e., the system must keep track which place is visited first, second, etc).

Our system also includes a social feature. Users are able to follow other users to receive notifications about their trips. For privacy reasons, the followee needs to approve follow requests and thus we need to keep track whether this approval already happened.

### 1.2.3 Tips

- We recommend to use the cross-platform yEd<sup>1</sup> modeling tool.
- To represent partial keys, you can "fake" a dashed underline by adding a second label consisting of one line of '-' characters and moving said label right below the attribute name label<sup>2</sup>.
- To illustrate specialization, the U-shape symbol on the line can be achieved by adding the letter 'U' as a label and dragging it onto the line (see the examples in 1.1).
- Holding the modifier key (e.g., ctrl on MacOS) allows for more precise placement of labels.

---

<sup>1</sup><http://www.yworks.com/products/yed/download#download>

<sup>2</sup><http://yed.yworks.com/support/qa/7885/update-please-need-this-feature-partial-key-representation>

## 2 Relational Algebra (RA)

The following tasks are based on the *Mondial III* database [1]. Refer to the appendix for its relational schema. You may also find the referential dependencies overview in the appendix helpful.

### 2.1 Formulating RA Expressions (30 points)

Formulate relational algebra expressions that answer the following queries. Ensure that your expressions are correct for all Mondial database instances (i.e., do not use any constants other than the ones provided in the descriptions).

- RA1. The name of all islands that are of type coral or atoll. You may use the constants 'coral' and 'atoll'.
- RA2. The name, province, country, and population of all cities above the arctic circle. You may use the constant 66.33.
- RA3. The country code of all countries belonging to the continent Europe (at least partially) that are not members of the European Union. You may use the constants 'Europe' and 'EU' (i.e., the abbreviation of the European Union).
- RA4. The name of all countries where more than half of the population speaks English. You may use the constant 'English' and 50.
- RA5. The name of all provinces that have waters (i.e., lake or sea) deeper than 1000 meters. You may use the constant 1000.
- RA6. The difference in population between 2001 and 2011 for every county (i.e., the country code). You may use the constants 2001 and 2011.
- RA7. The abbreviation and average GDP for every organization with at least one member country.
- RA8. The name of all lakes that are adjacent to at least two countries.
- RA9. The name, the number of lakes, and the number of rivers of all countries. Do also include countries with no lakes or rivers (you may leave them as NULL values).
- RA10. The name and elevation of the highest mountain in Sweden. You may use the constant 'S' (i.e., country code of Sweden). You don't need to handle the special case of multiple highest mountains. Tip: Understanding U1 might be beneficial to solving this task.

### 2.2 Understanding RA Expressions (5 points)

Explain precisely in natural language what the following relational algebra expressions return:

$$\begin{aligned} \text{U1. } MyAirport1 &\leftarrow \rho_{iatacode,x1} (\pi_{iatacode,|latitude-66.33|} (Airport)) \\ MyAirport2 &\leftarrow \rho_{iatacode2,x2} (MyAirport1) \end{aligned}$$

$$\pi_{iatacode} (Airport) - \pi_{iatacode} (MyAirport1 \bowtie_{x1 > x2} MyAirport2)$$

Notice that the  $|x|$  denotes the absolute value of  $x$ .

$$\text{U2. } \text{code} \mathcal{F}COUNT_{\text{code}} (\sigma_{length > 100} (borders \bowtie_{country1=code \vee country2=code} country))$$

### 3 Setup PostgreSQL (5 points)

The aim of this task is to setup everything for the next assignment. Therefore, everyone in your team should complete this task.

1. Download and install PostgreSQL 10<sup>3</sup>: <https://www.postgresql.org/download/>
2. Create a user called `mondial` and a database called `mondial`. This can be achieved using the `psql`<sup>4</sup> PostgreSQL client. Extended SQL code can be used to create a new user<sup>5</sup>/role<sup>6</sup> and database<sup>7</sup>.

```
psql
# Show help
\?
# List databases
\l
CREATE USER mondial;
CREATE DATABASE mondial WITH OWNER mondial;
# Quit interactive session
\q
```

Tips: You might consider creating your own user (the same as your laptop username) or configure `.pgpass`<sup>8</sup> for local testing to save you from specifying the user for `psql` with `-U mondial`.

3. Download the schema (`create_schema.sql`) and the input values (`insert_inputs.sql`) from GUL.
4. Import the `mondial` schema and values:

```
psql -q -U mondial -f ./create_schema.sql mondial
psql -q -U mondial -f ./insert_inputs.sql mondial
```

5. Connect to the `mondial` database with the terminal client `psql`:

```
psql -U mondial mondial
```

Tips: You can use `dropdb`<sup>9</sup> or `DROP DATABASE mondial;`<sup>10</sup> to destroy your database and start from scratch.

