Enabling distributed enterprise integration with WebSphere and DB2 Information Integrator

by C. M. Saracco
M. A. Roth
D. C. Wolfson

Information technology architects increasingly find themselves searching for better ways to access, integrate, and leverage their information, applications, and business processes. Information integration, in particular, is critical to the community of Web-based businesses, as firms that are able to leverage their information resources most effectively are best positioned to emerge as leaders in their industries. In this paper, we explore how companies can solve this complex business challenge by extending the reach of WebSphere® technology with DB2® Information Integrator (II). DB2 II offers WebSphere developers a new approach to coping with diverse and distributed information sources, enabling them to reduce programming costs, shorten development cycles, and attain reasonable levels of performance for server-side components that need to integrate information throughout their enterprises and partner networks.

Industry analysts project that corporate spending for enterprise business integration will represent a $5.6 billion industry by 2006.[1] These projections are driven by the large numbers of incompatible systems, software applications, databases, and deployed technologies commonly used within an enterprise. Nowhere is this integration challenge more evident than in the Web-based business environment, where information integration, in particular, can offer critical advantages.

Decades of advances in information technology (IT) have enabled many firms to improve business efficiency and pursue new business opportunities. Processes that once took days or weeks to complete may now take minutes or seconds. Business models that were once unthinkable are now standard, including just-in-time inventory management, real-time global collaboration, and 24-hour customer self-service. Yet along with the productivity gains, added revenues, and increased flexibility that these IT advances have brought, problems have been introduced as well.

Because many of the systems supporting these new business models were introduced sporadically and adopted on an as-needed basis, many companies are now faced with an array of disparate systems, applications, and data management software. A complex web of ad hoc integration frequently emerges to support the flow of information among these applications. Ongoing business changes continue to increase the tangle and complexity of interrelationships among applications. This situation is already impeding the ability of many companies to evolve their existing systems to accommodate new business requirements and organizational needs.

Recognizing this, a number of IT vendors have jumped into the fray and offered various software integration technologies and services. Unfortunately, many such technologies and services have been narrowly scoped and prohibitively expensive or require substantial staff training to implement. Recognizing this, many companies are demanding a strong, stan-

dards-based integration solution that enables them to leverage their existing IT assets as well as position themselves for future growth. Furthermore, given the current state of the global economy, firms are understandably reluctant to make heavy, up-front investments in integration technologies and services that may take years to pay off.

IT architects, managers, and developers responsible for Web-based applications are among the first to be confronted with this challenging business and technical reality. These professionals must cope with short development cycles, possess a high degree of technical expertise spanning multiple disciplines, and fulfill aggressive demands involving the scalability, availability, and reliability of their software environment. Business needs often dictate that Web applications and internal business processes must access, consolidate, integrate, and leverage information dispersed throughout a company, its business partners, and even public domain resources.

Critical technical challenges. To solve this business problem, IT architects must confront a variety of technical issues. Chief among these are:

1. Finding ways to present server-side-Java** and Web-application developers with a unified view of disparate data that is physically distributed across multiple systems, some of which may reside in the public domain or in partner-managed environments.

2. Bridging the disparate programming and data models favored by business analysts, Web developers, and database administrators.

3. Achieving acceptable levels of performance for mission-critical applications, each of which may employ distinct workloads.

In the following sections, we consider each of these issues in detail. Common application domains, such as customer relationship management (CRM), supply chain management (SCM), and enterprise resource planning (ERP), often seek to present a single, unified view of logical business entities to their users. Such entities might include customers, accounts, orders, and so on. Unfortunately, critical data appropriate to these business objects is often dispersed through different systems. Someone—or some software component—must locate the necessary data, access it using the appropriate application programming interface (API), resolve any discrepancies as needed, and consolidate the data into the desired form.

Locating the data can require modest or substantial effort, depending on how (and where) the data is managed and what degree of documentation is available about its characteristics. Retrieving (or modifying) the data almost always presents a challenge: syntactic, semantic, and performance issues must be taken into consideration to prevent runtime errors, ensure efficient use of system resources, and achieve acceptable levels of response time and throughput. However, even after the necessary remote data is successfully located and accessed, it often must be translated, transformed, and modified before being suitable for broader use. Such work may require substantial programming, as well as considerable knowledge about the source and target problem domains. Finally, merging, correlating, and consolidating the data effectively requires sophisticated design skills and complex programming logic, as this effort can easily encompass multiple join, sort, and aggregation techniques.

Of course, the problem of providing an integrated view of disparate data is further hindered by the varying ways in which consumers of this data want to view it. Two of the most popular ways for data consumers to view persistent information are as business objects and server-side Java objects (in particular, entity Enterprise JavaBeans** [EJBs**]). This can introduce an "impedance mismatch" due to differences between object-oriented programming languages, such as Java, and relational languages, such as SQL (Structured Query Language), particularly in the area of record versus set-based processing.[2,3] Commercial implementations of some relational database management systems (DBMSs) can present further challenges for object-oriented programmers if advanced features such as support for a flexible type system and complex data structures are absent or poorly implemented.

Fortunately, vendors and researchers have been exploring various techniques for coping with or overcoming this impedance mismatch, including advanced data modeling and mapping tools, DBMS enhancements for object/relational constructs, and greater collaboration on industry standards. Such efforts have helped improve the situation in recent years, and further improvements are likely in the future.

However, this impedance mismatch is not the only challenge associated with providing users with a unified view of disparate data. Quite often, data associated with business objects or server-side Java objects do not reside in a single, common data store. Instead, most critical business data are stored in one or more relational DBMSs, as well as pre-relational DBMSs (such as IMS* [Information Management System], IDMS [Integrated Database Management System], Adabas**, and Datacom/DB) and a variety of file systems. In some application domains, such as the life sciences, critical data may even be generated dynamically by search algorithms, public domain software, and proprietary partner-based applications. This diversity of actual or perceived data representations presents a further challenge that any integration technology must address. Extensibility, embeddable components, and industry-standard interfaces are some of the technical issues that apply to bridging this gap.

Finally, providing the base capability to integrate disparate data is of little use if the resulting environment will not perform well. For example, while a call center representative may clearly benefit from being able to view a customer's past problem reports and the status of each, this benefit is only achieved if the comprehensive customer data is available in real time. Waiting hours or days for such information renders it virtually useless.

