Virtual Database Technology

Ashish Gupta, Venky Harinarayan, Anand Rajaraman
Junglee Corporation

1. Introduction

Virtual Memory (VM) technology was invented to make your computer's disk drive behave as an extension of its main memory. Before the advent of Virtual Memory, most programs were limited by the physical main memory available on the computer. Sophisticated programmers used elaborate schemes – such as overlays and chaining – to work around this limitation. These schemes not only complicated the logic of the program, but were also error-prone and non-portable, and lengthened the development cycle for complex applications.

VM fundamentally transformed computing, by placing almost unlimited amounts of cheap memory at the disposal of computer applications. Most computers have limited amounts of main memory but virtually unlimited amounts of storage capacity in their disks, because secondary storage is an order of magnitude cheaper than main memory. All this storage is now available to the application programmer as if it were a part of the computer's physical memory. Thus, the VM breakthrough has led to the development of much more sophisticated programs than had been possible before.

Similarly, virtual database (VDB) technology makes external data – such as the World Wide Web – behave as an extension of an enterprise's relational database (RDBMS) system. According to some estimates, as much as 90% of the world's data is outside of relational database systems. Vital data is scattered across web sites, file systems, database systems, and legacy applications. These data sources differ in the way they organize the data, in the vocabulary they use, and in their data-access mechanisms. Many of them do not even support native query operations. Writing applications that combine data from these sources is a complex, often impossible, task because of the heterogeneity involved.

Junglee's VDB technology can fundamentally transform enterprise computing by providing a solution to this *data scatter* problem. VDB technology lets applications ask powerful queries of data that is scattered over a variety of data sources. The VDB gathers, structures and integrates the data from these disparate data sources and provides the application programmer with the appearance of a single, unified relational database system. VDB technology brings the Internet to enterprise applications and will lead to the development of a new breed of applications that can use *all* the data.

As an illustration of the applications enabled by VDB technology, consider job hunting on the Web. In order to make a meaningful career choice, a job seeker needs information on available opportunities as well as related data – such as

information on housing, school districts, and crime statistics in the job area. Information on job openings is scattered across thousands of different web sites – company home pages and several aggregate sites, such as newspaper classifieds sites. Keyword search capabilities on words appearing in the job listing are the only available search choice.

Using an application based on VDB technology, the job seeker can now obtain answers to the following query posed to the Web, "find marketing manager positions in a company that is within 15 miles of San Francisco and whose stock price has been growing at a rate of at least 25% per year over the last three years". This single query would span the Web employment listings of many corporations, in addition to web sites that have geographical mapping information and websites that contain historical records of corporate equity prices. The query would also return, for each position, related information including statistics on housing prices, school districts, and crime statistics.

VDB technology is particularly effective in enterprise settings when combined with data warehousing. Using VDB technology, corporations can include data from nontraditional sources (e.g., files in directory systems) and external sources (e.g., the World Wide Web) in the data warehouse, enabling key decision support applications.

2. Technology Architecture

Junglee's unique patent-pending VDB technology makes disparate, heterogeneous information sources behave like a single relational database system. As shown in Figure 1, VDB has two components: the *data integration system* and the *data publishing system.*

2.1 The Data Integration System

The data integration system combines data from several underlying data sources – web sites on the Internet, text files in a local directory hierarchy, legacy applications, and relational database systems – and provides a unified, consistent relational database interface. This complex task is performed by three functional elements:

Wrappers:
A wrapper interfaces with a data source (such as a World-Wide Web site) and makes it behave like a collection of tables in an RDBMS. Using a high-level site description in Junglee's *Source Description Language (SDL),* wrappers can be created with minimal effort and only descriptive programming. SDL is a language specifically designed to describe the behavior of data sources. A few lines of SDL can capture the structure of even the most complicated web site, including sites whose pages are dynamically generated in response to filled out forms. In

addition, it also captures relationships between hyperlinked pages in a web site and reflects the relationships in the contents of the virtual database tables corresponding to the site. Using a wrapper, any data source can be queried using the industry-standard SQL query language.

