UnityJDBC User Documentation

UnityJDBC User Documentation

Table of Contents

1.	General Information	
	Overview of UnityJDBC	1
2.	Installation	2
	Overview	2
	System Requirements	2
	Quick Setup and Installation	2
	Installation Walkthrough	8
	Configuring Data Sources	13
3.	Using SourceBuilder - Tutorial on Multiple Database Querying	15
	Using the Sample Databases	. 15
	Data Virtualization Using SourceBuilder	16
	Multiple Database Virtualization and Querying for HSQLDB	19
	Multiple Database Virtualization and Querying for IBM DB2	. 20
	Multiple Database Virtualization and Querying for Microsoft SQL Server	
	Multiple Database Virtualization and Querying for MySQL	
	Multiple Database Virtualization and Querying for Oracle	
	Multiple Database Virtualization and Querying for PostgreSQL	
	Multiple Database Virtualization and Querying for Sybase	
	Multiple Database Virtualization and Querying for MongoDB	
	Multiple Database Virtualization and Querying for Other JDBC/ODBC Sources	
4.	Multiple Database Programming with UnityJDBC	
	Using the Sample Programs	
	Using ExampleQuery.java	
	Using ExampleUpdate.java	
	Using ExampleMetadata.java	
	Using ExampleEngine.java	
	Using ExampleNoFileConnection.java	
5.	Using UnityJDBC with Query and Reporting Software	
	UnityJDBC Data Virtualization with SQuirreL SQL	
	UnityJDBC Data Virtualization with JasperReports	
	UnityJDBC Data Virtualization with Aqua Data Studio	
	UnityJDBC Data Virtualization with RazorSQL	
6.	Supported SQL Syntax	
	Overview	
	Data Types	
	Identifiers	
	Functions and Operators	
	Logical Operators	
	Comparison Operators	
	Arithmetic Functions and Operators	
	String Functions	
	Pattern Matching Operators	
	Data Type Conversion Functions	
	Date/Time Functions and Operators	
	Aggregate Functions	
	User-Defined Functions and Support for Other Functions	
	Function Translation	
	Non-parsed Functions	
	SELECT Statement	
	INSERT Statement	
	UPDATE Statement	
		. 55

DELETE Statement	65
EXPLAIN Statement	65
By-Pass Statement	65
7. Supported JDBC Methods	66
Overview	66
8. UnityJDBC Driver Internals	. 67
Overview	67
Embedded Relational Database Engine	67
9. History and Planned Features	68
History of UnityJDBC	68
Planned Features	
Feature List	68
Contacts and Support	69

List of Tables

6.1. Comparison Operators	58
6.2. Mathematical Operators	
6.3. Mathematical Functions	59
6.4. String Functions	60
6.5. Date Functions	61
6.6. Aggregate Functions	61
9.1. UnityJDBC Release History	68
9.2. Planned Features in Coming Versions	
9.3. UnityJDBC Feature List	

Chapter 1. General Information

Overview of UnityJDBC

UnityJDBC is a Type 4 JDBC driver capable of querying multiple databases in a single SQL query. The driver can be used similar to other JDBC drivers including with query, reporting, and business intelligence tools, application and web servers, or stand-alone Java programs. Internally, UnityJDBC contains a database engine and optimizer allowing it to efficiently join data from source databases to produce a single ResultSet. UnityJDBC supports updating data using results produced from cross-database queries and performs automatic dialect translation to convert queries into the proper dialect. A brief list of the major supported features is below:

- supports cross-database joins of any number or type of JDBC-accessible sources (Microsoft SQL Server, Oracle, DB2, Postgres, MySQL, Sybase, MongoDB, Splunk, ServiceNow, Cassandra, etc.)
- allows SQL-based comparison of data across databases to detect data inconsistencies, errors, or for synchronization of data between databases
- performs SQL dialect translation and automatically executes functions and features internally in the driver if the data source does not support them
- contains an advanced optimizer and query processor that performs efficient query processing by having each source process as much of the query as possible (e.g. "push-down filters")
- cross-database queries can be used to insert records into tables (INSERT INTO ... SELECT)
- supports cross-database PreparedStatements
- · has a driver by-pass feature to allow direct access to individual sources
- supports connection pools and connection properties
- supports user-defined functions
- · works with any data source that has a JDBC driver and will run on any Java supported platform
- works with all SQL query software including SQuirreL SQL, Aqua Data Studio, Toad, and RazorSQL
- works will business intelligence and reporting software including Tableau, JasperReports, Pentaho, and Splunk

Chapter 2. Installation

Overview

The free evaluation version of UnityJDBC is a fully functioning system. The only constraint with the evaluation version is that it will only produce the first 100 results in a ResultSet. The full version has no restrictions. The evaluation version can be downloaded at https://www.unityjdbc.com/download.php. You are free to distribute the evaluation version of the software.

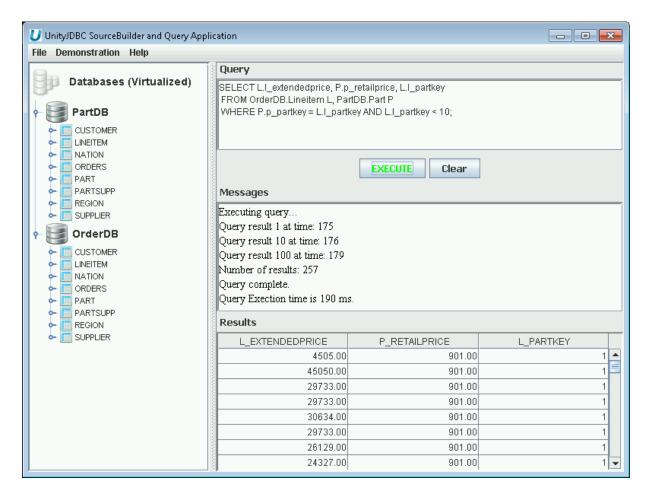
System Requirements

UnityJDBC requires a JRE of 1.6 or higher. UnityJDBC will run on a J2SE or J2EE platform.

Quick Setup and Installation

UnityJDBC can be downloaded, installed, tested, and configured for your environment in less than 10 minutes. Here are the easy steps:

- 1. **Downloading** First, download the UnityJDBC package to your computer from http://www.unityjdbc.com/ download.php.
- 2. Installation The UnityJDBC installation package contains the driver, a simple query GUI, and some test programs. The UnityJDBC_Trial_Install.jar can be run by typing the command: java -jar UnityJDBC_Trial_Install.jar. Follow the prompts to install UnityJDBC on your machine.
- 3. **Quick Start** The installation creates a shortcut to the UnityJDBC SourceBuilder which is a simple graphical query editor and configuration tool. You can also run it directly using initsources.bator initsources.sh in the installation directory. To try the sample queries, you will also want to click on the Start Database shortcut or run the script startDB.bat or startDB.sh in the directory sampleDB/hsqldb. Below is a screenshot of running a sample query that joins across two databases.



Multiple Database Query Example

- 4. Writing queries UnityJDBC uses the standard SQL language. You can include as many databases in the virtual database that you wish. To reference a table in a given database, prefix it with the database name. In the example above, OrderDB.Lineitem refers to the Lineitem table in OrderDB (a user-assigned name for the sample HSQL database), and PartDB.Part refers to the Part table in the PartDB database. Writing queries is as easy as prefixing the table name with the database name!
- 5. Creating Your Own Virtual Database A virtual database is defined using a sources XML file, often with the default filename of sources.xml. Each database also has a schema file storing information on its tables and fields. These XML files are automatically built by SourceBuilder, but can also be edited directly at any time. A few quick steps to produce your own virtual database:
 - a. Select File->New Source Group and use the default sources.xml file.
 - b. Select File->New Source and provide the JDBC connection information for your source. The screenshots below show MySQL and PostgreSQL.

U Source Connection Setup				
Sources file: sources.xml				
	Name :	mysql		
Source Type	Schema file:	mysql.xml		
Cassandra Generic JDBC	URL:	idbc:mysqt://cs-suse-4.ok.ubc.ca/tpch		
Generic ODBC	Driver:	com.mysql.jdbc.Driver		
HSQLDB IBM DB2	User:	rlawrenc		
Microsoft SQL Server	Password:	••••		
MongoDB	Catalog:			
MySQL Oracle	Schema:			
PostgreSQL	Tables included:			
ServiceNow Sybase	Tables excluded:	.*\\$.*		
Vectorwise	Statistics:	Row Counts 💌		
		Add Source		
Progress: 11 of 11	Success.			
Messages SELECT COUNT(") FROM tpch.table1 Done processing table: table1				
Table: table2 Schema: null Catalog: tpch Computing table statistics SELECT COUNT(*) FROM tpch.table2 Done processing table: table2				

Multiple Database Virtualization - Adding MySQL Database

U Source Connection Setup					
Sources file: sources.xml					
	Name:	postgres			
Source Type	Schema file:	postgres.xml			
Cassandra Generic JDBC	URL:	jdbc:postgresql://db.ok.ubc.ca/tpch			
Generic ODBC	Driver:	org.postgresql.Driver			
HSQLDB	User:	rlawrenc			
IBM DB2 Microsoft SQL Server	Password:	••••			
MongoDB	Catalog:				
MySQL	Schema:				
Oracle PostgreSQL	Tables included:				
ServiceNow					
Sybase	Tables excluded:	ANS.*			
Vectorwise	Statistics:	Row Counts			
	Add Source				
Progress: 10 of 10	Success.				
Messages					
Preparing to add data source					
Registering driver: org.postgresql.Driver					
Connecting to database Connection successful.					
Auto-generated database id: 40080408 Database name: PostgreSQL 8					
Table: customer Schema: null Catalog: null					
Computing table statistics SELECT COUNT(*)					
EPOM cuctomor					

Multiple Database Virtualization - Adding PostgreSQL Database

c. Write a query that can join across multiple databases. The only difference from standard SQL is that you prefix the table name with the database name.

U UnityJDBC SourceBuilder and Query Application				
Query Databases (Virtualized) Image: mysql				
 Image: part pp partsupp Image: partsupp partsupp Image: partsupp partsupartsupp partsupp partsupp partsupp partsupp partsupartsupp par	c_custkey	numOrders 18 13 22 14 16 16 19 17 17	totalOrderPrice	

Multiple Database Query Example that Joins Table in MySQL with a Table in PostgreSQL

- d. After creating your virtual database, you will have one sources.xml and an XML schema file for each database. These files can be moved to any location and are used to configure UnityJDBC when using it with other applications.
- 6. Using UnityJDBC Now that a virtual database is created, there are three general steps to using UnityJDBC common in all cases.
 - a. Move the sources.xml and the XML schema files for the databases into a desired directory. For this example, the directory is /unityjdbc.
 - b. Put the unityjdbc.jar into your CLASSPATH as well as the drivers for each database. (Some drivers are included with the UnityJDBC distribution in the directory drivers in the installation folder.) A common location is in <JAVA_HOME>/jre/lib/ext.
 - c. The connection information is the UnityJDBC driver class unity.jdbc.UnityDriver and the URL is jdbc:unity://<relative or absolute path to sources file>such as jdbc:unity:// unityjdbc/sources.xml.

The UnityJDBC driver can be used with any software that supports JDBC. A screenshot of configuring it in Aqua Data Studio is below. The same connection information applies to all JDBC-based software.

🛅 Edit Server Properties			×
蹐 General 🔐 Filter 🛛 🐺 Adva	nced 😡 Permissions		
RDBMS:			
Oracle 8i	Name:	UnityJDBC	
Oracle 8i - OCI Oracle 9i/10g Oracle 9i/10g - OCI	Туре:	Production	
MS SQL Server 2000/5 MS SQL Server 2000/5 - ODBC	Authentication		
MS SQL Server 7.0 MS SQL Server 7.0 - ODBC	Login Name:		
Sybase ASE 11/12/15 - TDS 4.5 Sybase ASE 11/12/15 - TDS 5.5	Password:		
Sybase Anywhere DB2 for LUW 7.x DB2 for LUW 8.x Informix 7/9/10	Save password:	v	
Informix 7/9/10 - ODBC MySQL	Location ———		
PostgreSQL Generic - JDBC	URL:	jdbc:unity://c:/unityjdbc/sources.xml	
Generic - ODBC	Driver:	unity.jdbc.UnityDriver	
	Driver Location:		
Mounted Scripts:			
Folder:			
		Ok Close	

Registering the UnityJDBC Driver in Aqua Data Studio

UnityJDBC can also be used with your own Java programs. There is sample code in the code directory in the installation folder. Here are two commands (executed from the code directory) to compile and run the sample code:

```
javac test/ExampleQuery.java
```

```
java test.ExampleQuery
```

If you have issues with compiling or running, try to explicitly indicate the location of the UnityJDBC JAR:

```
javac -cp .;./unityjdbc.jar;../sampleDB/hsqldb/hsqldb.jar test/
ExampleQuery.java
```

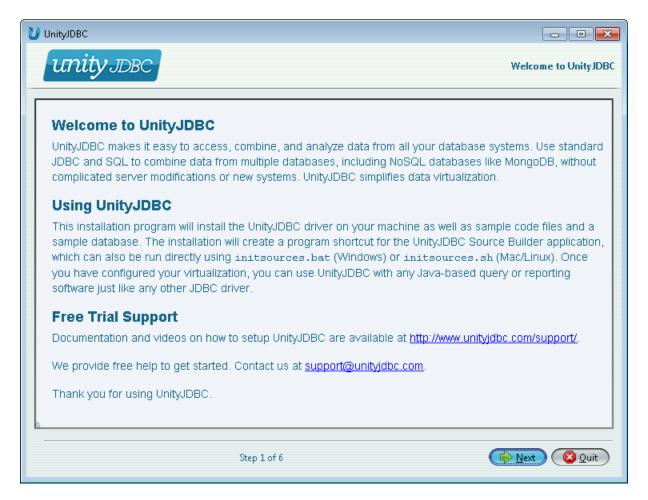
```
java -cp .;../unityjdbc.jar;../sampleDB/hsqldb/hsqldb.jar test.ExampleQuery
```

To create your own Java program, copy the file ExampleQuery.java to MyQuery.java. There are 2 lines that you must modify. The first line indicates where your new sources XML file is located on your machine. You may specify an absolute or relative path from the current directory. The second line you must modify is to change the SQL query to reference fields and tables in your data source(s). Compile and run the program. Queries can reference any table or field in any data source in your XML sources file as long as you prefix a table or field with the database name such as MyDB.MyTable.MyField.

Installation Walkthrough

Once you have downloaded UnityJDBC from http://www.unityjdbc.com/download.php, you will have downloaded a JAR file called UnityJDBC_Trial_Install.jar. First, you need to have Java previously installed in order to install UnityJDBC. Then, you can install UnityJDBC by either double-clicking the JAR file or running the following command: java -jar UnityJDBC_Trial_Install.jar. The installation steps are below.