To achieve acceptable performance for real-time information integration, solution providers must be capable of generating new, accurate data access strategies dynamically in response to incoming requests. Doing so implies the need to consult and leverage appropriate meta-data, including information about access methods available for different data sources, the quantity of data likely to be retrieved from a given data source, the degree to which such data must be cleansed or transformed before being ready for use, and so forth. Of course, this task is, in many respects, an extension of cost-based query optimization work that relational DBMS researchers have been developing for years.[4,5]

Background. For more than a decade, software vendors and researchers have explored various aspects of information integration and its challenges. For example, in the early 1990s, relational DBMS vendors such as IBM, Oracle, and Sybase began shipping products that supported database replication and distributed data access.[6] In the latter area, offerings of that time were often marketed as "next-generation gateways," "data access middleware," and "multidatabase servers."[7] These commercial products, although limited in scope, represented early attempts to resolve specific information integration problems facing customers. For example, multidatabase servers typically focused on providing a single SQL API to a select number of data sources (usually relational DBMSs) and enabling users to formulate a single query spanning multiple disparate systems.[8]

More recently, other forms of information integration technology have surfaced in commercial products. Many are designed to appeal to XML (eXtensible Markup Language) or Java programmers, and some minimize the degree of DBMS-based (or "schema-driven") technologies implemented in their integration architectures. BEA Systems[9] and Nimble Technology[10] both offer products that provide for XML-based integration of data residing in various sources. Snapbridge supports the definition of semantic objects containing rules and attributes for exposing disparate data as XML services and deriving composite data sets[11]; such technology is sometimes referred to as "schema-less" data federation. MetaMatrix's product line[12] focuses on meta-data management and distributed query support, relying on Web application servers (such as WebSphere*) for deployment services. These newer forms of information integration technology, although intriguing in many respects, are limited in scope compared to the architecture we describe in this paper. Their limitations include failing to adequately leverage the query filtering and optimization capabilities of remote data sources, restricting information access through an uncommonly used or relatively unstable API, providing only modest support for legacy data sources, and providing limited ability to effectively cache or replicate data as needed.

Broader technology initiatives, such as workflow processing and application integration frameworks, have also included mechanisms for integrating data.[13] While the level of data integration support in these initiatives varies depending on the specific technology domain and its commercial implementation, these initiatives typically provide limited transparency for data source access and require more manual effort to analyze, correlate, and manipulate data spanning multiple disparate sources.

Research in the broad field of information integration has been quite active. Such diverse projects as TSIMMIS[14] (The Stanford IBM Manager of Multiple Information Sources), HERMES[15] (Heterogeneous

Reasoning and Mediator System), DISCO[16] (Distributed Information Search Component), Pegasus**,[17] and Garlic[18] explored specific technologies critical to furthering the field of information integration. TSIMMIS and HERMES, for example, implemented special-purpose query processing systems to provide "mediators" capable of integrating data from specific sources. DISCO focused on scalability issues involved with supporting many heterogeneous data sources. Pegasus employed its own data model and query language to provide users with considerable flexibility for modeling various data sources, and Garlic featured support for object-oriented concepts, global query optimization, and an extensible wrapper architecture. Today, DB2 Information Integrator (II) has built upon Garlic technology, as well as earlier federated database technology supported in DB2 DataJoiner,[19] to deliver sophisticated information integration capabilities commercially.

Focus of this paper. While commercial and research efforts (such as those cited previously) have helped further the field of information integration technology in various ways, none provides a comprehensive solution to the integration challenges faced within many enterprises. In our view, coping with this challenge requires a solution spanning Web application servers, application integration technology, and information integration servers. In this paper, we define and explore a comprehensive integration architecture for the distributed enterprise. The first section describes a common business scenario that demands such an integration solution. With this in mind, we propose an appropriate division of responsibility and work to be delegated to the mid-tier server (a WebSphere Application Server instance) and the back-end information server (a DB2 II instance).

Defining a comprehensive enterprise information integration architecture

While many researchers and software vendors are pursuing various means of addressing the challenge of integrating information throughout a distributed enterprise, most have been taking a rather narrowly focused approach in doing so. Such an approach, possibly adequate several years ago, is not a viable long-term solution currently. In presenting an alternative, comprehensive approach, we address the following questions: What are the key components of a comprehensive, enterprise-wide architecture for integrating information? What division of processing responsibility is appropriate among its components? What commercial products might support such an architecture today, and what capabilities can they offer to IT organizations facing complex enterprise integration projects?

A sample business scenario. To better understand the need for a comprehensive integration architecture, it may be useful to consider a common business problem. Economic forces often drive companies to merge or establish close partnerships. As a result, business processes and information that were independently developed and managed must be integrated to support the new business model under which the linked companies must operate. Starting over from scratch is rarely an option, because firms must continue to function and secure new business while undergoing a merger or forming a partnership. This implies that existing data and applications must continue to be supported, forcing firms to assemble an integrated view of critical data that is spread across different systems, captured in different data formats (including different relational DBMSs, e-mail systems, Web pages, and the like), and accessible through different APIs. In some cases, access to this data is driven primarily through prepackaged or custom-built applications, which themselves define different data models and present different APIs.

The key to coping with this challenge involves developing an enterprise integration solution that will allow existing data and applications to continue to function unchanged while presenting a consolidated view of this disparate environment to all those who require it. This includes IT professionals developing new applications, business analysts developing corporate strategies, and even customers or third parties who need to conduct business with the new business entity as a whole.

In a sample business scenario, a parts distribution firm acquires two of its smaller rivals within the course of a year. The firm's top business priorities include providing a single on-line catalog for all available parts and developing new marketing and sales strategies appropriate for the company's entire customer base. Both tasks would be relatively straightforward to perform if data from all three original firms were stored in a single, centralized DBMS. But the underlying data is spread across multiple systems and stored in different formats: some is in DB2, some is in another relational DBMS, and some is in a spreadsheet or flat file system. Each has slightly different native APIs and different levels of natively supported

search capabilities. Thus, developing a single virtual representation of a customer and his or her order status (including prior order history) can be challenging. Similar situations arise when trying to pinpoint the cheapest supplier of a given part in a given region of the world or trying to develop a reward program for firms that make frequent, large purchases across the newly merged company.

Overview of a solution

To address the challenges just discussed, the firm would need to design and deploy an IT infrastructure architecture capable of supporting new Web applications spanning multiple divisions as well as legacy applications that may support only a single division (i.e., pre-merged firm). The use of a Web application server is likely to be considered, as such products enable firms to isolate and manage critical business services in a mid-tier server, exploit built-in support for common system services, and leverage industry-standard computing initiatives, such as J2EE** (Java 2 Platform, Enterprise Edition) and Web services. Furthermore, since critical data is spread across diverse sources, the firm would need to consider how best to fulfill the data access and integration requirements of components running in the Web application server tier.