6. Run the following query against the `mondial` database and report the result:

```
SELECT COUNT(iatacode) FROM airport;
```

---

<sup>3</sup>Tested with version 10 but version 9.x should be also fine.

<sup>4</sup><https://www.postgresql.org/docs/current/static/app-psql.html>

<sup>5</sup><https://www.postgresql.org/docs/current/static/sql-createuser.html>

<sup>6</sup><https://www.postgresql.org/docs/current/static/sql-createrole.html>

<sup>7</sup><https://www.postgresql.org/docs/current/static/sql-createdatabase.html>

<sup>8</sup><https://www.postgresql.org/docs/current/static/libpq-pgpass.html>

<sup>9</sup><https://www.postgresql.org/docs/current/static/app-dropdb.html>

<sup>10</sup><https://www.postgresql.org/docs/8.2/static/sql-dropdatabase.html>


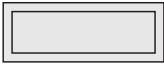
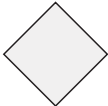
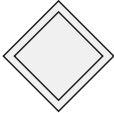

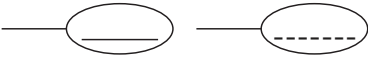

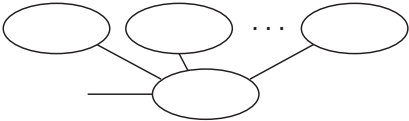

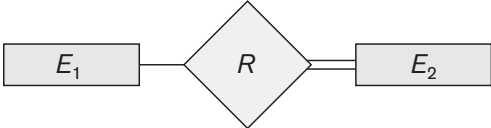
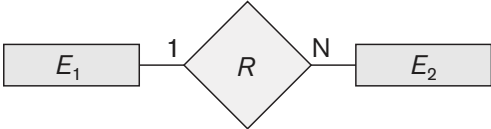
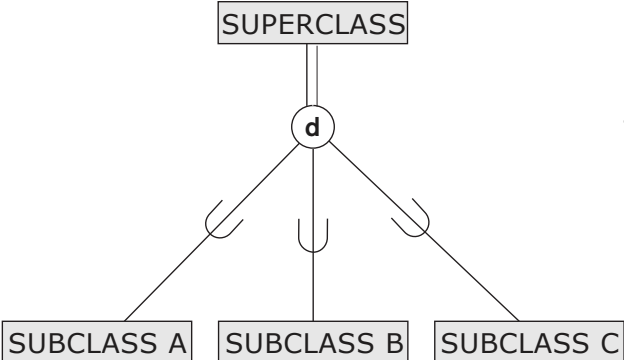
## References

- [1] Wolfgang May. Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999. Available from <http://dbis.informatik.uni-goettingen.de/Mondial>.

## Appendix

- A) EER Notation Cheatsheet
- B) RA Notation Cheatsheet
- C) ER-Diagram of the Mondial Database
- D) Referential Dependencies of the Mondial Database
- E) The relational schema of the Mondial database

