IBM® DB2® Universal Database™

Developing Enterprise Java Applications Using DB2 UDB, Version 7.2

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 21.

Contents

Developing Enterprise Java™ Applications Using DB2 UDB, Version 7.2

by: Grant Hutchison, DB2/IBM Integration Center

Introduction

DB2 Universal Database® (UDB) supports all the key Internet standards, making it an ideal database for use on the Web. It has in-memory speed to facilitate Internet searches and complex text matching combined with the scalability and availability characteristics of a relational database. DB2 Universal Database supports WebSphere™, Java, and XML Extender, which makes it easy for you to deploy your e-business applications. This paper discusses the features of DB2 UDB Universal Developer's Edition Version 7.2 (UDE), and included development tools such as VisualAge® for Java, Professional Edition and WebSphere Studio, for use in building Web applications.

Java enablement

DB2 Universal Database supports many types of Java programs. It provides driver support for client applications and applets written in Java using Java Database Connectivity (JDBC™). It also provides support for embedded SQL for Java (SQLj), Java user-defined functions (UDFs), and Java stored procedures.

JDBC driver support

The DB2 JDBC application (Type 2) driver (Figure 1 on page 2) enables Java applications to make calls to DB2 through JDBC. Calls to the JDBC application driver are translated to Java native methods. The Java applications that use this driver must run on a DB2 client, through which JDBC requests flow to the DB2 server.

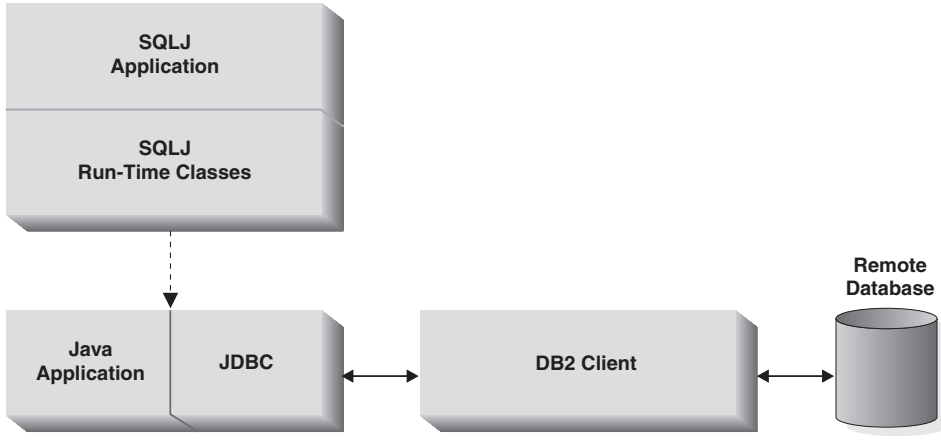

Figure 1. DB2 Java application implementation

The DB2 JDBC applet (Type 3) driver (Figure 2 on page 3) consists of a JDBC client and a JDBC server. The DB2 JDBC applet driver is loaded by the Web browser along with the applet. When the applet requests a connection to a DB2 database, the applet driver opens a TCP/IP socket to the DB2 JDBC applet server on the machine where the Web server and DB2 client are running. After a connection is set up, the applet driver sends each of the subsequent database access requests from the applet to the JDBC server through the TCP/IP connection. The JDBC server then makes corresponding DB2 calls to perform the task. Upon completion, the JDBC server sends the results back to the JDBC client through the connection.

