Data warehousing with IBM cloud db2 warehouse

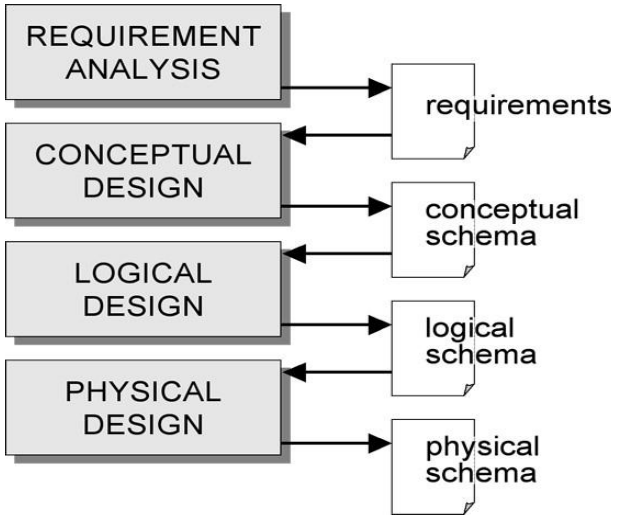
ABSTRACT

Multidimensional modeling requires specialized design techniques. Though a lot has been written about how a data warehouse should be designed, there is no consensus on a design method yet. This paper follows from a wide discussion that took place in Dagstuhl, during the Perspectives Workshop "Data Warehousing at the Crossroads", and is aimed at outlining some open issues in modeling and design of data warehouses. More precisely, issues regarding conceptual models, logical models, methods for design, interoperability, and design for new architectures and applications are considered.

1. INTRODUCTION

It is well known that data warehouses (DWs) are focused on decision support rather than on transaction support, and that they are prevalently characterized by an OLAP workload. Traditionally, OLAP applications are based on multidimensional modeling, that intuitively represents data under the metaphor of a cube whose cells store events that occurred in the business domain. Adopting the multidimensional model for DWs has a two-fold benefit. On the onehand, it is close to the way of thinking of data analyzers and, therefore, it helps users understand data; on the other hand, it supports performance improvement as its simple structure allows designers to predict users' intentions.

Multidimensional modeling and non-OLTP workloads require specialized design techniques. The most cited difference between design for transactional databases and DWs is denormalization, yet DW design has several other relevant peculiarities. Though a lot has been written about how a DW should be designed, there is no consensus on a design method yet. Most methods agree on the opportunity for distinguishing between a phase of *conceptual design* and one of *logical design*, like in Conceptual design aims at deriving an implementation-independent and expressive conceptual schema for the DW, according to the chosen conceptual model, starting from the user requirements and from the structure of the source databases. Logical design takes the conceptual schema and creates a corresponding logical schema on the chosen platform by considering some set of constraints (e.g., concerning disk space or query answering time). Several method also support a phase of *physical design*, that addresses all the issues specifically related to the suite of tools chosen for implementation – such as indexing and allocation. In some cases, a phase of *requirement analysis* is separately considered. From the functional point of view, the relationships between these phases can be summarized as in Figure 1 (in practice, this process will likely include feedback loops that allow to re-enter previous phases). Unfortunately, though most vendors of DW technology propose their own CASE solutions (that very often are just wizards capable of supporting the designer during the most tedious and repetitive phases of design), the only tools that currently promise to effectively automate some phases of design are just research prototypes this paper arises as an aterthought following a wide discussion that took place in Dagstuhl, during the Perspectives Workshop "Data Warehousing at the Crossroads" (August 2004). While the aim of the seminar was to discuss the current trends in data warehousing and to pave the way for future research in the whole field, here we will specifically focus on modeling and design, trying to answer the following question: "Has research on this topic come to an end? If not, what's left to do?" Thus, in this paper, benefiting from the fruitful discussions that took place there between



The core phases in DW design

all participants, we survey some topics related to DW modeling and design and outline the issues that, in our view, still need further exploration. More precisely, in Sections 2, 3, 4, 5, and 6 we address, respectively, conceptual models, logical models, methods for design, interoperability, and design for new architectures and applications.

2. CONCEPTUAL MODELING

Conceptual modeling provides a high level of abstraction in describing the warehousing process and architecture in all its aspects, aimed at achieving independence of implementation issues. Conceptual modeling is widely recognized to be the necessary foundation for building a database that is well-documented and fully satisfies the user requirements; usually, it relies on a graphical notation that facilitates writing, understanding, and managing conceptual schemata by both designers and users.

In the literature, conceptual modeling for DWs has been tackled from mainly two points of view so far:

- *Multidimensional modeling*. The existing approaches may be framed into three categories: extensions to the Entity-Relationship model extensions to UML While all models have the same core expressivity, in that they all allow the basic concepts of the multidimensional model to be represented, they significantly differ as to the possibility of representing more advanced concepts such as irregular hierarchies, many-to-many associations, and additivity.
- *Modeling of ETL*. The focus is to model the ETL process either from the functional the dynamic or the static point of view. Though the research on ETL modeling is probably less mature than that on multidimensional modeling, we believe that it will have a very relevant impact on improving the overall reliability of the design process and on reducing its duration.

While apparently a lot of work has been done in the field of conceptual modeling, we believe that some very important issues still remain open, as detailed in the following.

2.1 LOGICAL MODELING

Once the conceptual modeling phase is completed, the overall task of logical modeling is the transformation of conceptual schemata into logical schemata that can be optimized for and implemented on a chosen target system.

Considerable progress has been made in the area of multidimensional modeling, where target database systems are typically either relational or multidimensional. In relational implementations, the so-called star, constellation, and snowflake schemata are widely accepted to manage data cubes and are supported by various vendors. Concerning multidimensional implementations, several efficient multidimensional data structures such as condensed cubes and QC-Trees have been proposed to manage data cubes.

Nevertheless, we believe that some relevant challenges remain for future research, as summarized in the following subsections.

2.2 Semantic gap

With respect to fact modeling, there still is a semantic gap between advanced conceptual data models and relational or multidimensional implementations of data cubes. For instance, no commercial solutions can cope with generalization/specialization relationships in OLAP hierarchies . Additionally, it appears to be an open problem how to represent dimension constraints or even less expressive context dependencies , both of which explain the existence of null values in dimensions in logical implementations and allow to reason about summarizability with respect to *sets* of attributes. Moreover, a systematic treatment of summarizability addressing general aggregate functions beyond SUM remains an open issues. Consequently, future research is necessary to bridge this semantic gap, i.e., to preserve all information captured by advanced conceptual multidimensional models in logical implementations. To this end, research could either investigate how to enrich meta-data for tool support in a systematic way or, more ideally, look for more expressive logical models while preserving good query performance. Clearly, without the support of more expressive logical models we cannot expect to achieve a streamlined design process that guarantees quality criteria (e.g., avoidance of inconsistent queries, control over null values, reduction of sparsity to be satisfied and seriously takes security into account.

2.3ETL modeling

The transformation of conceptual ETL schemata into logical ones as well as their optimization are not very well understood. Indeed, present first steps towards the modeling and optimization of ETL processes at the logical level appears to be the only design method that includes an algorithmic transformation of conceptual into logical models.

Moreover, research on DW self-maintainability and independence has shown how to set up DWs in such a way that the maintenance processes can be simplified and made more efficient by avoiding maintenance queries. However, a combination of these results with ETL modeling techniques is still missing.