4.1 Overall Approach

In this present research pursuit, a web-based application (tool) was developed using Java to predict drug targets from bacterial genome. Using this web based application, targets were predicted and validated from 80 pathogenic microbes. The targets obtained were organized in a web based database which will facilitate next steps in drug discovery process. The application was tested using the genome data of muti-drug resistant Mycobacterium tuberculosis and the obtained drug targets were critically analyzed using bioinformatics tools and databases.

4.2 WEB-BASED APPLICATION

Algorithm for target prediction

An algorithm was designed based on the strategic approach derived to identify potential drug targets and validate them. The algorithm has three main steps:

1. Prediction of essential genes,
2. Exclusion of human homologs, and
3. Comparison with the existing drug targets.
Step-1: Prediction of Essential genes

The input (microbial genome) sequences are provided in fasta format.

Sequences are compared with the DEG database using blastall program.

The sequences matching (default cut off) with DEG is retrieved separately.

These essential genes are stored in a text file.

END

Fig. 4: Algorithm for prediction of essential genes

The DEG was manually downloaded from http://tubic.tju.edu.cn/deg/ and compiled as a database for comparison using the BLAST program against input sequences. The essential genes in the input pathogenic microbe are identified in Step-1. The resultant sequences (essential genes of the microbe) is used as an input sequence for the next step.
Step-2: Exclusion of human homologs

The predicted essential genes of the pathogenic microbe are compared against Homo sapiens. The protein sequences for Homo sapiens were downloaded from NCBI site (ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/protein/) and compared using the BLASTall program. The sequences which are not matching with the Homo sapiens are retrieved as a separate text file. This set of sequences will become the input file for the next step (i.e. target validation).

Fig. 5
Algorithm for predicting targets non homologous with Homo sapiens
Step-3: Comparison with existing drug targets

The third step involves comparison of the essential genes from the microbe which is not matching with the Homo sapiens are compared with the approved/proposed targets from Drug Bank. The gene/protein sequences of approved and proposed targets from Drug Bank is downloaded (http://redpoll.pharmacy.ualberta.ca/drugbank/) and used for the comparison.

Fig.6: Algorithm for validating the predicted targets
At each step the comparison was carried out by integrating the BLASTall exe. To compare the sequences, BLAST executable from NCBI (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/) were downloaded and customized.

The algorithm compares the input genome data with the essential genes and thereafter with the human genes to exclude the homologs based on the e-value cutoffs. Then these exclusive sequences from microbial genome which is not matching with the human homologs are compared with the approved and proposed targets from Drug Bank to validate them.

The web-based application was developed using JSP (View), Servlets (Control) and Struts (Framework). Essential BLAST parameters (-p, -i, -e, and -d) were customized to produce the desired results.

4.2.1 BLAST Parameters

BLAST - Basic Local Alignment Search Tool

Table-3 BLAST Parameters -p

<table>
<thead>
<tr>
<th>Parameter</th>
<th>-p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Specifies the type of search</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Input format</td>
<td>String</td>
</tr>
<tr>
<td>Example</td>
<td>To instructs blastall to run blastn program, use: -p blastn</td>
</tr>
</tbody>
</table>
Chapter IV

Methodology

Table 4 BLAST Parameters -d

<table>
<thead>
<tr>
<th>Parameter</th>
<th>-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Specifies the target database(s)</td>
</tr>
<tr>
<td>Default</td>
<td>Nr</td>
</tr>
<tr>
<td>Input format</td>
<td>String</td>
</tr>
<tr>
<td>Example</td>
<td>To search database named est_human, use: -d est_human</td>
</tr>
</tbody>
</table>

**Note:** Use database file name WITHOUT the file extension. Even though multiple databases can be specified in command line, using -d "nr est", database alias file is a better way to call multiple databases. For more information on database aliases, see: ftp://ftp.ncbi.nlm.nih.gov/blast/documents/formatdb.html.

Search multiple large databases together may encounter memory related problems.

