Teiid - Scalable Information Integration

1

Teiid Quick Start Example

7.0

1. What is Teiid?

Teiid is runtime for executing queries against a Virtual Database or VDB. Before you can access your data in a federated manner, use can use Teiid Designer or the Dynamic VDB feature to build a VDB. This picture shows the relationship between the tools involved.

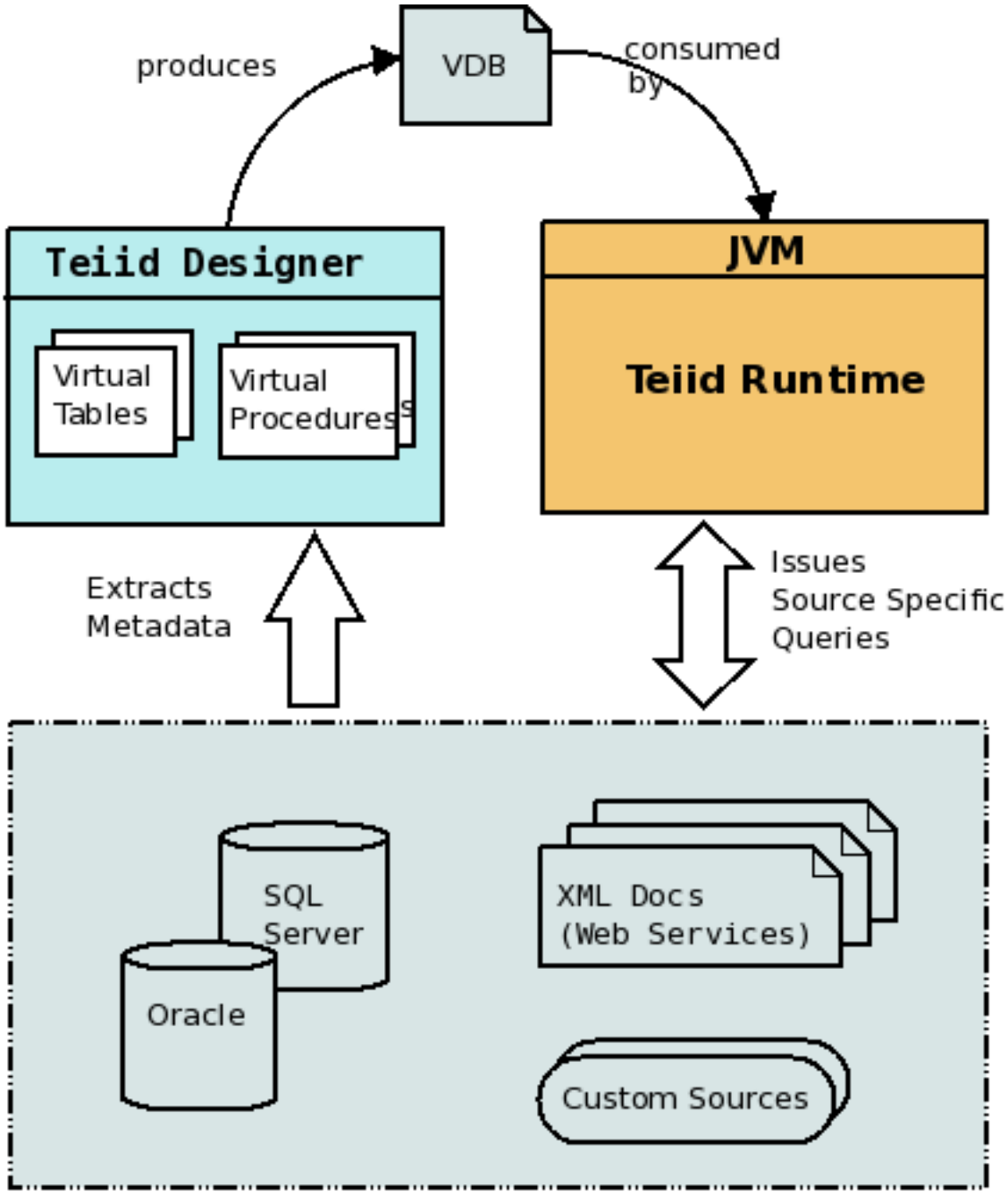

Figure 1. Lifecycle of Events

1.1. What is a Virtual Database?

A Virtual Database (VDB) is an artifact that combines one or more physical data sources to provide for easy data integration. Integration is encapsulated through view and procedures that Teiid will process in an optimized way against their respective sources. The physical sources can be JDBC sources, delimited text files, spreadsheets, or even Web services.