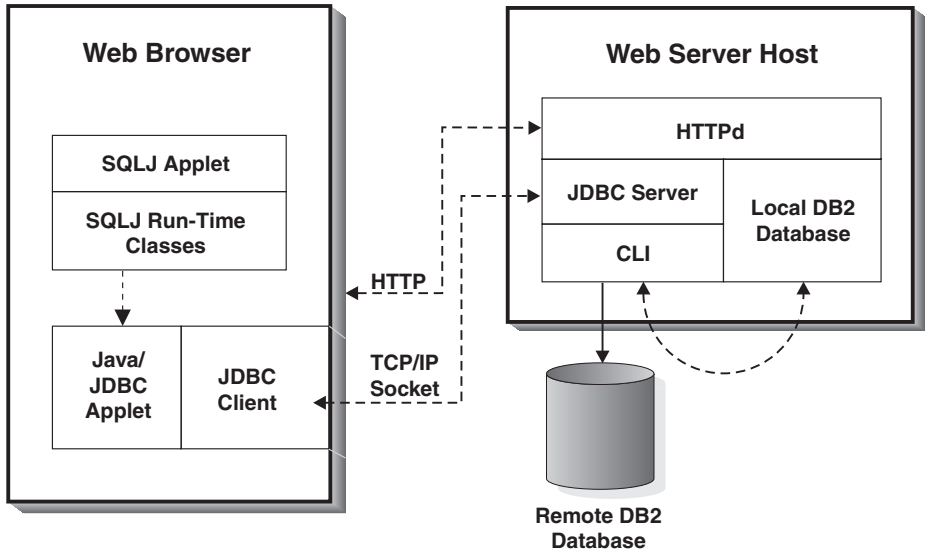

Figure 2. DB2 Java applet implementation

JDBC 1.0 defines the Java APIs for access to relational databases. With the introduction of JDBC 2.0, the APIs have been split into two parts:

JDBC 2.0 Core API
contains evolutionary improvements, but has been kept small and focused like the JDBC 1.0 API in order to promote ease of use. Code written for the 1.0 API continues to work with the 2.0 API. The 2.0 API classes are found in the java.sql package.

JDBC 2.0 Optional Package API
defines specific kinds of additional functionality when vendors are ready to provide the functionality and developers are ready to use the functionality. IBM has implemented these new classes and interfaces in new packages. You can use the new classes and interfaces in your applications by using the following import statements:

- import javax.sql.*;
- import COM.ibm.db2.jdbc.*;

The DB2 JDBC 2.0 driver supports the following features of the JDBC 2.0 Optional Package API:

- DataSource support
- Java Naming and Directory Interface™ (JNDI) for Naming Databases
- Java Transaction API (JTA) — available with the DB2 JDBC application driver only
- Database Connection Pooling

The JNDI and connection pooling interfaces are defined in the JDBC driver and their implementation is provided by WebSphere Application Server, Version 3.5.

Connection pooling is defined as part of the JDBC 2.0 Optional Package API. JNDI and DataSource objects are provided as an alternative to using DriverManager objects to access relational data servers.

JDBC 2.0 support is available on the AIX®, Solaris™, HP-UX, Linux, Linux for OS/390®, and Windows® 32-bit operating systems. To take advantage of the new features of JDBC 2.0, you must install both the JDBC 2.0 driver and the Java Development Kit (JDK™) 1.2, as the JDBC 1.2 driver is the default driver. To install the JDBC 2.0 driver, enter the **usejdbc2** command from the `sqllib/java12` directory. To switch back to the JDBC 1.2 driver, execute the **usejdbc1** command from the `sqllib/java12` directory.

For more detail, see:

- *DB2 Application Development Guide, Version 7*, chapter 21: "Programming in Java"
- http://www.software.ibm.com/data/db2/java

SQLj support

DB2 SQLj support enables you to build and run SQLj applets and applications. These Java programs contain embedded SQL statements that are precompiled and bound to a DB2 UDB database. SQLj applications use JDBC support, and require the SQLj run-time classes to authenticate and execute any SQL packages that were bound to the database at the precompiling and binding stage.

The SQLj standard has three components: embedded SQLj, a translator, and a run-time environment. The translator translates SQLj files to produce Java source files and profiles. The run-time environment performs the SQL operations in JDBC, and uses the profile to obtain details about database connections.

Java stored procedures and user-defined functions (UDFs)

You can create and use stored procedures in Java just as you would in other languages, with only a few minor differences. After you code the stored procedure, register it with the database by using the CREATE PROCEDURE statement. You can then call the stored procedure from your application. The stored procedure can be FENCED or NOT FENCED.

You can also create and use UDFs in Java just as you would in other languages, with only a few minor differences. After coding the UDF, register it

with the database using the CREATE FUNCTION statement. You can then refer to the UDF in the SQL of your application. The UDF can be FENCED or NOT FENCED.

You can use the DB2 Stored Procedure Builder (SPB) to develop Java stored procedures. To help you create stored procedures, DB2 SPB provides design assistants (wizards) that guide you through basic design patterns, help you create SQL queries, and estimate the performance cost of invoking a stored procedure.

DB2 SPB is implemented with Java, and all database connections are managed using JDBC. Using a JDBC driver, you can connect to any local DB2 alias or any other DB2 UDB database for which you can specify a host, port, and database name. Several JDBC drivers are installed with DB2 SPB for use on different operating systems.

To run your UDFs and stored procedures on the server, DB2 invokes the Java Virtual Machine (JVM). Before starting up the database, your database administrator must install and configure the appropriate JDK on your DB2 server. You can choose to use individual Java class files for your stored procedures and UDFs, or collect the class files into a Java Archive (JAR) file and install the JAR file in the database.

