Towards a Configurable Database Design: A Case of Semantic Data Warehouses

Selma Khouri[1,2] and Ladjel Bellatreche[1]

[1] LIAS/ISAE-ENSMA - University of Poitiers, 86960, Futuroscope Cedex, France
{selma.khouri,bellatreche}@ensma.fr
[2] National High School for Computer Science (ESI), Algiers, Algeria

Abstract. Many modern software systems are designed to be highly configurable. The configuration contributes in managing evolving software and controls the cost involved in making changes to software. Several standards exist for software configuration management (IEEE 828 and IEEE 1042). Unfortunately, making database configurable did not have the same spring as for software even though it can be seen as a software product. Nowadays, we are assisting to an explosion of new deployment layouts and platforms. This situation pushed the database community to admit the slogan: "one size no longer fits all". This motivates us to study the issue to make database design configurable. To satisfy this objective, we need to perform the following three tasks: **(i)** a deep understanding of the database design life-cycle, **(ii)** a formalization of each phase and **(iii)** an identification of the interactions between these phases. In this paper, we detail these tasks by considering the case of designing semantic data warehouses.

1 Introduction

Data are one of the most valuable resources that provide companies a serious competitive advantage. A wide range of DBMS products have been proposed in order to manage data efficiently. They are used in a large variety of new markets including conventional databases, data warehouses, scientific databases, semantic databases, social networking applications, etc. These different of databases can be summarized in one type that we call a *database product* ($\mathcal{DBP}$). The shareable design life-cycle of $\mathcal{DBP}$s includes three main phases: *conceptual design*, *logical design* and *physical design*. Its constitution has been made through a long evolution process. It starts from an *embryo* representing the physical design phase. This embryo underwent and is undergoing three main evolutions: *vertically*, *horizontally*, and *internally*. The vertical evolution adds new phases to the life-cycle. The addition of the conceptual and logical phases in the 70's to this embryo is an example of this evolution. Another example concerns the ETL phase (Extract, Transform, Load) that has been added to the life-cycle once the data warehouse technology has be launched. The *horizontal* evolution diversifies each design phase by adding new storage schemes (according to given data models), database architectures (conventional DB, *NoSQL* BD, Semantic DB,

R. Meersman et al. (Eds.): OTM 2014, LNCS 8841, pp. 760–767, 2014.
© Springer-Verlag Berlin Heidelberg 2014

etc.), deployment platforms (ex. Centralized, Cloud), etc. The *internal* evolution defines the steps of each design phase. We observe that the life-cycle is evolving and unfortunately most important design approaches and methodologies do not follow this evolution. Consequently, it is hard for a designer to understand and choose the alternatives offered by these evolutions, to meet her/his requirements.

A recent tendency emerging in the marketplace is to offer a *configurable storage engine*[1] (seen as ERP solution), which can be used across a broad application class, that can be tuned to the requirements of individual applications. Software configuration management[2] encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after the software engineering process. Making a configurable storage engine may be performed according two visions: (a) making the DBMS configurable or (b) making the design cycle configurable. We believe that the second option is more suitable. This vision is motivated by the citation of Michael Stonebraker who argues that "*one size no longer fits all*"[3]. We claim that a configurable database is feasible if the whole phases of the life-cycle are well formalized and their interactions are well captured. Our proposition in this paper is to bring us closer to this configurable engine solution. Because such a solution is very ambitious, we decided to start by focusing our study on *Semantic Data Warehouse* ($\mathcal{SDW}$) systems. We propose to revisit their proposed design cycle in order to answer the two following questions: **(Q1)**: what is the most relevant design choice for $\mathcal{SDW}$ systems? and **(Q2)**: how can designers deploy the $\mathcal{SDW}$ system using different logical/physical representations?

We believe that answering these two questions contributes in providing a configurable engine allowing designers to choose the most appropriate design option. The proposed design cycle dedicated to $\mathcal{SDW}$ systems is composed of five design phases [4]: requirements definition, conceptual design, logical design, ETL (Extract, Transform, Load) phase and physical design. The three central phases of $\mathcal{DW}$ design cycle have been borrowed from traditional database life-cycle (conceptual, logical, physical). We claim that the cycle revisit cannot concern these phases. One of the key lessons in the DBMS field is that logical and physical data independence ensured by these abstraction levels are guarantees of good design. The revisit of the $\mathcal{SDW}$ design cycle will concern the additional phases, namely requirements definition and the ETL phase. Because existing design studies do not have a global life-cycle vision, they omit the impact of these two phases on the design process. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the design cycle formalization. Section 4 revisits the design cycle. Section 5 presents a case study using LUBM benchmark, that illustrates the revisit and proposes a design process fitting this

[1] `http://cacm.acm.org/magazines/2008/7/5376-beyond-relational-databases/fulltext`

[2] `https://files.ifi.uzh.ch/rerg/amadeus/.../ch23-SCM-addendum.pdf`

[3] `http://cacm.acm.org/blogs/blog-cacm/32212-the-end-of-a-dbms-era-might-be-upon-us/fulltext`