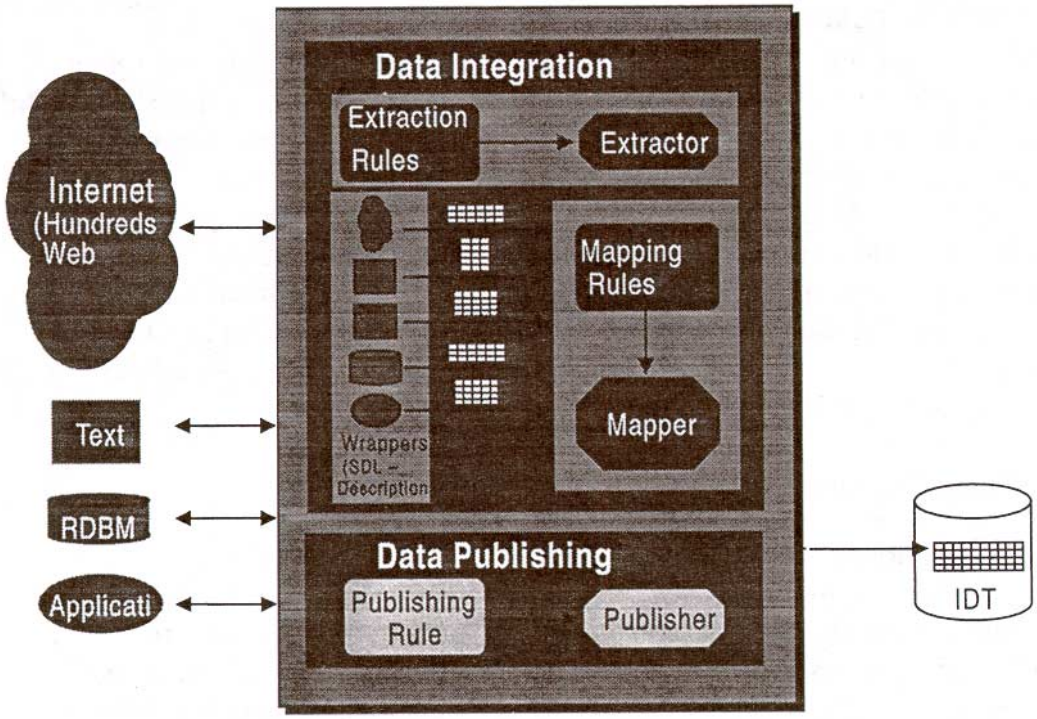

Figure 1

The Mapper:
Wrappers can make arbitrary data sources behave like relational data tables, but these tables are likely to have inconsistent schemas and vocabularies.
As a simple example, consider a scenario where some wrappers export *salary* in *$/month*, others export it in *$/week,* and the application wants *pay* in *$/year*. The required attribute name and unit conversions are handled by the mapper. The data transformations to be performed by the mapper are described as a collection of mapping rules in *Map Description Language (MDL)*. MDL can succinctly describe very powerful data transformations, including schema and vocabulary transforms that combine data from several tables, transforms that create new table columns based on the values of other columns, and data cleansing and formatting transforms.

The Extractor:
Data integration often involves extracting structure from unstructured textual data. For example, consider a newspaper web site that lists apartments for rent. The application

needs a table with columns for features such as number of bedrooms, number of bathrooms, location, and rent. However, each apartment classified listing is typically a block of undifferentiated text. Extraction rules describe how to extract the required features from the text. Extraction rules in *Extract Description Language (EDL)* can combine powerful text processing and linguistic analysis to deliver near-perfect feature extraction.

2.2 The Data Publishing System

While the data integration system is the heart of a VDB system, the data publishing system is its brain. The publisher uses *publishing rules* to schedule data acquisition, transformation, and dissemination. The publisher schedules the various data integration tasks – wrapping, mapping, and extraction – in order to publish integrated data tables (IDTs) at regular intervals (daily or weekly). An IDT is a snapshot of the data integrated by the data integration system from its data sources, and includes relational data tables and multimedia objects such as images. The IDT can be used directly within the VDB system, or it can be delivered over the Internet and can be used to populate and periodically update a data warehouse of integrated data.

The data publishing system has the following features:

Flexible scheduling:
Each data source has a different appropriate time for acquiring data from it, depending upon its update schedule and usage patterns over the day or week. The publishing system supports a very general notion of *tasks,* where a task can involve querying a wrapper, transforming its data using mapping rules, or populating RDBMS tables with transformed and cleaned data. Tasks can be scheduled to run when most appropriate, and to repeat at any required frequency (for example, daily or weekly).

Dependency tracking:
Publishing rules can capture inter task dependencies to create legal schedules. For example, the data from a wrapper needs to be acquired before any transformations are performed on it.

Failure recovery:
When autonomous, external data sources are involved, handling failures and unexpected events gracefully becomes important. Web sites can go down unexpectedly or change their structure. Network connections can break. Computer systems may crash. The publishing system guarantees that such failures are handled appropriately: when a data source is unreachable, for example, its wrapper is queried periodically until the source becomes available. And in all cases, including system crashes, the publishing system maintains the integrity of the acquired data.

Data Slicing:

Often, different applications require different slices of the same integrated data. The publishing system makes it easy to create slices tailored for each application.

2.3 Putting it all together: the application

A remarkable feature of VDB technology is the ease with which applications can be built. The simplest way to create an application based on VDB technology is to use the IDT. The IDT is a standard relational data table in a standard RDBMS – such as Oracle, Informix, Sybase, or Microsoft SQL Server. The publisher can be used to schedule periodic updates to the data snapshot. Standard Rapid Application Development (RAD) tools such as PowerBuilder, Delphi, or Visual Basic can now be used to build powerful applications on top of the RDBMS. Standard Web-database connectivity tools such as Netscape LiveWire and NetDynamics can be used to provide a web front-end to the data. The RDBMS will be transparently updated by the publishing system, so that the application will always see fresh data.

For some applications, the underlying data sources have extreme and rapid variations. For example, consider an application that uses stock quote data. The application needs real-time data. Such applications can bypass the publishing system and directly query the data integration system for the freshest data from the underlying sources. This bypass mechanism is called *dynamic querying*. The data integration system exposes the industry standard Java Database Connectivity (JDBC) API for dynamic querying.

3. Conclusion

VDB technology enables rapid deployment of applications with at least one of the following characteristics:

- Large numbers of data sources
- Data sources are autonomous, there is no centralized control
- Data sources can have a mixture of structured and unstructured data

The World Wide Web, and most Intranets, have all of these characteristics, and thus can benefit from Virtual Database technlogy.