1. Welcome Screen - Provides some background on UnityJDBC. Click Next.



UnityJDBC Installation Part #1 - Welcome Screen

2. End-User License Agreement - Read the EULA, select Accept, and click Next.

Please read the follow	ving license agreement carefully:		
UnitvJDBC E	ND-USER LICENSE AG	REEMENT (EULA)	
This End-User Lic person) and Unity Product") and its : portion of the Soft	ense Agreement (the "Agreeme Data Inc. ("UnityData") with res associated documentation. By a	nt") is between you (the "Customer", an organiza bect to the use of the software UnityJDBC (the "S ccepting this Agreement or by installing or using rees to enter into this Agreement. A contract is th	Software any
1. GRANT OF	LICENSE		
Software Product,		ferable, limited, and non-exclusive license to us usiness purposes, and subject to the terms and	e the
2. LICENSE F	ESTRICTIONS		
There cannot be	any reverse engineering or soft	ed, copied, distributed, and installed with no res /are modification that removes the trial restriction software are licensed on a per machine basis. /	ns of the
I accept the terms of	this license agreement.		

UnityJDBC Installation Part #2 - End-User License Agreement

3. Install Path - Select the installation path for UnityJDBC and then click Next.

UnityJDBC	
unity JDBC	Select an Installation Location
Select the installation path: C:\Program Files\UnityJDBC	Browse
Step 3 of 6	Previous Rext Quit

UnityJDBC Installation Part #3 - Select Installation Path

4. Install Progress - The installation will be performed and progress shown. Click Next when complete.

UnityJDBC	- • •
unity JDBC	Installing Unity JDBC Files
Pack installation progress:	
[Finished]	
Overall installation progress:	
1/1	
Step 4 of 6	Previous Kext Quit

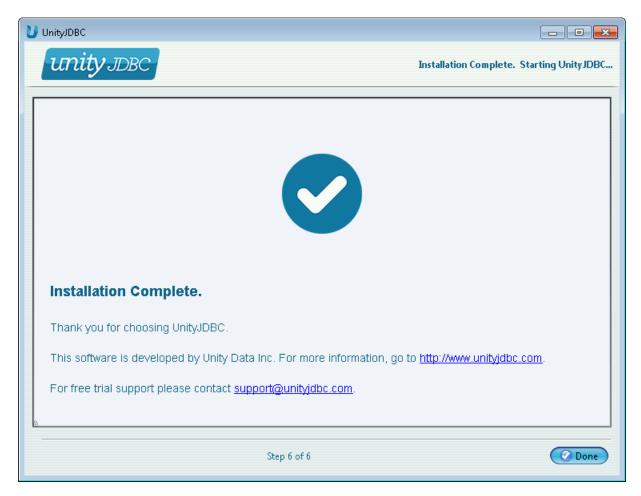
UnityJDBC Installation Part #4 - Installation Progress

5. **Install Shortcuts** - The installation will install shortcuts to start SourceBuilder and a sample database. This screen allows you to control if shortcuts are created and their location. When finished, click Next.

UnityJDBC	
unity JDBC	Specify Shortcuts for UnityJDBC
 Create shortcuts in the Start-Menu Create additional shortcuts on the desktop Select a Program Group for the Shortcuts: 	
7-Zip	create shortcut for:
Accessories	📃 🔘 current user
Administrative Tools	I users
Advantage Database Server 10.10	
Apache Tomcat 7.0	
Aptana	
ArgoUML	
Astah Community	
CamStudio	
DBEdit 2	
DbSchema)>
UnityJDBC	Default
Step 5 of 6	Previous 🕞 <u>N</u> ext Quit

UnityJDBC Installation Part #5 - Select Shortcuts

6. **Installation Complete** - The installation complete screen will be displayed if the installation is successful. UnityJDBC SourceBuilder will be automatically started. Click Done to close the install window.



UnityJDBC Installation Part #6 - Installation Complete

UnityJDBC will auto-start at the completion of the installation. If it does not, click on the shortcut created. If there is no shortcut, on Windows run initsources.bat and on Linux/Mac run initsources.sh in the installation directory.

Configuring Data Sources

UnityJDBC requires information about the data sources being queried in order to validate, optimize, and execute queries against those data sources. All source information is stored in XML files. There are two types of source information files: the *sources file* and *schema files*. A *sources file* with default name <code>sources.xml</code> provides information on all the sources that could be potentially queried. The location of this file is provided via the URL when initializing the driver. Inside the file is information on each source including its connection URL and parameters, JDBC driver, and schema file location. The sample sources file code\test\xspec\UnityDemo.xml is provided in the distribution package. Each data source requires a schema file. The *schema file* is an XML encoding of the schema information including table and field names, keys, joins, and relation sizes. It is used for validating queries and optimization. Two schema files provided in the distribution are: code\test\xspec\UnityDemoOrder.xml and code\test\xspec\UnityDemoPart.xml.

There are two ways to create your own sources file and associated schema files:

1. The easiest way is to use the SourceBuilder GUI that can be started using the installed shortcut or running initsources.bat or initsources.sh. This GUI will automatically extract source information and build

the necessary files. If you are using UnityJDBC as the multisource plugin in SQuirreL SQL, all features of the SourceBuilder are integrated into SQuirreL, and the XML files are generated automatically.

2. You can manually build a sources file using a text editor. To produce a schema file for a source, open up the program called com/unityjdbc/SourceBuilder/SchemaExtractor.java in the code directory. Modify the JDBC URL, driver path, and output directory accordingly and run the program. The account that you connect with must have read access to the database and associated tables that you want to access. After the XML schema file has been produced, move it to the directory where you want it and update the sources file to reference the correct location. In almost all cases, using the SourceBuilder utility will be faster and easier.

Chapter 3. Using SourceBuilder -Tutorial on Multiple Database Querying

Using the Sample Databases

SourceBuilder is a graphical query tool that allows you to define a virtual database consisting of multiple databases including MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Sybase, MongoDB, and others. SourceBuilder can be started using the shortcut produced during the installation, by running initsources.bat or initsources.sh in the installation folder, or directly from the driver using the command java -jar unityjdbc.jar. It is required that the unityjdbc.jar and all JDBC driver JAR files for databases used be in the Java classpath.

When SourceBuilder is first started, a welcome screen is displayed.

U UnityJDBC SourceBuilder and Query Application		
File Demonstration Help		
	Query	
	Query EXECUTE Clear Messages Welcome! To get started either: 1. Create your own sources by: a. File Menu->New Source Group and create new XML config file. b. File Menu->New Source and enter the JDBC connection information for each source. Results	

SourceBuilder Welcome Screen

To use the sample database and queries included, make sure to start the HSQLDB using the Start Database shortcut or run the script startDB.bat or startDB.sh in the directory sampleDB/hsqldb. Then, select Demonstration->Cross-database Join Test. Click the EXECUTE button to run the query and get results. If an error occurs, verify that the sample database is started and the hsqldb.jar file is in your classpath. You can try several other of the sample queries or write your own.

UnityJDBC SourceBuilder and Query App File Demonstration Help			
9-2	Query		
Databases (Virtualized) PartDB CUSTOMER CUSTOMER CUSTOMER ANATION CORDERS PART CARTSUPP CREGION CUSTOMER CUSTO	SELECT L.I_extendedprice, P.p_I FROM OrderDB.Lineitem L, PartI WHERE P.p_partkey= L.I_partke Executing query Query result 1 at time: 175 Query result 10 at time: 176 Query result 100 at time: 179 Number of results: 257 Query complete. Query Exection time is 190 ms.	DB.Part P	
🕶 🧰 REGION	Results		
- III SUPPLIER	L_EXTENDEDPRICE	P_RETAILPRICE	L_PARTKEY
	4505.00	901.00	1 🔺
	45050.00	901.00	1
	29733.00	901.00	1
	29733.00	901.00	1
	30634.00	901.00	1
	29733.00	901.00	1
	26129.00	901.00	1
	24327.00	901.00	1 🔻

SourceBuilder Multiple Database Query Example with a Cross-Database Join

Data Virtualization Using SourceBuilder

To build your own data virtualization of multiple data sources, create a new sources file by selecting File->New Source Group and providing a file name (the default is sources.xml).

UnityJDBC Source Builder and Query Appli	ication
File Demonstration Help	
New Source Group	Query
Load Source Group	
New Source	
Remove Source	
Remove Table	
Refresh Exit	EXECUTE
	Messages
	Welcome! To get started either: 1. Create your own sources by: a. File Menu->New Source Group and create new XML config file. b. File Menu->New Source and enter the JDBC connection information for each source. Results

Creating a New Source Group with SourceBuilder

🕖 Setup Sou	rce Group Configuration File	.
	Enter Location for Source Group XML Configuration File	
File name:	sources.xml]
	ΟΚ	

Prompt for Name of Sources File

Then, you will add each one of your sources by providing its JDBC connection information. To add a source you need the following information:

- 1. The JDBC Driver class name (e.g. com.mysql.jdbc.Driver).
- 2. The JDBC URL to connect to the source (e.g. jdbc:mysql://localhost/mydb).
- 3. User and password information if not specified in the JDBC URL.

4. A unique database name (does not have to be the same as the system database name) to refer to the data source in your data virtualization.

U Source Connection Setup			×
Sources file: sources.xml			
	Name:	mysql	
Source Type	Schema file:	mysql.xml	
Cassandra Generic JDBC	URL:	idbc:mysqt://cs-suse-4.ok.ubc.ca/tpch	
Generic ODBC	Driver:	com.mysql.jdbc.Driver	
HSQLDB IBM DB2	User:	rlawrenc	
Microsoft SQL Server	Password:	••••	
MongoDB MySQL	Catalog:		
Oracle	Schema:		
PostgreSQL	Tables included:		
ServiceNow	Tables excluded:		
Sybase	Tables excluded:		
Vectorwise	Statistics:	Row Counts 💌	
		Add Source	
Progress: 11 of 11	Success.		
Messages SELECT COUNT(*) FROM tpch.table1 Done processing table: table1 Table: table2 Schema: null Catalog: tpch Computing table statistics SELECT COUNT(*) FROM tpch.table2 Done processing table: table2 Success.			

An example of adding a MySQL source to the virtual database is below.

Adding a MySQL Data Source to the Data Virtualization

There are also several additional features that can be used to control the extraction.

- 1. Schema Specify a schema name or pattern (JDBC API) to only retrieve tables in the given schema. This is especially important for Oracle as by default tables from all schemas will be added to the virtualization.
- 2. **Tables included** Specify a JDBC API pattern (use % for wild card character match) to indicate which tables should be added to the extraction. For example, a pattern of R% will only add tables that begin with R.
- 3. **Tables excluded** Specify a Java string pattern (use .* for matching any sequence of characters) to indicate what tables to exclude from extraction. Each database has certain default exclusion patterns to avoid extracting system tables. It is recommended to modify the exclusion pattern if there are any issues extracting too many tables, especially system tables for your data source.
- 4. **Statistics** Statistics collection helps the UnityJDBC optimizer perform more efficient data virtualization queries. The default statistics collection is Row Counts which will calculate the number of rows for each extracted table.

The All statistics setting collects rows counts as well as value distribution information for each field. Some sources do not support collecting field distribution information in which case the Row Counts setting should be used. A setting of None will collect no statistics and is the fastest when performing extraction. UnityJDBC will execute queries with no statistics perfectly fine, although statistics do help in query planning for complex queries involving many tables.

Below is an example of extracting only the tables that end in 'ER' from an Oracle database. Note how the schema is also supplied as RLAWRENC.

U Source Connection Setup			×
Sources file: source	s.xml	Exit	
	Name :	oracle_tpch	
Source Type	Schema file:	oracle_tpch.xml	
Cassandra Generic JDBC	URL:	idbc:oracle:thin:rlawrenc/test@cssql.ok.ubc.ca:1521/xe	
Generic ODBC	Driver:	oracle.jdbc.driver.OracleDriver	
HSQLDB IBM DB2	User:	rlawrenc	
Microsoft SQL Server	Password:	••••	
MongoDB MySQL	Catalog:		
Oracle	Schema:	RLAWRENC	
PostgreSQL ServiceNow	Tables included:	%ER	
Sybase	Tables excluded:	.*\\$.*	
Vectorwise	Statistics:	Row Counts 💌	
		Add Source	
Progress: 2 of 2 Success.			
Messages			
Done processing table	CUSTOMER		
Table: SUPPLIER Schema: RLAWRENC Catalog: null Writing information into XML file. Updating sources file. SUCCESS. Data source successfully added. Computing table statistics SELECT COUNT(*) FROM RLAWRENC.SUPPLIER Done processing table: SUPPLIER			
Success.			

Adding an Oracle Data Source to the Data Virtualization (note use of schema name)

Multiple Database Virtualization and Querying for HSQLDB

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more HSQLDB databases with any other database system. To add a HSQLDB data source, the following information is used:

- 1. JDBC Driver class name: org.hsqldb.jdbcDriver.
- 2. JDBC URL: jdbc:hsqldb:hsql://<host address>/<database name>.

An example of adding a HSQLDB source to a data virtualzation is below.

U Source Connection Setup		
Sources file: sources.xml		
	Name:	hsqldb
Source Type	Schema file:	hsqldb.xml
Cassandra Generic JDBC	URL:	idbc:hsqldb:hsql://localhost/tpch
Generic ODBC	Driver:	org.hsqldb.jdbcDriver
HSQLDB IBM DB2	User:	
Microsoft SQL Server	Password:	
MongoDB MySQL	Catalog:	
Oracle	Schema:	
PostgreSQL	Tables included:	
ServiceNow Sybase	Tables excluded:	/\\$.*
Vectorwise	Statistics:	Row Counts 💌
		Add Source
Progress: 8 of 8 St	ICCESS.	
Progress: 8 01 8 St	iccess.	
Messages SELECT COUNT(*) FROM "PUBLIC".REGI Done processing table		▲
Table: SUPPLIER Schema: null Catalog: PUBLIC Computing table statistics SELECT COUNT(*) FROM "PUBLIC".SUPPLIER Done processing table: SUPPLIER Success.		

Adding a HSQLDB Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for IBM DB2

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more IBM DB2 databases with any other database system. To add an IBM DB2 data source, the following information is used:

- 1. JDBC Driver class name: com.ibm.db2.jcc.DB2Driver.
- 2. JDBC URL: jdbc:db2://<host name>/<database name>.

An example of adding an IBM DB2 source to a data virtualzation is below.

