DIT032 / DAT335 – DATA MANAGEMENT

ASSIGNMENT 3

Chalmers | University of Gothenburg Deadline: **So Feb 28th** 23:59

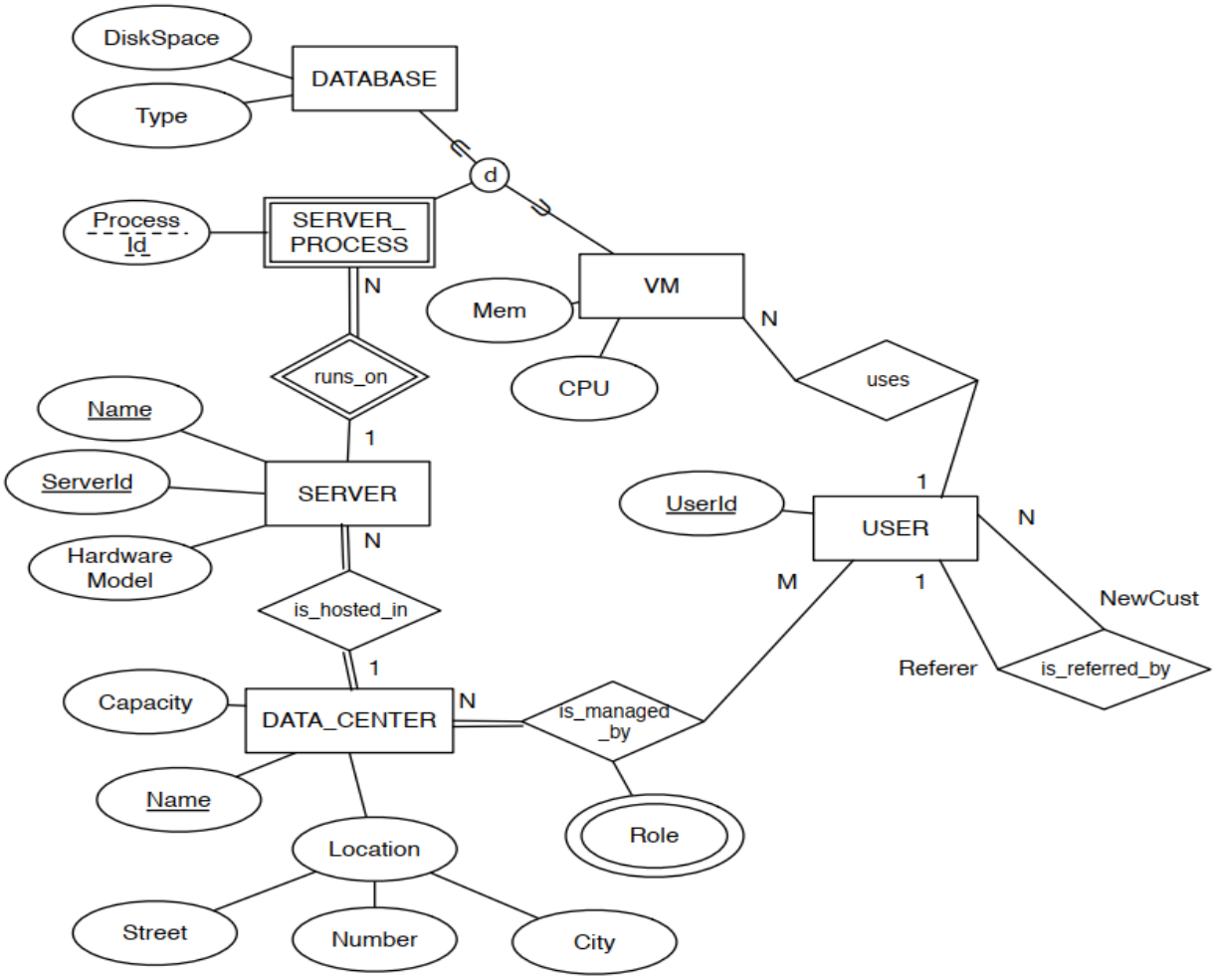
General Remarks

- There are in total 50 points to be reached for this assignment, and you need at least 35 points to pass the assignment. You need to pass 3/4, and you need at least 25 points on each individual assignment.
- Assignment solutions are to be produced in teams of two students. Discussions with other colleagues (e.g., through the forum in Canvas) are encouraged, but do not 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 are mandated to forward any suspected cases to the Disciplinary Committee (which may decide to suspend students from their studies).
- Remember to also hand in self- and peer assessment forms through Canvas *individually*.
- 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.
- 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. Please see Canvas for details on the re-submission procedure (you will also need to submit a detailed change log).