Migrating the data into a single data source is an unlikely short-term option, given that legacy applications employing source-specific APIs and data structures must be supported. Furthermore, maintaining multiple copies of the data may be an option for portions but not for all of the data, due to data latency issues. For at least some of this firm's planned work, real-time access to disparate data is necessary.

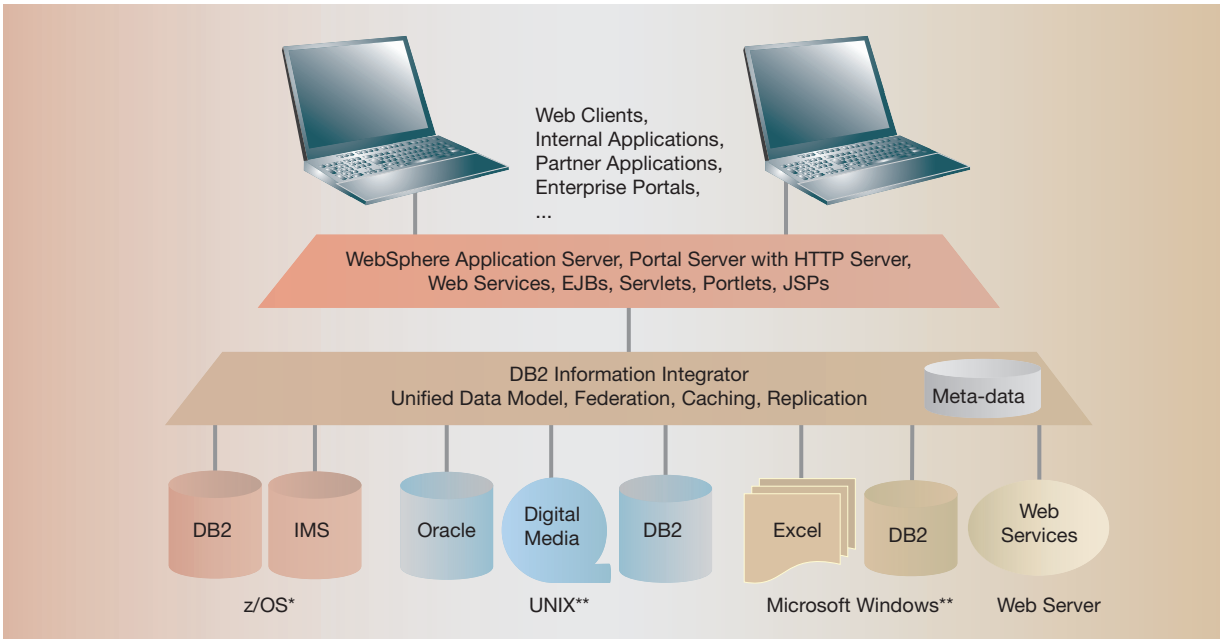
Combining a Web application server with an information integration server provides a powerful architectural solution for coping with the problems just discussed. Information integration servers such as DB2 II enhance and extend the role of relational DBMSs to include virtual data management services, global optimization of access to dynamically generated as well as persistent data, and extended metadata management support. Furthermore, information integration servers can support on demand computing initiatives, enabling administrators to dynamically reconfigure the server environment to access new data sources as needed.

Figure 1 illustrates a sample solution using the WebSphere Application Server and the WebSphere Portal Server to model business objects, host Web services, support custom portal applications, and deploy underlying J2EE components such as EJBs, servlets and JavaServer Pages (JSPs). Web clients, internal applications, business partner applications, and enterprise portals can all use XML or Java to access services deployed in the WebSphere tier. WebSphere services that need to work with external data exploit the unified data model provided by DB2 II to simplify (and potentially speed) retrieval and consolidation of disparate data, including persistent data or dynamically generated data. Note that the unified data model can even integrate external data exposed by Web services, which many search mechanisms, proprietary applications, workflow engines, and transformation engines now support.

This delineation of roles between the application server and the information integration data server enables each to assume responsibility for what it does best. Application servers, as the name implies, were originally designed to provide critical business services for specific applications. Such services include modeling business objects in a straightforward manner, providing for high levels of code reuse, promoting code portability, and offering built-in support for common infrastructure services, such as transaction management and security. Data servers, as one might imagine, were developed to manage shared access to data. Relational DBMSs—the most popular type of data servers for years—provide for data independence, efficient read/write access to persistent data, and support for data integrity.

If software vendors were starting from scratch, an ideal solution might be to construct a software offering that provides all of the services just described. However, given the current state of customer deployments, we believe such an attempt would be impractical. Application servers and data servers are widely deployed in many customer environments today, each fulfilling critical business needs. Migrating data, applications, and application components to a new hybrid server is a labor-intensive proposition that many customers would be loath to undertake. Indeed, the vast majority of medium- to large-sized firms rely on multiple data servers from multiple vendors to meet their diverse data management needs; a number also have multiple application servers from different vendors deployed to support different projects. Thus, a single application-server/data-server offering would only serve to duplicate critical functions which are already in place within many enterprises. Furthermore, as a practical matter, such an

Figure 1 Solution using WebSphere Application Server and WebSphere Portal Server



* Trademark or registered trademark of International Business Machines Corporation.
** Trademark or registered trademark of The Open Group or Microsoft Corporation.

offering would likely provide only a subset of the functions available through existing commercial application servers and data servers, as the expertise and effort required to design, develop, test, and deploy a superset of available capabilities would be cost-prohibitive for any vendor to develop in a short time.

Instead, we propose a more pragmatic approach, in which state-of-the-art application servers and information integration servers function cooperatively to fulfill the needs of a distributed enterprise. In time, we expect this cooperation to evolve and increase, enabling each type of server to pioneer technologies distinct to its domain and ultimately take advantage of the enhancements provided by the others. Such an approach promises to provide customers with a greater range of capabilities more quickly than would be possible by attempting to retrofit or duplicate function from one type of server to another. With this in mind, we explore in greater detail the roles we envision for the application-server and information-integration-server tiers.

