Teiid - Scalable Information Integration

1

Teiid Embedded Quick Start Example

6.0.0

Preface

Teiid Embedded offers the power of a federated relational query engine for use with any Java application - without the need for a separate server process. Accessing your federated data is as easy as making a JDBC connection to any relational data source. This guide will take you through

- Deployment of Teiid Embedded in your environment
- Building and deploying a Virtual Database(VDB)
- Connecting to your VDB and issuing federated queries against your VDB and getting results back

Commercial development support, production support, and training for Teiid is available through JBoss. Teiid is a Professional Open Source project and a critical component of the JBoss Enterprise Data Services Platform.

1. What is This Guide About?

This guide takes you through an introduction to the concepts important to Teiid Embedded, downloading the software, and building and deploying a demonstration virtual database in 60 minutes. There is a lot to cover, so let's begin!

Note

Please read *Federation Basics* [http://www.jboss.org/teiid/basics.html] and understand different terminologies used, resources needed and artifacts that need to be generated before developing a successful application. This example takes advantage of only a minimal set of features from Teiid Embedded for the sake of simplicity and time.

Download

You need to download the binaries for *Teiid Embedded* [http://teiid.org/Download.html] . Note that there are three different artifacts are available for download.

1. Teiid Source - contains all the source code for all modules
2. Teiid Binary - contains all the binary code for all modules
3. Teiid Embedded Kit - contains only required modules and their required 3rd party dependencies

For this Quick Start, download the *Teiid Embedded Kit* and extract this tar/zip file to a directory you can access.

In the expanded directory, you will find "teiid-${version}-client.jar", which is the main client binary jar file for Teiid Embedded, and the "lib" and "extension" directories containing binaries either directly or indirectly required to execute Teiid Embedded. Later sections describe installing these artifacts.

Note

Teiid Embedded requires *Java 6* [http://java.sun.com/javase/downloads] to run.

Access to physical data sources such as Oracle, MS-SQL Server, DB2, and Sybase through Teiid relies upon the user supplying their own JDBC drivers in the deployment. Teiid Embedded has been tested extensively with *DataDirect* [http://www.datadirect.com] If you have access to these JDBC drivers, we recommend their use.

Learning about Virtual Databases (VDB)

2.1. What is a Virtual Database?

A Virtual Database (VDB) is an artifact that defines the logical schema model combining one or more physical data sources to provide easy data integration. This is accomplished by use of virtual tables, virtual views and virtual procedures, transformed by queries to their underlying physical data sources. The physical sources can be JDBC sources, delimited text files, spreadsheets or even Web services.

Your JDBC application references the virtual (logical) schema defined in this VDB. When tables, or other objects, in the VDB's models are queried, the Teiid query engine interprets the virtual schema, makes the appropriate calls to the actual physical sources, combines and translates them as requested, and returns the top level results to the application.

2.2. What is Teiid Embedded?

Teiid Embedded is a container for deploying your VDB. Before you can access your data in a federated manner, use the Teiid Designer to build a VDB. This picture shows the relationship between the tools involved.

