Research Problems in Data Warehousing

Jennifer Widom

Department of Computer Science
Stanford University
Stanford, CA 94305-2140
widom@db.stanford.edu

Abstract

The topic of data warehousing encompasses architectures, algorithms, and tools for bringing together selected data from multiple databases or other information sources into a single repository, called a *data warehouse*, suitable for direct querying or analysis. In recent years data warehousing has become a prominent buzzword in the database industry, but attention from the database research community has been limited. In this paper we motivate the concept of a data warehouse, we outline a general data warehousing architecture, and we propose a number of technical issues arising from the architecture that we believe are suitable topics for exploratory research.

1 Introduction

Providing integrated access to multiple, distributed, heterogeneous databases and other information sources has become one of the leading issues in database research and industry [6]. In the research community, most approaches to the data integration problem are based on the following very general two-step process:

1. Accept a query, determine the appropriate set of information sources to answer the query, and generate the appropriate subqueries or commands for each information source.
2. Obtain results from the information sources, perform appropriate translation, filtering, and merging of the information, and return the final answer to the user or application (hereafter called the *client*).

We refer to this process as a *lazy* or *on-demand* approach to data integration, since information is extracted from the sources only when queries are posed. (This process also may be referred to as a *mediated* approach, since the module that decomposes queries and combines results often is referred to as a *mediator* [20].)

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.
CIKM '95, Baltimore MD USA
© 1995 ACM 0-89791-812-6/95/11..$3.50

The natural alternative to a lazy approach is an *eager* or *in-advance* approach to data integration. In an eager approach:

1. Information from each source that may be of interest is extracted in advance, translated and filtered as appropriate, merged with relevant information from other sources, and stored in a (logically) centralized repository.
2. When a query is posed, the query is evaluated directly at the repository, without accessing the original information sources.

This approach is commonly referred to as *data warehousing*, since the repository serves as a warehouse storing the data of interest.

A lazy approach to integration is appropriate for information that changes rapidly, for clients with unpredictable needs, and for queries that operate over vast amounts of data from very large numbers of information sources. However, the lazy approach may incur inefficiency and delay in query processing, especially when queries are issued multiple times, when information sources are slow, expensive, or periodically unavailable, and when significant processing is required for the translation, filtering, and merging steps. In cases where information sources do not permit ad-hoc queries, the lazy approach is simply not feasible.

In the warehousing approach, the integrated information is available for immediate querying and analysis by clients. Thus, the warehousing approach is appropriate for:

- clients requiring specific, predictable portions of the available information
- clients requiring high query performance (the data is available locally at the warehouse), but not necessarily requiring the most recent state of the information
- environments in which native applications at the information sources require high performance (large multi-source queries are executed at the warehouse instead)
- clients wanting access to private copies of the information so that it can be modified, annotated, summarized, and so on, or clients wanting to save information that is not maintained at the sources (such as historical information)

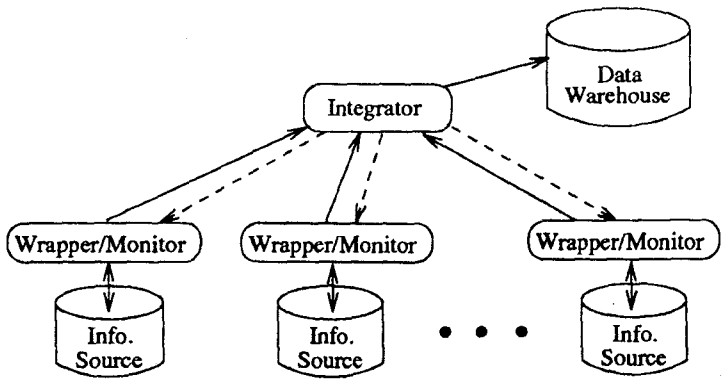

Figure 1: Basic architecture of a data warehousing system

The lazy and warehousing approaches are each viable solutions to the data integration problem, and each is appropriate for certain scenarios.[1] The database research community has focused primarily on lazy approaches to integration. In this paper we consider research problems associated with the warehousing approach.