The JVM is loaded on the first call to a Java UDF or stored procedure. In all cases, the JVM stays loaded until the embedding process ends. For NOT FENCED UDFs and stored procedures, one JVM is loaded per database instance, and runs inside the database engine's address space for best performance. For FENCED UDFs, a distinct JVM is loaded inside the db2udf process; similarly, FENCED stored procedures load a distinct JVM inside the db2dari process.

Java application development for Web servers

There are several tools provided with UDE Version 7.2 which provide Web enablement support. VisualAge for Java, Version 3.5 is an integrated development environment (IDE) that enables you to build, test, and deploy Java applications to WebSphere Application Server and DB2 Universal Database. WebSphere Studio is a suite of tools that brings all aspects of Web site development into a common interface. WebSphere Application Server Standard Edition version 3.5 provides a robust deployment environment for e-business applications. Its components let you build and deploy personalized, dynamic Web content quickly and easily.

VisualAge for Java, Professional Edition

VisualAge for Java, Professional Edition Version 3.5 (provided with UDE Version 7.2) contains features and performance improvements that make it

easier than ever to create scalable, hardworking e-business applications. Tight integration with IBM WebSphere Application Server, WebSphere Studio, and DB2 Universal Database reduces development time and improves productivity, while providing easier, secure access to enterprise data.

IBM Data Access JavaBeans™, an optional installable feature of VisualAge for Java, allows application developers to easily access JDBC-enabled relational databases. The IBM Data Access JavaBeans, found in the *com.ibm.db* package, include classes to simplify access to relational databases and provide the following enhanced features:

Caching of query results
SQL query results can be retrieved all at once and placed into a cache. The application or servlet can move forward and backward through the cache or jump directly to any result row in the cache. Contrast this to facilities in the java.sql package, in which rows are retrieved from the database one at a time, in the forward direction only, and a newly retrieved row overlays the last retrieved row unless additional code is written to expand functionality. For large result sets, the IBM Data Access JavaBeans provide ways to retrieve and manage packets, which are subsets of the complete result set.

Updates through the results cache
The servlet can use standard Java methods to change, add, or delete rows in the result cache. Changes to the cache can be propagated to the underlying relational table.

Query parameter support
The base SQL query is defined as a Java String, with parameters replacing some of the actual values. When the query is run, the IBM Data Access JavaBeans provide a way to replace the parameters with values made available at run time. For example, the parameter values might be submitted by a user on an HTML form.

Metadata support
A StatementMetaData object contains the base SQL query. Higher levels of metadata can be added to the object to help pass parameters into the query and work with the returned results. When the query is run, the parameters are automatically converted between Java datatypes and the corresponding SQL datatypes.

VisualAge for Java, Enterprise Edition

VisualAge for Java, Enterprise Edition Version 3.5 (not provided with UDE Version 7.2) enables you to build, test, and deploy Java applications within a single environment. In addition to its visual programming features, VisualAge for Java provides wizards to lead you quickly through many tasks, including the creation of applets, servlets, and applications. It also enables you to import existing code and export code, as required, from the underlying file system.

VisualAge for Java provides an SQLj Tool that implements the SQLj standard, enabling you to simplify database access. The translator component is integrated into the IDE, enabling you to import, translate, and edit SQLj files. The run-time environment is an installable feature that is added to your workspace. The original SQLj source files are maintained in your project resources directory, as are the profiles. The VisualAge for Java SQLj Tool creates profiles in your project resources directory.

Persistence management

The Persistence Builder is a feature of VisualAge for Java that generates a layer of code that implements all of the JDBC access calls necessary to insert, update, or retrieve the data for an object from the database. The Persistence Builder is a productivity tool that simplifies many of the challenges when developing database applications. If the application was developed using object modeling, the Persistence Builder Schema Browser can be used to generate the necessary tables to persist the objects, known as *top-down* mapping. If the database exists then the Schema Browser can be used to reverse-engineer the tables into objects, known as *bottom-up* mapping. The third technique is called *outside-in* or "meet in the middle," as each model is maintained and adaptive changes are made within the object model or the relational model. Using the VisualAge for Java Persistence Builder can improve programmer efficiency and separate the data model and the object model. WebSphere Application Servers can provide connection pooling for DB2 UDB database connections, while the persistence builder code can be generated to utilize the connection pool instead of directly connecting to the database. Used primarily in the development of EJBs, the Persistence Builder can also provide links between object associations and transactions using the equivalent constraints and transactions in DB2.