Role of the application server tier. The application server tier plays a critical role in a comprehensive integration solution, providing services for deploying reusable business objects and their underlying components, managing workflows that orchestrate business processes, and accessing persistent data stores as needed to support these tasks. It is this last role that overlaps most significantly with the role of an information integration server, and we will pay particular attention to this area in this paper. However, to establish the context in which this work is relevant, we discuss the broad role of the application server tier first. Popular J2EE programming constructs, such as EJBs and servlets, enable application developers to isolate and represent common business entities and related services in a manner that separates reusable business logic from application-specific needs. Session beans, for example, can contain logic representing common business functions, such as checking a customer's credit or registering a customer's complaint. Because such functions may be useful to a number of applications within an enterprise, it is preferable to develop and deploy such services only once, thereby enhancing the consistency of corporate processes, minimizing redundant code, and simplifying software maintenance. The application server is well suited to providing runtime ser-

vices for these business functions: the object-oriented programming model employed by Web application servers facilitates easy transformation of business object models to software objects; built-in infrastructure services for transaction management, security, and the like keep coding efforts relatively low; and standard APIs employed by these servers, including those for XML, Java, and Web services, support client access from a variety of traditional and emerging platforms.

With common business services represented and managed at the application server tier, it makes sense to manage the integration of these services at this software layer as well. Such integration can be managed through workflow technology,[20] such as WebSphere MQSeries* Workflow, and through enterprise application integration (EAI) offerings,[21] such as the WebSphere Business Integration product suite, which combines workflow technology with a broad range of ready-made interfaces to popular data sources and third-party applications. Flow models or activity graphs commonly employed by these technologies orchestrate (and enforce) the steps necessary to execute a business process, which is typically a complex, multistep series of activities.

Of course, simple business functions as well as complex workflows often need to work with persistent data managed by a variety of systems. For this reason, application servers enable J2EE programmers to deploy entity EJBs, which directly represent persistent data to server-side Java programmers. Entity EJBs with container-managed persistence (CMP) relieve programmers from hand-coding basic data access logic, as the EJB container running in the application server automatically generates the necessary code to perform these services.

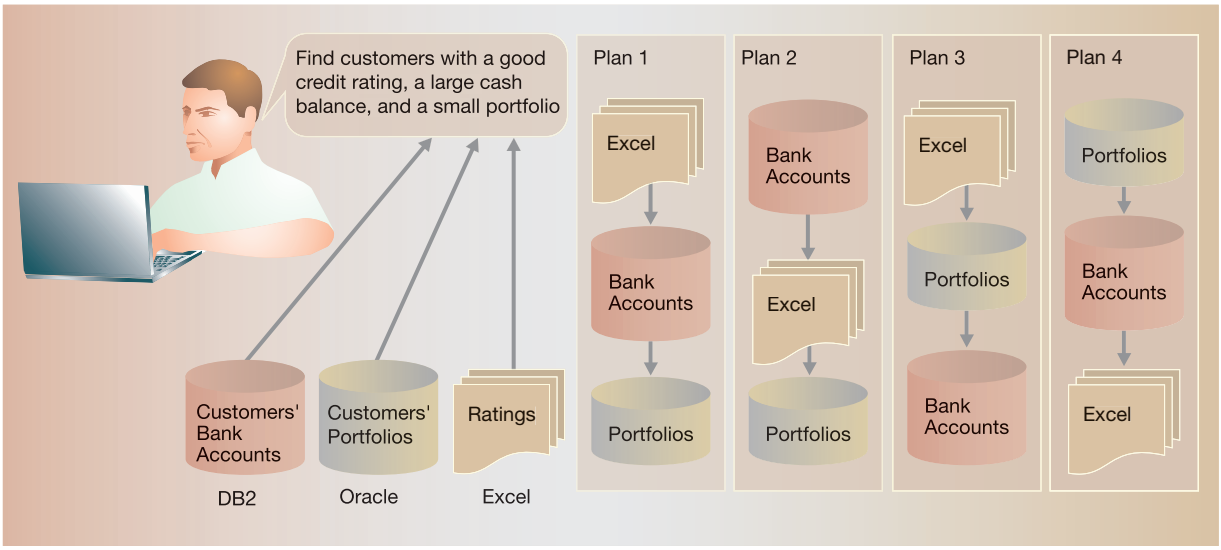
This raises a significant question: what data access and integration work should Web application servers undertake, and what should they delegate to an information integration server? We believe Web application servers can quite adequately assume responsibility for working directly with persistent data when a very limited number of data sources are involved and access requirements are rather simple, such as inserting a row into a given database table or searching for a specific object (row) based on a given primary key value. Indeed, such operations are characteristic of entity EJBs, and Web application servers support these functions quite well. However, as the number and types of data sources increase, the task of integrating them becomes more complex. Clearly, as the data models, semantics, and capabilities of the sources to be integrated begin to diverge, the task of integrating that data at the application server tier becomes more challenging. Indeed, as we discuss later, undertaking this effort with as few as three data sources can result in substantial effort and a significant increase in skills when multiple joins, unions, and/or aggregations are necessary. In the following two sections, we will discuss some of the features of an information integration server tier and the benefits such a tier brings to enterprise-integration development projects, including greater application flexibility and improved data abstraction through the use of views.

Role of the information integration server tier. As discussed earlier, the application server tier is a reasonable platform for integrating business logic, orchestrating business processes, and supporting portal-based applications. Such activities depend heavily on accessing and manipulating relational data, which typically exists in large volumes and resides in diverse sources. The work of managing large-scale relational databases is indeed itself a complex task that is outside the scope of the application server. The role of the information integration server tier is to ease the burden of integrating, managing, and querying heterogeneous, distributed data by providing a scalable, robust integration infrastructure for all data within an enterprise, regardless of its source, format, or location.

A powerful architecture for an information integration server is one that leverages mature data management technology that has been at the core of most enterprise software systems. DB2 II is such an information integration server. A relational DBMS provides storage management as well as update and retrieval capabilities for relational data. DB2 II extends these core functions to handle data that may be stored, managed, and accessed by autonomous systems, both relational and non-relational. Building an integration solution using an existing DBMS platform provides significant advantage over other approaches. A DBMS-based approach extends decades of large-scale data management technology to heterogeneous data and can transparently exploit innovations in heterogeneous data as the core relational DBMS technology evolves. Some of the advantages of a DBMS-based approach include the following.

Providing a single interface to heterogeneous data. ODBC (Open Database Connectivity), JDBC** (Java

Figure 2 Customer financial account information



Database Connectivity), and SQL are among the most prevalent APIs that applications employ. Building an integration platform within the existing database management framework enables applications to use a standard API and a familiar data model to access data and systems that may support a variety of APIs and data models. Remote data sources are mapped to table-like objects, and database views that combine data from remote sources provide a powerful abstraction to shield applications from the details of working with diverse sources. Providing an SQL-based API to heterogeneous data sources has the added benefit of enabling database administration tools (such as DB2 Control Center), development tools (such as WebSphere Studio), and object-oriented components (such as ready-made portlets) to work transparently with disparate data. Looking to the future, as DBMSs acquire the capability to natively support an XML data model and the XQuery Language,[22] this function will extend to heterogeneous data.