2 Industrial Perspective

Before considering the research problems associated with data warehousing, we note that there has been great interest in the topic within the database industry over the last several years [12]. Most leading vendors claim to provide at least some "data warehousing tools," while several small companies are devoted exclusively to data warehousing products. Despite rapid advances in commercial data warehousing tools and products, most of the available systems are relatively inflexible and limited in their features. We believe that a truly general, efficient, flexible, and scalable data warehousing architecture requires a number of technical advances, outlined below.

The importance of data warehousing in the commercial segment appears to be due to a need for enterprises to gather all of their information into a single place for in-depth analysis, and the desire to decouple such analysis from on-line transaction processing systems. Analytical processing that involves very complex queries (often with aggregates) and few or no updates—usually termed *decision support*—is one of the primary uses of data warehouses, hence the terms data warehousing and decision support often are found together, sometimes interchanged.[2] Since decision support often is the goal of data warehousing, clearly warehouses may be tuned for decision support, and perhaps vice-versa. Nevertheless, decision support is a very broad area, so we focus this paper specifically on research issues associated with the warehousing approach to integration.

[1] Another promising and relatively unexplored approach to information integration is a *hybrid* approach, in which some information is stored in a centralized repository while other information is fetched on demand, e.g., [21].

[2] Other relevant terms include *data mining*, *on-line analytical processing* (*OLAP*), and *multidimensional analysis*, which we view as refinements or subclasses of decision support.

3 Architecture of a Data Warehousing System

Figure 1 illustrates the basic architecture of a data warehousing system. The bottom of the diagram shows the information sources. Although the traditional disk shapes connote conventional database systems, in the general case these sources may include non-traditional data such as flat files, news wires, HTML and SGML documents, knowledge bases, legacy systems, and so on. Connected to each information source is a *wrapper/monitor*. The *wrapper* component of this module is responsible for translating information from the native format of the source into the format and data model used by the warehousing system, while the *monitor* component is responsible for automatically detecting changes of interest in the source data and reporting them to the integrator.

When a new information source is attached to the warehousing system, or when relevant information at a source changes, the new or modified data is propagated to the *integrator*. The integrator is responsible for installing the information in the warehouse, which may include filtering the information, summarizing it, or merging it with information from other sources. In order to properly integrate new information into the warehouse, it may be necessary for the integrator to obtain further information from the same or different information sources. This behavior is illustrated by the downward dashed arrows in Figure 1.

The data warehouse itself can use an off-the-shelf or special purpose database management system. Although in Figure 1 we illustrate a single, centralized warehouse, the warehouse certainly may be implemented as a distributed database system, and in fact data parallelism or distribution may be necessary to provide the desired performance.

The architecture and basic functionality we have described is more general than that provided by most commercial data warehousing systems. In particular, current systems usually assume that the sources and the warehouse subscribe to a single data model (normally relational), that propagation of information from the sources to the warehouse is performed as a batch process (perhaps off-line), and that queries from the integrator to the information sources are never needed.

anges at the ware-
ges should be han-
or modifications to
housing system as

formation: As in
ltiple, heterogene-
is the likelihood of
e information from
n the same or dif-
tion from multiple
rlier, we described

clients will demand schen
house itself. All of these
dled with as few disrupti
other components of the
possible.

- **Duplicate and inconsiste** any environment involvin ous information sources, encountering copies of th multiple sources (represe ferent ways), or related in sources that is inconsister

When the integrator receives a change notification, in order to integrate that change into the warehouse the integrator may need to fetch additional data from the same or different sources. (As a simple example, if the warehouse joins two relations R and S, and there is a notification of an insert to relation R, then the inserted tuple must be joined with the contents of relation S.) Issuing queries to sources can incur a processing delay, the queries may be expensive, and such queries are the basis of the warehouse maintenance "anomalies" alluded to in Section 4.2 [22]. Even worse, when information sources are highly secure or when they are legacy systems, administrators will not allow anything to be installed that may affect the performance of the systems, or that may require modifications to the way they are used. Such sources may only support dumps, or *snapshots*, of the data are provided off-line, and changes are detected by comparing successive snapshots.