U Source Connection Setup			×
Sources file: sources.xml			
	Name:	db2	
	Schema file:	db2.xml	
Source Type	URL:	jdbe:db2://essql.ok.ube.ca/tpeh	
Generic JDBC			
Generic ODBC	Driver:	com.ibm.db2.jcc.DB2Driver	
HSQLDB	User:	rlawrenc	
IBM DB2 Microsoft SQL Server	Password:		
MySQL	Schema:		
Oracle			
PostgreSQL	Tables included:		
Sybase	Tables excluded:	.*\\$.*	
	Statistics:	Row Counts	
Add Source			
Progress: 9 of 9 Su	Progress: 9 of 9 Success.		
Progress, o or o or o			
Messages			
SELECT COUNT(*)			-
FROM Partsupp Done processing table	· Partsunn		
Done processing table. Partsupp			
Table: Region Schema: null Catalog: tpch			
Computing table statistics			
SELECT COUNT(*) FROM Region			
Done processing table: Region			
			=
Table: Supplier Schema: null Catalog: tpch Computing table statistics			-
Computing table statistics Writing information into XML file.			-

Adding an IBM DB2 Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for Microsoft SQL Server

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more Microsoft SQL Server databases with any other database system. To add a Microsoft SQL Server data source, the following information is used:

1. JDBC Driver class name: com.microsoft.sqlserver.jdbc.SQLServerDriver.

```
2. JDBC URL: jdbc:sqlserver://<host>;DatabaseName=<database name>.
```

An example of adding a Microsoft SQL Server data source to a data virtualzation is below.

Sources file: sources.xml Exit Sources Type Schema file: sqlserver Cassandra URL: idbct.sqlserver.kml Cassandra URL: idbct.sqlserver.kml Cassandra URL: idbct.sqlserver.kml Generic JDBC Driver: commicrosoft.sqlserver.phbc.Sol.ServerDriver HS0LDB User: rlsvrenc Microsoft SQL Server Password:	U Source Connection Setup			
Source Type Schema file: sqlserver.xml Cassandra URL: idbc:sqlserver.i/cssql.ok.ubc.ca,DatabaseName=TPCH100 Generic JDBC Driver: com microsoft.sqlserver.jdbc.SQLServerDriver HSQLDB User: rlawrenc IBM DB2 Password: •••• MongoDB Catalog:	Sources file: sources.xml			
Cassandra Generic JDBC Generic JDBC Generic JDBC Generic ODBC Driver: com.microsoft sqlserver.i/cssql.ok.ubc.ca,DatabaseName=TPCH100 Generic ODBC Driver: com.microsoft sqlserver.i/dbc.SQLServerDriver HSQLDB User: rlawrenc MongoDB Catalog: MySQL Oracle PostgreSQL Schema: ServiceNow Sybase Vectorwise Statistics: Row Counts Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(°) FROM tpch100.dbc.REGION Done processing table: REGION Table: SUPPLIER Schema: dbc Catalog: tpch100 Computing table statistics		Name :	sqlserver	
Generic JDBC URL: idbc:sqlserver://cssql.ok.ubc.ca,DatabaseName=TPCH100 Generic ODBC Driver: com.microsoft.sqlserveridbc.SQLServerDriver HSQLDB User: rlawrenc IBM DB2 Password: •••• MongoDB Catalog:		Schema file:	sqlserver.xml	
Generic ODBC Driver: com.microsoft.sqlserver.jdbc.SQLServerDriver HSQLDB User: rlawrenc IBM DB2 Password: •••• MongoDB Catalog: ····· MySQL Catalog: ····· Oracle Schema: ····· PostgreSQL Tables included: ····· ServiceNow Tables excluded: *%.* ····· Sybase Tables excluded: *%.* ····· Vectorwise Statistics: Row Counts ▼ Add Source Porgress: 8 of 8 Success.		URL:	idbc:sqlserver://cssql.ok.ubc.ca;DatabaseName=TPCH100	
IBM DB2 User: rlawrenc Microsoft SQL Server Password: ●●●● MongoDB Catalog: ●●●● MySQL Oracle Schema: ●●●● PostgreSQL Tables included: ●●●● ServiceNow Tables included: ●●●●● Sybase Tables excluded: ^*%.* Vectorwise Statistics: Row Counts ▼ Add Source Progress: 8 of 8 Success.		Driver:	com.microsoft.sqlserver.jdbc.SQLServerDriver	
Microsoft SQL Server Password: ●●●●● MongoDB Catalog:		User:	rlawrenc	
MySQL Oracle Schema: PostgreSQL Tables included: ServiceNow Sybase Tables excluded: *\\$.* Vectorwise Statistics: Row Counts Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics		Password:	••••	
Oracle Schema: PostgreSQL Tables included: ServiceNow Sybase Sybase Tables excluded: Vectorwise Statistics: Row Counts Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics	-	Catalog:		
PostgreSQL Tables included: ServiceNow Tables excluded: Sybase Tables excluded: Vectorwise Statistics: Row Counts Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics		Schema:		
Sybase Tables excluded: Vectorwise Statistics: Row Counts Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics		Tables included:		
Sydase Vectorwise Statistics: Row Counts Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics		Tables excluded:	* 2/*	
Add Source Progress: 8 of 8 Success. Messages SELECT COUNT(*) FROM tpch100.dbo REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics				
Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics			Add Source	
Messages SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo Catalog: tpch100 Computing table statistics	Progress: 8 of 8 Su	ICCESS.		
SELECT COUNT(*) FROM tpch100.dbo.REGION Done processing table: REGION Table: SUPPLIER Schema: dbo_Catalog: tpch100 Computing table statistics	i rogicoor e ere e e			
Computing table statistics	SELECT COUNT(*) FROM tpch100.dbo.RE			
FROM tpch100.dbo.SUPPLIER Done processing table: SUPPLIER Success.	Computing table stati SELECT COUNT(*) FROM tpch100.dbo.SU Done processing table	PPLIER	h100	

Adding a Microsoft SQL Server Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for MySQL

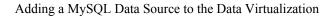
Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more MySQL databases with any other database system. To add a MySQL data source, the following information is used:

1. JDBC Driver class name: com.mysql.jdbc.Driver.

2. JDBC URL: jdbc:mysql://<host address>/<database name>.

An example of adding a MySQL source to a data virtualzation is below.

U Source Connection Setup		
Sources file: sources	s.xml	Exit
	Name:	mysql
Source Type	Schema file:	mysql.xml
Cassandra Generic JDBC	URL:	idbc:mysql://cs-suse-4.ok.ubc.ca/tpch
Generic ODBC	Driver:	com.mysql.jdbc.Driver
HSQLDB	User:	rlawrenc
IBM DB2	Password:	
Microsoft SQL Server MongoDB		
MySQL	Catalog:	
Oracle	Schema:	
PostgreSQL ServiceNow	Tables included:	
Sybase	Tables excluded:	.*\\$.*
Vectorwise	Statistics:	Row Counts 💌
		Add Source
Progress: 11 of 11 S	Success.	
Messages Select COUNT(")		
FROM tpch.table1 Done processing table	: table1	
Fable: table2_Schema: null_Catalog: tpch		
BELECT COUNT(*)		
FROM tpch.table2 Done processing table	: table2	
	n - Anna an an Anna	
Success.		



Multiple Database Virtualization and Querying for Oracle

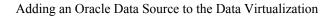
Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more Oracle databases with any other database system. To add an Oracle data source, the following information is used:

1. JDBC Driver class name: oracle.jdbc.driver.OracleDriver.

2. JDBC URL: jdbc:oracle:thin:<user>/<password>@<server>:1521/<service>.

An example of adding an Oracle data source to a data virtualzation is below.

U Source Connection Setup			×
Sources file: sources	Sources file: sources.xml		
	Name:	oracle	
Source Type	Schema file:	oracle.xml	
Cassandra Generic JDBC	URL:	idbctoracletthintrlawrenc/xxx@cssql.ok.ubc.cat1521/xe	
Generic ODBC	Driver:	oracle.jdbc.driver.OracleDriver	
HSQLDB IBM DB2	User:		
Microsoft SQL Server	Password:		
MongoDB MySQL	Catalog:		
Oracle	Schema:	RLAWRENC	
PostgreSQL	Tables included:		
ServiceNow Sybase	Tables excluded:	.*\\$.*	
Vectorwise	Statistics:	Row Counts 💌	
		Add Source	
Progress: 9 of 9 Su	ICCESS.		
Messages Done processing table	REGION		
Table: SUPPLIER Sch Writing information into Updating sources file. SUCCESS. Data sour SELECT COUNT(*) FROM RLAWRENC.SU Done processing table) XML file. ce successfully added. IPPLIER	alog: null Computing table statistics	
Success.			-



Multiple Database Virtualization and Querying for PostgreSQL

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more PostgreSQL databases with any other database system. To add a PostgreSQL data source, the following information is used:

1. JDBC Driver class name: org.postgresql.Driver.

2. JDBC URL: jdbc:postgresql://<server>/<database>? user=<userId>&password=<password>.

An example of adding a PostgreSQL source to a data virtualzation is below.

U Source Connection Setup			x
Sources file: sources	s.xml	Exit	
	Name:	postgres	
Source Type	Schema file:	postgres.xml	
Cassandra	URL:	idbe:postgresgl://db.ok.ube.ca/tpch	
Generic JDBC Generic ODBC	Driver:	org.postgresgl.Driver	
HSQLDB			
IBM DB2	User:	rlawrenc	
Microsoft SQL Server	Password:	••••	
MongoDB MySQL	Catalog:		
Oracle	Schema:		
PostgreSQL	Tables included:		
ServiceNow	Tables excluded:	^\\$.*	
Sybase Vectorwise		Row Counts V	
Vectorwise	Statistics:		
Add Source			
Progress: 10 of 10 S	Success.		
Messages Preparing to add data s	source		-
Registering driver: org.postgresql.Driver Connecting to database Connection successful.			=
Auto-generated database id: 40080408 Database name: PostgreSQL 8			
Table: customer_Schema: null_Catalog: null Computing table statistics SELECT COUNT(*) EROM customer			•

Adding a PostgreSQL Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for Sybase

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more Sybase databases with any other database system. To add a Sybase data source, the following information is used:

- 1. JDBC Driver class name: com.sybase.jdbc4.jdbc.SybDriver.
- 2. JDBC URL: jdbc:sybase:Tds:<server>:<port>/<database>? user=<userid>&password=<password>.

An example of adding a Sybase source to a data virtualzation is below.

U Source Connection Setup		
Sources file: sources	s.xml	Exit
	Name:	sybase
Source Type	Schema file:	sybase.xml
Cassandra Generic JDBC	URL:	idbc:sybase:Tds:cssql.ok.ubc.ca:5000/tpch
Generic ODBC	Driver:	com.sybase.jdbc4.jdbc.SybDriver
HSQLDB	User:	
IBM DB2		
Microsoft SQL Server MongoDB	Password:	•••••
MySQL	Catalog:	
Oracle	Schema:	
PostgreSQL	Tables included:	
ServiceNow Sybase	Tables excluded:	.*\\$.*
Vectorwise	Statistics:	Row Counts 💌
		Add Source
Progress: 9 of 9 Su	ICCESS.	
Progress, sors 30	100033.	
Messages Computing table stat SELECT COUNT(*) FROM tpch.dbo.region Done processing table Table: supplier Schem Computing table stat SELECT COUNT(*) FROM tpch.dbo.suppli	e: region ha: dbo_Catalog: tpch istics er	
Done processing table	e: supplier	
•		

Adding a Sybase Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for MongoDB

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization of one or more MongoDB databases (collections) with any other database system. Note that a MongoDB JDBC driver is built into UnityJDBC and can be used separately from UnityJDBC directly. For more information go to http://www.unityjdbc.com/mongojdbc To add a MongoDB data source, the following information is used:

1. JDBC Driver class name: mongodb.jdbc.MongoDriver.

```
2. JDBC URL: jdbc:mongo://<server>:1521/<database>.
```

An example of adding a MongoDB data source to a data virtualzation is below.

U Source Connection Setup			
Sources file: sources.xml			
	Name:	mongo	
Source Type	Schema file:	mongo.xml	
Cassandra Generic JDBC	URL:	idbe:mongo://ds029847.mongolab.com:29847/tpch	
Generic ODBC	Driver:	mongodb.jdbc.MongoDriver	
HSQLDB IBM DB2	User:	dbuser	
Microsoft SQL Server	Password:	•••••	
MongoDB MySQL	Catalog:		
Oracle	Schema:		
PostgreSQL	Tables included:		
ServiceNow	Tables excluded:	*\\$.*	
Sybase Vectorwise	Statistics:	Row Counts V	
	Statistics.	Add Source	
Progress: 14 of 14 Success.			
Messages			
Registering driver: mongodb.jdbc.MongoDriver Connecting to database			
Connection successful.			
Auto-generated database id: 80020202 Database name: MongoDB 2.2			
Table: _schema Schema: null Catalog: null			
Computing table statistics			
SELECT COUNT(*) FROM "_schema"			

Adding a MongoDB Data Source to the Data Virtualization

Multiple Database Virtualization and Querying for Other JDBC/ODBC Sources

Using UnityJDBC SourceBuilder, it is possible to build a data virtualization for any data source that supports JDBC or ODBC including Microsoft Access databases, Excel files, text files, and any data source that has a JDBC driver.

Chapter 4. Multiple Database Programming with UnityJDBC

Using the Sample Programs

UnityJDBC is a universal query translator. It allows you to develop your programs without worrying about the underlying database. All SQL statements executed with UnityJDBC are translated for the database used. Even if you do not need multiple database queries, data virtualization, or queries that span multiple different databases, UnityJDBC simplifies your development by handling all the issues with SQL dialects. If a function is not supported by your database, UnityJDBC will execute it internally. If you forget a function name for a database, it will translate to the correct function call for the particular source. This allows you the freedom to write your SQL code in a database independent way. You no longer have to make major changes to your code if you change database systems. If you want full control, you can use the UnityJDBC database engine directly in your code to join ResultSets with each other regardless of their source, and perform dynamic filtering, ordering, and analysis.

Sample programs are provided in the directory code. Here is a list of the programs and the features they demonstrate:

- 1. ExampleQuery. java a query example that joins data across two databases
- 2. ExampleUpdate.java demonstrates INSERT/UPDATE/DELETE and how to store a cross-database query result into a table
- 3. ExampleMetaData.java query example showing how to extract metadata information
- 4. ExampleEngine.java an advanced example that shows how users can use the UnityJDBC database engine directly
- 5. ExampleNoFileConnection.java example showing how to configure UnityJDBC in code without using XML files

All of these examples use a local HSQL database that can be started using the script startDB.bat or startDB.sh in the directory sampleDB/hsqldb in the installation folder.