Exploiting query processing and execution techniques with heterogeneous data. Significant investments have been made in query rewrite, optimization, and execution techniques to handle large-scale query processing for relational DBMSs. Enterprises with significant heterogeneous data integration challenges should be able to exploit these same techniques to combine data from multiple sources. As an example, consider the value that query optimizers provide. There are often many alternatives for retrieving and combining data to fulfill a particular request. The best alternative usually depends upon a variety of data characteristics, including cardinality, data location, and index availability. It is often a non-trivial task to determine the right strategy for a given application.

In Figure 2, we present a scenario in which customer account information for a financial institution is distributed among a DB2 database, an Oracle database, and an Excel spreadsheet. The investment broker wishes to identify new sales opportunities by finding customers with a large bank account balance, a small stock portfolio, and a good credit rating. The right side of this figure shows four alternatives for computing the results of the query. Plan 1 finds the set of customers with good ratings, checks to see if they have a large account balance, and then checks to see if they have a small stock portfolio. Plan 2 first finds customers with large account balances, verifies that they have a good rating, and then checks to see if they have a small stock portfolio. Other plans in the figure represent additional alternatives. The best strategy for a given query depends on how many customers have large account balances, how many have small portfolios, and how many have good credit ratings. DBMS query optimizers have been refined over the years to analyze numerous strategies and use a

variety of techniques to select an efficient data access strategy for a given query. An information integration server can extend and exploit these capabilities with heterogeneous data.

Providing multiple application-transparent options for data location. Integration projects have a variety of functional and performance requirements. As a result, they may require a variety of ways to place or locate the data to be integrated in a way that is transparent to applications. For example, two autonomous departments may need to share data, but neither may be willing to relinquish control over that data or allow it to be copied to a central repository. In another example, a business unit with a mainframe environment may need to provide Web-based access to its systems without affecting the load on the host system. In these and similar cases, an information integration server provides the following multiple options to manage the placement of heterogeneous data:

1. Consolidate external data into a centralized database. This is a traditional approach for data integration and may be appropriate for certain applications, particularly those that use small, unchanging data sets.

2. Replicate the data. This option allows multiple, nearly current copies of the data to reside in various data stores, enabling firms to geographically disperse applications that work with the same data to improve response time and availability.

3. Federate the data. This option provides real-time access to heterogeneous data as though it were stored in a single virtual database, even though the data remains remote and under the control of external systems.

4. Cache the data. This option allows an application to transparently access a nearly up-to-date copy of a data set or access data at the original source. The DBMS engine determines if the query can be satisfied from the cached copy or requires access to the original source and routes the query appropriately. This option may be appropriate for firms that wish to transparently locate heavily accessed data near the application but still have the flexibility to access the original source for less frequently used data.

Shifting the responsibility for managing data integration to the DBA, rather than the application developer. For typical enterprise applications, the tasks of modeling data, designing databases, managing data access, and tuning for performance, capacity, and availability require a special set of skills and the attention of a database administrator (DBA). An enterprise integration project involves similar tasks and poses additional challenges. Among these challenges are determining the best way to model data from disparate sources, the data access strategy that will yield reasonable performance, and how to shield applications from potential schema changes or API changes at each of the data sources. By using the information integration tier as the focal point for integration, firms can shield application developers from this complexity.

Leveraging the multitiered architecture

Earlier, we presented a business scenario involving a corporate merger that could force WebSphere developers to build J2EE-compliant applications that would support a single, virtual image of critical information that was dispersed across disparate systems. We now revisit this situation and consider how the architecture we defined can help simplify the development and deployment requirements of an integrated business solution.

By using DB2 II for data access and integration needs, IT organizations can avoid the expensive and time-consuming tasks of manually performing this work inside custom-built J2EE components, Web services, or portlets. Instead of connecting to multiple data sources individually, implementing the logic necessary to correctly retrieve the desired information from each in an efficient manner, and consolidating this data manually within the application space, programmers can issue a single query using nicknames or views (virtual tables) in DB2 II that span all necessary data sources. DB2 II takes care of the complexities involved with connection management services, query decomposition and rewrite services, global query optimization, and resolution of semantic differences between different data sources (which might include, for example, the treatment of nulls).

Figure 3 illustrates the enterprise integration architecture involving WebSphere Application Server and DB2 II that can be used to meet the business requirements presented in our sample scenario.

In the following discussion, we explore this architecture in more detail, concentrating on the interop-

Figure 3 Enterprise integration architecture

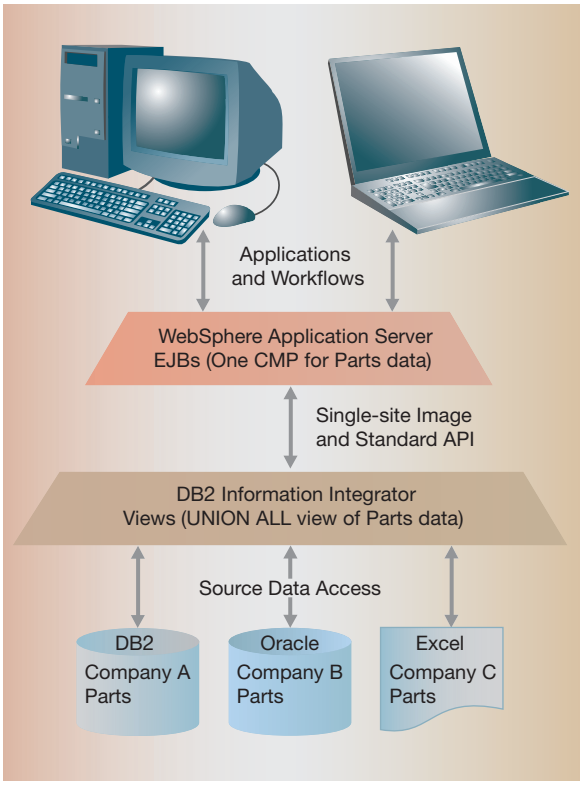
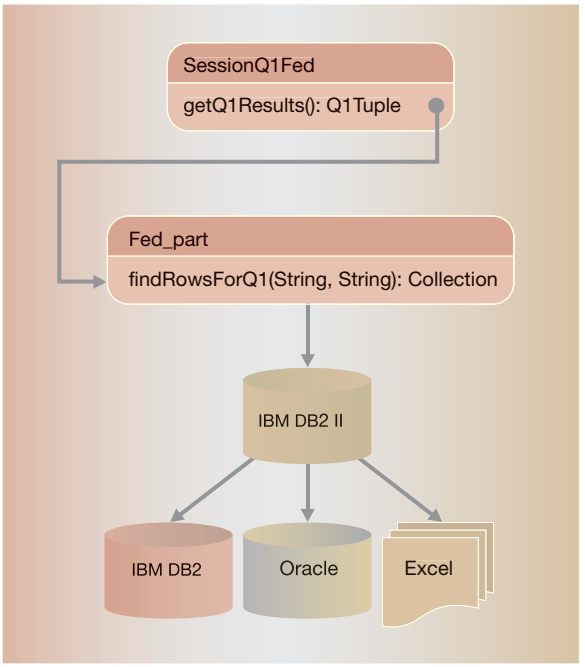


