CHAPTER-6

DATA WAREHOUSE
CHAPTER 6

DATA WAREHOUSE

6.1 INTRODUCTION

Data warehousing is gaining in popularity as organizations realize the benefits of being able to perform sophisticated analyses of their data. Recent years have seen the introduction of a number of data-warehousing engines, from both established database vendors as well as new players. The engines themselves are relatively easy to use and come with a good set of end-user tools. However, there is one key stumbling block to the rapid development of data warehouses, namely that of warehouse population. Specifically, problems arise in populating a warehouse with existing data since it has various types of heterogeneity. Given the lack of good tools, this task has generally been performed by various system integrators, e.g. software consulting organizations which have developed in-house tools and processes for the task. The general conclusion is that the task has proven to be labor-intensive, error-prone, and generally frustrating, leading a number of warehousing projects to be abandoned mid-way through development.

However, the picture is not as grim as it appears. The problems that are being encountered in warehouse creation are very similar to those encountered in data integration, and they have been studied for about two decades. However, not all problems relevant to warehouse creation have been solved, and a number of research issues remain. The principal goal of this paper is to identify the common issues in data integration and a data-warehouse creation. Developers of warehouse creation tools to examine and, where appropriate, incorporate the techniques developed
for data integration, and researchers in both the data integration and the data warehousing communities to address the open research issues in this important area.

6.2 ENTERPRISE DATA MANAGEMENT ARCHITECTURE

The Standard modern enterprise data management architecture for a large organization i.e., day-to-day, as well as strategic, i.e., long term, data management and analysis needs. It consists of operational data management systems which support existing applications by managing current data. It also consists of a corporate data warehouse and a number of data marts on which various kinds of strategic analyses are performed. In the following, we briefly describe the various elements of the architecture.

• Operational Data System

These are traditional relational and other data systems, which are used for the day-to-day operations of the organization. They contain up-to-date, and provide interactive access, both for transaction and decision-support systems. The database size can range from tens of megabytes to a few gigabytes, which though large in itself is quite small compared to the size of the corresponding warehouse. Since these databases are regularly updated, usually by concurrently executing applications, a critical goal for them is to maintain transaction consistency. In addition, since many of the application are mission critical, e.g., airlines and banking, there is usually a need to mask system failures from the application. Finally, due to interactive access, response time and throughput requirements are stringent.
• **Warehouse Creation**

For a warehouse to be used effectively, it is important to pay sufficient attention to its creation. As shown in Fig. 1 this includes architecture selection, warehouse schema creation, and warehouse population. These issues are discussed in detail in subsequent sections.

• **Corporate Data Warehouse**

This is the principal repository of historical information in the organization, and stores data instances for the enterprise data model. Data is entered into this repository periodically, usually in an appended-only manner. Sometimes the data here may directly be used for analysis. However, in most cases focused sets of data are extracted into smaller data marts where they are analyzed. Since this is often the definitive data in the organization, in many instances such as warehouse has also been called the foundation data. Since data is usually entered to extracted in a batch mode, interactive response is not a significant issue.

• **Data Marts**

Data marts are smaller data warehouses, usually focusing on a small subset of the enterprise data. Typically each mart is used by a particular unit of the organization for various strategic analyses relevant to its goals. Data is extracted from the corporate warehouse into the data mart periodically, and used for analysis. Interactive response is an issue in data marts as interactive analysis tools work directly on the data.
• **Warehouse Analysis Tools**

A number of tools have been built for performing strategic analysis of warehouse data. As shown in Fig. 2 these include trend analysis, data mining, simulation, forecasting, and on line analytical processing.

6.3 **ISSUES IN DATA WAREHOUSE CREATION**

A number of issues must be addressed in building a data warehouse. Some are associated with the creation of the warehouse and other with its operation. In the following, we discuss the issues affecting warehouse creation.

• **Warehouse Architecture Selection**

A number of architectural approaches to data warehousing are possible, and selection between them is affected by factors such as size, nature of use, etc.

• **Database Conversion**

Shown in Fig. 2, this architectural choice involves taking all the data in the source systems and performing a conversion to the (integrated) target.

**Fig. 21 Database conversions**
System. This approach is applied when: the source system are to be retired and all the data is being moved to the target system, and both the source and target systems will continue to be in operation, and this is the initial population of the target database. The data in the course systems is mapped into a global model and then written out/exported to the target database. In this approach, bulk data conversion is being performed, and hence optimization of the processing is important. Integrating data from multiple sources can require comparison between all pairs of data items, i.e., the cost can in general be quadratic with respect to the input size. Since input data can typically be in the order of tens of millions or higher this optimization is quite critical.

Fig. 3. Database Synchronization

Fig. 3 shows the synchronization architecture. In this case, there is an existing warehouse, and data is extracted periodically from the operational source systems to update the target system, thus synchronizing the target system to the source system.

The key differences between this architecture and the previous one are:

1. The volume of data handled in much smaller, and
2. Information being produced for the target database already has its counterpart from before, with which it may require integration.

The main issue is one of using incremental algorithms to optimize the processing. A lot of recent work in materialized view maintenance is relevant to this problem.

- **Federated Database**

In some cases the numbers of users of the enterprises-wide integrated data are few, and having a separate full-fledged data warehouse may be overkill. In addition, there may sometimes be a requirement for having the target database be completely up-to-date. In such cases, a federated database (FDB) may be the best architectural choice, as shown in Fig. 4 in this architecture, only the data required by a query is integrated. Hence, the optimization focuses on handling small amounts of data as well as fast interactive response time. Since query processing and data integration must be handled together, the optimizer must consider both simultaneously.