Table 5 BLAST Parameters -i

<table>
<thead>
<tr>
<th>Parameter</th>
<th>-i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Specifies input query file</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Input format</td>
<td>String</td>
</tr>
<tr>
<td>Example</td>
<td>To search query file my_query.txt, use: -i my_query.txt</td>
</tr>
</tbody>
</table>

**Note:** Use the complete file name WITH its extension. The query must be in FASTA format. If multiple entries are in the input file, all queries will be searched. To use stdin (default), omit the -i and redirect the file using '<', like in blastall -pblastp -d refseq_protein <my_aa_query
### Table-6 BLAST Parameters -e

<table>
<thead>
<tr>
<th>Parameter</th>
<th>-e</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Specifies Expectation value cutoff</td>
</tr>
<tr>
<td>Default</td>
<td>10</td>
</tr>
<tr>
<td>Input format</td>
<td>Real</td>
</tr>
<tr>
<td>Example</td>
<td>To specify an e-value cutoff of $2 \times 10^{-20}$, use: -$e 2e-20$</td>
</tr>
</tbody>
</table>

**Note:** Accepted formats are integer (100), fraction (1/500), decimal (0.005), and exponential (5e-5). This parameter controls the search sensitivity. To see less significant hits, increase the setting. Relaxed settings will decrease the search speed.

The application takes the input (genome of microorganisms) in .ffn or .txt file format and provides target sequences (potential drug targets) as a single text file. The input genome sequences should contain all the sequences in fasta format in a single text file.

#### 4.3 SELECTED PATHOGENIC MICROBES

Data mining and web search was carried out to list all the pathogenic microbes. Based on prevalence and severity, 80 pathogenic microorganisms (Annexure-I) were selected for this present research work. The genome of these 80 pathogenic microbes were downloaded from NCBI ftpsite (ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria). These sequence data were stored locally for data analysis. The targets obtained from these 80
organisms were analyzed for their gene name, protein encodes, function, and pathway for further validation. These analyses were carried out manually using BLAST program and pathway search in KEGG database.

4.4 COMPREHENSIVE DATA ANALYSIS

From these 80 pathogenic microbes, multi-drug resistant strain (*Mycobacterium tuberculosis*) was selected for critical analysis. The targets obtained from the pathogenic organism were compared with the existing and proposed targets (Drug Bank) to explore new targets for drug discovery. Also the targets were analyzed for its conservity among other pathogens for its narrow or broad spectrum ranges bioinformatics tools.

4.5 WEB-BASED DATABASE

The targets obtained from the rest of the pathogenic microbes were organized in a web based database to serve as a data resource for drug discovery. The various fields in the database includes the organism name, gene name, gene sequence, protein product, pathway, Enzyme Commission Number (EC No.) and its function. The following figure shows the schema used of the database.
Identification and Validation of Drug Targets

Based on these data types, a web-based database containing potential and validated drug targets were developed using JSP (frontend) and MySQL (Backend). The database has the following fields as shown in Fig.7 and the data can be searched using organism name, gene or protein name either singly or in combination. AJAX concepts are applied to database search options to fasten the query search process. The data in the database can be updated manually or as a single upload from the spreadsheet. The spreadsheet with following template directly uploads the data to the web based database.

**Fig. 7: List of database fields and its description**

![Diagram of database fields](image.png)

**Organism**
- Name of the organism (Ex. Mycobacterium tuberculosis)

**Gene name**
- Name of the gene (Ex. nifE3)

**Gene sequence**
- A string of nucleotide sequences

**Protein products**
- Name of the protein (Ex. NifE3 protein)

**Pathway**
- Name of the pathway (Amino acid transport and metabolism)

**E.C.No**
- Enzyme Commission Number (2.8.1.7)

**Function**
- Function of the protein (Transfer the N-acyl diglyceride moa to the esterotic N-terminal cysteine in proteoprotein)

**Protein sequence**
- A string of amino acid sequences
4.6 TOOLS AND DATA RESOURCES

4.6.1 Servlet

Servlet is a server side program and web application, Servlet take the request from client and processes it and gives the response. Java Servlet has security, session management and instance persistence.

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the servlet interface, which defines life-cycle methods.

When implementing a generic service, you can use or extend the Generic servlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

4.6.2 JSP

JavaServer Pages technology put snippets of servlet code directly into a text-based document. A JSP page is a text-based document that contains two types of text: static template data, which can be expressed in...
any text-based format such as HTML, WML, and XML, and JSP elements, which determine how the page constructs dynamic content.

JavaServer Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.