Figure 4 Enterprise Java bean integrating disparate data



erability possible today between application servers and information integration servers. As Figure 3 illustrates, J2EE developers can use an information integration server to model disparate data within a single CMP entity EJB. Attributes of this EJB can span data in relational and non-relational sources on remote systems—a modeling capability not supported natively in application server environments. This option is possible because the information integration server enables administrators to create views that union or join data managed by different systems. The following SQL code illustrates how a DB2 II user can create nicknames representing remote data objects (such as tables or files) and easily combine these in a view. This example creates a single logical image of parts data stored in DB2, Oracle, and Microsoft Excel spreadsheets:

```
CREATE NICKNAME db2_part FOR mydb2.companyA.
  part;

CREATE NICKNAME ora_part FOR myora.companyB.
  part;

CREATE NICKNAME odbc_part FOR myodbc.part$;

CREATE VIEW fed_part AS

SELECT db2_part.*, 'db2' AS p_server
  FROM db2_part UNION ALL

SELECT ora_part.*, 'ora' AS p_server
  FROM ora_part UNION ALL

SELECT odbc_part.*, 'xls' AS p_server
  FROM odbc_part;
```

J2EE developers can then map this view to a single CMP entity EJB, using two attributes (p_server and the primary key column of each underlying table) as the primary key of the EJB. This will ensure the uniqueness of each EJB instance, given the view definition above. (Other alternatives are possible, involving UNION-based view definitions, but a full discussion of this topic is beyond the scope of this paper.) As Figure 4 shows, fed_part is a CMP entity bean based on a view spanning data in DB2, Oracle, and Excel.

Certain implementation issues arise when mapping views that join data to EJBs. For example, such views

Figure 5 Example of "instead of" trigger

```
CREATE TRIGGER EMPVIEW_INSERT INSTEAD OF INSERT ON EMPVIEW
REFERENCING NEW AS NEWEMP FOR EACH ROW . . .
INSERT INTO EMPLOYEE (EMPNO, NAME, SALARY, WORKDEPT)
VALUES(EMPNO, NAME, SALARY,
   COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
   WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
   RAISE_ERROR('70001', 'Invalid or unknown department')))
```

may be read-only by default, and designers may address this issue in two potential ways: at the application server tier, by flagging the EJB as read-only (which WebSphere developers can do by setting the "access intent" attribute in the EJB deployment descriptor) or at the information integration tier, by writing "instead-of" triggers to operate against the view and instruct the server how to process insert, update, or delete requests. As of this writing, the latter option applies only to non-federated views. Instead-of triggers consist of relatively simple SQL code. See Figure 5, which shows a DB2 UDB (Universal Database) example that enables insert operations to be processed against a view that joins two local tables (EMPLOYEE and DEPARTMENT).

In some cases it may be difficult to create views that correlate data from disparate sources. This can occur if primary key values for the same entity need to be represented differently on each remote data source. Using our example involving parts data, this might occur if one data source employed Universal Product Codes (UPCs) for primary key values, and another data source used a company-specific tracking number. In such cases, a mapping table could be created for the DB2 II instance to retain information about the identity relationships that exist between entities represented by different primary key values on different data sources, and this mapping table could be referenced in a view definition to appropriately join data from different sources.

By employing an information integration server to present J2EE programmers with a unified view of disparate data, IT architects not only have greater object modeling capabilities (such as the ability to define a single CMP entity EJB with attributes spanning multiple data sources), but can also transparently extend the reach of WebSphere to data sources that may be difficult to support natively. Our previous example included Excel spreadsheets, which are difficult to represent through DataSource objects (pooled connection objects) using standard WebSphere tools. (In connection pooling, a system component maintains a pool of available connections that it can assign to clients as they request them, rather than having each client application undertake the overhead of initiating a fresh connection to the target source.) With DB2 II, Excel spreadsheets—as well as many other types of persistent and dynamically generated data—appear as standard DB2 tables to WebSphere, making it simple to create CMP entity EJBs that map to these spreadsheets by using WebSphere's built-in support for DataSource objects.

Perhaps the simplest way to see the advantages of pairing an information integration tier with an application server tier is to consider the alternative of managing the integrated data access directly from application server tier components. Returning to our previous merger scenario involving parts distributors, the work involved to support an on-line catalog search that spans all three distribution companies might include locating the first 20 parts that match a certain name and type, as specified by a potential customer. This search function, similar in nature to some of the searches found in e-commerce workloads,[23] could be written in one of two ways. In a CMP entity EJB environment, the search would be written in the EJB Query Language as part of a custom finder method; in an environment that relied on servlets, Web services, or session EJBs, the search would be written as a SQL statement in a method. In the latter case, the SQL code necessary to support this search is quite simple if all necessary data resides in a single DB2 UDB database:

Figure 6 Example of the handling of SQL statements from various sources

```
// ------------------------------------- DB2 remote -------------------------------------
// select for query #1
query1[1]= "SELECT DISTINCT p_partkey, p_name, p_mfgr, p_type " +
"FROM companyA.part " +
"WHERE p_type LIKE ? " +
"AND p_name LIKE ? " +
"ORDER BY p_partkey " +
"FETCH FIRST 20 ROWS ONLY";
// ------------------------------------- Oracle remote -------------------------------------
// select for query #1
query1[2]= "SELECT * FROM (" + "SELECT p_partkey, p_name, p_mfgr, p_type " +
"FROM part " +
"WHERE p_type LIKE ? " +
"AND p_name LIKE ? " +
"ORDER BY p_partkey" +
") WHERE ROWNUM <= 20";
// ------------------------------------- Excel local -------------------------------------
// select for query #1
query1[3]= "SELECT p_partkey, p_name, p_mfgr, p_type " +
"FROM [part$] " +
"WHERE p_type LIKE ? " +
"AND p_name LIKE ?";
```

```
SELECT p_name, p_mfgr, p_type, p_partkey

FROM part WHERE

p_type LIKE '%burnished%' AND

p_name LIKE '%lavender%'

ORDER BY p_partkey

FETCH FIRST 20 ROWS ONLY;
```

