Université Catholique de Louvain

LINGI2172 - Databases

# Mission 3 - Database Design

*Auteur :*
Baugnies Benjamin (6020-10-00)
Colson Olivier (5039-10-00)
Vanwelde Romain (3143-10-00)

Group 3

*Superviseurs :*
Pr. Bernard Lambeau
Antoine Cailliau

4 avril 2014

# 1 Introduction

Telling a story is challenging. Indeed, in order to build a “good" scenario, one must think of a lot of different aspects and ensure coherence between all these points.

Some can manage it using diagrams, other can rely on tons of paper sheets referencing each other or, more reasonably, use a computer to store all their documents. However, with all these ways of working, the same problem arises : as the storyline and background get denser, it becomes more and more difficult to ensure that no contradiction appears. This is a big problem, since contradictions ruin the feeling of reality that must always be given by a good scenario. How about asking the computer to gather, interpret and display all this information in a clean and understandable way ?

Our project can be defined as a “narration manager". Its goal is to make it easier for people to write coherent and complex scenarios without either becoming mad or cancelling their project because of its increasing complexity. It is intended for all "story makers" (videogame makers, film makers, roleplayers, writers, . . .), and is thus meant to be generic and conveniently adapt to various situations, as well as user preferences and priorities in the story (for example, some users could want to define a precise date for each event happening in their story, as others could prefer to focus on the relations binding all the characters together).

Possibilities of telling a story are infinite, yet time and coordination constraints often limit what is actually possible to achieve. It is now time to push these limits away.

# 2 Elementary Facts

Bellow are some elementary facts we wrote to better understand what to do, which relation exists between all the entities.

## 2.1 About characters

Pierre is from Bruxelles.
Pierre is born on 28/12/1992
Jean is born on 01/04/1992
Benjamin is born on 03/06/1992
Jean is melancholic
Benjamin is member of the association "Les Petits Riens"

## 2.2 Characters relations

Pierre liked Benjamin from 9/12/2002 to 13/7/2007.
Benjamin liked Pierre from 10/11/2003 to 12/8/2009.
Jean doesn't like Benjamin from 10/11/2003 to 12/8/2009.
Paul is Pierre's father.

Pierre is Paul's son since 28/12/1992.

### 2.3 About events

Jean attended the event "The beer festival"
The "beer festival" took place at LLN
The "beer festival" lasted from 08/03/2014 to 18/03/2014.
The "beer festival" is "blablablablablablablabla" as description.

### 2.4 About places

Intel room is a sub-location of Réaumur's Map and is represented on square number 10.
Réaumur is a sub-location of LLN's Map and is represented on square number 5.
The LLN's Map represents the location "LLN"
The Réaumur's Map represents the location "Réaumur"
LLNMap has 10 square width, and represents a 5km distance.
LLNMAP has 20 square length, and represents a 10km distance.

## 3 ORM schema

The ORM schema is shown on the Figure 1. If some relations are too difficult to read, the numeric version is available in the Annexes directory of the zip file.

As we can see on the schema, there are 5 main entities which are `"Characters"`, `"Date"`, `"Events"`, `"Place"`, and `"Map"`.
We will explain here above three specific case of the diagram :

**Character - Relation relations involving time range, time or timeless notion.**
The relations explain the relationships between different characters. They are uni-directional and the different kinds of relations can be defined by the user. We split these relations into three types to represent the fact that some relations are time-independent (e.g "... is my father"), some can start at a given time and be permanent and/or open-ended (e.g "... is my godfather since ... "), and finally some can last for only a while (e.g "... was my friend from ... to ... ").

**Pseudo relation.**
This relation involves two characters and a pseudonym. It describes the name used by the first character for the second one. This represents the fact that during the story, a given character might not know another's real name. We added this relation since this can have an impact on the story (he wouldn't realize others were talking about someone he knows for example). The corresponding table will also allow us the find all the pseudonyms a given person might go by.