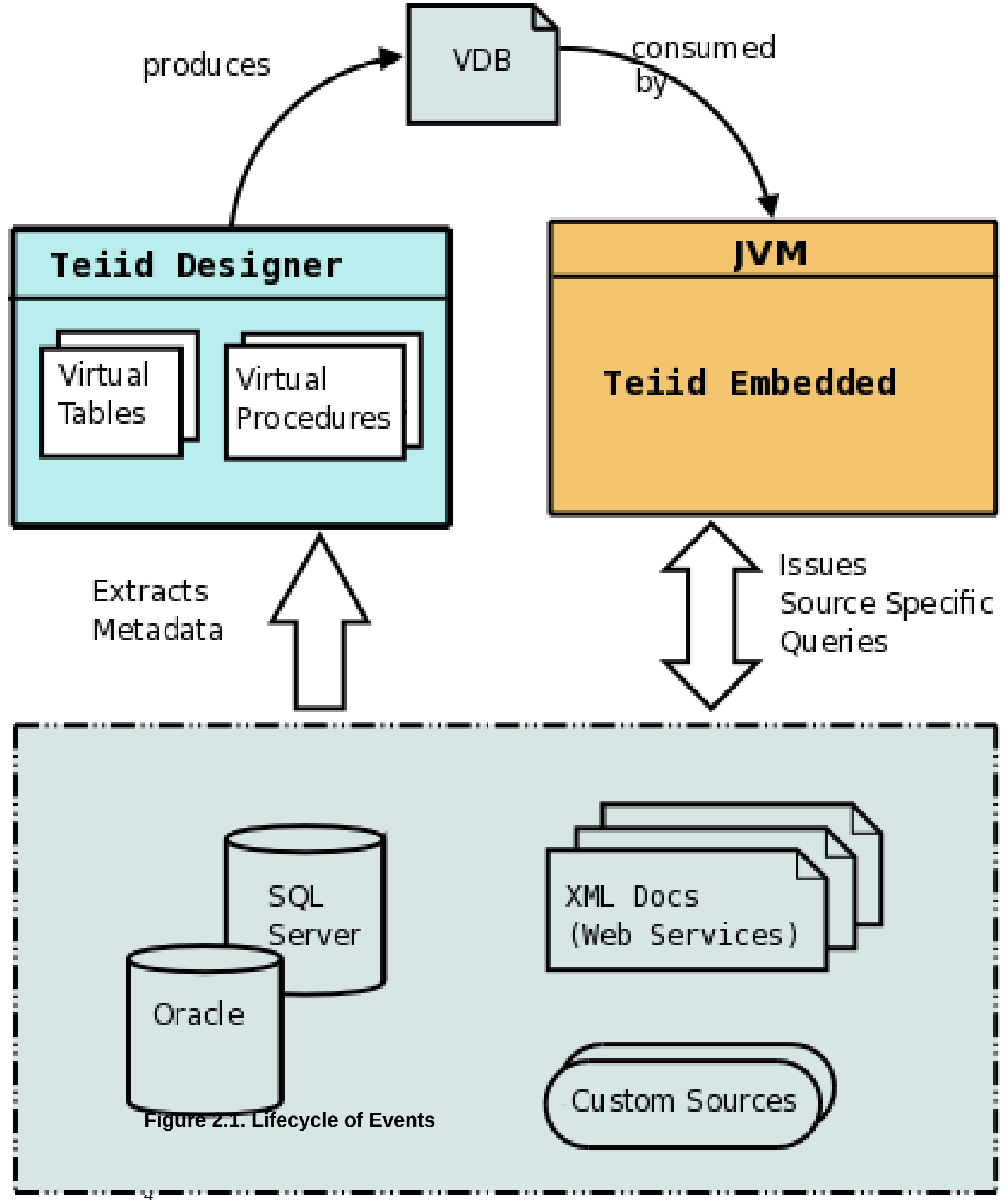

Figure 2.1. Lifecycle of Events

The Teiid Designer tool lets you define physical data sources by importing the required metadata (schema) from these sources. Once the metadata is imported, use the Designer to build a virtual data model. The collection of physical and logical models form a virtual schema. Once the user has completed the models, the Designer is used to package them in a Virtual Database, or VDB. So, a VDB can be further defined as a collection of logical (virtual) data models.

Example Explained

3.1. Portfolio Application Explained

To demonstrate how Teiid Designer and Teiid Embedded work together, follow these steps to build a simple portfolio valuation virtual database. The investor's portfolio information is stored in a Derby database and "current" stock prices are stored in a delimited text file. When completed, a single query will cause Teiid Embedded to access the relational and non-relational sources, calculate the portfolio values, and return the results. Here are some of key points to consider as you build this example:

- It's pretty simple to do
- You're querying a text file as though it was a relational database
- Imagine how much more complete this would be if you replaced the text file with a connection to a financial Web site. (Hint: You can with Teiid Embedded and a custom connector.)