Unfortunately, this simple query will not solve the problem presented in our sample scenario because the required data is spread across DB2, Oracle, and Excel spreadsheets. Without an information integration server such as DB2 II, a programmer would have to connect to each data source individually and then rewrite the query to execute against each source, amounting to considerably more work. Given our scenario, the programmer is actually somewhat fortunate because a JDBC/ODBC bridge is available to enable the programmer to "query" Excel spreadsheets from a Java component using SQL statements. With other non-relational sources, such bridges may not be available, forcing the programmer to translate the fragment of the original query into the native API required for data retrieval.

However, even in our more simplified scenario, the programmer still has to contend with the different SQL dialects supported by the three required data sources. More importantly, this individual also must determine how to manually decompose this query so that the Java component will push down as much data filtering work as possible to each data source without compromising the semantics of the overall query. Thus, the component might require SQL statements similar to those shown in Figure 6 just to retrieve the necessary data from each source.

Note that the DB2 and Oracle queries express the result set size restriction differently and that Excel does not support this capability at all. Furthermore, note that referencing a "table" name in the FROM clause for Excel requires a special syntax. These syntactic and functional differences are among the simplest that J2EE programmers will need to confront if they have to integrate disparate information manually in their components. The situation can easily

Table 1 Logic to be implemented in J2EE components or Web services working with disparate data, with and without DB2 Information Integrator.

With Information Integration Server	Without Information Integration Server
1. Connect to one data source.	1. Connect to *N* data sources.
2. Issue one SELECT statement.	2. Issue multiple SELECT statements, at least one per data source
	3. Insert results into temporary table(s) or private application structure(s).
	4. Issue SELECT statement(s) against temporary table(s) or merge/sort/transform data in private structures.
	5. Delete data from temporary table(s) or release resources associated with private structures.
3. Release connection to one data source.	6. Release connections to *N* data sources.

become more complex, as studies have shown.[24] Moreover, at this point, the sample code has simply retrieved different result sets. These still need to be merged and integrated, which can itself be a substantial task.

Table 1 compares and contrasts the logic necessary to build servlets, session EJBs, or Web services that read and integrate disparate data with and without an information integration server. The virtual database environment provided by an information integration tier relieves programmers from needing to connect individually to appropriate data sources, retrieve data from these sources by using each source's native API, and transform, merge, and consolidate the resulting interim data into the desired target format.

The advantages of an architecture like that of DB2 II are considerable, particularly when retrieving and consolidating data from multiple sources. Studies involving servlet[25] and entity EJB[26] programming efforts show a code reduction in the range of 40 to 65 percent with DB2 II, compared with native data access implementations that required a consolidated view of data from three sources. Development time savings were even greater due to simplifications in design, development, and testing efforts. Other software components, such as workflows, that delegate information integration to DB2 II, are likely to realize similar benefits as well because they likewise will be relieved of the difficult task of connecting to, retrieving, and unifying disparate data in an efficient manner.

Early performance work involving the architecture described in this paper has been encouraging. An analysis of servlets needing to integrate data from three sources showed that the implementations tested using DB2 II for data access were generally competitive with those that used native data access mechanisms.[27] Furthermore, performance of the native data access servlets depended greatly on the degree of expertise and quantity of time the developer was able to dedicate to designing and coding the data access logic.

As business needs cause changes in the nature of the data distribution scheme, in the database schemas, or even in the queries themselves, more code changes may be required for Web components that use native data access than for those that use an information integration server. This is because the information integration server's global optimizer assumes responsibility for assessing the costs of different data access strategies and selecting an efficient approach based on available statistics about remote data sources. While not infallible, such optimizers play an important role in minimizing the skill and labor required to achieve reasonable performance in a distributed and heterogeneous computing environment. Other information integration server technologies, such as data caching supported through material query tables (MQTs), also offer potential performance gains.[28]

Conclusions

Defining and constructing effective solutions for enterprise information integration enables firms to capitalize on existing information assets while keeping development and maintenance costs within reason. By combining a Web application server tier (such as WebSphere Application Server) with an information

integration server tier (such as DB2 II), firms can readily develop Java components, Web applications, and enterprise portals that present a unified view of disparate data that comprise critical business entities, such as customers and orders. Using these two server tiers together enables IT architects to leverage the benefits of each without attempting to force-fit one into assuming an architectural role for which it was not originally designed.

In the previous section, we cited early studies that have shown a clear productivity benefit associated with the joint deployment of these two technologies, including a 40 to 65 percent code savings for certain server-side Java objects and the potential for reduced maintenance costs as business requirements evolve. Much of this is due to providing programmers with the ability to employ a common API against a virtual database, which eliminates the need to manually establish multiple data source connections, determine efficient source-specific data access strategies, write source-specific data retrieval requests, and manage the final correlation and aggregation of the resulting data. Furthermore, the variety of data placement options supported by an information integration server enables IT architects to design their environments as needed, which might include some combination of data replication, data caching, and data federation.

Data management responsibilities for this integrated environment remain with database administrators, who have the most skills in database design, modeling, and tuning techniques. This relieves Web programmers from having to take on this burden. The standards-based nature of both Web application servers and information integration servers, combined with the years of development and research that underlie them, offer firms a solid foundation upon which to build critical integrated solutions.

Acknowledgments

The authors would like to thank the reviewers for their thoughtful comments relating to an early draft of this article.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Software AG Limited Liability Company, Sun Microsystems, Inc., or Pegasus Solutions, Inc.