**Place - Map relation.**
This is a rather complex relation that we introduced to keep track of a story's geography at different levels. The first use is to allow users to situate events or characters. We can also define sub-places to refine locations. We can see that a place's map is optional. However, since a place's sub-places are linked to its map, it implicitly becomes required when we want to add levels. This structure allows us to chain an arbitrary number of levels with a "place - map - place - map - . . . " hierarchy. This relation has two constraints that are not expressed in the database and will have to be verified in the software implementation. Firstly, the map square on which the sub-place is located must belong to the map's domain of possible squares (between 1 and (# square width)*(# square length)). Secondly, it is required that two places of the same level do not overlap.

## 4 Tutorial-D script

In the zip file, you can find files `structure_rel.d` and `data_rel.d`.

**structure_rel.d** has three differents parts. The **first** one is the definition of special types (All the entities ID, and one for some names too). Then, the **second** part is all the relvars, with all primary keys, and also unique keys (e.g. `MAPPEDPLACE` which has one key on the PLACEID and another on the MAPID). The third part is all the constraints which are foreign keys. We express first all the constraints implying CHARID, then implying ASSOCIATIONID, and so on.

**data_rel.d** fills the database with all elementary facts expressed before in the report.

## 5 Relvar predicates

In this section, we will give the relvar predicates that our database represents. The attributes of a relation are in bold, and the attributes that form the key are underlined.

### About characters :

— The character <u>**CHARACTERID**</u> is named **NAME**. <u>**ASSOCIATIONID**</u>.
— The character <u>**CHARACTERID**</u> has a behavior that is <u>**BEHAVIOR**</u>.
— The character <u>**CALLERID**</u> knows the charactert <u>**CALLEID**</u> by the pseudonym <u>**PSEUDONYME**</u>.

### About time :

— The date <u>**DATEID**</u> is on the day **DAY** of month **MONTH** in the year **YEAR**.
— The character <u>**CHARACTERID**</u> was born on **BIRTH**.
— The character <u>**CHARACTERID**</u> died on **DEATH**.

### About Associations :

- The association **ASSOCIATIONID** is named **NAME**.
- The association **ASSOCIATIONID** is described as **DESCRIPTION**.
- The character **CHARACTERID** is a member of the association.

### About character relations :

- The relation **RELATIONID** is of the type **RELATIONSHIP**.
- The character **SOURCE** has a timeless relation with **TARGET** of the type **RELATIONID**.
- The character **SOURCE** started a permanent relation with **TARGET** of the type **RELATIONID** at the time **DATEID**.
- The character **SOURCE** had a relation with **TARGET** of the type **RELATIONID** that started on **START** and ended on **ENDDATE**.

### About places :

- The place **PLACEID** is called **PLACENAME**.
- The map **MAPID** has a width of **WIDTH** split into **NUMWIDTH** sections, and a length **LENGHT** split into **NUMLENGTH** sections.
- The place **PLACEID** is represented by the map **MAPID**.
- The place **PLACEID** is locate on the square **SQUAREID** of the **MAPID** map.
- The character **CHARACTERID** originates from **PLACEID**.

### About events :

- The event **EVENTID** is named **NAME**.
- The event **EVENTID** is described as **DESCRIPTION**.
- The character **CHARACTERID** attended **EVENTID**.
- The event **EVENTID** happened at **PLACEID**, started on **BEGINNING**, and ended on **ENDDATE**.

## 6 SQL script

In the zip file, you can find files `structure.sql` which builds the entire structure of the database in SQL. This script is idempotent, and is build with the same structure as the tutorial-D script. Indeed, we first create all the tables with the primary keys/unique keys, then we alter them to add the foreign key constraints.

The file `data.sql` is also available in the zip file. This one is also idempotent. It builds the structure of the database as the previous file, then it fills the database with all the elementary facts presented above in the report.