The JavaServer Pages™ (JSP) technology provides a simplified, fast way to create web pages that display dynamically generated content. JSP technology was designed to make it easier and faster to build web-based applications that work with a wide variety of web servers, application servers, browsers and development tools.

**The JSP Approach to Web Application Development**

In developing the JSP specification, Sun Microsystems worked with a number of leading web server, application server and development tool vendors, as well as a diverse and experienced development community. The result is an approach that balances portability with ease-of-use for application and page developers.
JSP technology speeds the development of dynamic web pages in a number of ways

Separating content generation from presentation

Using JSP technology, web page developers use HTML or XML tags to design and format the results page. They use JSP tags or scriptlets to generate the dynamic content (the content that changes according to the request, such as requested account information or the price of a specific bottle of wine). The logic that generates the content is encapsulated in tags and JavaBeans components and tied together in scriptlets, all of which are executed on the server side. If the core logic is encapsulated in tags and beans, then other individuals, such as web masters and page designers can edit the JSP page without affecting the generation of the content.

On the server, a JSP engine interprets JSP tags and scriptlets, generates content (for example, by accessing JavaBeans components, accessing a database with JDBC™ technology, or including files), and sends the results back in the form of an HTML (or XML) page to the browser. This helps authors protect proprietary code while ensuring complete portability for any HTML-based web browser.

Emphasizing Reusable Components

Most JSP pages rely on reusable, cross-platform components (JavaBeans or Enterprise JavaBeans™ components) to perform the more complex processing required of the application. Developers can share and
exchange components that perform common operations, or make them available to larger user or customer communities. The component-based approach speeds overall development and lets organizations leverage their existing expertise and development efforts for optimal results.

**Simplifying Page Development with tags**

Web page developers are not always programmers familiar with scripting languages. The JavaServer Pages technology encapsulates much of the functionality required for dynamic content generation in easy-to-use, JSP-specific XML tags. Standard JSP tags can access and instantiate JavaBeans components, set or retrieve bean attributes, download applets, and perform other functions that are otherwise more difficult and time-consuming to code.

**The JSP Page Components**

A JSP directive passes information to the JSP engine. In this case, the first line indicates the location of some Java programming language extensions to be accessible from this page. Directives are enclosed in `<%@` and `%>` markers.

Fixed template data: Any tags that the JSP engine does not recognize it passes on with the results page. Typically, these will be HTML or XML tags. This includes the Unordered List and H1 tags in the example above.
JSP actions, or tags: These are typically implemented as standard tags or customized tags, and have XML tag syntax.

An expression: The JSP engine evaluates anything between <%= and %> markers. A scriptlet is a small script that performs functions not supported by tags or ties everything together. The native scripting language for JSP 1.0 software is based on the Java programming language.

**JSP Directives**

JSP pages use JSP directives to pass instructions to the JSP engine. These may include the following:

- **JSP Page Directives** communicate page-specific information, such as buffer and thread information or error handling.
- **Language Directives** specify the scripting language, along with any extensions.
- **The Include Directive** (shown in the example above) can be used to include an external document in the page. A good example is a copyright file or company information, file -- it is easier to maintain this file in one central location and include it in several pages than to update it in each JSP page. However, the included file can also be another JSP file.

- **A taglib Directive** indicates a library of custom tags that the page can invoke.
**JSP Tags**

Most JSP processing will be implemented through JSP-specific XML-based tags. JSP 1.0 includes a number of standard tags, referred to as the core tags. These include:

- **jsp:useBean** This tag declares the usage of an instance of a JavaBeans component. If the Bean does not already exist, then the JavaBean component instantiates and registers the tag.
- **jsp:setProperty** This sets the value of a property in a Bean.
- **jsp:getProperty** This tag gets the value of a Bean instance property, converts it to a string, and puts it in the implicit object "out".
- **jsp:include**
- **jsp:forward**

The 1.1 releases will include additional standard tags. The advantage of tags is that they are easy to use and share between applications. The real power of a tag-based syntax comes with the development of custom tag libraries, in which tool vendors or others can create and distribute tags for specific purposes.

**Scripting Elements**

JSP pages can include small scripts, called scriptlets, in a page. A scriptlet is a code fragment, executed at request time processing. Scriptlets may be combined with static elements on the page (as in the
example above) to create a dynamically generated page. Scripts are delineated within <%= and %> markers. The scripting language engine will evaluate anything within those markers. The JSP specification supports all of the usual script elements, including expressions and declarations.