Cited references

1. E. Morphy, "Business Integration Spending to Surpass IT," *CRM Daily* (January 27, 2003).
2. C. J. Date, *An Introduction to Database Systems*, Sixth Edition, Addison-Wesley Publishing Company, Inc., Reading, MA (1995), p. 670.
3. C. M. Saracco, *Universal Database Management: A Guide to Object/Relational Technology*, Morgan Kaufmann Publishers, San Francisco, CA (1998), pp. 1–19.
4. P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, *Access Path Selection in a Relational Database Management System*, IBM Research Report RJ-2429 (January 1979).
5. L. D. Shapiro, "Join Processing in Database Systems with Large Main Memories," in *Readings in Database Systems*, Third Edition, M. Stonebraker and J. M. Hellerstein, Editors, Morgan Kaufmann Publishers, San Francisco, CA (1998), pp. 128–140.
6. C. J. Bontempo and C. M. Saracco, *Database Management Principles and Products*, Prentice Hall, Upper Saddle River, NJ (1995), pp. 140–160.
7. C. J. Bontempo and C. M. Saracco, "Data Access Middleware: Seeking Out the Middle Ground," *InfoDB* **9**, No. 4, pp. 7–13 (August 1995).
8. C. M. Saracco, op. cit., pp. 183–197.
9. "BEA WebLogic Server 8.1 Overview," BEA Systems white paper, 2003, http://www.bea.com/framework.jsp?CNT=white_papers_index.jsp&FP=/content/news_events/white_papers.
10. "Introductory Guide to Information Integration and Nimble Technology," Nimble Technology white paper, 2003, http://www.nimble.com/solutions/literature/.
11. "Snapbridge FDX, the Data Federation Product Suite," Snapbridge white paper, 2003, http://www.snapbridge.com/company/news/white_papers.html.
12. "Leveraging Enterprise Data Assets," MetaMatrix white paper, 2002, http://www.metamatrix.com/13_whtpprs.jsp.
13. L. M. Haas, E. T. Lin, and M. A. Roth, "Data Integration through Database Federation," *IBM Systems Journal* **41**, No. 4, 2002, pp. 578–596.
14. H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papkonstantinou, J. Ullman, and J. Widom, "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS," *Proceedings of the AAAI Symposium on Information Gathering*, Stanford, CA; AAAI Press (1995), pp. 61–64.
15. S. Adali, K. Candan, Y. Papkonstantinou, and V. S. Subrahmanian, "Query Caching and Optimization in Distributed Mediator Systems," *Proceedings of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada; ACM, New York (1996), pp. 137–148.
16. A. Tomasic, L. Raschid, and P. Valduriez, "Scaling Heterogeneous Databases and the Design of DISCO," *Proceedings of the 16th International Conference on Distributed Computer Systems*, Hong Kong; IEEE, New York (1996), pp. 449–457.
17. M.-C. Shan, "Pegasus Architecture and Design Principles," *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C.; ACM, New York (1993), pp. 422–425.
18. M. Tork Roth, P. Schwarz, and L. Haas, "An Architecture for Transparent Access to Diverse Data Sources," in *Component Database Systems*, K. R. Dittrich, A. Geppert, Editors, Morgan-Kaufmann Publishers, San Francisco, CA (2001), pp. 175–206.

19. C. J. Bontempo, *DataJoiner—A Multidatabase Server*, IBM White Paper, IBM Programming Systems Division (October 1994).
20. F. Leymann and D. Roller, "Using Flows in Information Integration," *IBM Systems Journal* **41**, No. 4, 2002, pp. 732–742.
21. D. S. Linthicum, *Enterprise Application Integration*, Addison-Wesley Publishing Company, Inc., Reading, MA (1999).
22. S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Simeon, Editors, *X Query 1.0: An XML Query Language*, W3C working draft (May 02, 2003), http://www.w3.org/TR/xquery.
23. *TPC Benchmark W Specification Version 1.8*, Transaction Processing Performance Council, San Jose, CA (Feb. 19, 2002), http://www.tpc.org/tpcw/.
24. C. M. Saracco, S. Englert, and I. Gebert, "J2EE Development Across Multiple Data Sources: Digging into the Details," *DB2 Developer Domain* (May 2003), http://www-106.ibm.com/developerworks/db2/library/techarticle/0306saracco/0306saracco.html.
25. C. M. Saracco, S. Englert, and I. Gebert, "Using DB2 Information Integrator for J2EE Development: A Cost/Benefit Analysis," *DB2 Developer Domain* (May 2003), http://www-106.ibm.com/developerworks/db2/library/techarticle/0305saracco/0305saracco.html.
26. C. M. Saracco and T. J. Rieger, "Our Experience with Developing Entity EJBs over Disparate Data Sources," *DB2 Developer Domain* (May 2003), http://www-106.ibm.com/developerworks/db2/library/techarticle/0305saracco/0305saracco.html.
27. S. Englert, C. M. Saracco, and I. Gebert, "Performance of DB2 Information Integrator in a J2EE Environment with Multiple Data Sources," *DB2 Developer Domain* (June 2003), http://www-106.ibm.com/developerworks/db2/library/techarticle/0306saracco1/0306saracco1.html.
28. A. Kurnetsov, "Using Materialized Query Tables to Speed Up Queries in DB2 UDB," *DB2 Developer Domain* (August 2002), http://www7b.boulder.ibm.com/dmdd/library/techarticle/0208kuznetsov/0208kuznetsov.html.

Accepted for publication September 25, 2003.

Cynthia M. Saracco *IBM Software Group, Silicon Valley Laboratory, 555 Bailey Ave., San Jose, CA 95141 (saracco@us.ibm.com).* Ms. Saracco is a senior software engineer with more than 15 years of experience spanning various technology disciplines, including database management, Web application servers, workflows, and server-side Java. A former software technology instructor at the University of California at Santa Cruz extension program, Ms. Saracco has published more than 50 technical reports and journal articles internationally and lectured throughout North America, South America, Europe, and the Middle East. She has written two books on database management software, one of which she co-authored with Charles J. Bontempo.

Mary A. Roth *IBM Software Group, Silicon Valley Laboratory, 555 Bailey Ave., San Jose, CA 95141 (torkroth@us.ibm.com).* Ms. Roth is a senior manager and architect at IBM's Silicon Valley Lab. She has more than 13 years of experience in database research and development. As a researcher and member of the Garlic project at IBM's Almaden Research Center, she contributed key advances in heterogeneous data integration techniques and federated query optimization and led efforts to transfer Garlic support to DB2. Ms. Roth is leading a team of developers to deliver a key set of components for IBM's information integration initiative.

Daniel C. Wolfson *IBM Software Group, 11501 Burnett Road, Austin, TX 78758 (dwolfson@us.ibm.com).* Mr. Wolfson is a Distinguished Engineer and the Chief Technical Officer of IBM's information integration initiative. With more than 16 years of experience in distributed computing, his interests range broadly across database management, messaging, and transaction systems. His current responsibilities include DB2 integration with the WebSphere family, Web services, and asynchronous client protocols.