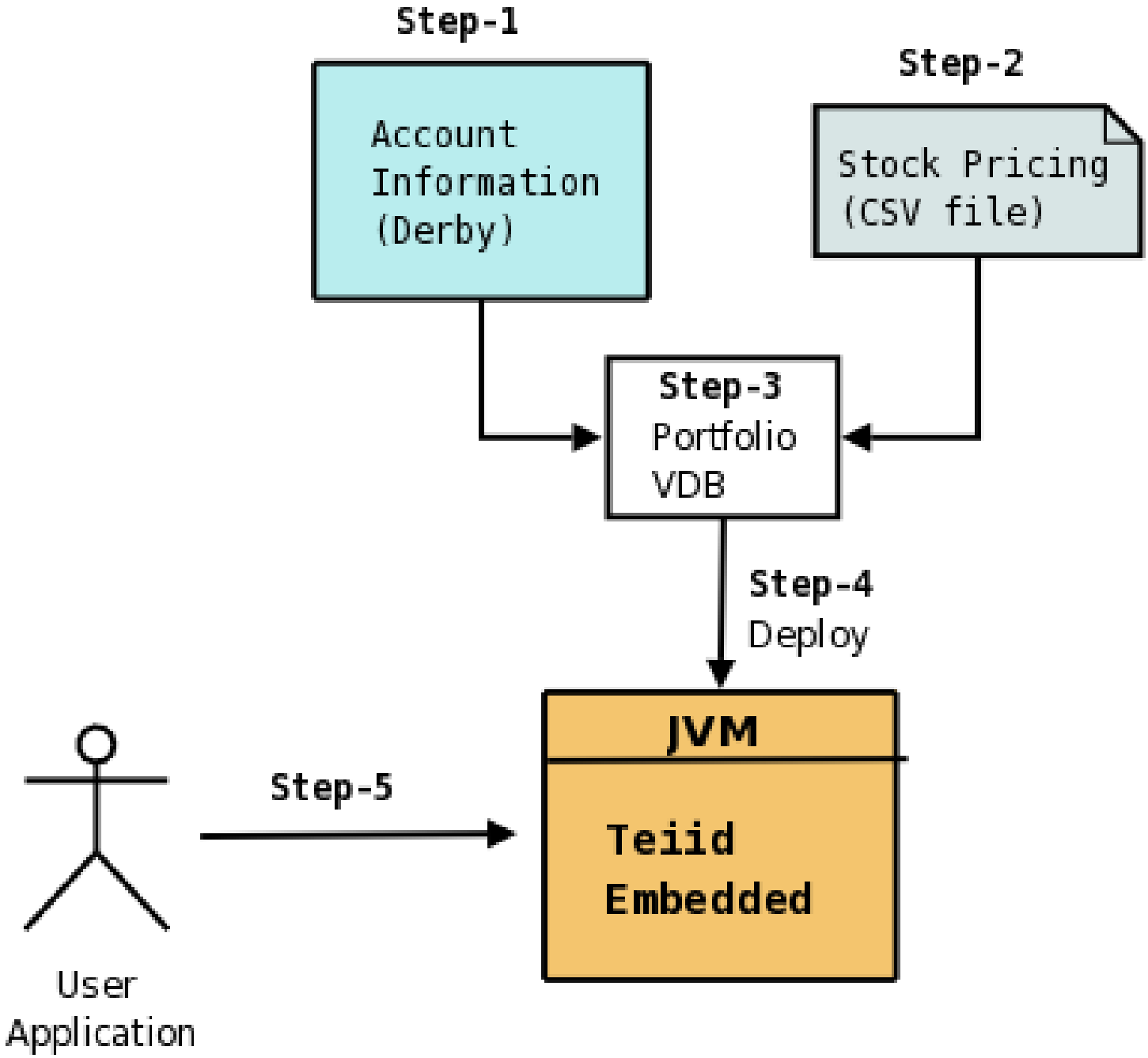

Figure 3.1. Various Steps involved in creating the example

- *Step-1: Create the Relational Database's schema and load the sample data*
- *Step-2: Describe the CSV file and its contents*
- *Step-3: Build a VDB*
- *Step-4: Deploy the VDB in Teiid Embedded*
- *Step-5: Access the VDB using JDBC API*

Note

Derby [http://db.apache.org/derby/] is suggested as it is Open Source, easily obtained, and light-weight. You can substitute any other relational database, as long as you have a suitable JDBC driver. The schema file provided, and described below, is specific to Derby, but can be easily converted for use with other databases.

3.2. Step-1: Create the Relational Database's schema and load the sample data

This example is written using *"Derby"* [http://db.apache.org/derby/] as the relational database. *Download* [http://db.apache.org/derby/derby_downloads.html] and install Derby on your machine, or if you have access to an installed instance use that. It is expected that you create a database called "accounts" in this Derby instance that is used by this example application.

Below find the corresponding schema. For the complete schema, please refer to *"examples/portfolio/customer-schema.sql"* file of the downloaded kit.

```
- -Contains the name and address of a Customer who owns portfolio account
CREATE TABLE CUSTOMER
...

--Contains Customer's account number and its current status
CREATE TABLE ACCOUNT
...

--Contains information about stock symbol, company name etc.
CREATE TABLE PRODUCT
...

--Contains each Account's holdings of Stocks
CREATE TABLE HOLDINGS
...
```

We need to start the Derby RDBMS and create the "accounts" database with the below schema. These commands are intended for a Linux environment. For starting the Derby instance on another platform, you will need to use commands appropriate to that platform.