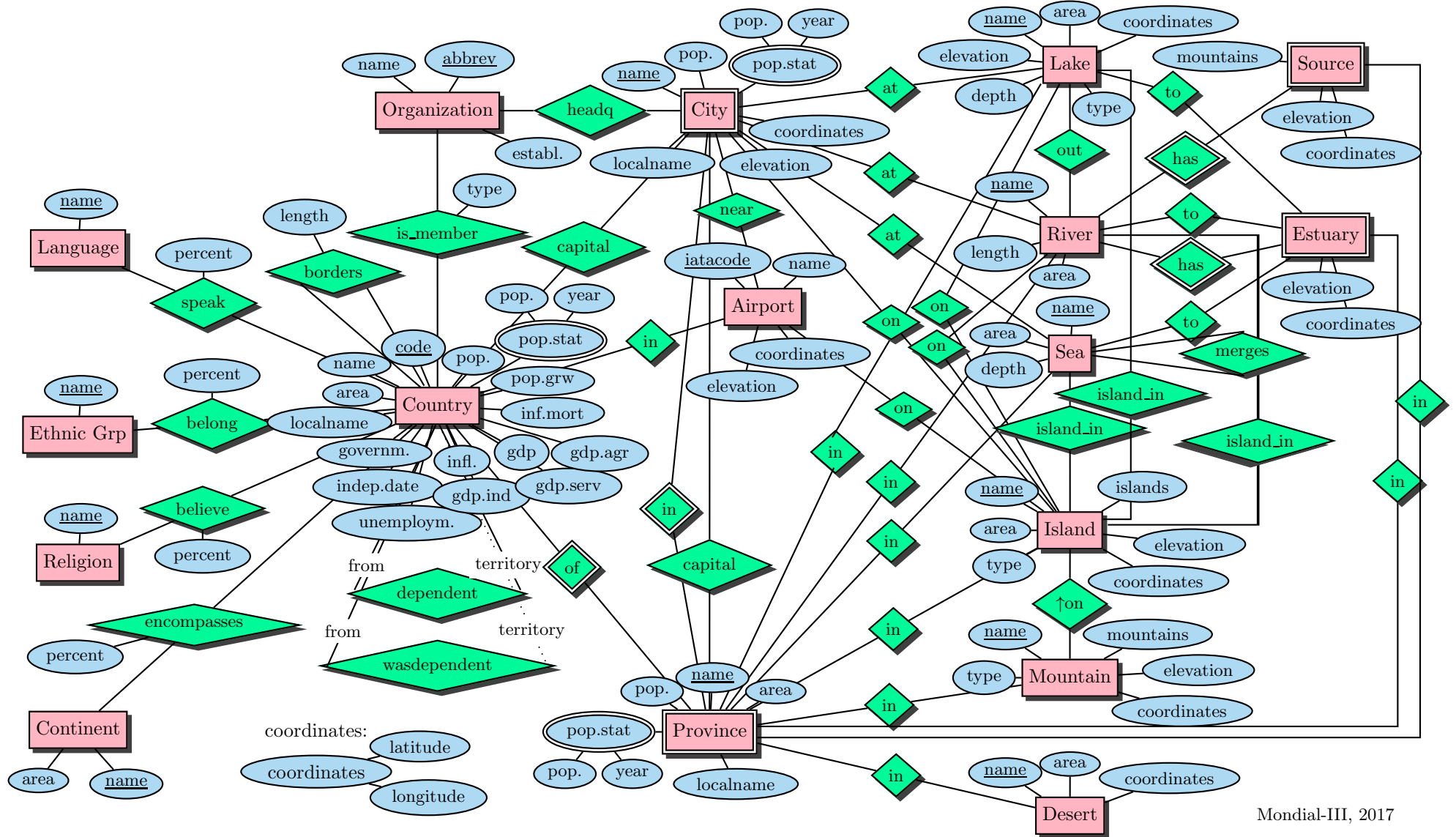


Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute / Dashed Underline for Partial Key
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1 : N for $E_1 : E_2$ in $R$
	Total Disjoint Specialization