To compile and run any of these sample programs make sure you are in the code directory and execute the following commands:

javac test/ExampleQuery.java

java test.ExampleQuery

If you have CLASSPATH issues, you can explicitly indicate the location of the HSQL JDBC driver and the UnityJDBC driver by:

javac -cp.;../UnityJDBC.jar;../sampleDB/hsqldb/hsqldb.jar test/ ExampleQuery.java

java -cp.;../UnityJDBC.jar;../sampleDB/hsqldb/hsqldb.jar test.ExampleQuery

Using ExampleQuery.java

The ExampleQuery. java demonstrates the basic features of the UnityJDBC driver. The code is below.

```
import java.sql.*;
public class ExampleQuery
{
// URL for sources.xml file specifying what databases to integrate.
// This file must be locally accessible or available via http URL.
static String url="jdbc:unity://test/xspec/UnityDemo.xml";
public static void main(String [] args) throws Exception
Connection con = null;
Statement stmt = null;
ResultSet rst;
try {
   // Create new instance of UnityDriver and make connection
   System.out.println("\nRegistering driver.");
   Class.forName("unity.jdbc.UnityDriver");
   System.out.println("\nGetting connection: "+url);
   con = DriverManager.getConnection(url);
   System.out.println("\nConnection successful for "+ url);
   System.out.println("\nCreating statement.");
   stmt = con.createStatement();
   // Unity supports scrollable ResultSets,
   // but better performance with FORWARD_ONLY
   // stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
   11
                                 ResultSet.CONCUR_READ_ONLY);
   // A query is exactly like SQL.
   // Attributes should be FULLY qualified: database.table.field
   // Statement must end with a semi-colon ;
   // This query performs cross-database join on the client-side
   String sql =
     "SELECT PartDB.Part.P_NAME, OrderDB.LineItem.L_QUANTITY,"
   + " OrderDB.Customer.C Name, PartDB.Supplier.s name"
   + " FROM OrderDB.CUSTOMER, OrderDB.LINEITEM, OrderDB.ORDERS,"
   + "
            PartDB.PART, PartDB.Supplier"
   + " WHERE OrderDB.LINEITEM.L_PARTKEY = PartDB.PART.P_PARTKEY AND"
   + "
             OrderDB.LINEITEM.L ORDERKEY = OrderDB.ORDERS.O ORDERKEY"
   + "
             AND OrderDB.ORDERS.O_CUSTKEY = OrderDB.CUSTOMER.C_CUSTKEY"
   + "
             AND PartDB.supplier.s_suppkey = OrderDB.lineitem.l_suppkey"
   + " AND OrderDB.Customer.C_Name = 'Customer#000000025';";
   // Note: Client's local JVM is used to process some operations.
   // For large queries, this may require setting a large heap space.
   // JVM command line parameters: 0 -Xms500m -Xmx500m
   // These parameters set heap space to 500 MB.
  rst = stmt.executeQuery(sql);
   System.out.println("\n\nTHE RESULTS:");
   int i=0;
   long timeStart = System.currentTimeMillis();
```

```
long timeEnd;
   ResultSetMetaData meta = rst.getMetaData();
   System.out.println("Total columns: " + meta.getColumnCount());
   System.out.print(meta.getColumnName(1));
   for (int j = 2; j <= meta.getColumnCount(); j++)</pre>
      System.out.print(", " + meta.getColumnName(j));
   System.out.println();
   while (rst.next()) {
      System.out.print(rst.getObject(1));
      for (int j = 2; j <= meta.getColumnCount(); j++)</pre>
         System.out.print(", " + rst.getObject(j));
      System.out.println();
      i++;
   }
   timeEnd = System.currentTimeMillis();
   System.out.println("Query took: "+
                      ((timeEnd-timeStart)/1000)+" seconds");
   System.out.println("Number of results printed: "+i);
   stmt.close();
   System.out.println("\nOPERATION COMPLETED SUCCESSFULLY!");
catch (SQLException ex)
{
    System.out.println("SQLException: " + ex);
finally
   if (con != null)
   try{
        con.close();
                          }
   catch (SQLException ex)
       System.out.println("SQLException: " + ex); }
   {
```

}

}

{

} }

The UnityJDBC driver behaves exactly like other JDBC drivers. The basic steps for querying a database with a JDBC driver are:

1. Load the driver (optional) - This is done by Class.forName("unity.jdbc.UnityDriver");

- 2. Make a connection A connection is made to a database by providing the database URL and other properties including user id and password. This example is using the DriverManager to make the connection (con = DriverManager.getConnection(url);). Note that the URL is of the form jdbc:unity://<path to sources file>. In this case, the URL is jdbc:unity://test/xspec/ UnityDemo.xml.This path may be an absolute or relative path on the machine. It is also possible to retrieve encrypted XML files from a network source. The sources file provides the connection information for the individual data sources for use by UnityJDBC.
- 3. Execute a statement UnityJDBC follows the JDBC API for creating statements and executing queries and updates. There are some methods unique to UnityJDBC which are covered in a later section. Standard SQL syntax is supported. The major difference is that tables in different databases can be referenced in the same query. This is accomplished using the syntax database.table to refer to tables and database.table.field to refer to fields. (Note that aliasing using AS is supported.) If full names are not provided, UnityJDBC will attempt to match as appropriate, but it will generate errors if the provided names are not unique.

This file is a good one to modify to start your own program. Simply change the class and file name, the URL to the location of your source list file, and the query executed, and you are done.

Using ExampleUpdate.java

UnityJDBC natively supports INSERT, UPDATE, and DELETE statements on a single database. These statements can be executed in by-pass mode in which case UnityJDBC does not parse or validate the statement and passes it straight to the JDBC driver for the corresponding database. In native mode, UnityJDBC will parse and validate the statement before passing it to the data source. Note that the basic INSERT, UPDATE, and DELETE statements operate only on a single table in SQL, so no cross-database query processing is necessary. A sample of the code in ExampleUpdate.java is below.

```
Class.forName("unity.jdbc.UnityDriver");
con = DriverManager.getConnection(url);
stmt = con.createStatement();
// Example #1: Basic query
String sql = "SELECT * FROM mydb.Customer;";
rst = stmt.executeQuery(sql);
printResult(rst);
// Example #2: DELETE using native parsing
String databaseName = "mydb";
sql = "DELETE FROM mydb.customer WHERE id = 51 or id=52;";
stmt.executeUpdate(sql);
// Example #3: INSERT (by-pass method)
sql = "INSERT INTO Customer (id,firstname,lastname,street,city) "
     + " VALUES (51, 'Joe', 'Smith', 'Drury Lane', 'Detroit')";
((UnityStatement) stmt).executeByPassQuery(databaseName,sql);
// Example #4: INSERT - Unity Parsed
sql = "INSERT INTO mydb.Customer (id, firstname, "
          + " lastname, street, city) "
          + " VALUES (52, 'Fred', 'Jones', 'Smith Lane', 'Chicago'); ";
stmt.executeUpdate(sql);
// Example #5: INSERT INTO (SELECT...) across databases
sql = "INSERT INTO emptydb.customer (SELECT * FROM mydb.customer);";
stmt.executeUpdate(sql);
// Prove that we transferred the data
sql = "SELECT * FROM emptydb.Customer;";
rst = stmt.executeQuery(sql);
printResult(rst);
```

Note that you can use the by-pass feature to execute any statement on a source database that UnityJDBC does not natively support. Experimental results show that the by-pass features adds insignificant overhead compared to calling the source JDBC driver directly. Thus, client code only needs to load and use the UnityJDBC driver directly. This results in more portable code that can be more easily moved between database systems.

When UnityJDBC parses the SQL, you can use table and field references that are prefixed with the database name. This is optional if the table and field names are unique across all databases, otherwise the database name is required.

The database name is assigned in the schema file describing the source and does not have to be the same as the system name used by the database system itself. That is, the name can be set by the developer using UnityJDBC.

Multiple source UnityJDBC queries can be used with an INSERT INTO statement to populate a table in the database. This allows a user to write a cross-database query to collect information from multiple sources and then insert the result back into a table in any one of the sources. Currently, the only restriction is that the table that will be inserted into must exist and must be present in the schema file describing the source.

Using ExampleMetadata.java

ExampleMetadata.java demonstrates UnityJDBC's support for the DatabaseMetaData interface. This interface functions exactly according to the standard with the major difference that metadata is returned for all databases in the data virtualization rather than from a single database. That is, all your "integrated" databases really do appear as a single database to your application.

Using ExampleEngine.java

Embedded in the UnityJDBC driver is a complete relational engine. This is required to process cross-database join queries. Most users will not interact with the engine directly, and their only contact with the engine may be to increase the JVM heap sizes for processing large cross-database queries. However, all of the relational operators of selection, projection, and join are available for direct use in your programs. The join algorithms support sources larger than main memory, and allow you the full power of combining ResultSets from multiple databases. It is also possible to explicitly track global query progress on a per operator basis or perform your own optimization of queries after the UnityJDBC optimizer has built an execution tree.

Using ExampleNoFileConnection.java

ExampleNoFileConnection.java demonstrates UnityJDBC's ability to be dynamically configured at run-time including adding, removing, or updating sources. It is possible to dynamically build a virtual database without using XML configuration files by interacting with UnityJDBC through its metadata API.

Chapter 5. Using UnityJDBC with Query and Reporting Software

The UnityJDBC SourceBuilder is a simple query utility for cross-database joins and data virtualization. In most cases, you will use UnityJDBC to perform data virtualization with reporting software such as JasperReports and Splunk or query software such as SQuirreL SQL or Aqua Data Studio. Once you have used SourceBuilder to build the virtual source and schema XML configuration files, you use UnityJDBC like any other JDBC driver. This section contains examples on how to install and use UnityJDBC data virtualization in popular software systems.

UnityJDBC Data Virtualization with SQuirreL SQL

UnityJDBC is integrated into SQuirreL SQL allowing users to build SQL queries that join data from multiple sources directly within SQuirreL. The multiple source query plugin allows SQuirreL users to create a virtual data source that may consist of multiple data sources on different servers and platforms. The user can enter one SQL query to combine and join information from multiple sources.

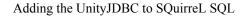
Benefits:

- The multisource plugin powered by UnityJDBC allows SQuirreL SQL to support multiple source queries.
- No data source or server changes are required.
- The plugin supports standard SQL including joins, group by, aggregation, LIMIT, and ordering where tables may come from one or more sources.
- The plugin will perform function translation where a user requests a function that is not supported on a certain source.

UnityJDBC can be installed directly as a plug-in through the SQuirreL SQL plug-in interface. It is also possible to download the UnityJDBC distribution and replace the unityjdbc.jar in the plug-in with the latest from UnityJDBC.

1. Registering the UnityJDBC Driver - By adding a driver.

Change Driver: UnityJDBC
Change Driver: UnityJDBC
Driver
Name: UnityJDBC
Example URL: jdbc:unity://virtual
Website URL: www.unityjdbc.com
Java Class Path Extra Class Path
C:\Program Files (x86)\squirrel-sql-3.2.1\squirrel-sql.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\antlr.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\aopalliance.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\asm-attrs.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\asm.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\asm.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\axis-jaxrpc.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\axis-jaxrpc.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\axis-saaj.jar C:\Program Files (x86)\squirrel-sql-3.2.1\lib\axis-wsdl4j.jar
Class Name: unity.jdbc.UnityDriver
OK Close



2. **Registering your Data Sources** - Register your data sources as usual. In this example, we will perform data virtualization of multiple databases on Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. Any database with a JDBC driver is supported including those accessible using the JDBC-ODBC bridge.

🥭 Change Alias: MS	Change Alias: MSSQL									
Change Alias: MSSQL										
Name:	MSSQL									
Driver:	✓ Microsoft MSSQL Server JDBC Driver ▼ New									
URL:	dbc:sqlserver://cssql.ok.ubc.ca:1433;databaseName=tpch									
User Name:	sa									
Password:	•••••									
Auto logon	Connect at Startup									
	Properties									
Warning - Pass	words are saved in clear text									
	OK Close Test									

Registering a Microsoft SQL Server Source in SQuirreL SQL

Change Alias: mysql_tpch										
Change Alias: mysql_tpch										
Name:	mysql									
Driver:	✓ MySQL Driver ▼ New									
URL:	jdbc:mysql://cs-suse-4.ok.ubc.ca/tpch									
User Name:	rlawrenc									
Password:	••••									
🗌 Auto logon	Connect at Startup									
	Properties									
Warning - Pass	words are saved in clear text									
		_								
	OK Close Test									

Registering a MySQL Source in SQuirreL SQL

🍼 Change Alias: ora	cle 💌										
Change Alias: or	Change Alias: oracle										
Name:	oracle										
Driver:	✓ Oracle Thin Driver ▼ New										
URL:	jdbc:oracle:thin:system/password@cssql.ok.ubc.ca:1521/x										
User Name:	system										
Password:	•••••										
Auto logon	Connect at Startup										
	Properties										
Warning - Pass	words are saved in clear text										
	OK Close Test										

Registering an Oracle Source in SQuirreL SQL

👏 Change Alias: po:	Change Alias: postgres									
Change Alias: postgres										
Name:	postgres									
Driver:	✓ PostgreSQL ▼ New									
URL:	jdbc:postgresql://db.ok.ubc.ca/tpch									
User Name:	rlawrenc									
Password:	••••									
Auto logon	Connect at Startup									
	🖹 Properties									
Warning - Pass	words are saved in clear text									
	OK Close Test									

Registering a PostgreSQL Source in SQuirreL SQL

3. Create a Multiple Database Alias - Make sure you have registered the UnityJDBC driver (during installation). Create an alias consisting of virtual sources. The name field can be any name. It does not have to be virtual. If you are using the virtualization embedded into the plugin, the URL is jdbc:unity://virtual. If you have previously created a data virtualization using the SourceBuilder utility, then the URL should be the file location of the sources file previously created.

🥭 Add Alias	
Add Alias	
Name:	virtual
Driver:	✓ UnityJDBC
URL:	jdbc:unity://virtual
User Name:	
Password:	
Auto logon	Connect at Startup
	🕆 Properties
Warning - Pass	words are saved in clear text
	OK Close Test

Add a Data Virtualization Alias for Multiple Sources in SQuirreL SQL

4. Add Microsoft SQL Server to Data Virtualization - Right-click on the root object in the object tree, and select (Virtualization) Add Source. User selects the Microsoft SQL Server source to add to the data virtualization.