**Application Models for JSP Pages**

A JSP page is executed by a JSP engine, which is installed in a web server or a JSP-enabled application server. The JSP engine receives requests from a client to a JSP page, and generates responses from the JSP page to the client. JSP pages are typically compiled into Java Servlets. Java Servlets are a standard Java extension, described in more detail at www.java.sun.com. The page developer has access to the complete Java application environment, with all of the scalability and portability of the Java technology-enabled family. When a JSP page is first called, if it does not yet exist, it is compiled into a Java Servlet class and stored in the server memory. This enables very fast responses for subsequent calls to that page. (This avoids the CGI-bin problem of spawning new processes for each HTTP request, or the runtime parsing required by server-side includes.)

JSP pages may be included in a number of different application architectures or models. JSP pages may be used in combination with different protocols, components and formats. The following sections describe a few of the possibilities.
The Future for JSP Technology

JSP technology is designed to be an open, extensible standard for building dynamic web pages. Developers will use JSP pages to create portable web applications that can run with different web and application servers for different markets, using whatever authoring tools fit their market and their needs. By working with a consortium of industry leaders, Sun has ensured that the JSP specification is open and portable. You should be able to author JSP pages anywhere and deploy them anywhere, using any client and server platforms. Over time, tool vendors and others will extend the functionality of the platform by providing customized tag libraries for specialized functions.

Java Script

JavaScript may be considered a derivative of the programming language Java. But while both are tools for providing interactivity into web pages, they are as different as bananas and papayas. Java is a complex programming environment where you create packaged ("compiled") software applications that you can insert into a web page. The learning curve for Java is monumental at best (despite claims of the expanding number of software tools). On the other hand, JavaScript offers a simpler set of programming instructions that you can enter directly among the HTML formatting of your web pages, and code that can be easily accessed and modified.
Before JavaScript, to create interactive forms (web pages with fields, buttons, and menus) we needed to write computer programs ("CGI" scripts) that resided on and ran from a web server. But with JavaScript, you can perform many form tasks without connecting to a web server. In the jargon, we are processing on the "client-side".

Even better, JavaScript allows you to create content that is dynamic, so that the code inside one web page can produce many different types of displays and features depending on the viewer's actions, including the images that change when you move the mouse over a graphic.

JavaScript combined with the absolute screen positioning available in web browsers that support HTML 4.0 provide what is known as Dynamic HTML, or DHTML.

**Advantages of JavaScript**

As stated above, JavaScript provides interactivity for your web pages without relying on server-side "CGI" (Common Gateway Interface) programming, which means your pages can be interactive even when you are not connected to the Internet. Since the code is typed directly into your HTML files, you can create Javascript with software as simple as a plain text editor. You can quickly test and modify JavaScript code. JavaScript functionality is built into most newer web browsers since 1996, so there is no extra software for the viewer to download or install.
JavaScript also provides useful commands for testing the viewer’s capability to view other types of web multimedia (i.e. whether they have Shockwave installed). Although not all web browsers may support JavaScript, there are fairly reliable methods for you to direct viewers to alternative pages.

Because of its wide use, there are numerous reference sites for learning about JavaScript as well as many sites to download free code that you can use. We’ll share a few with you in the next section.

**Disadvantages of JavaScript**

If you lack experience in programming, JavaScript will look daunting to you. Often, it is not clear how the code works when you examine the HTML source. Long, complicated JavaScript can add quite a bit of download time to your HTML page.

Although JavaScript is supported on the two major web browsers, there are a few differences that may cause problems. Many of the graphic HTML creation tools do not handle JavaScript very well.

### 4.7 MICROSOFT SQL SERVER

#### 4.7.1 Relational Database Concepts

Databases contain data that’s specifically organized. A database can be as simple as a flat file (a single computer file with data usually in a tabular form) containing names and telephone numbers of one’s friends, or
as elaborate as the worldwide reservation system of a major airline. During the 90s, the relational data access scheme came to the forefront. Relational scheme views data as rows of information.

Each row contains columns of data, called fields. The main concept in the relational scheme is that the data is uniform. Each row contains the same number of columns. One such collection of rows and columns is called a table. Many such tables form a relational database. Tables contain records. There are three types of data record relationships between records:

1. **One-to-one**. One record in a table is related to at least one record in another table.