Start a terminal session, and change directory to where Derby is installed and execute following commands

```
export DERBY_HOME=`pwd`
./bin/startNetworkServer
```

The above starts the Derby in network mode. Now, start another terminal and we will use Derby''s 'ij' tool (like SQL*PLus for Oracle) to create the schema, using the "customer-schema.sql" file in "examples/portfolio" directory

```
export DERBY_HOME=`pwd`
./bin/ij /path/to/customer-schema.sql
```

Make sure you did not have any issues when creating the schema as it is needed for going forward in this example. You can use 'ij' tool to verify the tables were created. As an alternative, you may use other tools like *SQuirreL* [http://www.squirrelsql.org/] , or *Eclipse's Data Tools* [http://www.eclipse.org/datatools/] plugin to connect to the Derby instance to check the schema and sample data.

3.3. Step-2: Describe the CSV file and its contents

In order to use a Text file as the source, we need to define two different types of files: A descriptor file and one or more data files. Conceptually, a Descriptor file defines a schema, and each data file defines data inside a table

1. Text Descriptor File - The Text Descriptor file defines the structure of the text file you want to use as a data source. The path to the data file, delimiter character used,and the number of header lines are attributes of the Descriptor file. A sample descriptor file is shown below.

```
MarketData.Price.location = /path/to/marketdata-price.txt
MarketData.Price.delimiter = ,
MarketData.Price.headerLine = 1
```

The table shown below details some of the available properties that can be used in a Descriptor file. For full details look in Text File Connector documentation

Table 3.1. Connection Properties

Property	Description
location	The path to the file on the local system or URL to remote file.
delimiter	The character the file uses to delimit the fields within each line in the file
headerLine	The line number in which the names of the columns are defined
skipHeaderLines	The number of top lines to skip in a text file (include one for the header).

Note that each property line starts with the "schema" information and followed by the logical "table" name and then the "property" itself. In the above sample file "MarketData" is the schema and "Price" is the table. The sample Descriptor file is for a 'single' table. You can define multiple tables in the same Descriptor file, but each "location" property for a given "table" must point to different data file.

2. Data File: This is the file that is identified by the "location" property in the Descriptor file. Each data file contains column information for the table. The column information is typically defined on line 1 as header line in the file, and all the following lines contain the actual rows of data. Each single line corresponds to single row. A portion of the sample file is shown below. The complete sample file is "examples/portfolio/marketdata-price.txt".

```
SYMBOL,PRICE
IBM,83.46
RHT,11.84
BA, 44.58
ORCL,17.37
```

Just locate the sample data files provided or create your own data files. Make sure that the descriptor file contains the full path to the data file. Now, both our sources are ready to be used to build a VDB

Using Teiid Designer to build a VDB

4.1. Download Teiid Designer

Note

If you would like to skip building your own VDB, you can safely go to next page and continue with the exercise as there is a pre-built VDB available for your convenience in the examples/portfolio/PortfolioModel directory of the download.

The Teiid Designer project provides a Eclipse based designer tool for this purpose of building a VDB. You can download Teiid Designer *here* [http://teiid.org/teiiddesigner.html] Once downloaded, please follow the installation instructions. Start the Teiid Designer and return here to continue building the VDB.

4.2. Building a VDB

Once you start the Designer, there are several steps you must perform to create a VDB. These steps are summarized in the following table. There is a video at the end of the list that will take you though all the steps involved.

1. Switch to the Designer Perspective, by opening the "Window->Open Perspective.."
2. Create new "Model Project", using "File->New->ModelProject"
3. Import the Derby database's "account" schema into a source model.
4. Manually create the source model for the "Text File". Create a Table called "Price" with "stock" and "price" as column names
5. Create a view model called "AccountView"
6. In the "AccountView" model, create a virtual table (called as Base Table in the menu), and build the transformation query combining the "Accounts" tables with "Price" table
7. In the "AccountView" model, create a virtual procedure called "buyStock", to learn about procedure language.
8. Create a VDB based on the models created
9. Test the created VDB with some ad-hoc queries inside Designer.
10. Export or copy the VDB for use in your application.