1. [SQL1] Creating a SQL Database (10 points)

This task is based on knowledge from **Lecture 6**. You can start immediately after this lecture.

Consider again the simplified grid management system that we worked with for Assignment 1 and 2



Your task now is to actually implement this database (i.e., the database tables and some sample content) in the relational database management system PostgreSQL.

Important: you can "try out" commands in the psql command line, but you need to deliver all code that you write to create your database as a text file. Save all SQL code to a single text file called grid_management.sql, which you will submit as part of your delivery in Canvas. Make sure that your solution can be executed via:
psql -f grid_management.sql grid postgres
(test this prior to your submission)
We will not accept database dumps, you need to send us the actual code you used to create your database!

SQL1.1 Create a new database called "grid".

SQL1.2 In this new database, create SQL tables using the appropriate DDL statements, such as

CREATE TABLE for all relations. Add data types as appropriate, and make sure to identify and implement primary and foreign key constraints as well "not null" and "unique" constraints. Also make sure that you are handling the weak entity "SERVER_PROCESS" appropriately.

SQL1.3 Add a custom constraint that stipulates that the capacity of any data center cannot be more than 10000 (servers).

SQL1.4 Insert at least three rows of test data into each of those tables.

- Tips
  - The relational schema you created in Assignment 2 will be an excellent starting point for SQL1.2. If you have done a good job in writing down the relational schema, you essentially only need to transcribe the textual representation into SQL code.
  - The following PostgreSQL commands may be useful to explore your database, and to validate that you created your tables correctly:
    \c <db_name> to connect to a database,
    \d to list all tables in the database,
    \d+ <tbl_name> to see the detailed schema of a specific table, and
    SELECT * from <tbl_name>; to see all content (all rows) in a table.

2. [SQL2] Simple SQL Queries (6 points)

Most of this task you can already do after **Lecture 6**. If we have not covered some of the required constructs they will follow in **Lecture 7**.

Write 6 simple SQL queries on the database you created. You can decide on your own what to query for exactly, but the queries should "do" something interesting (i.e., it should be a query for something that somebody could theoretically want to find out). The following SQL constructs should be used at least once in at least one of your queries:

1. A selection (a WHERE clause)
2. A projection (a SELECT with a defined attribute list, not just "*")
3. An ORDER BY clause
4. An aggregation with grouping (GROUP BY)
5. A join over more than two (i.e., 3+) tables (any type of join)
6. A union, intersect, or except operator

Submit a separate text file called trips_queries.sql containing each query, and an explanation for each query what it is intended to do. Example:

```
-- SQL2.1
-- This query finds all foos from bar.
SELECT foo FROM bar;
```

3. [SQL3]: Given the tables in Figure 1, answer the following queries: (16 points)

Lecturer

LecturerID	Name	PositionID	DeptID
LC09	Nizam	DS54	CS
LC42	Bariah	DS52	GM
LC35	Ahmad	DS51	SE
LC51	Navin	DS45	SE
LC78	Faizal	DS52	GM
LC80	Kenneth	DS52	SN

Student

StudentID	Name	BirthPlace	Gender
CD12021	Harithah	Bohuslän	F
CB13002	Padli	Dalsland	M
CA12201	Harith	Gotland	M
CC12300	Gareth	Halland	M
CD13200	Rymes	Öland	F
CB13031	May Lin	Östergötland	M
CA13500	Shankar	Närke	M
CC13111	Aminah	Skåne	M

Appointment

AptID	AptDate	StudentID	LecturerID	AptTime
10	8/2/14	CC12300	LC78	1200
20	1/3/14	CD13200	LC42	1500
30	7/3/14	CB13031	LC35	0900
40	1/4/14	CB13031	LC51	1000
50	2/4/14	CA13500	LC09	1200
60	3/4/14	CA12201	LC42	1000
70	5/5/14	CC12300	LC35	1500

Figure 1: Tables of Lecturer, Student and Appointment

SQL3.1 Construct a query to display the following result, which lists the appointment in the month of April **only** with names of the students and lecturers.