Debugging

VisualAge for Java provides several tools for developing JavaServer Pages™ (JSPs). The Servlet Launcher allows you to start a Web server, open your Web browser, and launch a servlet. The JSP Execution Monitor allows you to monitor the execution of JSP source, generated servlets, and HTML source as it is generated. VisualAge for Java also allows you to set breakpoints within servlet code, dynamically update the servlet at breakpoints, and continue running the servlet with the incorporated changes .

The IBM WebSphere Test Environment is an optional installable feature of VisualAge for Java. This feature allows you to run servlets and Enterprise JavaBeans™ (EJBs) within the VisualAge for Java environment prior to deployment to WebSphere Application Servers. The environment provides a servlet engine and the required services for EJB applications. Since this test environment is provided within the IDE, breakpoints can be set in servlets or JSPs and a client application or browser can be invoked to test all aspects of

the server-side Java programs. The remote debugger can be used to set breakpoints in EJBs that are executing within an EJB Server such as WebSphere Application Server.

Using the DB2 Stored Procedure Builder (SPB) and the VisualAge remote debugger, you can remotely debug a stored procedure installed on a DB2 server. To debug a stored procedure, run the stored procedure in debug mode. You are not required to debug the stored procedures from within an application program. You can test your stored procedure separately from the calling application program.

Using the Debug Properties notebook in SPB, you can change, add, or remove debug records in the stored procedures debug table. If you are a database administrator and you created the selected stored procedure, you can grant database authorization to other users to debug the stored procedure.

WebSphere Studio

WebSphere Studio Version 3.5 is a suite of tools that brings all aspects of Web site development into a common interface. The WebSphere Studio makes it easier than ever to cooperatively create, assemble, publish, and maintain dynamic Web applications. Content authors, graphic artists, programmers, and Web masters are able to work on the same projects, each having access to the files they need. The Studio is composed of the Workbench, the Page Designer, the Remote Debugger, and wizards, and it comes with trial copies of companion Web development products, such as Macromedia Flash, Fireworks, Freehand, and Director. WebSphere Studio enables you to do everything you need to create interactive Web sites that support your advanced business functions, including the following:

- Use Studio wizards to create Java beans, database queries, and Java servlets.
- Group your Web site files into projects and folders. Filters and global search capabilities let you find just the files you need. Projects can be exported to VisualAge for Java and deployed to WebSphere Application Server.
- Maintain the files individually or in a shared version control system.
- Edit and update the files with your preferred tools. When opening a Studio file, you can quickly launch your default selection or you can choose one of your alternative tools.
- Quickly assess file relationships and find broken links.
- Publish your Web site during any stage of development on any of your WebSphere Application Servers. Go directly from site development to site publishing from within the Studio Workbench.

The Studio Workbench helps you manage and maintain your Web site applications and files and provides the following capabilities and features:

- A graphical display of the link relationships between the files in a project.
- The automatic updating of links whenever files change or move.
- The ability to stage your Web site production cycle and publish various stages to different (and to multiple) servers.
- An import wizard that simplifies the transfer of existing site content directly into a Studio project.
- A quick way to archive Web sites or sub-sites in a single file.
- The ability to easily integrate third-party tools into the Workbench environment.
- An enhanced team environment with a common view of work-in-progress, through the integration of popular source control-management software such as IBM VisualAge TeamConnection®.

The Studio Page Designer provides a visual design environment that enables you to create JSP, Java servlets, and other Java-based Web tools. For example, you can use the visual environment to drag and drop JavaBeans into JSP applications. The Studio Page Designer can also be used to create Dynamic Hyper Text Markup Language (DHTML) and HTML pages and includes the capability to easily edit and toggle between the HTML or DHTML source and the browser view. The Studio Remote Debugger provides source-level debugging of JSP files and Java servlets within the Studio environment.

WebSphere Application Server

The WebSphere Application Server combines the portability of server-side business applications with the performance and manageability of Java technologies to offer a comprehensive platform for designing Java-based Web applications. It enables powerful interactions with enterprise databases and transaction systems.