🥭 sQi	uirreL SQL Client Snapshot-20120501_0909												
<u>F</u> ile	Drivers <u>A</u> liases <u>P</u> lugins <u>S</u> ession	<u>W</u> indows <u>H</u> elp											
Con	nect to: MSSQL 🔻 🙈 😹 A	ctive Session: 4 - vir	tual	•	1	× 📬 🛱							
	× 🗍 4 - virtual												
Aliases													
Alia			T + 1	3									
	Objects SOL												
	Q 💎 🕻 🗖	Metadata Status	Schemas	Table Types	Data Ty	pes Numeric Func	tions	String Functions	System Fund	tions	Time/Date Functions	Keywords	
Drivers	♀		Property Nam					Value					
Ĩ,	P 🔠 💞 Refresh Tree and Cache	e /er CLASSNAM		•		unity.jdbc.UnityDriver		value					
		ver CLASSPAT				C:\eclipse\workspace	\UnitvJE	DBC Live\unitvidbc.i	ar				
	💎 Edit Objects Filter					<null></null>			-				
	(Virtualization) Add sou	rce ily				false							
	Copy Object (Virtualizati	ProductNa	me			UnityJDBC							
	Copy Qualified Object N												
			sion			4.1.109							
		tDriverName ProceduresAreCallab	1			UnityJDBC							
		TablesAreSelectable	le			false true							
		toCommitFailureClos	as All ResultSe	ite		false							
		taDefinitionCausesTr				false							
		taDefinitionIgnoredIn				false							
		esMaxRowSizeInclud				false							
	ge	tCatalogSeparator											
	ge	tCatalogTerm				catalog							
		tClientInfoProperties											
		tDatabaseMajorVersi				4							
		tDatabaseMinorVersi				1							
		tDefaultTransactionIs	olation			TRANSACTION_NON	E						
		tDriverMajorVersion tDriverMinorVersion				4							
		tDriverVersion				4.1.109							
		tExtraNameCharacter	·e			4.1.109							
		tidentifierQuoteString	3			-							-
L D	/virtual											1.20/3	20
	, en tual											, ,	
	tate: null												
	Code: 0												
Please	e try out the Tools popup by hitting ctrl+t in	n the SQL Editor. Do it	three times to	stop this messa	age.								-
						Logs: Erre	ors 176,	Warnings 0, Infos 4,12	8	76 of 1	54 MB	8:09:14 AI	M PDT

Adding Microsoft SQL Server Source to the Data Virtualization in SQuirreL SQL

🍰 Select an Alias to add	d to the Virtualization									
Select source to add to	o the virtualization: mssql 💌 Exit									
Name:	mssql									
Catalog:										
Schema:										
Tables included:	%									
Tables excluded:	_*\\$.*									
Statistics:	Row Counts 💌									
	Add									
Progress: 8 of 8 Su	ccess.									
Messages										
Computing table statis	stics									
SELECT COUNT(*) FROM tpch100.dbo.SUF	PPLIER									
Done processing table:										
Success.										

	QuirreL SQL Client Snapshot-20120510_09										
<u>F</u> ile	Drivers Aliases Plugins Session	<u>W</u> indows <u>H</u> elp									
C	Connect to: cssql_unity 🔻 🧀 Active Session: 1 - virtual 🔍 🗞 🕟 🛛 📬										
s.	g 1 - virtual										
liase											
٩	2 Dipiects SQL										
	🔍 🚺 Metadata Status Schemas Table Types Data Types Numeric Functions String Functions System Functions Time/Date Functions Keywords										
Drivers	P ■ virtual	Property Nam	e			Value					
E E	P 🗄 MSSQL	JDBC Driver CLASSNAME		unity	.jdbc.UnityDriver						
-	P III TABLE	JDBC Driver CLASSPATH		C:\ee	clipse\workspace\UnityJ	DBC_Live\unityjdbc.j	ar				
	CUSTOMER (1500	getURL		<nul< th=""><th>></th><th></th><th></th><th></th><th></th></nul<>	>						
		isReadOnly		false							
		getDatabaseProductName		Unity	JDBC						
		getUserName									
		getDriverVersion	4.1.1								
		getDatabaseProductVersion			4.1.109						
					UnityJDBC						
					false						
		autoCommitFailureClosesAllResultS	to	true faise	•						
		dataDefinitionCausesTransactionCor		false							
		dataDefinitionIgnoredInTransactionCol	mm	false							
		doesMaxRowSizeIncludeBlobs		false							
		getCatalogSeparator			, 						
		getCatalogTerm		catal	og						
		getClientInfoProperties									
		getDatabaseMajorVersion		4							
					1						
		getDefaultTransactionIsolation			NSACTION_NONE						
		getDriverMajorVersion		4							
		getDriverMinorVersion		1							
		getExtraNameCharacters							-		
		aetIdentifierQuoteStrina									
	/virtual								1,1/1		
Ple	ase try out the Tools popup by hitting ctrl+	t in the SQL Editor. Do it three times to	stop this messa	ge.							
-								of 121 MB 0			
1					Logs: Error	s 3, Warnings 0, Infos 5	2 📮 41	of 121 MB 0	2:30:46 PM PDT		

Resulting Data Virtualization

5. Add Oracle Database to Data Virtualization - User can add as many sources as they wish. You can also rename the source in the virtual view. It does not have to be the same as the alias name used by SQuirreL. When adding Oracle sources, make sure to specify a schema so that system tables and tables from all schemas are not extracted.

d to the Virtualization										
Select source to add to the virtualization: oracle										
Name: oracle										
Catalog:										
RLAWRENC										
%										
ables excluded: *\\$.*										
Row Counts 💌										
Add										
ccess.										
stics										
PPLIER										
SUPPLIER										

Prompt to add Oracle Source to the Data Virtualization (Note use of SCHEMA).

SQuirreL SQL Client Snapshot-20120510_09 <u>Eile Drivers Aliases Plugins Session</u>	<u>W</u> indows <u>H</u> elp									
Connect to: cssqLunity 🔻 🦓 🖉 Active Session: 1 - virtual 🔍 😚 🕞 🛇 📑 🗄										
serection of the second secon	I R & X	† + =								
	Metadata Status	Schemas	Table Types	Data Ty	Numeric Functions	String Functions	System Functio	ns Time/Date Functions	Keywords	
		Property Nam	e			Value				
E P MSSQL	JDBC Driver CLASSNAM	E			unity.jdbc.UnityDriver					
	JDBC Driver CLASSPATH	4			C:\eclipse\workspace\UnityJ	IDBC_Live\unityjdbc.j	ar			
	getURL				<null></null>					
	sReadOnly				false					
	getDatabaseProductNar	ne			UnityJDBC					
	getUserName									
	getDriverVersion				4.1.109					
	getDatabaseProductVersion getDriverName				4.1.109 UnityJDBC					
	allProceduresAreCallable				false					
🔍 🔍 🗠 🚫 UDT	allTablesAreSelectable				true					
🗢 📇 mysql 📰 🔛	autoCommitFailureClos		te		false					
🗣 🖾 oracle	dataDefinitionCausesTra				false					
🗢 🥅 TABLE	dataDefinitionIonoredInTransactions				alse					
🔍 🔍 🗠 🚫 UDT	doesMaxRowSizeInclude				false					
🗣 🛅 postgres	getCatalogSeparator									
P 🛗 TABLE	getCatalogTerm				catalog					
	getClientInfoProperties									
	getDatabaseMajorVersio				4					
	getDatabaseMinorVersio				1					
- ALTER AND ALTE	getDefaultTransactionIs	plation			TRANSACTION_NONE					
	getDriverMajorVersion				4					
	getDriverMinorVersion				1					
	getExtraNameCharacter: petIdentifierQuoteString	8							-	
	rendenimerQUotestrind									
/virtual									1,1/1	
Please try out the Tools popup by hitting ctrl+l	t in the SQL Editor. Do it	three times to	stop this mess	age.						
					Logs: Errors	3, Warnings 0, Infos 20	3 📮	28 of 122 MB 0	2:33:51 PM PDT	

A Data Virtualization in SQuirreL SQL with Databases MySQL, Oracle, PostgreSQL, and Microsoft SQL Server

6. Execute a Multiple Database Query - The user can execute an SQL query that spans multiple sources and get a single result. The virtualization is transparent to the user and SQuirreL. Below is an example of a query that joins two tables in different databases.

			La 4			
× 🗍 4 - vir	tual					
a 🌼 🗦	* 🔅 🗅 🕒 🖡] 🖳 🗁 🗶 🗘 🕂 🖶				
Ohiasta	201					
	-					
select * from	MSSQL.Region R INNER JOIN	mysql.Nation N ON R.r_regionkey = N.n_regionkey				▼ ↓ 🗓 🗹 Limit rows: 100
select * fro	mM					
	1	ion R INNER JOIN mysql.Nation N ON R.r_regionkey = 1	N.n_regionke	N_NAME N_REGIO N_COMMENT ALGERIA 0 2Cxh17 L 11wk6hMh300irngN32CPwClkyLk6khM → ARGENTINA 1 2Cn30kwz1wLn7FL330hCgn6k/r5PyRikg1B7 BRAZIL 1 gLm50nACAmnBC)2kli/R7RCPNgPxnC0yLk6khM → CANADA 1 4JM0 AhnO5Lh wo2Ml652kw1B7 EGYPT 4 11im5126 Cxj.NMCmLx0kni02[2m3An4/WR10 EGYPT 4 11im5126 Cxj.NMCmLx0kni02[2m3An4/WR10 FFHOPIA 0 NS7n.LS0P Ozin14B25320n1MH458b7 inthB FRANCE 3 3mjmuli 9.3.32thNNINIP4W370r8ynH15wn GERMANY 3 2 n04RknwC m2BB 516mAg IBywd0M30/HP INDIA 2 MNIRSRC/RMj111wjN7Mm H1MjMIMmB017PL INDIA 2 S)Fm0071Lj7AB/MA(Al31Lur/13BPzZcwjjMr IRAN 4 h532433Bg/ND005095B2h06x0301w4mmP IRAQ 4 N gWR604Rny05005095B2h06x0301w4mmP IRAQ 4 N gWR604Rny05005095B2h06x0301w4mmP JPAN 2 Y PRn604Rny0F0000m5B1x04x04mJmL102101000		
Results	MetaData Info Overvie	N				
R_REGIO	R_NAME	R COMMENT	N NATION	N NAME	N DECIO	NL CONVENT
0	AFRICA	special Tiresias about the furiously even dolphins are furi	0	ALGERIA	0	2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6l
0	AMERICA	even, ironic theodolites according to the bold platelets wa	0	ARGENTINA	0	2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6l zQn30kwz1wLn7PLS30hCgn56kP5PyRikgi
0 1 1 1	AMERICA AMERICA	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa	0 1 2	ARGENTINA BRAZIL	0 1 1	2CxhI7 L1iwk6hMh300izngN32CPwCikyLk6i zQn3Okwz1wLn7PLS3OhCgn56kP5PyRikgi gLmS0nACAmnBCj2klki7RCPNgPxnC0jNg
1 1 1	AMERICA AMERICA AMERICA	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa	0 1 2 3	ARGENTINA BRAZIL CANADA	0 1 1 1	2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6i zQn3Okwz1wLn7PLS3OhCgn56kP5PyRikgi gLmS0nACAmnBCj2klki7RCPNgPxnC0jNg 4yMO AhnQ5Lh wzQAM662Aw1ByCI7CxmzF
1 1 1 4	AMERICA AMERICA AMERICA MIDDLE EAST	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual,	0 1 2 3 4	ARGENTINA BRAZIL CANADA EGYPT	0 1 1 1 1 4	2Cxhl7 L1lwk6hMh300lzngN32CPwClkyLk6l zQn30kwz1wLn7PLS30hCgn56kP5PyRikgi gLmS0nACAmnBCj2klki7RCPNgPxnCOjNg 4yM0 AhnQ5Lh wzQAM662Aw19yC17Cxmz2 11lm5126 Cxj NMQmLx0ikni02j2m3Ah4yNR
1 1 1 4 0	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tiresias about the furiously even dophins are furi	0 1 2 3 4 5	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA	0 1 1 1 4 0	2Cxhl7 L1iwk6hlh300izngN32CPwCikyLk6 2Cn30lwz1wLn7PL330hCgn56k655ryRicg JLmS0nACAmBCj2kliN7CPNgPwnC0jNg 4yM0 Ahn05Lh wc0All662Aw18pC/TCxm25 11im5126 Cxj NMCmLx0ikn02j2m3Ah4yNR NS7n LSOP 025n1AlB2S02nN0IMH45BxP if NS7n LSOP 025n1AlB2S02nN0IMH45BxP if
1 1 1 4 0 3	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA EUROPE	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tiresias about the furiously even dolphins are furi special, bold deposits haggle foxes, platelet	0 1 2 3 4 5 6	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA FRANCE	0 1 1 1 4 0 3	2Cxh17 L1iwk6hMh300izngN32CPwCikyLk6i 2Gn30kwz1wLn7PL330hCgn56kr5PyRikgi gLm30nACAmbG)2klk17RCPNgPmCO[Ng 4)M0 AhnG5Lh wz0Al662Aw1ByC17Cxm2 11im5126 CyLNMCmLx0ikn102/2m3Ah4yNR NS7n LSOP 0z5n AlB2S02nN0IMh45BxP iF 3mjmizi 3 SJ.SI2hNNNNP4w370xRxN15w
1 1 1 4 0 3 3	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA EUROPE EUROPE	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tiresias about the furiously even dophins are furi special, bold deposits haggle foxes, platelet special, bold deposits haggle foxes. platelet	0 1 2 3 4 5 6 7	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA FRANCE GERMANY	0 1 1 4 0 3 3	2Chi17 L1iwk6hM300izngN32CPwClikyLk8 zGn30kwz1wLn7PL330hCgn56kr56p,Rikgi gLm30nAChmBCj2klia7RCPMp2mC010g 4jM0 AhnQ5Lh wz0Al662Aw1ByC17CxmzF 11im5126 GUNMCmL20ina02g2m3AhyMr NS7n LSOP Oz5n1AlB2S02nN0IM4SBxP iF 3mjmLi S 3L3/zhNNhIP4w370xRyN15wr z n0P4Rkw0 Cm2BB 516mág IByw40M30y
1 1 4 0 3 3 2	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA EUROPE EUROPE ASIA	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tiresias about the furiously even dolphins are furi special, bold deposits haggle foxes, platelet special, bold reguests haggle foxes, platelet silent, bold reguests haggle foxes platelet	0 1 2 3 4 5 6 7 8	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA FRANCE GERMANY IINDIA	0 1 1 1 4 0 3 3 2	2Cxh17 L11wk6hM300izngN32CPwCikyLk6i zGn30kwz1wLn7PL330hCgn5kkr5PyRkgi JLm30nACamBCj2klia7RCPNgPmCOl0y 4yMO Ahn05Lh wz0All662Aw1ByCl7Cxm2F I1im5126 Cxj NMGMLc0im02j2m3Ah4yR MS7h LS0P C25n1Al82S02n01MiM458kr2 3mjmiz S 3L3Ch1NhNIP4w370rRyh15wr z n0P4Rkw0 Cm2BB 516mAg Byw40M30y MNIRSRCIRMJ1111wjN7Mm M11yNIMBE017
1 1 4 0 3 3 2	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA EUROPE EUROPE ASIA ASIA ASIA	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tiresias about the furiously even dolphins are furi special, bold deposits haggle foxes, platelet special, bold deposits haggle foxes, platelet silent, bold requests sleep slyb across the quickly sly depend.	0 1 2 3 4 5 6 7 8 9	ARGENTINA BRAZIL CANADÀ EGYPT ETHIOPIA FRANCE GERMANY INDIA INDONESIA	0 1 1 4 0 3 3 2 2 2	2Cxh17L1iwk6hM1300cngN32CFw0ipLk8 z0n30/wz1wLn7PL330hCgn56hv6jP,Kikgi gLm30nACmmB02kik17RCPNgPmCO1g 4JM0 Ann05Lh wz0AM662Aw1ByC17CxmzF 11m5126 0zyNM0mL20ind02gm3An4yhv NS7n LSOP 0z5n14B2S02nN0IMI458xP iF 3mjmizi 3 L3L2iNNNhIIP4w370rRxyh15wr zn0P4Rkw0 CrnzB6 516mag (Byw40Ild30/ MNIRSRCIRMj111 wJN7Mn H1ylNIIMm017 5JPm00711,74Bj6MA402LmV37B9rzc00/ H30187C1RMj111 wJN7Mn H1ylNIIMm017
2 1 1 4 0 3 3 2 2	AMERICA AMERICA AMERICA MIDDLE EAST AFRICA EUROPE EUROPE EUROPE ASIA ASIA MIDDLE EAST	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual, special Tresias about the furiously even dolphins are furi special, bold deposits haggle foxes, platelet special, bold deposits haggle foxes, platelet silent, bold requests sleep slyly across the quickly sly depend. silent, bold requests sleep slyly across the quickly sly depend. silent, bold requests use carefully above the unusual,	0 1 2 3 4 5 6 7 8 9 9	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA FRANCE GERMANY INDIA INDONESIA IRAN	0 1 1 4 0 3 3 2 2 2 4	2Ch17 L11wk6hM300izngN32CPwClikyLk6i 2Gn30kwz1wLn7PL330hCgn5kir56P,Rkgi gLm30nACamBCj2klir7RCPNgPmC00j4 4jM0 Ahn05Lh wc0Al6624w1BycTCXmzF 11im5126 Cxj NMGmLx0lini02j2m3Ah4yNR NS7n LS0P 025n1Al82502nN0IMA588zi 3mjmi3 S 3L3/zhNNhNIP4w370xRyN15wr z n0P4Rkw0 Cm2B 516m8 [Byw40130y MNIR5RCIRMj1111wjN7Myn M1bjNIMB007 SjPm0071Lj7A8J6MAACI3nLw738PrzCwj h532q438gN05000g582h050kn3014W1
2 1 1 4 0 3 3 2 2 2 4	AMERICA AMERICA AMERICA AMERICA EUROPE EUROPE EUROPE ASIA ASIA MIDDLE EAST MIDDLE EAST	even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa even, ironic theodolites according to the bold platelets wa furiously unusual packages use carefully above the unusual special Tiresias about the furiously even dolphins are furi special, bold deposits haggle foxes, platelet special, bold deposits haggle foxes, platelet silent, bold requests leep slyly across the quickly sly depend. silent, bold requests sleep slyla across the quickly sly depend. furiously unusual packages use carefully above the unusual furiously unusual packages use carefully above the unusual.	0 1 2 3 4 5 6 7 8 9 10 11	ARGENTINA BRAZIL CANADA EGYPT ETHIOPIA FRANCE GERUANY INDIA INDONESIA IRAN IRAN	0 1 1 1 4 0 3 3 2 2 4 4 4	2Coh17 L1wk6hM1300cngN32CPw0irpL68 2Gh30kw21wLn7PL330hGgh56kP5P;Rikgi GLm50nACAmBCj2kin7RCPNgPmCOh2 4jkl0 Ahn05Lh w20Al662Aw1ByC17Cxm2 11im5126 CJNMCmL20ina012gm3Ah4JW NS7n LSOP 025n1Al82S02nN0IMA5BxP iF 3mmiat 33.23kN1NNIPMP40730r6Xyn15w zn0P4Rkw0 Cm2BB 516mdg IByw040M30y MNIR5RCIRMJ1111wjN7Mym M1yNIMB017 519rm0071 Lj7A8jMk1AcAnLw173PrzCM h52g43BgSh0560.0g82h06jw07g27/iPHI5Rx8
1 1 1 4 0 3 3 2 2 2 4 4 4 2 4	Next Busins Session Windows Heip Vertice Missee Pusins Session Vertice Missee Vertice Note Vertice Missee Vertice Note <					
Physics Pugins Session Wintow Pugins Next MSSQL Pugins A virtual Pugins Pugins Pugin						
Control						