2. **One-to-many relationship**. One record in a table could be associated with many records in another table.

3. **Many-to-many relationships**. Many records in a table could be associated with many records in another table.

### 4.7.2 Features of RDBMS

A relational DBMS must be able to manage databases entirely through its relational capabilities. All information in a relational database (including table and column names) is represented explicitly as values in tables.
Guaranteed access: Every value in a relational database is guaranteed to be accessible by using a combination of the table name, primary key value, and column name.

Systematic null value support: The DBMS provides systematic support for the treatment of null values (unknown or inapplicable data), distinct from default values, and independent of any domain.

Active, online relational catalog: The description of the database and its contents is represented at the logical level as tables and can therefore be queried using the database language. View updating—All views that are theoretically updateable can be updated through the system.

Set-level insertion, update, and deletion: The DBMS supports not only set-level retrievals but also set-level inserts, updates, and deletes.

Physical data independence: Application programs and ad hoc programs are logically unaffected when physical access methods or storage structures are altered.

Logical data independence: Application programs and ad hoc programs are logically unaffected, to the extent possible, when changes are made to the table structures.

Integrity independence: The database language must be capable of defining integrity rules. They must be stored in the online catalog, and they cannot be bypassed.
Distribution independence: Application programs and ad hoc requests are logically unaffected when data is first distributed or when it is redistributed.

No subversion: It must not be possible to bypass the integrity rules defined through the database language by using lower-level languages.

This method has several advantages and many disadvantages. In its favor is the fact that the physical structure of data on a disk becomes unimportant. The programmer simply stores pointers to the next location, so data can be accessed in this manner. Also, data can be added and deleted easily. However, different groups of information could not be easily joined to form new information. The format of the data on the disk could not be arbitrarily changed after the database was created. Doing so would require the creation of a new database structure.

4.7.3 Microsoft SQL Server

Microsoft SQL Server is a relational database management system produced by Microsoft. It supports Microsoft's version of Structured Query Language (SQL), the most common database language. It is commonly used by businesses for small- to medium-sized databases, and - in the past five years - some large enterprise databases.
Brief History of SQL Server

The code base for Microsoft SQL Server (prior to version 7.0) originated in Sybase SQL Server, and was Microsoft's entry to the enterprise-level database market, competing against Oracle, IBM, and Sybase. Microsoft, Sybase and Ashton-Tate teamed up to create and market the first version named SQL Server 4.2 for OS/2 (about 1989) which was essentially the same as Sybase SQL Server 4.0 on Unix, VMS, etc. Microsoft SQL Server for NT v4.2 was shipped around 1992 (available bundled with Microsoft OS/2 version 1.3) and was a simple port from OS/2 to NT. Microsoft SQL Server v6.0 was the first version of SQL Server that was architect for NT and did not include any direction from Sybase.

About the time Windows NT was coming out, Sybase and Microsoft parted ways and pursued their own design and marketing schemes. Microsoft negotiated exclusive rights to all versions of SQL Server written for Microsoft operating systems. Later, Sybase changed the name of its product to Adaptive Server Enterprise to avoid confusion with Microsoft SQL Server. Until 1994 Microsoft's SQL Server carried three Sybase copyright notices as an indication of its origin.

Several revisions have been done independently since with improvements for SQL Server. SQL Server 7.0 was the first true GUI based database server and was a rewrite away from the legacy Sybase code, a variant of SQL Server 2000 was the first commercial database for the Intel
IA64 architecture. During this time there was a rivalry between Microsoft and Oracle’s servers for winning the market over enterprise customers.

The current version, Microsoft SQL Server 2005, was released in November of 2005. The launch took place alongside Visual Studio 2005 and BizTalk Server 2006. The SQL Server 2005 Express edition is currently available for free download. The Microsoft SQL Server product is not just a database, it also contains (as part of the product) an enterprise ETL tool (Integration Services), Reporting Server, OLAP and messaging technologies specifically Service Broker.