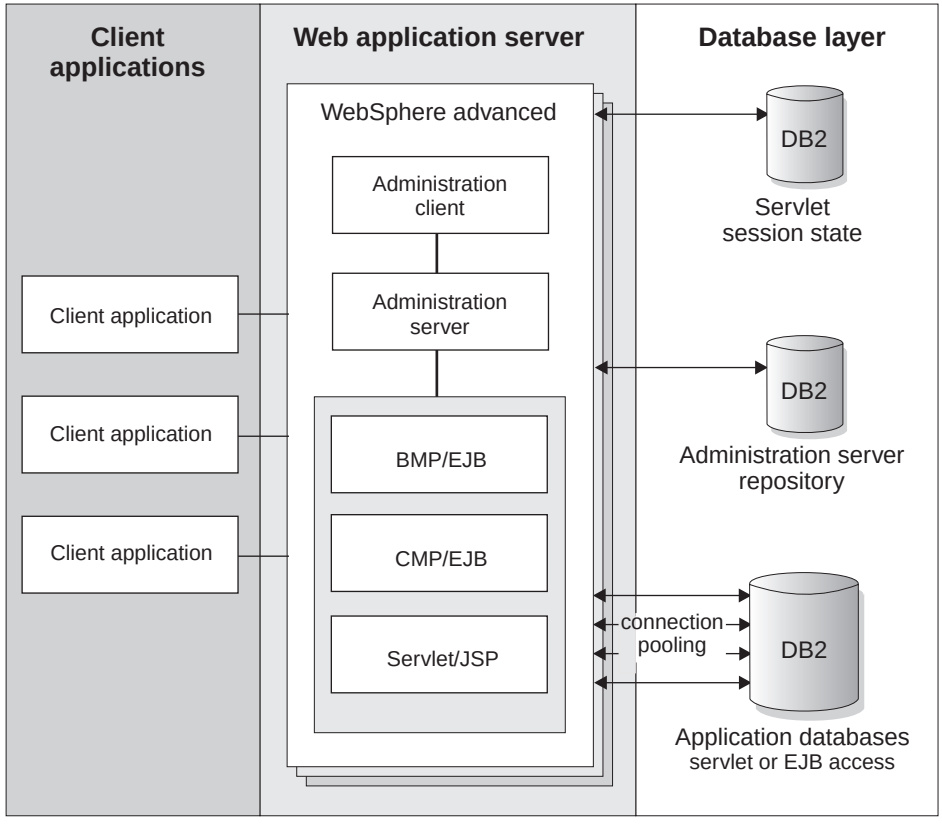

Figure 3. WebSphere Application Server

Application servers extend the capabilities of a Web server to handle application requests. Given an application comprised of HTML pages, servlets, and enterprise beans, the application server makes the following exchange possible:

1. A user at a Web browser on the public Internet visits a company Web site. The user requests use of an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing servlets and enterprise beans. It forwards the request to IBM WebSphere Application Server.
4. The IBM WebSphere Application Server product forwards the request to one of its application servers on which the application is running.
5. The application processes the user request.
6. The application server collaborates with the Web server to return the results to the user's Web browser.

The WebSphere Application Server provides a servlet engine that implements the Java Servlet API 2.1. It includes its own packages that extend and add to the Java Servlet API, and leverage JDK 1.2 across all supported operating systems. The extensions and additions make it easier to manage session state, create personalized Web pages, generate better servlet error reports, and access databases.

The Application Server supports JSPs, a powerful approach to dynamic Web page content. One of the advantages of JSPs is that they enable you to effectively separate the HTML coding from the business logic in your Web pages. The IBM extensions to the JSP specification include HTML-like JSP tags that make it easy for HTML authors to add the power of Java to their Web pages.

Connection pooling

Connection pooling support is provided with WebSphere Application Server. Connection pooling enables you to control and reduce the resources used by your Web-based applications. Web-based applications accessing data servers incur higher and less predictable overhead than non-Web applications, because Web users connect and disconnect frequently. Often, more resources are spent connecting and disconnecting than are spent during the interactions themselves.

Connection pooling spreads the connection overhead across several user requests by establishing a pool of connections which servlets can use. After the initial resources are spent to produce the connections in the pool, additional connect/disconnect overhead is insignificant because the existing connections are reused repeatedly.

Connection pooling as provided by WebSphere Application Server specifies how relational data servers are accessed by a servlet using the new JDBC 2.0 Standard Extension APIs. This is the new model. Servlets running under WebSphere Application Server Version 3.x can be coded to make efficient use of data server connection resources.

Connection pooling advantages: Each time a resource attempts to access a database, it must connect to that database. A database connection incurs overhead – it requires resources to create the connection, maintain it, and then release it when it is no longer required. The overhead is particularly high for Web-based applications because of the frequency of connection and disconnection. User interactions are typically short, due to the transient nature of Web browsing. Because Internet requests can arrive from virtually anywhere, usage volumes can be large and difficult to predict. To address the problem, WebSphere Application Server establishes a pool of database connections that is shared by applications on application servers.