Deployment

Having built the VDB, it must be deployed so it can be access through a JDBC connection. Note there are two steps involved in this deployment:

1. Deploying Teiid Embedded
2. Deploying VDB

Teiid Embedded is typically used in one of two manners:

- As a *Stand-alone* Java application
- In a JEE Application Server (Deploy and Create connection pool).

This example deploys Teiid Embedded as a stand-alone Java application. The sample deployment is shown below. You can find more details about deploying to application servers on the Teiid web-site

5.1. Stand-alone Application Deployment

1. Add the "${embedded root}/teiid-${version}-client.jar" to your application classpath
2. Place your VDB in a the "${embedded root}/deploy" or another location referenced by the deploy.properties file.
3. Now, go to the next section to learn how to *connect to the VDB*

Connecting to a VDB through JDBC

At this point you have deployed Teiid Embedded and your VDB. Now it's time to connect the sample application to this VDB, issue SQL queries, and view the returned, integrated data. Note that this process is no different than connecting to any other JDBC source.

6.1. Stand-alone Java Application Deployment

For a Java application to connect to a JDBC source, it needs a URL, user-id, and password. To connect to your VDB all you need is URL and any additional optional properties that you would like to set such as log file, number of threads etc. Currently the Teiid Embedded is not associated with any user authentication system, so user-id and password are not required. The JDBC connection is supported through *"com.metamatrix.jdbc.EmbeddedDriver"* driver requires the URL syntax of

jdbc:metamatrix:<VDB-Name>

to connect to the VDB. You can add any additional optional properties at the end after the semi-colon(;) using name=value format. If multiple properties are defined, they should be separated additional semi-colons. For example

jdbc:metamatrix:<VDB-Name>;dqp.logFile=/home/query.log

You can use any of these *optional connection properties* [https://www.jboss.org/community/docs/DOC-13157] in your URL. Here is sample code showing how to make JDBC connection.

```
public void execute() throws SQLException {
    String url = "jdbc:metamatrix:Accounts";
    String sql = "select firstname, lastname from customer";

    Class.forName("com.metamatrix.jdbc.EmbeddedDriver");

    Connection connection;
    try{
        connection = DriverManager.getConnection(url);
        Statement statement = connection.createStatement();
        ResultSet results = statement.executeQuery(sql);
        while(results.next()) {
          System.out.println(results.getString(1));
          System.out.println(results.getString(2));
          ...
        }
```

```
            results.close();
            statement.close();
        } catch (SQLException e){
            e.printStackTrace();
            throw e;
        } finally {
            try{
              connection.close();
            }catch(SQLException e1){
              // ignore
            }
        }
    }
```

You can also use *com.metamatrix.jdbc.EmbeddedDataSource* to make connection in your Java application. For example, you can use following code fragment to make a connection to the VDB and issuing the query exactly same as in the above example

```
EmbeddedDataSource ds = new EmbeddedDataSource();
ds.setDatabaseName("Accounts");

Connection connection = ds.getConnection();
...
```

EmbeddedData source also provides an option to set optional parameters using the "set" methods on the data source look. For all the allowable properties at the *data source properties* [https://www.jboss.org/community/docs/DOC-13158].

6.2. Testing Your Teiid Embedded Deployment

The Teiid Embedded installation includes a simple Java class which demonstrates JDBC access of the deployed VDB. To execute this demonstration, follow these steps:

1. Ensure Derby is running
2. Change to the /examples/portfolio directory within your Teiid Embedded installation
3. Execute the run script (either for Linux or Windows)

The sample query, "select * from CustomerAccount", queries the view model and, from that, queries the two underlying data sources: one relational, one file-based. All the sample Java class

does is connect to the VDB, issue the query, and print the results. Teiid Embedded does the "heavy lifting". (For the complete code used in this example, look in the "examples/portfolio" directory.)

You are encouraged to experiment with queries that go beyond the simple "select * from CustomerAccount". Additionally, there is a sample Yahoo connector that returns stock market data. It can be used in place of the text file so the market data is more timely. Go to the Teiid Connector Sandbox projects for more information.

If your application is Web based, you can also deploy Teiid Embedded in an application server and treat it as any other JDBC source by creating a connection pool with the *com.metamatrix.jdbc.EmbeddedDataSource* and assigning it a JNDI name. Refer to *deployment to application server* [https://www.jboss.org/community/docs/DOC-13183] for more information.