The Teiid Designer tool lets you define physical data sources by importing the required metadata (schema) from these sources. Once the metadata is imported, Designer can assist in building additional view layers. The collection of physical and view models will form a VDB.

2. What is This Guide About?

This guide takes you through an introduction to the concepts important to Teiid, downloading the software, and building and deploying a 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] to understand different terminologies used, resources needed, and artifacts to be generated before developing a successful application. This example takes advantage of only a minimal set of features from Teiid for the sake of simplicity and time.

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.

Download

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

1. Teiid Source - contains all of the source code
2. Teiid AdminShell - contains the admin client
3. Teiid Runtime - contains the Teiid engine and required 3rd party dependencies

For this Quick Start, download and install *JBoss AS 5.1.0* [http://www.jboss.org/jbossas/downloads.html]. Then download the Teiid runtime and unzip the contents under any of the JBoss AS profiles, such as "default" or "all". The default profile is the typical installation location, for example "<jboss-install>/server/default". The Teiid runtime directory structure matches JBoss profiles directly - it is just an overlay.

In the "<jboss-install>/server/<profile>/lib" directory, you will find "teiid-7.0-client.jar", which is the main client binary jar file for Teiid. This jar file contains the Teiid's JDBC driver and data source classes.

Note

JBoss AS 5.1 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. Copy the JDBC driver files into "<jboss-install>/server/<profile>/lib" before you create any data sources.

Example Explained

2.1. Portfolio Application Explained

The investor's portfolio information is stored in a Derby database and "current" stock prices are stored in a delimited text file. When the VDB is completed, a single query will cause Teiid to access the relational and non-relational sources, calculate the portfolio values, and return the results.