Connection pooling lets the administrator control and reduce the resources used by Web-based applications, spreading the connection overhead across several user requests. Connection pooling can also improve the response times of Web-based applications.

When a user makes a request over the Web to a resource, the resource accesses a data source. Because the data source locates and uses an existing connection from the pool, the user request does not incur the overhead of creating a new connection.

Each connection is associated with a particular user request. When the request is satisfied and the response is returned to the user, the data source returns the connection to the connection pool for reuse. Again, the overhead of a disconnect is avoided.

Each user request incurs a fraction of the cost of connecting or disconnecting. After the initial resources are used to produce the connections in the pool, additional overhead is insignificant because the existing connections are reused.

How WebSphere Application Server manages connection pools: WebSphere Application Server establishes and maintains pools of connections as specified by the administrator. Once the connections are established, they are distributed in response to user requests, and then perform housekeeping operations to maintain a balance between available connections and the demand for connections. This ensures that an existing connection is available when a servlet or application server needs one.

For example, the connection pool periodically identifies idle or orphaned connections. It terminates idle connections and returns orphaned connections to the connection pool. This means fewer connections are available (and fewer resources are used) when demand for connections is low. An idle connection is one that has not been used for the amount of time specified in the Idle Timeout property of the data source. A connection is orphaned when its owning servlet or application server terminates or does not respond.

WebSphere Standard Edition

The WebSphere Application Server, Standard Edition Version 3.5 (provided with UDE Version 7.2) is a component of the WebSphere Studio. InstantDB is the default database used for WebSphere Standard Edition repository. However, DB2 Universal Database can be configured to be WebSphere's administrative server repository. You can run the DB2 server on the same machine as WebSphere Application Server or on a different Web server.

WebSphere Advanced Edition

The WebSphere Application Server Advanced Edition (not provided with UDE Version 7.2) builds on the Standard Application Server, as it requires a database for operation. DB2 Universal Database is provided with the WebSphere Application Server Advanced Edition Version 3.5, to be used as the administrative server repository. It introduces server capabilities for applications built to the EJB Specification from Sun Microsystems, which provides support for integrating Web applications to non-Web business systems.

The EJB server is the application server of WebSphere Application Server's three tier architecture. It connects the client tier of Java servlets, applets, applications, and JSPs with the resource management tier, the data source.

There are two types of enterprise beans: *session* beans and *entity* beans. Session beans encapsulate temporary data associated with a particular client. Entity beans encapsulate permanent data that is stored in a data source. The persistence service ensures that the data associated with entity beans is properly synchronized with their corresponding data in the data source. To accomplish this task, the persistence service works with the transaction service to insert, update, extract, and remove data from the data source at the appropriate times.

There are two types of entity beans: those with container-managed persistence (CMP) and those with bean-managed persistence (BMP). In entity beans with CMP, the persistence service handles nearly all of the tasks required to manage persistent data. CMP entity beans can be implemented using DB2 UDB (including DB2 for OS/390) as the persistent data store. In entity beans with BMP, the bean itself handles most of the tasks required to manage persistent data.

Application development extensions

XML Extender

Extensible Markup Language (XML) is the accepted standard technique of data exchange between applications. An XML document is a tagged document consisting of character data and markup tags. The markup tags are definable by the author of the document. A Document Type Definition (DTD) is used to declare the markup definitions and constraints. The DB2 XML Extender provides a mechanism for programs to manipulate XML data using SQL extensions.

Typically, XML documents are stored as individual files in a file system. The DB2 XML Extender provides an alternative storage and manipulation environment. XML documents can be stored as a single column or as a collection, using a set of columns.

The DB2 XML Extender introduces three new data types: XMLVARCHAR, XMLCLOB, and XMLFILE. The Extender provides UDFs to store, extract and update XML documents stored within a single column. Searching can be performed on the entire XML document or based on structural components using the location path, which uses a subset of the abbreviated syntax defined by the XML Path Language (XPath). Side tables can be used to improve search performance for elements or attributes that are frequently queried.

To facilitate storing XML documents as a set of columns, the DB2 XML Extender provides an administration tool to aid the designer with XML-to-relational database mapping. The Document Access Definition (DAD) is used to maintain the structural and mapping data for the XML documents. The DAD is defined and stored as an XML document, making it simple to manipulate and understand. New stored procedures are available to compose or decompose the document.

MQSeries® enablement

A set of MQSeries functions are provided with DB2 Universal Database Version 7.2 to allow DB2 UDB applications to interact with asynchronous messaging operations. This means that MQSeries support is available to applications written in any programming language supported by DB2 UDB. For simplicity, all examples shown in this section are SQL statements. A WebSphere application can utilize these MQSeries SQL statements.