4.1 Wrapper/Monitors

The *wrapper/monitor* components illustrated in Figure 1 have two interrelated responsibilities:

1. **Translation:** Making the underlying information source appear as if it subscribes to the data model used by the warehousing system. For example, if the information source consists of a set of flat files but the warehouse model is relational, then the wrapper/monitor must support an interface that presents the data from the information source as if it were relational. The translation problem is inherent in almost all approaches to data integration—both lazy and eager—and is not specific to data warehousing. Typically, a component that translates an information source into a common integrating model is called a *translator* or *wrapper* [3, 20].[3]

2. **Change detection:** Monitoring the information source for changes to the data that are relevant to the warehouse and propagating those changes to the integrator. Note that this functionality relies on translation since, like the data itself, changes to the data must be translated from the format and model of the information source into the format and model used by the warehousing system.

One approach is to ignore the change detection issue altogether and simply propagate entire copies of relevant data from the information source to the warehouse periodically. The integrator can combine this data with existing warehouse data from other sources, or it can request complete information from all sources and recompute the warehouse data from scratch. Ignoring change detection may be acceptable in certain scenarios, for example when it is not important for the warehouse data to be current and it is acceptable for the warehouse to be off-line occasionally. However, if currency, efficiency, and continuous access are required, then we believe that detecting and propagating changes and incrementally folding the changes into the warehouse will be the preferred solution.

In considering the change detection problem, we have identified several relevant types of information sources:

- **Cooperative sources:** Sources that provide triggers or other *active database* capabilities [19], so that notifications of changes of interest can be programmed to occur automatically.

- **Logged sources:** Sources maintaining a log that can be queried or inspected, so changes of interest can be extracted from the log.

Each type of information source capability provides interesting research problems for change detection. For example, in cooperative sources, although triggers and active databases have been explored in depth, putting such capabilities to use in the warehousing context still requires addressing the translation aspect; similarly for logged sources. In queryable sources, in addition to translation, one must consider performance and semantic issues associated with polling frequency: If the frequency is too high, performance will degrade, while if the frequency is too low, changes of interest may not be detected in a timely way. In snapshot sources, the challenge is to compare very large database dumps, detecting the changes of interest in an efficient and scalable way [13]. An important related problem in all of these scenarios is to develop appropriate representations for the changes to the data, especially if a non-relational model is used [4].

Finally, we note that a different wrapper/monitor component is needed for each information source, since the functionality of the wrapper/monitor is dependent on the type of the source (database system, legacy system, news wire, etc.) as well as on the data provided by that source. Clearly it is undesirable to hard-code a wrapper/monitor for each information source participating in a warehousing system, especially if new information sources become available frequently. Hence, a significant research issue is to develop techniques and tools that automate or semi-automate the process of implementing wrapper/monitors, through a toolkit or specification-based approach [16].

4.2 Integrator

Assume that the warehouse has been loaded with its initial set of data obtained from the information sources. (The task of setting up and loading the data warehouse is discussed in Section 4.5 below.) The ongoing job of the integrator is to receive change notifications from the wrapper/monitors for the information sources and reflect these changes in the data warehouse; see Figure 1.

At a sufficiently abstract level, the data in the warehouse can be seen as a *materialized view* (or set of views), where the *base data* resides at the information sources. Viewing the problem in this way, the job of the integrator is essentially to perform *materialized view maintenance* [9]. Indeed, there is a close connection between the view maintenance problem and data warehousing [15]. However, there are a number of reasons that conventional view maintenance techniques cannot be used, and each of these reasons highlights a research problem associated with data warehousing:

- In most data warehousing scenarios, the views stored at the warehouse tend to be more complicated than conventional views. For example, even if the warehouse and

[3] Most commercial data warehousing systems assume that both the information sources and the warehouse are relational, so translation is not an issue. However, some vendors do provide wrappers for other common types of information sources