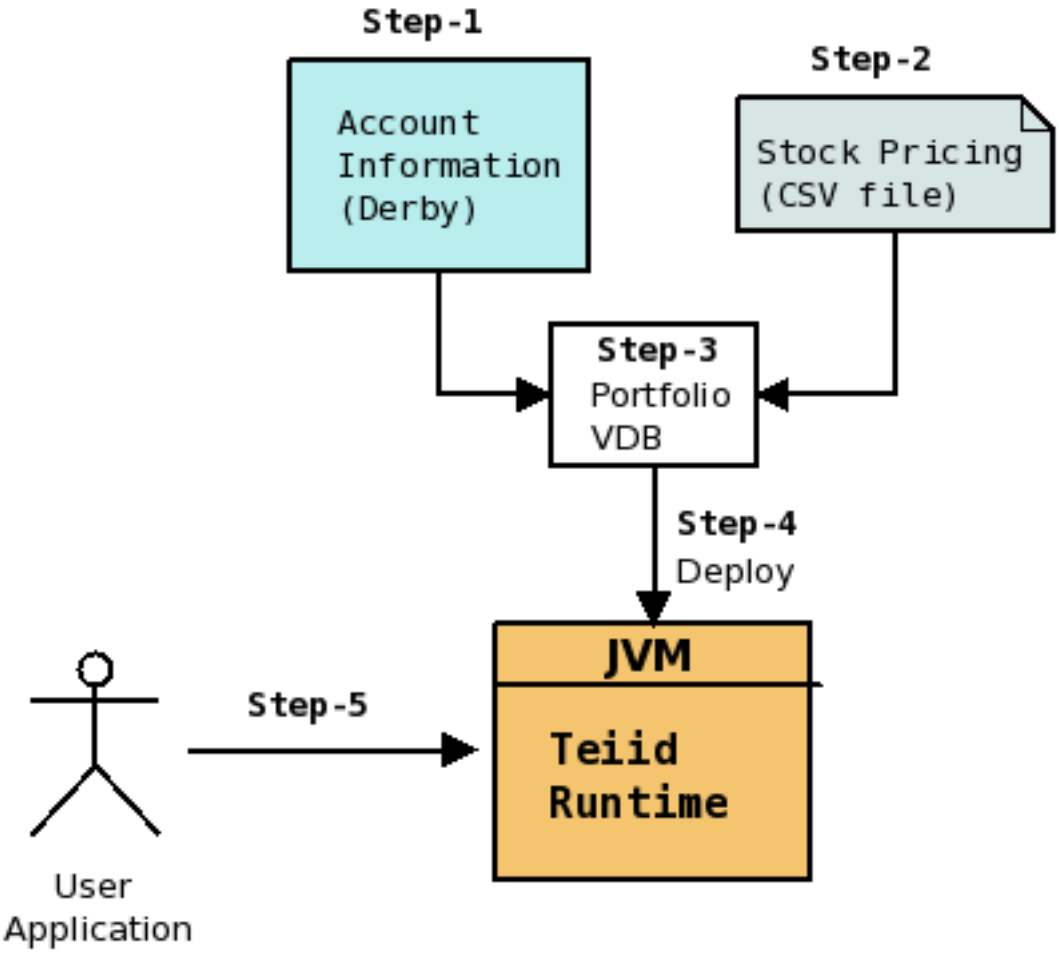

Figure 2.1. Various Steps involved in creating the example

- *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*
- *Step-5: Access the VDB using the JDBC API*

Note

Derby [http://db.apache.org/derby/] is used here since 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.

2.2. 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. An existing local or remote instance can be used if it exists.

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
```

This 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
```

This will create the accounts schema. It's abbreviated ddl is shown below.

```
--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
...
```

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.

2.3. Describe the CSV file and its contents

In order to use a Text file as the source, we need a data file defines data inside a table

1. Data 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. Now, both our sources are ready.

Building a VDB

A VDB can be built with either the *Designer* [http://www.jboss.org/teiiddesigner.html] tool or through a simple XML file called a dynamic VDB. See the "dyanmicvdb-portolio" for an example dynamic VDB. for this example purpose we will use the a dynamic VDB. If you would like to use the Designer to build your VDB, check out the Designer examples. If you need to build any view layers using your source, you must use Designer based approach to building the VDB. A sample Designer based VDB is available in the "teiid-examples/portfolio/PortfolioModel" directory.

3.1. Building Dynamic VDB

This XML file is defines a set of sources that need to treated as single source by the client application. Dynamic VDB does not yet allow for the creation of view layers. Below XML file defines "dynamicvdb-portfolio" example vdb.

portfolio-vdb.xml (copy available in "teiid-examples/dynamicvdb-portfolio" directory)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb name="DynamicPortfolio" version="1">

    <description>A Dynamic VDB</description>

    <!--
      Setting to use connector supplied metadata. Can be "true" or "cached".
      "true" will obtain metadata once for every launch of Teiid.
      "cached" will save a file containing the metadata into
      the deploy/<vdb name>/<vdb version/META-INF directory
    -->
    <property name="UseConnectorMetadata" value="cached" />


    <!--
      Each model represents a access to one or more sources.
      The name of the model will be used as a top level schema name
      for all of the metadata imported from the connector.

      NOTE: Multiple model, with different import settings, can be bound to
      the same connector binding and will be treated as the same source at
      runtime.
    -->
    <model name="MarketData">
        <!--
```

```
        Each source represents a translator and data source. There are
        pre-defined translators, or you can create one. ConnectionFactories
        or DataSources in JBoss AS they are typically defined using "xxx-ds.xml" files.
      -->
                    <source name="text-connector" translator-name="file" connection-jndi-
name="java:marketdata-file"/>
  </model>

  <model name="Accounts">
    <!--
      JDBC Import settings

      importer.useFullSchemaName directs the importer to drop the source
      schema from the Teiid object name, so that the Teiid fully qualified name
      will be in the form of <model name>.<table name>
    -->
    <property name="importer.useFullSchemaName" value="false"/>

     <!--
       This connector is defined in the "derby-connector-ds.xml"
      -->
                <source name="derby-connector" translator-name="derby" connection-jndi-
name="java:PortfolioDS"/>
  </model>