A Multiple Database Query with Join Results Expressed on Previous Data Virtualization

7. **Perform SQL Query Translation** - The UnityJDBC driver used to perform the virtualization will also translate functions that are not implemented by certain sources. For example, Microsoft SQL Server does not support TRIM(), but you can do the same result using RTRIM(LTRIM()). Unity will automatically translate a TRIM() function specified in a MSSQL query to the correct syntax supported by the database.

Ø s	QuirreL SQL Client Version 3.3.0		
<u>F</u> ile	Drivers Aliases Plugins Session Windows Help		
C	onnect to: cssql_unity 🔻 🥰 Active Session: 1 - test 🔍 📢 🕞 🙁 📑 🗄		
	× 1 1-test		
Aliases	₽¢ ★ ⇒ C⊳⊾∎∎≙× ♀↓∃ = < @@@ ₽00		
	Objects SQL Hibernate		
	select trim("+n_name+") as trimmedName from MSSQL Nation	▼ ↓ 🞚 🗹 Limit rows:	100
Drivers	select trim(' '+n_name+' ') as trimmedName from MSSQL.Nation		<u>^</u>
ā			
			-
	A.7		
	select trim(c_n select trim(' A select trim('		
	Rows 25; select trim(' '+n_name+' ') as trimmedName from MSSQL.Nation		≭⊠⊠
	Results MetaData Info Overview		
	trimmedNa		
	ALGERIA		
	ARGENTINA BRAZIL		
	Err&zL CANADA		
	EGYPT		=
	ETHIOPIA		
	FRANCE GERMANY		
	INDONESIA		
	IRAN		
	JAPAN JAPAN		
	JORDAN		
	KENYA		
	/rest/NSSQL		1,38 / 38
A.T.	SQL selected for execution.		
	Sub Senetation execution: wy 1 of 1, Rows read: 25, Elapsed time (seconds) - Total: 0.026, SQL query: 0.006, Building output: 0.02		
Que	ny 1 of 1, Rows read; 25, Elapsed time (seconds) - Total: 0.022, SQL query: 0.005, Building output: 0.017		-
	Logs: Errors 4, Warnings 0, Infos 20 📃 💂	0 of 127 MB 0 🧊	3:48:35 PM PDT

An example of SQL Query and Dialect Translation - Converting TRIM() function for Microsoft SQL Server

The plugin source code, like all of SQuirreL, is released under the GNU Lesser General Public License. The UnityJDBC virtualization driver is released under a commercial license. However, the UnityJDBC driver included in the plugin is fully functioning with no time limits allowing an unlimited number of sources and queries. The only limitation is the size of the result set is limited to the first 100 rows. (Note there is no limit on the number of rows extracted from each source. So select count(*) from table with a 1 million row table is fine as it only returns one result row.) Use LIMIT 100 to get the first 100 results of a query. A full version of the UnityJDBC driver can be purchased as www.unityjdbc.com [http://www.unityjdbc.com].

UnityJDBC Data Virtualization with JasperReports

The JasperReports library and community version of JasperReports server does not support data virtualization allowing one SQL query to extract data from multiple databases. UnityJDBC can be used with JasperReports to enable this data virtualization which simplifies the construction of reports, especially reports that would usually use subreports.

UnityJDBC Data Virtualization with Aqua Data Studio

Aqua Data Studio can query multiple databases with different SQL queries. However, you cannot query more than one database at the same time in one SQL query. UnityJDBC allows a user to write one query to join, aggregate, and summarize data across any number of databases. It also performs translation of SQL dialects and functions.

Benefits:

• UnityJDBC allows Aqua Data Studio to support multiple source queries.

- No data source or server changes are required.
- UnityJDBC supports standard SQL including joins, group by, aggregation, LIMIT, and ordering where tables may come from one or more sources.
- UnityJDBC will perform function translation where a user requests a function that is not supported on a certain source.

This example assumes that the installation of UnityJDBC has already been completed. The following is an example of creating sources and executing multiple database queries with Aqua Data Studio.

1. Create a New Source Group - Under File Menu, Select New Source Group.

UnityJDBC Source Bu	uilder and Query Appl	lication		
File Demonstration	Help			
New Source Group Load Source Group New Source Remove Source Remove Table Refresh Exit	Virtualized)	Query SELECT L.I_extendedprice, P.p. FROM OrderDB.Lineitem L, Par WHERE P.p_partkey = L.I_partk	tDB.Part P	
PART CUSTOMER SUPPLIER]	Messages		
NATION OrderDB REGION PARTSUPP ORDERS TestTime LINEITEM PART		Executing query Query result 1 at time: 168 Query result 10 at time: 171 Query result 100 at time: 172 ERROR: java.sql. SQLExceptio version at: www.unityjdbc.com Number of results: 100		in the trial version. Full
CUSTOMER		Results L_EXTENDEDPRICE 45050.00 29733.00 29733.00 30634.00 29733.00 2129733.00 29733.00 29733.00 29733.00 29733.00 29733.00 29733.00 29733.00 29733.00 29733.00 24327.00	P_RETAILPRICE 901.00 901.00 901.00 901.00 901.00 901.00 901.00 901.00	L_PARTKEY

Aqua Example: Creating a New Source Group Data Virtualization

2. Select Sources File Name and Location - Use the default sources.xml or select a file location.

U Setup Sour	rce Group Configuration File	×
	Enter Location for Source Group XML Configuration File	
File name:	name: sources.xml	
	OK	

Aqua Example: Specifying a Source File Location

3. Select New Source - Under File Menu, Select New Source.

UnityJDBC Source B	uilder and Query Apj	plication 🗖 🗖	x
File Demonstration	Help		
New Source Group Load Source Group New Source Remove Source	Virtualized)	Query	_
Remove Table Refresh			
Exit		EXECUTE Clear	
		Messages	
		Welcome! To get started either: 1. Create your own sources by: a. File Menu->New Source Group and create new XML config file. b. File Menu->New Source and enter the JDBC connection information for each source. Results	

Aqua Example: Creating a New Source

4. Add a Microsoft Database - Enter connection information for Microsoft SQL Server to add it to the data virtualization.

Sources file: sources.xml					
Source Type	Name :	db1_sqlserver			
Generic JDBC	URL:	jdbc:sqlserver://cssql.ok.ubc.ca;DatabaseName=tpch100;responseBuffering=adaptive			
Generic ODBC HSQLDB	User:	rlawrenc			
IBM DB2	Password:				
Microsoft SQL Serv MySQL	er Schema:				
Oracle	Driver:	com.microsoft.sqlserver.jdbc.SQLServerDriver			
PostgreSQL	Schema file:	db1_sqlserver.xml			
Sybase		Add Source			
Preparing to add da Registering driver: c		rver.jdbc.SQLServerDriver	4		

Aqua Example: Adding a Microsoft SQL Server Database to the Data Virtualization

5. Add a MySQL Database - Enter connection information for MySQL database to add it to the data virtualization.

Sources file: sour	ces.xml	Exit	
Source Type	Name :	db2_mysql	
Generic JDBC	URL:	jdbc:mysql://cs-suse-4.ok.ubc.ca/tpch	
Generic ODBC HSQLDB	User:	rlawrenc	
IBM DB2	Password:	test	
Microsoft SQL Serve	er Schema:		
MySQL			
Oracle	Driver:	com.mysql.jdbc.Driver	
PostgreSQL	Schema file:	db2_mysql.xml	
Sybase		Add Source	
_	a source		

Aqua Example: Adding a MySQL Database to the Data Virtualization

6. Viewing Data Virtualization of Sources - Select Exit to return to the main screen and see data virtualization of MySQL and Microsoft SQL Server sources.

UnityJDBC Source Builder and Query App File Demonstration Help	olication
Databases (Virtualized) Carbon Supple Customer Custom	Query EXECUTE Clear Messages Welcome! To get started either: 1. Create your own sources by: a. File Menu->New Source Group and create new XML config file. b. File Menu->New Source and enter the JDBC connection information for each source. Results

Aqua Example: Resulting Data Virtualization of Two Database Sources

7. Multiple Database Query and Cross-Database Join Example - Type in a cross-database query and execute it to view results.

UnityJDBC Source Builder and Query Ap	plication	×
File Demonstration Help		
8	Query	
Databases (Virtualized)	select count(*) from db1_sqlserver.Orders O1 INNER JOIN db2_mysql.Orders O2 ON O1.o_orderkey = O2.o_orderkey EXECUTE Clear Messages Executing query Query result 1 at time: 134 Number of results: 1 Query complete. Query Exection time is 135 ms.	
► III PART ► III CUSTOMER	Results	
	MatchFunc1	
	· · · · · · · · · · · · · · · · · · ·	15000

Aqua Example: Executing a Cross-Database Join of Two Databases

8. **Finding Sources and Schema Files** - Now that the data virtualization is complete, find the sources.xml file and schema files for your sources. The image below shows the default location which would be inside the UnityJDBC installation directory.

Organize 🔻 🛛 🔀 Oper	Burn	New folder				0
Name		Date modified	Туре	Size		
퉬 code		7/3/2012 8:57 PM	File folder			
🐌 OtherJDBCDrivers		7/3/2012 8:57 PM	File folder			
퉬 sampleDB		7/3/2012 8:57 PM	File folder			
퉬 Uninstaller		7/3/2012 8:57 PM	File folder			
퉬 webserver		7/5/2012 12:32 PM	File folder			
📄 .installationinformatio	n	7/10/2012 10:47 PM	INSTALLATIONIN	7 KB		
📄 db1_sqlserver.xml		7/10/2012 11:02 PM	XML File	50 KB		
📄 db2_mysql.xml		7/10/2012 11:01 PM	XML File	46 KB		
📄 def.xml		7/10/2012 9:47 PM	XML File	50 KB		
🍇 initsources.bat		7/1/2012 11:39 PM	Windows Batch File	1 KB		
🍇 initsources.sh		7/4/2012 3:57 PM	Shell Script	1 KB		
📄 README.txt		6/7/2012 7:58 PM	TXT File	7 KB		
📄 sources.xml		7/10/2012 11:02 PM	XML File	1 KB		
🕖 unity.ico		6/29/2012 8:13 AM	Icon	4 KB		
📧 unityjdbc.jar		7/10/2012 10:48 PM	Executable Jar File	920 KB		
🔁 UnityJDBCDoc.pdf		9/30/2011 2:38 PM	Adobe Acrobat D	211 KB		

Aqua Example: Finding Source and Schema Files

9. Moving Sources and Schema Files - Optionally, move the sources and schema files to a permanent location. In this example, they are moved to C:/unityjdbc.