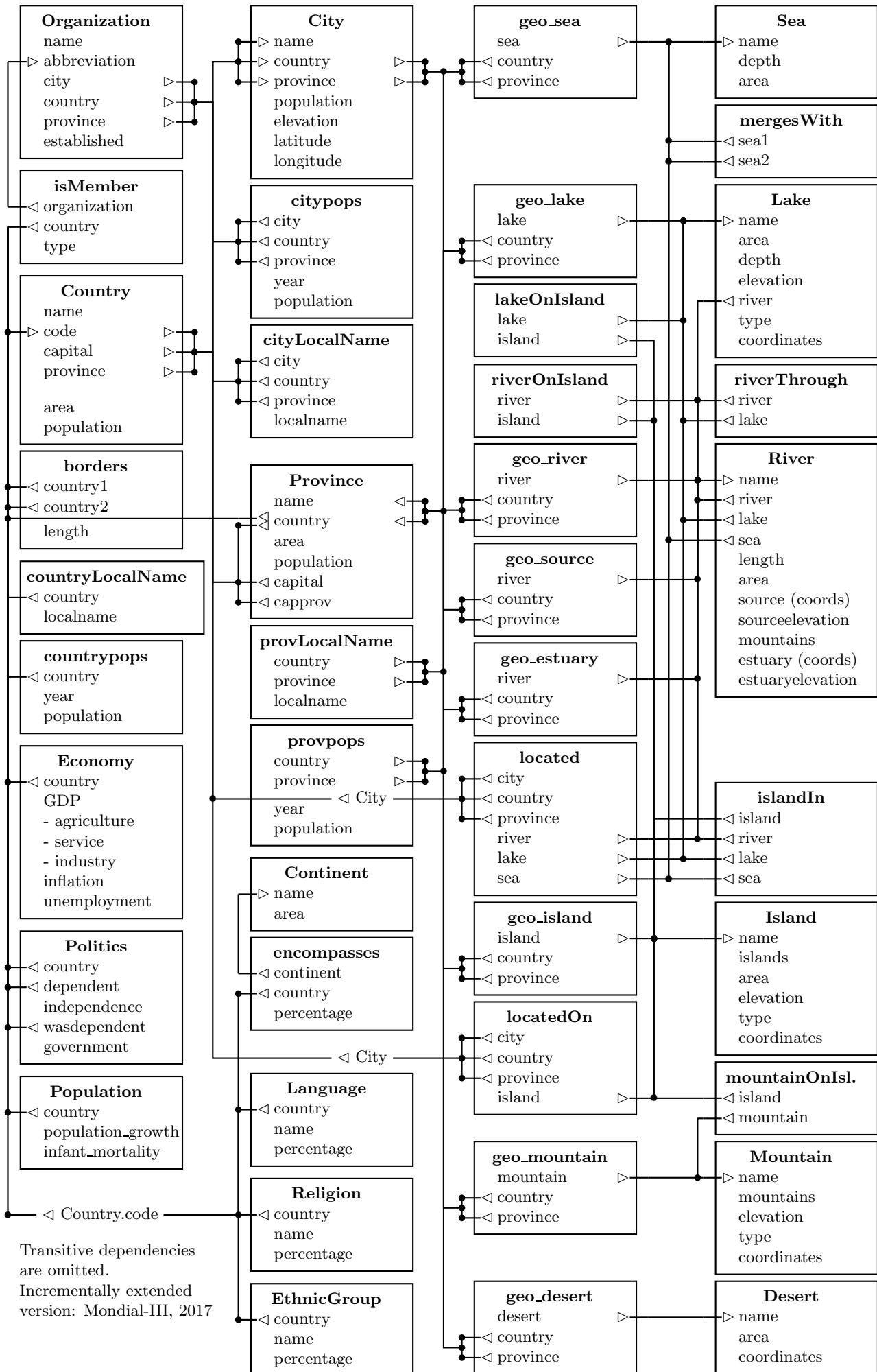
**Table 8.1** Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle, \langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \star_{(\langle \text{join attributes 1} \rangle, \langle \text{join attributes 2} \rangle)} R_2$
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$ .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	$R_1(Z) \div R_2(Y)$

# ER-Diagram of the Mondial Database



# Referential Dependencies of the Mondial Database



## The relational schema of the Mondial database

**Country:** the countries (and similar areas) of the world with some data.

name: the country name

code: the car code

capital: the name of the capital

province: the province where the capital belongs to

area: the total area

population: the population number

**Economy:** economical information about the countries.

country: the country code

GDP: gross domestic product (in million \$)

agriculture: percentage of agriculture of the GDP

service: percentage of services of the GDP

industry: percentage of industry of the GDP

inflation: inflation rate (per annum)

unemployment: unemployment rate

**Politics:** political information about the countries.

country: the country code

independence: date of independence (if independent)

wasdependent: the political body where the area was dependent of; usually a country (but not always).

dependent: the country code where the area belongs to

government: type of government

**Population:** information about the population of the countries.

country: the country code

population\_growth: population growth rate (per annum)

infant\_mortality: infant mortality (per thousand)

**Countrypops:** information about the population number of the countries in different years.

country: the country code

population: number of inhabitants

year: in which year

**CountryLocalName:** information about the local name of the country.

country: the country code

localname: the local name, usually in a local alphabet (UTF-8)

**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 language in this country

**EthnicGroup:** information about the ethnic groups in a country

country: the country code

name: name of the religion

percentage: percentage of the language in this country

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

country1: a country code

country2: a country code

length: length of the border between country1 and country2

**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

**City:** information about cities.

name: the name of the city

country: the code of the country where it belongs to

province: the name of the province where it belongs to

population: population of the city

elevation: the elevation (above sea level) of the city

latitude: geographic latitude

longitude: geographic longitude

**Citypops:** information about the population number of the cities in different years.

city: the name of the city

province: the name of the province

country: the code of the country where it belongs to

population: number of inhabitants

year: in which year

**CityLocalName:** information about the local name of the city.

city: the name of the city

province: the name of the province

country: the code of the country where it belongs to localname:  
the local name, usually in a local alphabet (UTF-8)

**Province:** information about administrative divisions.

name: the name of the administrative division

country: the country code where it belongs to

area: the total area of the province

population: the population of the province

capital: the name of the capital

capprov: the name of the province where the capital belongs to

note that *capprov* is not necessarily equal to *name*. E.g., the municipality of *Bogota (Colombia)* is a province of its own, and *Bogota* is also the capital of the surrounding province *Cundinamarca*.