- **Enterprise Schema Creation**

With any of the data warehouse architectural choices discussed above, the first step is to develop an enterprise schema. The enterprise schema is the output of enterprise data modeling, and captures the main entities about which the enterprise maintains information and relationships between them. It is invariably the case that various units of an organization have existing schemas for their respective portion of the enterprises-wide information. These are used as the starting point for the enterprise schema. Thus, an important task for enterprise it, and filling out any missing elements, In addition, data warehouses provide explicit
support for the historical dimension as well as aggregation, which must be modeled as special entities. Finally, the union of the enterprise data may give rise to new entities which must be modeled in the schema.

- **Enterprise Data Model**

A data model for the enterprise must be first developed, to identify the key entities in the organization and their relationships to each other. In an ideal situation, such modeling would begin by analyzing the various processes in the organization and extracting the relevant entities and relationships from them. However, in most situations it is much more realistic to use the existing entities and relationships, modeled by the schemas of existing components data sources, as a starting point. Analysis of various organizational processes helps in refining the enterprise data model. Experience has shown that there is no perfect enterprise data model, and hence this task can never be 100 percent complete. Thus, once a reasonable model has been obtained it must be put into operation, with provision for subsequent modifications.

- **Schema Integration**

In creating the enterprises data model, an all too often encountered situation is one shear multiple, independently developed database model different aspects of the same set of real-world entities. However, having been developed independently, the schemas do not match well. Structural heterogeneity is one class of discrepancies which arise, e.g., different field sizes, units, or scales. Semantic heterogeneity is another class of discrepancies, which includes the synonym and the homonym problems. An example of the former is the names EMP and EMPLOYEE being used to refer to the same class of real-world employees in two different databases. An example of the latter is the name EMP being used
to refer to real-world employees in one database, and to real world employers in another.

The reason for these discrepancies is that the designers of the respective databases were targeting different sets of applications, even though for the same set of real-world entities, and hence captured different (but overlapping) models of reality in their databases. Hence, overall the schema integration problem is one of integrating multiple overlapping models of a set of real world-entities. Specific technical approaches include automatic and semiautomatic ways to determine synonyms, antonyms, IS-A relationships, etc. A large body of related work exists in the database and knowledge representation community.

- **Constraints**

In addition to the structural and semantic mismatches of schema entities, there is a additional problem of constraint mismatches, which are often not evident from the definitions of entity types. For example, one database may have a constraint such as EMP. Age<= 60 while another may have the constraint on age as EMP. Age <=65. In integrating such databases there seems to be in general no right approach to resolving such constraint incompatibilities. For example, we can take the approach that the new constraint should be EMP. Age <= 65 since it is the eaker of the two, and hence all data is guaranteed to satisfy it. However, tightness of constraints in a schema is usually an indication of the quality of the data in the database, and such an approach eventually degrades the overall quality of the data to its lowest common denominator.

An alternative approach is to select some tighter constraint at integration time, based on existing constraints in participating databases, and
perform the requisite data quality improvement through the use of tools and human intervention, to ensure that the integrated data has at least this data quality. Using an object model for the integrated database, another possible approach is to define two employee sets, namely one with EMP. Age <= 60 and the other with EMP. Age <= 65 with an IS-A relationship between the two sets. There has been some initial work in addressing these issues.

Warehouse Population

One the component schemas have been integrated to develop the global schema of the warehouse, with acceptable resolution of schema mismatches, the next step is to populated the warehouse with data. Experience has shown that even though the schemas may have been integrated, there may still be problems in integrating the specific data instances.

Fig. 4 Federated database.
Semantic Issues

A number of semantic discrepancies arise in integrating data instances from multiple sources. A specific problem is entity identification, namely determining whether a pair of records coming from two different databases represents the same or different real-world entities. This becomes a difficult problem because usually there is no common key which can be used as an identifier for the union of the two sets of records.

Another is the attribute value conflict resolution problem. This occurs when records from different databases have been matched, i.e., entity identification has occurred, but it is found that values of the same attribute for an entity instance, coming from different data sources, are different. There can be a number of possible sources of such errors, e.g., data-entry errors, different policies of database maintenance, and modeling of slightly different concepts. As data warehouses are being built, experience with such problems is being generated, and various ad hoc solutions have been proposed. A systematic study of semantic heterogeneity issues in warehouse population has only just begun.

- Scalability

Since warehouses store information about the database as it progresses over time, they tend to grow much more rapidly than on-line databases. It is quite common to start with an initial warehouse of size 10-100 gigabytes, and subsequently have a periodic (weekly or monthly) update rate of 1-10 gigabytes. The processing described above for warehouse population must be carried out for the initial population as well as for every periodic update. The data integration tasks typically range in complexity from \(O(n \log n)\) to \(O(n^2)\) for \(n\) data items. Given tens to hundreds of gigabytes of data, this can be very time consuming, and
hence there is a need for improved algorithms for data integration tasks. Furthermore, since these tasks are heavily set-oriented, data-parallel computing techniques appear promising, and should be investigated.

- **Incremental Updates**
  As data is added to an existing warehouse, it must be integrated with pre-existing data. The success of incremental update techniques in on-line databases must be extended to this type of processing. Examples of such techniques include incremental algorithms and indices. As of now hardly any work has been done in this area.