(5 points)

AptID	AptDate	StudentID	StudentName	LecturerID	LecturerName
40	1/4/14	CB13031	May Lin	51	Navin
50	2/4/14	CA13500	Shankar	09	Nizam
60	3/4/14	CA12201	Harith	42	Bariah

SQL3.2 Construct a query to display the LecturerID, Lecturer Name, Postion_ID, AptID and StudentID for Lecturers with PositionID = 'DS52' sorting the data by Lecturer Name and AptID as shown in the following result.

(4 points)

LecturerID	Name	PositionID	AptID	StudentID
LC42	Bariah	DS52	20	CD13200
LC42	Bariah	DS52	60	CA12201
LC78	Faizal	DS52	10	CC12300

SQL3.3 Construct a query to display the name, birthplace and gender of all students that have appointment with lecturer named Ahmad (assuming that Ahmad's LecturerID is LC35).

(3 points)

SQL3.4 Assuming that Ahmad's LecturerID is not known, construct a query to display the name, birth place and gender of all students that have appointment with lecturer named Ahmad. *Hints: Use the Query of the SQL4.3.*

(4 points)

4. [SQL4] Querying Mondial (18 points)

The following tasks are again based on the Mondial III database, which you have already used in the previous assignment. Refer again to the appendix for its relational schema. You may also find the referential dependencies overview in the appendix helpful. However, oftentimes it will be easier to just explore the database content directly in the psql command line client (e.g., by using the \d+ TABLENAME command or by querying the content of tables using select * from TABLENAME). However, make sure to consider the various special cases that can come up (e.g., countries can be on multiple continents, mountains may have the same height, etc.). There will often be multiple correct ways to formulate a query. Formulate SQL queries that answer the following questions. Save all your queries into a SQL file called mondial_queries.sql, which you submit as part of your solution. Label each query with an SQL comment as in the previous task.
Important: We only accept queries tested with PostgreSQL. Do **not** use the SQL editor in RelaX!

SQL4.1 Write a query to list all volcanic mountains

SQL4.2 Write a query to show how many mountains of type "volcanic" or "Plateau"

SQL4.3 Return the names of all mountains of unspecified type ordered in reverse alphabetical order.

SQL4.4 Return the names of all deserts, and the country codes of all countries bordering them. Ensure that the result does not contain duplicate rows.

SQL4.5 Write a query to show how many countries are members in "G-24" and located in Asia. Please display all the country names and codes that are members in "G-24" and not located in Asia.

SQL4.6 Return the names of "waters" of all kinds (i.e., lakes, rivers, and seas), ordered by their name. Add a column "type" that indicates what type of water it is (either lake, river, or sea).

SQL4.7 The continent name and average GDP for each continent. If a country is on multiple continents, count their GDP towards the average calculation for all their continents.

SQL4.8 Write a query to display the name of all countries that only have borders longer than 100 km.

SQL4.9 Write a query to return all countries with a higher percentage of English speakers than in the United States of America. Use the country code "USA". Implement this as a single query, and don't use the percentage of English speakers in the US as a constant in this query.
Solution.

5. [SQL5] Query Optimization (for the glory)

In this assignment, we have so far only cared about whether queries work, not whether they are actually fast. In this task, we will now investigate the performance of the queries you wrote in Task SQL2-4. Our three main methods to improve the performance of SQL queries are query rewriting, adding appropriate indexes, and creating views.

[SQL5.1] Query Plans and Query Rewriting

The basic Postgres command you can use to visualize the query plan for a given query is EXPLAIN ANALYZE (to use it just prefix your query with this command, e.g., EXPLAIN ANALYZE SELECT * FROM country;). Test this with a few queries that you have written earlier in the assignment (the more complex queries will tend to lead to more interesting execution plans). The textual output of Postgres is not easy to understand, but there are web-based tools that you can try that visualize your traces (e.g., here4). Play around with different types of queries (ones that you have written previously, but you can also experiment with new queries). Do you understand why and in which order Postgres executes different query components? Can you see patterns of which types of queries are typically slower than others?

Use the knowledge you have gained with these experiments to revisit some of the queries you have written in SQL2, SQL3 and SQL4. Can you rewrite them so that they do the same thing, but more efficiently? Or, if they are already fairly fast, can you construct a query that is particularly slow? What are the key characteristics of such queries?

[SQL5.2] Indexes

Now revisit your queries and consider for which (if any) a well-placed index may improve performance. Add an index and test using EXPLAIN ANALYZE (but keep in mind that for low execution times, say a few ms, the natural variation between identical executions will be very large). If you do not have a query that would profit from adding an index, come up with such a query, add the index to your database, and test it.