**Provpops:** information about the population number of the provinces in different years.

province: the name of the province

country: the code of the country where it belongs to

population: number of inhabitants

year: in which year

**ProvinceLocalName:** information about the local name of the province.

province: the name of the province

country: the code of the country where it belongs to localname:

the local name, usually in a local alphabet (UTF-8)

**Organization:** information about political and economical organizations.

name: the full name of the organization

abbreviation: its abbreviation

city: the city where the headquarter is located

country: the code of the country where the headquarter is located

province: the name of the province where the headquarter is located

established: date of establishment

**isMember:** memberships in political and economical organizations.

organization: the abbreviation of the organization

country: the code of the member country

type: the type of membership

**Lake:** information about lakes.

name: the name of the lake

area: the total area of the lake

depth: the depth of the lake

elevation: the elevation (above sea level) of the lake

river: the river that flows out of the lake (may be null)

type: the type of the lake, e.g., salt, caldera, ...

coordinates: its geographical coordinates as (latitude, longitude)

**Sea:** information about seas.

name: the name of the sea

depth: the maximal depth of the sea

area: the total area of the sea

**River:** information about rivers.

name: the name of the river

length: the length of the river

area: the size of its catchment area

river: the river where it finally flows to

lake: the lake where it finally flows to

sea: the sea where it finally flows to;

(note that at most one out of {river,lake,sea} can be non-null)

source: the coordinates of its source

sourceElevation: the elevation (above sea level) of its source

mountains: the mountains where its source is located

estuary: the coordinates of its estuary

estuaryElevation: the elevation (above sea level) of its estuary

**RiverThrough:** information about rivers flowing through lakes.

river: the name of the river

lake: the lake where it flows through

**Mountain:** information about mountains

name: the name of the mountain

mountains: the mountains where it belongs to

elevation: the maximal elevation of the summit of the mountain

type: the type of the mountain, e.g. volcanic, (active) volcano, ...

coordinates: its geographical coordinates as (latitude, longitude)

**Island:** information about islands

name: the name of the island

islands: the group of islands where it belongs to

area: the area of the island

elevation: the maximal elevation of the island

type: the type of the island, e.g. volcanic, coral, atoll, ...

coordinates: its geographical coordinates as (latitude, longitude)

**Desert:** information about deserts.

name: the name of the desert

area: the total area of the desert

coordinates: its geographical coordinates as (latitude, longitude)

**mergesWith:** information about neighboring seas. Note that in this relation, for every pair of neighboring seas (A,B), only one tuple is given – thus, the relation is *not* symmetric.

sea1: a sea

sea2: a sea

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

city: the name of the city

country: the country code where the city belongs to

province: the province 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

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

**locatedOn:** information about cities located in islands.

city: the name of the city

country: the country code where the city belongs to

province: the province where the city belongs to

island: the island it is (maybe only partially) located on

Note that for a given city, there can be several islands where it is located on.

**islandIn:** information the waters where the islands are located in.

island: the name of the island

sea: the sea where the island is located in

lake: the lake where the island is located in

river: the river where the island is located in

Note that an island can have coasts to several seas.

**MountainOnIsland:** information which mountains are located on islands.

mountain: the name of the mountain

island: the name of the island

**RiverOnIsland:** information which rivers are located on islands.

river: the name of the river

island: the name of the island



**LakeOnIsland:** information which lakes are located on islands.

lake: the name of the lake

island: the name of the island

**Airport:** information about airports

iatacode: the IATA code of the airport

name: the name of the airport

country: the country code where the airport is located

city: in case the airport is associated with a city, the name of the city

province: the province where the city belongs to

island: if it is located on an island, the name of this island

latitude: geographic latitude

longitude: geographic longitude

elevation: the elevation (above sea level) of the city

gmtOffset: the GMT offset of the local time

**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

**geo\_estuary:** geographical information about the estuary of rivers.

river: the name of the river

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\_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\_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\_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\_sea:** geographical information about seas.

sea: the name of the sea

country: the code of the country to which the sea is adjacent

province: the province of this country

**geo\_source:** geographical information about sources of rivers.

river: the name of the river

country: the country code where it is located

province: the province of this country

Incrementally extended version: Mondial-III, 2017

2017-11-17: Underlined primary keys

2017-11-21: Added geo\_\* explicitly