</vdb>
```

The XML file is explained below.

1. The above XML file defines a "vdb" with name "DynamicPortfolio" with version "1"

2. A "model" XML element represents a schema that being integrated. This sample defines two sources, "MarketData" that represents the schema for the text file that has the stock price information and "Accounts" that represents the "portfolio" schema in the Derby database.

3. The "source" element inside the "model" element defines name of the source (can be any name), and name of the translator (defines the type of the source like oracle, db2, mysql, h2, file, ws etc..) and the "connection-jndi-name" defines the source's JNDI name in the JBoss AS container.

4. Also note that inside the "model" elements, some "property" elements are used to define how metadata can be imported from source. For more information check out the Reference Guide's Dynamic VDB section.

5. Note that you need to also create the necessary deployment files for the data sources (Connection Factories) too.

Deployment

Having built the VDB, it must be deployed into Teiid server, so it can be accessed through a JDBC connection.

This example deploys Teiid within a JBoss AS. The sample deployment is shown below. You can find more details about deploying to application servers on the Teiid web-site

4.1. JBoss AS Application Deployment

1. Before we can deploy the VDB to the server, we need to create and deploy the required Connection Factories for Derby and File sources. Connection Factories are sources that provide the data then their data is integrated through Teiid. If you are familiar with creating data sources or connection factories in JBoss AS, then this is exactly the same operation.

   For this example we need to create three (2) different data sources. You would first need to create a data source to Derby, then to File source. See or copy the following

   ```
   ${teiid-examples}/portfolio/portfolio-ds.xml
   ${teiid-examples}/portfolio/marketdata-file-ds.xml
   ```

   files into "${jboss-install}/server/{profile}/deploy" directory.

2. The above data sources provide the data to your VDB. Now to deploy your VDBl place your VDB file (either the dynamicvdb-portfolio/portfolio-vdb.xml file or the full .vdb file) in a the "${jboss-install}/server/{profile}/deploy" directory. If the JBoss AS is not already started, start by running the "${jboss-install}/bin/run.sh" or "${jboss-install}/bin/run.bat" scripts. Check the logs or the console to make sure there were no errors during the deployment of the VDB.

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 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 like Oracle.

5.1. Stand-alone Java Application Deployment

Before you can make a JDBC connection to the Teiid VDB, add the Teiid's driver jar file to your application's classpath

```
${jboss-install}/server/${profile}/lib/teiid-7.0-client.jar
```

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 a URL and any additional optional properties that you would like to set. Teiid defaults to allowing the "admin" as user with password as "teiid". Additional user accounts can be added. A JDBC connection can be obtained through the Teiid driver `"org.teiid.jdbc.TeiidDriver"` with the URL syntax of

jdbc:teiid:<VDB-Name>@mm(s)://<host name>:<port>

You can add optional properties at the end of the URL using a semi-colon(;) name=value format. For example

jdbc:teiid:<VDB-Name>@mm(s)://<host name>:<port>;autoCommitTxn=DETECT

Check out Client Developer's guide for all the optional connection properties in your URL. Here is sample code showing how to make JDBC connection.

```
public void execute() throws SQLException {
   String url = "jdbc:teiid:Portfolio@mm://localhost:31000";
   String sql = "select firstname, lastname from customer";

   Class.forName("org.teiid.jdbc.TeiidDriver");

   Connection connection;
   try{
      connection = DriverManager.getConnection(url, "admin", "teiid");
      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 `org.teiid.jdbc.TeiidDataSource` 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

```
TeiidDataSource ds = new TeiidDataSource();
ds.setDatabaseName("Portfolio");
ds.setUser("admin");
ds.setPassword("teiid");

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

TeiidDataSource source also provides an option to set optional parameters using the "set" methods on the data source look. For all the allowable data source properties check out Client Developer's Guide.

5.2. Testing Your Teiid Deployment

The Teiid 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 ${jboss-install}/server/profile/teiid-examples/portfolio directory
3. Execute the run script (either for Linux or Windows)

Depending on the VDB you deployed, see the relevant README file for example queries. If you are using a graphical client, such as *SQuirreL* [http://www.squirrelsql.org/], have a look at the metadata tree to see not only what is exposed by your VDB, but also the SYS schema tables.

If your application is Web based, you can create data source for your VDB using the above and treat it as any other JDBC source using `org.teiid.jdbc.TeiidDataSource` and assigning it a JNDI name. Refer to Client Developer's Guide deployment for more information on creating a DataSource.

Note

"embedded" mode is only available in versions of Teiid up to 6.2