[SQL5.3] Views

Repeat the same experiment with views. Which of your queries may profit from adding a view, and under which conditions? Experiment with different types of views (e.g., materialized views). What are the different advantages and disadvantages of different types of views?

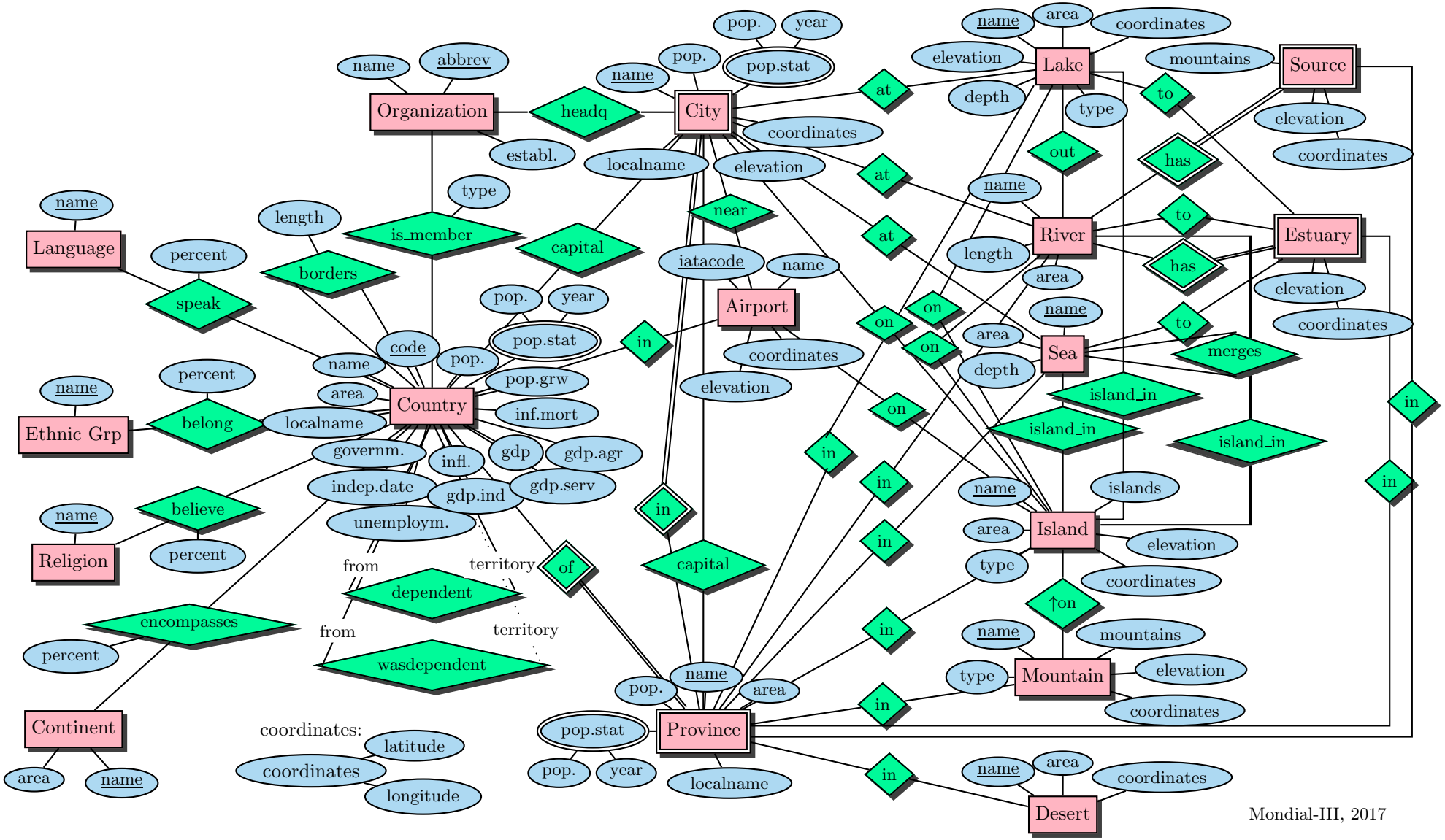
Appendix

A. ER-Diagram of the Mondial Database.

B. Referential Dependencies of the Mondial Database.

C. The relational schema of the Mondial database.

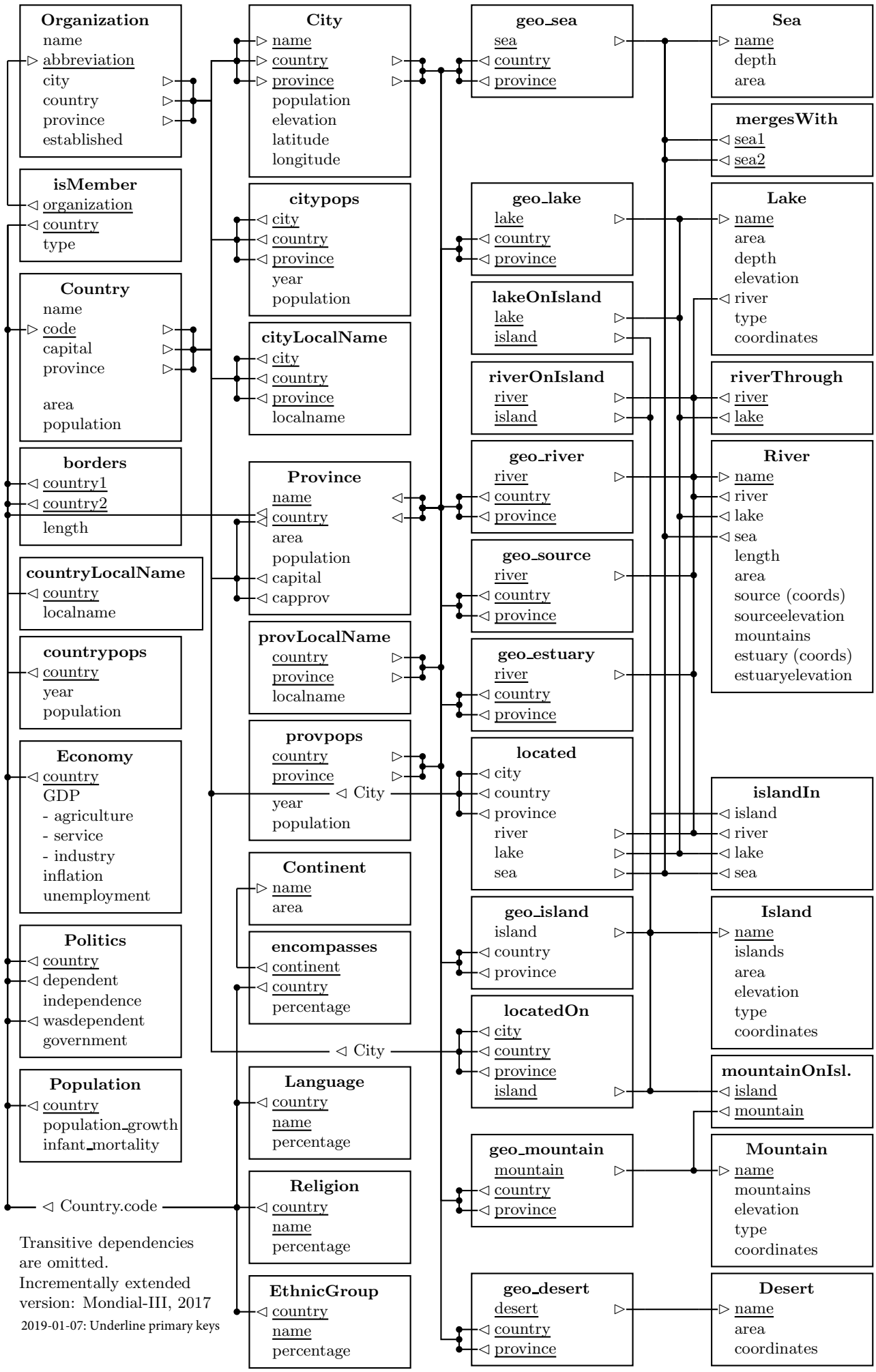
ER-Diagram of the Mondial Database



Mondial-III, 2017

Updates:
2018-01-24 add total participation for weak entity types

Referential Dependencies of the Mondial Database



Transitive dependencies are omitted.
Incrementally extended version: Mondial-III, 2017

2019-01-07: Underline primary keys

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 country 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: infomation about the languages spoken in a country
country: the country code
name: name of the languange
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 montains 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 montains where the 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: Addded geo_* explicitly
2019-01-07: Fixed primary keys for Language, Religion, EthnicGroup
2020-02-15: Fixed misplaced linebreak in CityLocalName