🕽 🔵 🗢 📙 🕨 Con	nputer 🕨 Local Di	sk (C:) ト unityjdł)C		▼ 47	Search u	mityjdbc		۲
Organize 🔻 🛛 Inclu	ıde in library 🔻	Share with 🔻	Burn	New folder					0
Name	*	Date mod	lified	Туре	Size				
📄 db1_sqlserver.xml		7/10/2012	2 11:02 PM	XML File		50 KB			
📄 db2_mysql.xml		7/10/2012	2 11:01 PM	XML File		46 KB			
📄 sources.xml		7/10/2012	2 11:02 PM	XML File		1 KB			

Aqua Example: Moving Source and Schema Files to Another Location

10.Adding UnityJDBC data source to Aqua Data Studio - Copy unityjdbc.jar and all JDBC drivers for databases into jre/lib/ext in Aqua Installation directory.

					_	4
Organize 🔻 🏼 🔮 Open 🛛 Burn	New folder	1	1	• ==	•	
Name	Date modified	Туре	Size			
📧 dnsns.jar	7/10/2012 11:58 PM	Executable Jar File	9 KB			
📧 localedata.jar	7/10/2012 11:58 PM	Executable Jar File	825 KB			
📄 meta-index	7/10/2012 11:58 PM	File	1 KB			
📧 mysql.jar	7/2/2012 3:07 PM	Executable Jar File	728 KB			
📧 sqljdbc.jar	7/2/2012 9:55 PM	Executable Jar File	570 KB			
📧 sunjce_provider.jar	7/10/2012 11:58 PM	Executable Jar File	167 KB			
📧 unityjdbc.jar	7/10/2012 9:42 PM	Executable Jar File	920 KB			

Aqua Example: Installing unityjdbc.jar in Aqua Data Studio

11.Start Aqua and Register a Generic JDBC Source - The configuration information is:

- Driver: unity.jdbc.UnityDriver
- URL: jdbc:unity://<path to sources.xml file>

🕤 General 🛛 🐺 Filter 🛛 🐺 Advanced		ions 🛛 🕼 Script	
Oracle 8i 🔺	Name:	unity_multiple	
Oracle 8i - OCI	Type:	Production	-
Oracle 9i/10g/11g Oracle 9i/10g/11g - OCI			•
MS SQL Server 2000/5/8/12	Tab Color:		
MS SQL Server 2000/5/8/12 - ODBC	Tab Title Format:	{0}@{1}	plain
MS SQL Server 7.0			
MS SQL Server 7.0 - ODBC	- Authentication		
Sybase ASE 11/12/15 - TDS 4.5	Login Name:		
Sybase ASE 11/12/15 - TDS 5.5	Login Name;		
Sybase Anywhere	Password:		
Sybase IQ DB2 - LUW 7.x		Save password	
DB2 - LUW 8.×/9.×		V Save password	
DB2 - iSeries			
DB2 - z/OS			
Informix	- Location		
Informix - ODBC	URL:	jdbc:unity://c:/unityjdbc/sources.xml	
Teradata			
Netezza	Driver:	unity.jdbc.UnityDriver	
Aster nCluster ParAccel Analytic Platform	Driver Location:	ar;C:\Program Files\AguaDataStudio11\jre\lib\ext\unityjdbc.jar	owse
MySQL			
PostgreSQL			
Apache Derby			
Generic - JDBC			
Generic - ODBC 🔹			
- Mounted Scripts			
Folder:		P	owse

Aqua Example: Adding a UnityJDBC Virtual Data Source in Aqua Data Studio

12. Multiple Database Query in Aqua Data Studio - Create a query like usual except it can contain multiple databases.

Aqua Data Studio 11.0.9 - [Evaluation] [Untitled]		
<u>File Edit Server Query Automate Tools</u>	D <u>B</u> A Tools ER Modeler <u>Wi</u> ndow <u>H</u> elp	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		
Servers 🗇	🙀 @unity_multiple [Untitled]* 🛛 🛛	
Catalog Unity_multiple Gradeage Unity_multiple Gradeage Unity	Image: Linkey: Image: Linkey: Image: Linkey: Image: Linkey: L	
🖻 🐺 db2_mysql	1:30 INS [7/11/2012 12:15:12 AM] Script executed - No Errors [Time: 117ms]	
Customer ⊕ ILinetem ⊕ ILine	Image: second	
Local Database Servers / unity_multiple / Catalog / unity	/ Database / db1_sqlserver / Table / CUSTOMER	124 : 200 : 455 MB 👘

Aqua Example: Executing a Multiple Database Join Query and Displaying Results

13.Function and SQL Dialect Translation with Aqua Data Studio - Forget what functions you can use on each database? No problem – UnityJDBC will translate automatically. This translation is supported for common databases and can be freely extended by user-defined functions and translations for each database dialect.

🛃 Aqua Data Studio 11.0.9 - [Evaluation] [Untitled]*		×
Eile Edit Server Query Automate Tools DBA Tools ER Modeler Window Help		
i Servers تا الالله المالي المالية المالي المالية المالي المالية المالي المالية المالي المالية المالي المالية ا المالي المالي المالية ال	4 ▷ 🗉	
		Detail
- ⊕ Catalog ⊖ @ unity → ₩ ⊠ № ⊠ № ⊠ № ⊠ № ≣ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩ ₩		
🔓 Database unity 🗸 Schema: db1_sqlserver 🗸 Username:		
⊖ = dbl_sdeever i select trin(N_NAME) as n_name left(R_NAME,5) as r_name from dbl_sdiserver.Nation INNER JOIN db2_mysdi.Region ON n_nationkey = r	_reg	
In the two second secon		
te Te orders		
In the second se		
User Defined Types	Þ	
⊕ Interem E Interem E Interem Sum: 0 Q-		
B- I Nation n_name r_name		
Image: Conders 1 ALGERIA AFRIC Image: Part 2 ARGENTINA AMERI		
B E Part2 3 BRAZIL ASIA		
Partsupp A CANADA ZUROP FOYT HIDL		
🔂 🌐 Supplier		
🕀 🋐 UnityJDBC_cssql 🔹 👻		
🗄 Servers 🗇 Files 😂 Projects 🖽 Grid in Messages		
Local Database Servers / unity_multiple / Catalog / unity / Database / db1_sqlserver / Table / CUSTOMER 121: 208:	455 MB	Û

Aqua Example: Performing SQL Dialect and Function Translation

UnityJDBC Data Virtualization with RazorSQL

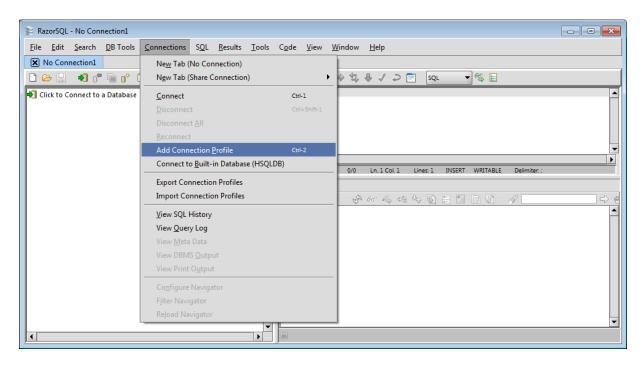
UnityJDBC allows RazorSQL users to create a virtual data source that may consist of multiple data sources on different servers and platforms. The user can enter one SQL query to combine and join information from multiple sources.

Benefits:

- UnityJDBC allows RazorSQL to support multiple database queries.
- No data source or server changes are required.
- UnityJDBC will perform function translation where a user requests a function that is not supported on a certain source.

This example assumes that the installation of UnityJDBC has already been completed, and that the user has already created a data virtualization using the UnityJDBC SourceBuilder. The following is an example of creating sources and executing multiple database queries with RazorSQL.

- 1. Installing the Jars Copy the unityjdbc.jar into the RazorSQL JRE directory (e.g. C:\Program Files (x86)\RazorSQL\jre\lib\ext).
- 2. Setup RazorSQL Start RazorSQL. Add a UnityJDBC connection profile under menu Connection->Add Connection Profile.



RazorSQL: Adding a New Connection Profile

3. Setup RazorSQL - Part 2 - Select OTHER then press the CONTINUE button.

nection Profiles	Add Connection Profile	
DAT	ABASE TYPE	
	the (InterSystems)	
	ASE	
DB		
De		
	Maker	
	ebird	
	ntBase	
	enplum	
H2		
	QLDB / HyperSQL	
	res	
	ormix	
	erbase	
	crosoft Access	Select a database type from the list on the left.
	crosoft SQL Server	
	mer SQL	If the database type is not in the list, select OTHER
	ngoDB	a the database type is not in the inst, select of mert
	SQL	Then, select the Continue button to enter connection information
	tezza	men, select the continue button to enter connection mormation
	enBase	
	acle	Continue
Par	adox	
Per	vasive	
Poi	ntBase	
	tgreSQL	
	pleDB (Amazon)	
So		
	Lite	
-	L Azure	
	ase (ASE)	
	ase (IQ)	
	ase (SQL Anywhere)	
	adata	
OT	HER	

RazorSQL: Adding a UnityJDBC Virtual Connection using OTHER Option

4. Setup RazorSQL Connection Profile - When setting up the connection profile, the profile name can be anything you wish. The driver location is where you just put the unityjdbc.jar. The driver class is unity.jdbc.UnityDriver. There is no login or password. The JDBC URL is the location of the configuration files for the virtual sources. In this example, we put the files UnityDemo.xml, UnityDemoOrder.xml, UnityDemoPart.xml that came in the distribution in the directory c:/temp/unityjdbc. You will change this to your source files that you built using the SourceBuilder utility.

Connection Wizard			
Connection Profiles Add Connection Pr	ofile		
	Profile Name	myconn	
CONNECTION TYPE	DRIVER INFO. Driver Location	C:\Program Files (x86)\RazorSQL\jre\lib\ext\unityjdbc.jar	Browse
ODBC	Driver Class	unity.jdbc.UnityDriver	
*RazorSQL JDBC Bridge	AUTHENTICATION Login Password Save Password DATABASE INFO.]
* Driver is shipped with RazorSQL	JDBC URL	jdbc:unity://c:/temp/unityjdbc/UnityDemo.xml	
	Auto Commit	⊙ On ○ Off ○ Smart Commit	
	SQL Restrictions	None ○ Read ○ Read / ○ Read / Write ○ Read / Write ○ / Delete / Delete	
	<u>SSH Tunnel</u> <u>Connect at Startup</u>	Configure	
		CONNECT BACK	

RazorSQL: Creating a UnityJDBC Virtual Connection

5. **RazorSQL Multiple Database Connection and Query** - Connect to the connection profile. You can then see tables from all your virtual sources and built a multiple database query.

RazorSQL - myconn File Edit Search DB Tools Connections SQL	Results Tools Code View Win	dow Help		
x *myconn				
🗅 🗁 🔛 🔞 🕐 🖷 🕐 🕕 🔗 🗄 🗟 🥸	🗐 🧇 🗄 🔜 🚐 🚎 ★ 🖾 🔿	\$\$\$√ ⊅ 🖹	SQL • 65	
Conterd Customer Custom		m OrderDB.NATION N nkey = N2.n_nation Ln.2 Col 34 Lines: 2	k e y INSERT WRITABLE	n Cp1252 Delimiter; ;
	N NATIONKEY	N NAME	N REGIONKEY	NC
	1 0	ALGERIA	-	final accounts wake quickly, special
	2 1	ARGENTINA	1	idly final instructions cajole stealthil
	3 2	BRAZIL	1	always pending pinto beans sleep si
	4 3	CANADA	1	foxes among the bold requests
	5 4	EGYPT	4	pending accounts haggle furiously.
⊕ [™] MATION	6 5	ETHIOPIA		fluffily ruthless requests integrate fl
🕀 🗀 ORDERS	7 6	FRANCE	3	even requests detect near the pendi
🖶 🗁 PART	8 7	GERMANY	3	blithely ironic foxes grow. quickly p
Der Partsupp	9 8	INDIA	2	ironic packages should have to are :
region	10 9	INDONESIA	2	unusual excuses are quickly request
🗄 🗀 SUPPLIER	▼			
•	14:48:33:752 Loading	Tables Done.		

RazorSQL: Executing a Multiple Database Join with UnityJDBC

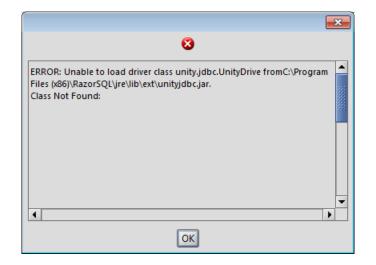
A few possible setup errors and their resolution:

1. **Unable to make a connection** - If you do not type the connection string correctly or you have not put the XML files in the right location, you will get the error below. To fix, verify the location of the files.

	×
8	
ERROR: An error occurred while trying to make a connection to the database:	
JDBC URL: jdbc:unity://c/temp/unityjdbc/UnityDemo.xm	
Unable to retrieve connection information at: c:/temp/unityjdbc/UnityDemo.xm	
•	•
OK	

RazorSQL: Error When Unable to Find Sources File

2. Unable to find driver - If you did not put the correct class name or location of the unityjdbc.jar, you will get the following error. Verify the class name and jar location to resolve.



RazorSQL: Error When Unable to Find Driver or Incorrect Driver Class Name

Chapter 6. Supported SQL Syntax

Overview

UnityJDBC supports a cross-database SELECT statement. The SELECT statement has the standard SQL-92 syntax and supports WHERE, ORDER BY, GROUP BY, and HAVING. UnityJDBC supports subqueries on a single database and cross-database including subqueries in the FROM clause and WHERE clause. entire query is on a single database. SQL functions are supported using a function syntax with parameters rather than using SQL keywords and syntax. Table and fields often should be prefixed with the database name they originate from. This database name is provided in the schema file for the data source.

Data Types

The standard SQL data types are supported. Since UnityJDBC uses the JDBC drivers provided by database vendors, non-standard data types may not be universally supported.

Identifiers

An *identifier* is a string used to reference a database, table, or field. Identifiers follow the standard SQL rules. Since a UnityJDBC query may span multiple databases, table and field identifiers defined in a data source may not be unique across all data sources. In which case, the database name should be added to the identifier to create a unique system-wide identifier. For instance, consider an order database given the name OrderDB with a table called Orders and fields id and orderDate. The Orders table may be referred to using only Orders or OrderDB.Orders. Similarly, the field id may be referred to as Orders.id or OrderDB.Orders.id. Standard aliasing using AS in the FROM and SELECT clauses is supported. Delimited identifiers are supported by enclosing in double quotes (e.g. "from" or "my field with spaces"). Delimited identifiers must be used for SQL reserved words.

