DIRECT: A Query Facility for Multiple Databases

ULLA MERZ and ROGER KING
University of Colorado

The subject of this research project is the architecture and design of a multidatabase query facility. These databases contain structured data, typical for business applications. Problems addressed are: presenting a uniform interface for retrieving data from multiple databases, providing autonomy for the component databases, and defining an architecture for semantic services.

DIRECT is a query facility for heterogeneous databases. The databases and their definitions can differ in their data models, names, types, and encoded values. Instead of creating a global schema, descriptions of different databases are allowed to coexist. A multidatabase query language provides a uniform interface for retrieving data from different databases. DIRECT has been exercised with operational databases that are part of an automated business system.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages; H.2.7 [**Database Management**]: Database Administration

General Terms: Languages, Management

Additional Key Words and Phrases: Data models, design, heterogeneous databases, query languages

1. INTRODUCTION

This article presents the architecture and design of a multidatabase query facility known as DIRECT. DIRECT, an interactive software system for querying heterogeneous databases, has been the vehicle for exploring and realizing this design.

It is not unusual that a company maintains order information, inventory data, and customer credit ratings in different databases. Depending on how long ago these databases and their applications were developed and on the requirements that governed the choice of database management system, the inventory data may be stored in an IMS database and the customer credit ratings in a DB2 database. Generating a report that lists all items on order,

This work was funded by an IBM Shared University Research Grant.
Authors' addresses: U. Merz, Micro Decisionware, 3035 Center Green Drive, Boulder, CO 80301-5404; R. King, Department of Computer Science, University of Colorado, Campus Box 430, Boulder, CO 80309.

their shipping date, and the total charge given the customer's credit conditions requires retrieving and merging data from three databases.

A software system is needed that retrieves data from heterogeneous databases. Many services, such as network communication services, transaction-processing services, and semantic services are needed to automate the retrieval of data stored in multiple databases. Each service has to resolve different aspects of heterogeneity.

The focus of this research effort is on the semantic services [Segev 1991] whose objective is to provide a uniform application and user interface in spite of heterogeneous database schemas and data manipulation languages. In addition, the semantic services must address the problem of identifying data with the same meaning duplicated and distributed across several databases.

It is possible to reconcile differences in data models and query languages by defining a global schema using a common data model. This solution favors centralized control. A second solution defines a uniform query language; it is characterized by providing autonomy and extensibility. Inevitably, new database technologies with new data models will be introduced as software systems adjust to changing business needs. Therefore, it is necessary that an architecture for semantic services provides autonomy and extensibility.

Independent of the solution chosen, semantically equivalent data elements must be identified and specified to make it possible to join and merge data from different databases. Two or more data elements containing atomic printable data values that represent the same real-world fact are semantically equivalent[1] if their data values belong to the same domain. Also, data values in different domains are semantically equivalent when functions exist (not necessarily invertible) for mapping or converting them from one domain into the other.

Users familiar with the application are in the best position to identify semantically equivalent data elements. Their understanding of the data names and comments in the data definitions, and their knowledge about the applications using the data are important for judging which data elements contain semantically equivalent or related data.

Given the considerations stated above, the proposed solution is based on the following design decisions:

(1) Allow the coexistence of different data models

—to guarantee extensibility and
—to provide autonomy for the individual databases.

(2) Unify heterogeneous databases through multidatabase queries

—to retrieve data as needed and
—to create alternative views of existing data definitions.

[1] This definition does not consider semantic equivalence of database constructs, such as sets, tables, or relations.

(3) Define an architecture for semantic services

—to help a user to specify a multidatabase query and

—to provide a cooperative problem-solving environment for identifying semantically equivalent and related data elements.

DIRECT is a query facility for heterogeneous databases. Since coexistence is the basis for autonomy, allowing different database descriptions and data models to coexist is at the center of this solution. Coexistence also provides extensibility, because new data-modeling constructs do not have to be mapped into the semantically equivalent data-modeling constructs of a global schema. In contrast to previous solutions, DIRECT does not create a global schema, but maintains the syntax and semantics of the data definitions for the individual databases. Instead of defining static functions for mapping semantically equivalent data elements, DIRECT provides human-computer interaction techniques that assist users in identifying them.

The proposed architecture for semantic services focuses on the user, whose objective is to report data stored in multiple databases and whose judgment is needed to identify semantically equivalent data elements. The architecture is based on an architecture for cooperative problem-solving systems. Their goal is to assist users in tasks that are based on human judgment rather than on analytical rules.

DIRECT has been developed using an iterative design process [Gould 1988]. Three prototypes have been developed, two of which were evaluated for their usability in informal experiments. The purpose of the usability tests was to understand the user's task of specifying a multidatabase query. The usability tests played an invaluable role in defining the solution. Understanding the user's task of specifying a multidatabase query led to identifying a suitable user interface architecture. In turn, this architecture served as the basis for the proposed architecture for semantic services.