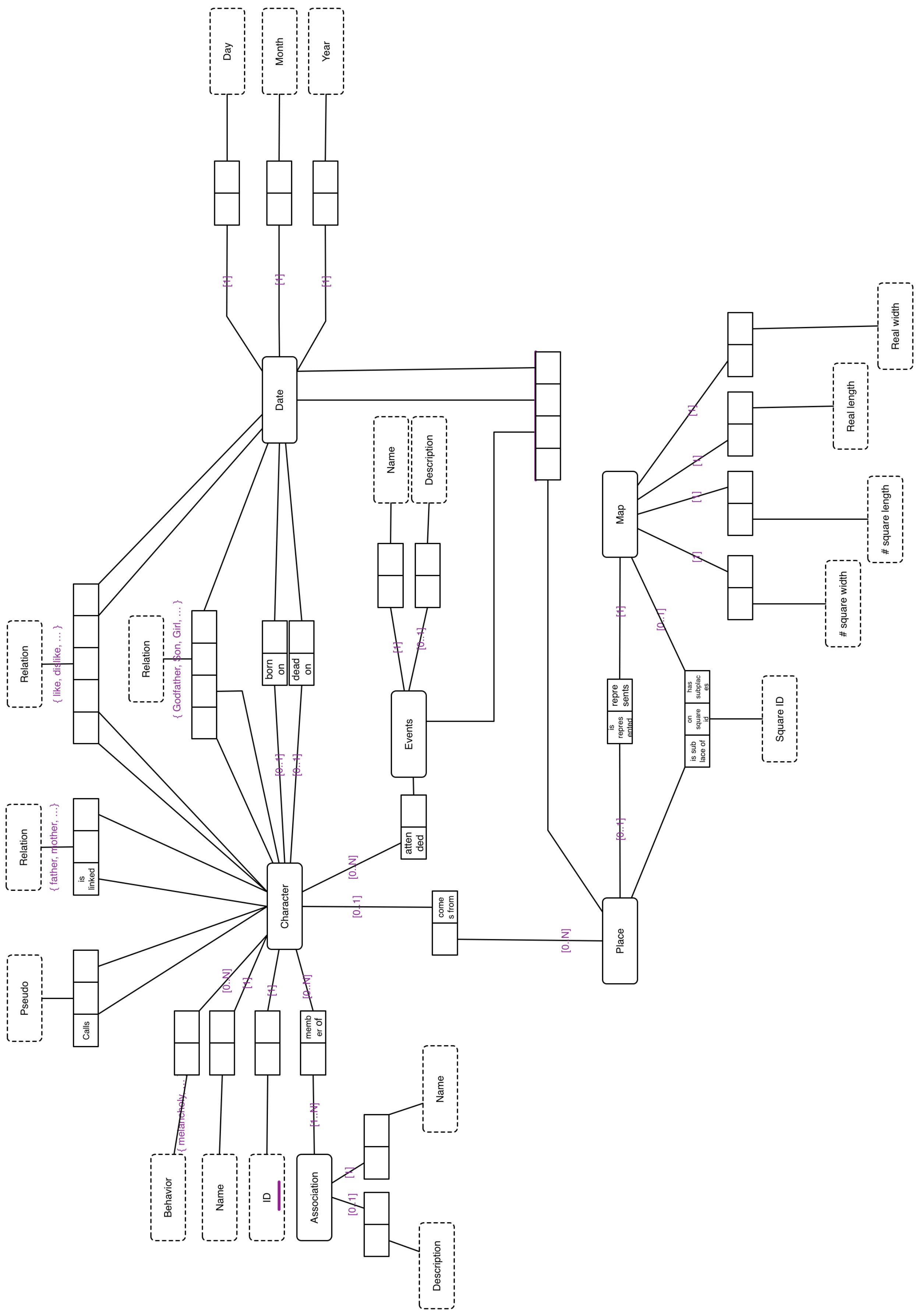

**Figure 1** – ORM Schema