Messaging Styles

MQSeries does not mandate that the messages it transports adhere to a particular structure. XML messages typically have a self-describing message structure. Messages can also be unstructured, requiring user code to parse or construct the message content. Such messages are often semi-structured, that is, they use either byte positions or fixed delimiters to separate the fields within a message.

MQSeries supports three messaging models: datagrams, publish/subscribe (p/s), and request/reply (r/r). Messages sent as datagrams are sent to a single destination with no reply expected. In the p/s model, one or more publishers send a message to a publication service that distributes the message to one or more interested subscribers. Request/reply is similar to datagram, but the sender expects to receive a response. The supplied DB2 MQSeries functions support all three message models.

MQSeries is used in a wide variety of ways. Simple datagrams are exchanged to coordinate multiple applications, exchange information, request services, and provide notification of interesting events. The publish/subscribe style is most often used to disseminate real-time information in a timely manner. The

request/reply style is generally used as a simple form of pseudo-synchronous remote procedure call (RPC). More complex models can be constructed by combining these basic styles.

DB2 UDB Version 7.2 provides a new MQSeries Assist wizard. This wizard creates a table function that reads from an MQSeries queue using the MQSeries UDFs, which are also new in DB2 UDB Version 7.2. The wizard can treat each MQSeries message as a delimited string or a fixed length column string. The created table function parses the string according to your specifications, and returns each MQSeries message as a row of the table function. The wizard also allows you to create a view on top of the table function and to preview an MQSeries message and the table function result.

DB2 / MQ Infrastructure

In a basic configuration, as shown in Figure 4, an MQSeries server is located on the database server machine along with DB2 Universal Database. The MQSeries functions are available from a DB2 server and provide access to other MQSeries applications. Multiple DB2 clients can concurrently access the MQSeries functions through the database. The MQSeries operations allow DB2 applications to asynchronously communicate with other MQSeries applications. For instance, the new functions provide a simple way for a DB2 application to publish database events to remote MQSeries applications, initiate a workflow through the optional MQSeries Workflow product, or communicate with an existing application package with the optional MQSeries Integrator product.

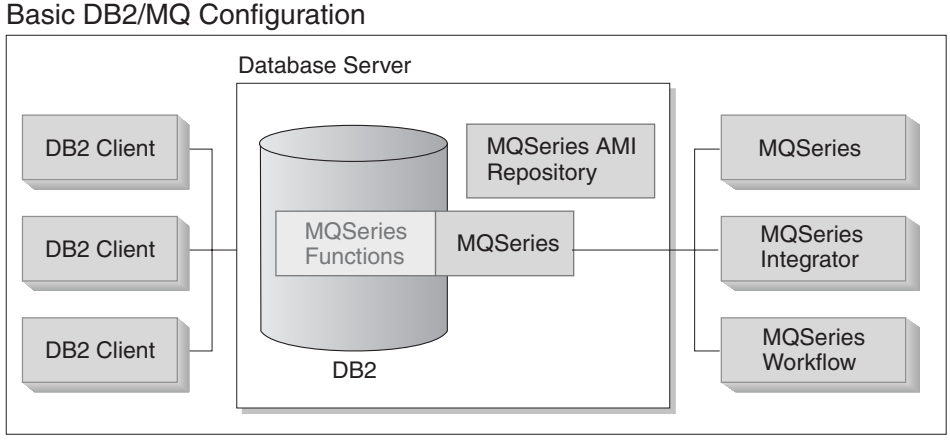

Figure 4. Basic DB2/MQ configuration

When MQSeries support is installed as part of DB2 Universal Database, a configuration script automatically establishes a default configuration that client applications can use with no further administrative action. The default

configuration enables application programmers to get started quickly and provides a simpler development interface.

Messaging examples

The following SQL statement sends a basic message using the default configuration:

```
values MQSEND('simple message')
```

This will send the message `simple message` to the MQSeries queue manager and queue specified by the default configuration.

The MQSeries functions provided with DB2 Universal Database are based on the Application Messaging Interface (AMI). AMI supports the use of an external configuration file, called the AMI Repository, to store configuration information. The default configuration includes an MQSeries AMI Repository configured for use with DB2 UDB. For more information, refer to: http://www-3.ibm.com/software/ts/mqseries/api/.