The following section describes related research efforts. Section 3 provides an overview of the proposed architecture for semantic services and its particular realization in DIRECT. Section 4 describes a sample scenario using DIRECT, and Section 5 provides an assessment and summary of the research results.

2. DIRECT IN PERSPECTIVE

This section summarizes solutions offered by research in database management systems. Differences to the proposed solution are outlined.

2.1 Schema Integration

The principles of removing redundancy and imposing centralized control have influenced the research in schema integration. The global schema trades autonomy of the individual databases for centralized control. Schema integration methodologies [Batini et al. 1986] achieve uniformity by creating a virtual, global schema based on a semantic data model. The individual database schemas are mapped into the constructs and notation of this semantic data model and are then merged into a global schema. The result is a single description in a uniform notation that assumes a single representation will meet all data needs.

The capability to change existing database schemas or the extensibility to add databases with new data-modeling constructs is not easily achieved by schema integration methodologies. Because global schemas are static, schema integration methodologies do not easily accommodate changes to existing database schemas or the addition of new schemas. Also, there is no guarantee that all future data-modeling constructs can be mapped into the constructs of a particular semantic data model.

Allowing different database descriptions and data models to coexist has several advantages. The autonomy of the individual databases is preserved by retaining the original data names and constructs. Additionally, new data-modeling constructs are more easily accommodated, because existing data definitions do not have to be mapped into semantically equivalent constructs of a particular data model.

2.2 Multidatabase Query Languages

Because database autonomy is important, Litwin and Abdellatif [1987] propose to dispense with the global schema. Instead of creating a uniform description of the different database schemas, a multidatabase query language provides a uniform interface for joining and merging data values from different databases. Multidatabase query languages refer to data names as defined in the individual databases rather than the data names of a global schema.

Current proposals [Krishnamurthy et al. 1991; Litwin and Abdellatif 1987] extend the relational query model with new language features to specify semantically equivalent data elements, to resolve schematic differences, and to convert data values. In particular, the multidatabase query language MDSL [Litwin and Abdellatif 1987] uses multiple identifiers and semantic variables to define semantically equivalent data elements. The design of the multidatabase query language IDL [Krishnamurthy et al. 1991] focuses on creating different views for resolving schematic differences.

These proposals do not address the problem users have identifying semantically equivalent data elements and creating multidatabase queries from English statements. They also do not discuss the issues involved in extending these languages to data models other than the relational model.

This research project defines a multidatabase query language based on relational algebra. Also, it is shown that semantically equivalent data elements can be defined implicitly through query functions such as join and union. Instead of providing statements for specifying semantically equivalent data elements, human-computer interaction techniques are provided that help users to identify them.

3. THE ARCHITECTURE OF DIRECT

Part of the research effort involved implementing a prototype to validate the feasibility of the design decisions stated above. The prototype DIRECT was used to explore an architecture for semantic services. An architecture for semantic services addresses the following problems:

—differences in query languages,

—differences in data models, and

—semantically equivalent data elements duplicated and distributed across several databases.

A number of research projects and commercial products define approaches to resolve the semantic heterogeneity of different databases (see Figure 1). Schema integration methodologies, surveyed by Batini et al. [1986], define a global schema with a uniform data model and a methodology for defining semantically equivalent data elements. Multidatabase query languages, such as MDSL and IDL, define a common high-level query language with extensions for specifying semantically equivalent data elements. The commercial product Ingres/Star [Ingres Corp. 1991] provides a relational data dictionary that creates a uniform name space for all data elements. A software system, called Carnot [Rasmus 1991], unifies all database descriptions in a knowledge-based system that includes functions to match and map semantically equivalent data elements automatically; database-specific query languages are mapped into a canonical representation. This research effort investigates yet another alternative by keeping the database schemas and their data models separate and by making it a joint effort of the software system and the user to identify interactively the semantically equivalent data elements.

The following sections propose an architecture for the semantic services. Particular components of this architecture, as implemented in DIRECT, are discussed also.

3.1 An Architecture for Semantic Services

The functional components of an architecture for the semantic services (see Figure 2) are similar to those of Design Environments as described in Lemke [1989]. The architecture of Design Environments defines interaction techniques and visual representations that help users to solve problems and make decisions; it allows users to bring their application domain knowledge into the solution process. A Design Environment consists of the following functional components:

Parser. The parser makes previously created design specifications available by transforming them and importing them into the Design Environment.

Work Area. The work area provides a visual representation of the design specification under construction.

Palette. The palette provides a visual representation of the parts or tools that can be used to create a design specification.

Catalog. The catalog stores a collection of design specifications that can be reused and modified.

Critic. The critic alerts users to invalid descriptions and suggests alternative solutions during the construction process.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.