**Versions for Windows**

- 1993 - SQL Server 4.21 for Windows NT
- 1995 - SQL Server 6.0, codenamed *SQL95*
- 1996 - SQL Server 6.5, codenamed *Hydra*
- 1999 - SQL Server 7.0, codenamed *Sphinx*
- 1999 - SQL Server 7.0 OLAP, codenamed *Plato*
- 2000 - SQL Server 2000 32-bit, codenamed *Shiloh*
- 2003 - SQL Server 2000 64-bit, codenamed *Liberty*
- 2005 - SQL Server 2005, codenamed *Yukon*
- Next release - codenamed *Katmai*
Description

MS SQL Server uses a variant of SQL called T-SQL, or Transact-SQL, an implementation of SQL-92 (the ISO standard for SQL, certified in 1992) with some extensions. T-SQL mainly adds additional syntax for use in stored procedures, and affects the syntax of transactions support. (Note that SQL standards require (ACID) Atomic, Consistent, and Isolated, Durable transactions.) MS SQL Server and Sybase/ASE both communicate over networks using an application-level protocol called Tabular Data Stream (TDS). The TDS protocol has also been implemented by the FreeTDS project ((1)) in order to allow more kinds of client applications to communicate with MS SQL Server and Sybase databases. MS SQL Server also supports Open Database Connectivity (ODBC).

Variants

A stripped-down version of Microsoft SQL Server known as MSDE (Microsoft SQL Server Desktop Engine) is distributed with products such as Visual Studio, Visual FoxPro, Microsoft Access, ASP.NET Web Matrix, and other products. MSDE has some restrictions: a limit of 2 GB databases, and it comes with no GUI tools to administer it. It also has a workload governor, which reduces its speed once you exceed 8 concurrent workloads on the engine.

Microsoft recently released the successor to MSDE, dubbed SQL Server Express. Similar to MSDE, SQL Express includes all the core
functionality of SQL Server and the workload governor was removed, but places restrictions on the scale of databases. It will only utilize a single CPU, 1 GB of RAM, and imposes a maximum size of 4 GB per database (log sizes don’t count). Microsoft provides a separate download (“feature pack”) for the Express edition that includes less feature rich version of Reporting Services. SQL Express also doesn’t include enterprise features such as Analysis Services, Data Transformation Services, and Notification Services. Unlike MSDE, SQL Express includes a management console, called SQL Server Management Studio Express.

4.8 JAVA DATABASE CONNECTIVITY (JDBC)

The Java Database Connectivity (JDBC) of the Java Enterprise APIs is the first of such cross-platform, cross-database approaches to database access from Java programs. The Enterprise APIs also consist have Remote Method Invocation (RMI) and serialization APIs (for Java programs to Marshall objects across namespaces and invoke methods in remote objects), Java IDL (Interface Definition Language) for communicating with CORBA, and other object-oriented systems. The JDBC APIs and the JDBC-ODBC Bridge, you can access and interact effectively with almost all databases from Java applets and applications.
4.8.1 JDBC Drivers

JDBC Drivers are set of classes that enable the Java application to communicate with databases. Java.sql that ships with JDK contains various classes for using relational databases. But these classes do not provide any implementation only the behaviors are defined. The actual implementations are done in third-party drivers. Third party vendors implement the java.sql.Driver interface in their database driver.

**JDBC technology drivers fit into one of four categories:**

1. A JDBC-ODBC bridge provides JDBC API access via one or more ODBC drivers. Note that some ODBC native code and in many cases native database client code must be loaded on each client machine that uses this type of driver. Hence, this kind of driver is generally most appropriate when automatic installation and downloading of a Java technology application is not important. For information on the JDBC-ODBC Bridge driver provided by Sun.

2. A native-API partly Java technology-enabled driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.

3. A net-protocol fully Java technology-enabled driver translates JDBC API calls into a DBMS-independent net protocol which is then translated to a DBMS protocol by a server. This net server middleware
is able to connect all of its Java technology-based clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC API alternative. It is likely that all vendors of this solution will provide products suitable for Intranet use. In order for these products to also support Internet access they must handle the additional requirements for security, access through firewalls, etc., that the Web imposes. Several vendors are adding JDBC technology-based drivers to their existing database middleware products.

4. A native-protocol fully Java technology-enabled driver converts JDBC technology calls into the network protocol used by DBMSs directly. This allows a direct call from the client machine to the DBMS server and is a practical solution for Intranet access. Since many of these protocols are proprietary the database vendors themselves will be the primary source for this style of driver. Several database vendors have these in progress.