Service points and *policies* are two key concepts in MQSeries AMI, carried forward into the DB2 UDB MQSeries functions. A service point is a logical endpoint from which a message can be sent or received. In the AMI repository, each service point is defined with an MQSeries queue name and queue manager. Policies define the quality-of-service options that should be used for a given messaging operation. Key qualities of service include message priority and persistence. Default service points and policy definitions are provided and may be used by developers to further simplify their applications. The previous example can be rewritten as follows to explicitly specify the default service point and policy name:

```
values MQSEND('DB2.DEFAULT.SERVICE',
'DB2.DEFAULT.POLICY', 'simple message')
```

Queues can be serviced by one or more applications at the server where they reside. In many configurations, multiple queues will be defined to support different applications and purposes. For this reason, it is often important to define different service points when making MQSeries requests. This is demonstrated in the following example.

```
values MQSEND('ODS_Input', 'simple message')
```

Note that in this example, the policy is not specified, so the default policy will be used.

For further information:

http://www.software.ibm.com/data/integration/MQSeries

Net.Data®

Net.Data is an application that connects Web applications to DB2 Universal Database. It now provides XML support which allows you to generate XML tags as output from your Net.Data macro, instead of manually entering the tags. You can also specify an XML style sheet (XSL) to be used to format and display the generated output.

Net Search Extender

The Net Search Extender (not provided with UDE Version 7.2) uses an indexing technique, known as n-gram index, to provide a new high speed text search extender. There are many uses for this extender in the area of Web applications, as text fields are commonly queried by end-users. Finding relevant documents based on text field indexes can improve Web user satisfaction. Any columns based on CHAR, VARCHAR, or LONG VARCHAR can be indexed using an n-gram index. When the index has been created and activated, searches can be performed using a new stored procedure. Active indexes are stored in shared memory to optimize search performance.

Spatial Extender

The Spatial Extender (not provided with UDE Version 7.2) allows users to integrate spatial data into their queries. It supports spatial types to model real-world entities such as the location of customers, boundaries of parks, and paths of cable lines.

DB2 Integrated Web Services Tutorial

A new DB2 tutorial, called eVideoCentral, is available for download at http://www.ibm.com/software/data/developer/samples/evideo. The DB2 eVideoCentral tutorial gives a sample solution for a company that provides business services to another company over the web. This concept is known as a Business-to-Business application. The eVideoCentral tutorial demonstrates the integration of DB2, WebSphere, and MQSeries. Many technologies are used to design and build eVideoCentral, including: IBM's WebSphere Application Server, VisualAge for Java, and the DB2 XML Extender interaction tool. A set of centralized services for individual retail video stores is accessed via Simple Object Access Protocol (SOAP) (see http://www.w3.org/2000/xp/).

SOAP is a lightweight protocol for exchanging information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it
- A set of encoding rules for expressing instances of application-defined datatypes
- A convention for representing remote procedure calls and responses

SOAP services can provide a mechanism for one company to access DB2 services from another company.

The eVideoCentral tutorial demonstrates the design, development, and implementation of simple data repository (insertion/modification) and query services. The query services use the existing DB2 XML Extender. The first version of eVideoCentral is provided via the web, and it includes: Java Servlets, JSP (Java Server Pages), schema for the DB2 database objects, and the accompanying tutorial documentation.

Designing applications using Unified Modeling Language (UML)

DB2 UDB data models can now be accessed from within the popular UML modeling product, Rational Rose from Rational Software. This program can extract existing schemas into a logical model or can generate schemas directly from the model. There are many benefits in using a common design language (such as UML) for both object models and data models during the application development process.

Reference: *Database Design for Smarties: Using UML for Data Modeling*; Robert J. Muller; ISBN 1558605150

Summary

DB2 UDB Universal Developer's Edition delivers all the tools you need to rapidly build and deploy applications. The package includes a full-function integrated development environment, a scalable Web application server, and DB2 UDB features such as the XML Extender. DB2 Universal Database is a scalable, industrial-strength database that will be the data management foundation for your e-business.

Additional information

For additional information, please refer to the following Web sites:

DB2 Universal Database Resources:

- http://www.software.ibm.com/data/developer
- http://www.software.ibm.com/data/db2/java

Visual Age Developer Domain:

- http://www.software.ibm.com/vadd

WebSphere Developer Domain:

- http://www.software.ibm.com/webservers/appserv
- http://www.ibm.com/websphere/developer

Accessing DB2 Stored Procedures from EJBs:

- http://www.software.ibm.com/developer/library/j-spejb/?dwzone=java

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms, which may be denoted by an asterisk(*), are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries or both and is licensed exclusively through X/Open Company Limited.

Other company, product, or service names, which may be denoted by a double asterisk(**) may be trademarks or service marks of others.