Functions and Operators

Arithmetic operators +, -, /, %, * are supported as well as generic expressions. Functions are not specified according to SQL keyword syntax but rather as a function identifier with parameters similar to programming languages. The format of functions is: function (param1, param2, ...).

Logical Operators

The logical operators of AND, OR, NOT, and XOR are available.

Comparison Operators

The following comparison operators are available:

Table 6.1. Comparison Operators

Operator	Description
<	less than
>	greater than
<=	less than or equal to

Operator	Description
>=	greater than or equal to
=	equal
!=	not equal
IS [NOT] NULL	tests if value is NULL
IS [NOT] [TRUE FALSE]	tests if value is true or false

Arithmetic Functions and Operators

The following mathematical operators are supported:

Operator	Description		
+	addition (and string concatenation for strings)		
-	subtraction		
/	division		
%	modulus (remainder of integer division)		
*	multiplication		

The following are a few of the mathematical functions supported. A complete list of functions is available on the web site.

 Table 6.3. Mathematical Functions

Function	Return Type	Example	Result	Description
abs(x)	Same as x	abs(-17.4)	17.4	Absolute value
ceil(x)	Same as input	ceil(-42.8)	-42	Smallest integer not less than argument
exp(x)	Same as input	exp(1.0)	2.718	exponential
floor(x)	Same as input	ln(2.0)	0.69314	natural logarithm
log(x)	Same as input	log(100.0)	2	base 10 logarithm
power(a, b)	double precision	power(9,3)	729	a raised to the power of b
random()	double precision	random()		random value between 0.0 and 1.0
sqrt(x)	double precision	sqrt(2.0)	1.4142	square root

String Functions

The following are a few of the string functions supported. A complete list of functions is available on the web site.

Function Ret		Example	Result	Description	
<str> + <str> Stri</str></str>		'Unity' + 'JDBC'	UnityJDBC	String concatenation	
ascii(string) in		ascii('xyz')	120	ASCII code of the first character of the input string	
length(string)	int	length('UnityJDBC') 9		Length of string in characters	
lower(string)	String	lower('JDBC')	jdbc	Convert string to lower case	
position(search, target)	int	position('J','UnityJDBC')	5	Location of search in target (indexed from 1)	
replace(source, search, replace)	String	replace('abUnityabJDBC', 'ab', 'XX')	XXUnityXXJDBC	Replace all occurrences of search string in source string with replace string	
substring(string, Strin start)		substring('UnityJDBC',6)	JDBC	substring starting at position start	
substring(string, start, count)	String	substring('UnityJDBC,6,2)	JD	substring starting at position start and taking count characters	
trim(string)	String	trim(' UnityJDBC ')	UnityJDBC	remove leading and trailing spaces from string	
ltrim(string) OR trim(string, 'LEADING')	String	trim(' UnityJDBC ')	'UnityJDBC '	remove leading spaces from string	
rtrim(string) OR trim(string, 'TRAILING')	String	trim(' UnityJDBC ')	' UnityJDBC'	remove trailing spaces from string	
trim(string, ['BOTH', 'LEADING', 'TRAILING'], [<chars>])</chars>	String	trim('aaaUnityJDBCbbb', 'BOTH', 'ab')	UnityJDBC	remove leading, trailing or both from string where characters removed may be optionally specified in <chars></chars>	
upper(string)	String	upper('jdbc')	JDBC	Convert string to upper case	

Table 6.4. String Functions

Pattern Matching Operators

Pattern matching is supported using the LIKE operator.

For example, 'abcdef' LIKE 'ab%' is true. The '%' is used to match one or more characters, and '_' is used to match a single character.

Data Type Conversion Functions

Data type conversions are performed using the CAST(x, y) function. The CAST function takes any object as the first parameter and takes a string literal representation of the type to cast to as the second parameter. Note that the type must be put in single quotes as a string literal. Example:

CAST(45, 'VARCHAR') creates '45'

```
Possible type names are: 'VARCHAR', 'CHAR', 'INT', 'FLOAT', 'DOUBLE', 'DATE', 'TIMESTAMP', 'TIME'.
```

Date/Time Functions and Operators

The following are a few of the date functions supported. A complete list is on the website.

Table 6.5. Date Functions

Function	Return Type	Example	Result	Description
CURRENT_TIMESTAMP	TIMESTAMP	CURRENT_TIMESTAMP	2011-07-06 12:53:45	Returns the current date. Format: "yyyy- MM-dd HH:mm:ss"
CURRENT_TIME	TIME	CURRENT_TIME	12:53:45	Returns the current time. Format: "HH:mm:ss"
CURRENT_DATE	DATE	CURRENT_DATE	2011-07-06	Returns the current date. Format: "yyyy-MM-dd"
YEAR	INT	YEAR('2011-07-06)	2011	Returns the year of the given date expression.
MONTH	INT	MONTH('2011-07-06)	7	Returns the month of the given date expression.
DAY	INT	DAY('2011-07-06)	6	Returns the day of the given date expression.
DATEADD	TIMESTAMP	DATEADD('2011-07-06', INTERVAL 3 days)	2006-07-06 12:53:45	Allows the addition of a given date field to a datetime expression. Intervals are supported and are translated as necessary for systems that do not support them.

Aggregate Functions

The following aggregate functions are supported:

Table 6.6. Aggregate Functions

Function	Argument Type	Return Type	Description
avg(x)	int, float, double precision type	int for integer types, double precision for float/double types	Average of all input values
count(*)	N/A	int	Count of number of input values
count(x)	any	int	Count of number of non-null input values
group_concat(x)	any	varchar	Returns a comma-separated list of all input values.
max(x)	any comparable type	same as input	Maximum of all input values

Function	Argument Type	Return Type	Description
min(x)	any comparable type	same as input	Minimum of all input values
sum(x)	int, float, double precision type	int for integer types, double precision for float/double types	Sum of all input values

User-Defined Functions and Support for Other Functions

For queries on a single database, UnityJDBC parses functions and passes them directly to the database engine for execution. Thus, all functions that can be executed at the source are available. UnityJDBC and user-defined functions are used only when applying functions to data **after** it is extracted from the sources. UnityJDBC will parse queries containing functions that it itself cannot process in its internal database engine. These functions are passed down to the database engine and executed locally. Only functions that require inputs from more than one database are processed in the UnityJDBC database engine. All other functions are passed down to the sources.

UnityJDBC supports user-defined functions (UDFs). Adding your own user-defined function is easy. There are two types of functions: row functions and aggregate functions. A row function operates on one row at a time for its data and includes functions like SUBSTRING() and ABS(). An aggregate function is used in GROUP BY queries and aggregates an expression (usually a column) across multiple rows in a group to produce a single value. Examples include MAX() and COUNT().

To create a row function, you must create a Java class that extends the Function class. A template example is in the file F_Function_Template.java. This class must implement a constructor, an evaluate() method, and provide information on the parameters it requires. Once completed, as long as this function is available in the CLASSPATH, UnityJDBC will search for it when called. A similar template is available for aggregate functions, A_Aggregrate_Template.java. Sample code is provided in the directory unity/functions.

Function Translation

UnityJDBC has a database of known functions. This database contains information on what functions are supported on each data source. This is how UnityJDBC processes functions:

- 1. UnityJDBC does not support function If a function is not in the UnityJDBC database, it is passed down as-is to the underlying source. If the source is able to execute it successfully, the query continues. If not, an error is thrown.
- 2. UnityJDBC supports function, data source requires translation If the function requested in the query is not directly supported by the data source (different name, different parameters, etc.), but UnityJDBC contains a mapping in its database, the function is translated to the correct form on the data source and executed on the data source.
- 3. UnityJDBC supports function, data source does not support function If UnityJDBC supports the function but not the data source, then the query is optimized to perform as much of the processing as possible on the source, but the function execution is performed internally in UnityJDBC. This way your query can execute on data sources with the help of UnityJDBC that do not support the required functions.
- 4. UnityJDBC is running with local execution If the local execution flag is set for the UnityStatement object executing the query, all functions except aggregate functions are executed by UnityJDBC. This setting may be useful to reduce load on the source or to guarantee absolute consistency of function execution across different sources.

The UnityJDBC function database is encrypted and stored in the unityjdbc.jar. To add user-defined functions to the function database, create a mapping.xml file in the JRE classpath (execution directory, etc.) that stores the information on the function. An example is included in the release and more information is available on the web site.

Non-parsed Functions

UnityJDBC attempts to support most of the SQL standard. If there is a function or feature not supported, it is possible to use the NP() function to pass the query string directly to the data source by-passing UnityJDBC validation. This may be used to support non-standard functions or SQL syntax. Below are several examples.

```
Query:
SELECT N1.n_nationkey, NP('OrderDB', 'n_name', 'varchar')
FROM OrderDB.Nation N1 WHERE N1.n_nationkey = 1;
Result: (n_name is substituted directly into the query)
SELECT n_nationkey, n_name
FROM Nation N1 WHERE N1.n_nationkey = 1
Query:
SELECT N1.n_nationkey, NP('OrderDB','(select n_name from nation n2
where N1.n_nationkey = N2.n_nationkey)', 'varchar') as name
FROM OrderDB.Nation N1 WHERE N1.n_nationkey = 1
Result:
SELECT N1.N_NATIONKEY,
(select n_name from nation n2 where N1.n_nationkey = N2.n_nationkey) name
FROM NATION N1 WHERE N1.N_NATIONKEY = 1
Query:
SELECT N2.*
FROM NP('OrderDB',
 '(select n_name,n_nationkey from nation)','n_name,n_nationkey') N1,
NP('PartDB',
 '(select n_name,n_nationkey from nation)','n_name,n_nationkey') as N2
where N2.n_nationkey < 2 and N1.n_nationkey = N2.n_nationkey;
Result:
// Substitutes subquery for each of the two data sources (OrderDB and PartDB).
// The result of the two subqueries is then joined at the UnityJDBC level.
// OrderDB:
SELECT N2.n_name, N2.n_nationkey
FROM (select n_name,n_nationkey from nation) N2\n WHERE N2.n_nationkey < 2
// PartDB:
SELECT N1.n_nationkey FROM (select n_name,n_nationkey from nation) N1
```

More information on non-parsed functions is available on the web site.

SELECT Statement

The SELECT statement supported by UnityJDBC has the following syntax.

```
SELECT [ALL | DISTINCT ] <exprList>
    [FROM <tableList>]
    [WHERE <condition>]
    [GROUP BY <exprList>
    [HAVING <condition>]
    [ORDER BY <expr> [ASC | DESC],...]
```

[LIMIT <expr> [OFFSET <expr>]]

- An <exprList> is a list of expressions. Each individual expression <expr> may be a column identifier, a literal constant, or some expression consisting of operators, functions, constants, and column identifiers. Recall that a column identifier may often need to be prefixed by its database name and table name.
- A <tableList> is a list of table references. Each table reference can be aliased using the AS operator. A table reference may also be a named subquery such as SELECT * FROM (SELECT * FROM T1) AS R WHERE R.val > 50.
- A <condition> is a boolean condition that may contain multiple subconditions related using AND, OR, and XOR.
- If the GROUP BY clause is used, no attributes should be present in the SELECT <exprList> that are not in an aggregate function or are GROUP BY attributes.
- The HAVING <condition> filters groups and typically should contain only aggregate functions.
- The ORDER BY clause can order results on any number of attributes in either ascending or descending order.
- The LIMIT clause allows paging of results. The OFFSET clause determines the first row of the result with the first row numbered as 1.

Some examples using the TPC-H schema follow. The database name for these examples is 'OrderDB'.

Return all nations with their key and name:

SELECT OrderDB.Nation.n_nationkey, OrderDB.Nation.n_name
FROM OrderDB.Nation;

Return the nations and their regions. Only return nations in the region name of 'AMERICA'. Note the use of table aliasing using AS.

```
SELECT N.n_nationkey, N.n_name, R.r_regionkey, R.r_name
FROM OrderDB.Nation as N, OrderDB.Region as R
WHERE N.n_regionkey = R.r_regionkey AND R.r_name = 'AMERICA';
```

Calculate the number of countries in each region. Only return a region and its country count if it has more than 4 countries in it. Order by regions with most countries.

```
SELECT R.r_regionkey, R.r_name, COUNT(N.n_nationkey)
FROM OrderDB.Nation as N, OrderDB.Region as R
WHERE N.n_regionkey = R.r_regionkey
GROUP BY R.r_regionkey, R.r_name
HAVING COUNT(N.n_nationkey) > 4
ORDER BY COUNT(N.n_nationkey) DESC;
```

INSERT Statement

The INSERT statement supported by UnityJDBC has the following syntax:

INSERT INTO <tbl_name> [(<col_name>,...)] VALUES <exprList>;

Specifying column names is optional. An example is below:

INSERT INTO mydb.Customer (id,firstname,lastname,street,city)
VALUES (52,'Fred','Jones','Smith Lane', 'Chicago');

UnityJDBC also supports INSERT INTO ... SELECT with the following syntax:

```
INSERT INTO <tbl_name> [(<col_name>,...)] VALUES <exprList>
(SELECT <query>);
```

This is useful for storing query results into another table. Note that this table and all its column must already exist or an error will be returned. Here is an example:

INSERT INTO emptydb.customer (SELECT * FROM mydb.customer);

UPDATE Statement

The UPDATE statement supported by UnityJDBC has the following syntax:

UPDATE <tbl_name> SET coll=expr1, col2=expr2, ... [WHERE <condition>];

An example is below:

UPDATE Employee SET salary=salary*1.10 WHERE age > 50;

DELETE Statement

The DELETE statement supported by UnityJDBC has the following syntax:

DELETE FROM <tbl_name> [WHERE <condition>];

An example is below:

DELETE FROM Employee WHERE salary > 100000;

EXPLAIN Statement

The EXPLAIN statement supported by UnityJDBC has the following syntax:

EXPLAIN <query>

The EXPLAIN statement provides an explanation of the how UnityJDBC will execute a given query include the translated queries to be executed on each source, the operations performed by UnityJDBC, and the expected cost of each query operation. Using EXPLAIN is a great way to determine the performance of queries and improve their execution speed.

By-Pass Statement

You can use methods to by-pass or flow through the driver to execute an untranslated query directly on a single source. In the UnityStatement class are these two methods:

```
ResultSet executeByPassQuery(String dbName, String sql)
int executeByPassUpdate(String dbName, String sql)
```

These methods will execute a query or update on a single source (given by name). The SQL statement provided is not parsed or validated and passed directly to the source driver. There is no overhead in this type of query as it is equivalent to invoking the source's JDBC driver directly.

Chapter 7. Supported JDBC Methods Overview

UnityJDBC supports the majority of the methods in the Driver, Connection, Statement, ResultSet, and ResultSetMetaData interfaces. UnityJDBC supports the PreparedStatement interface but not the CallableStatement interface. UnityJDBC supports native updates using INSERT, DELETE, and UPDATE. It is also possible to use INSERT INTO to insert query results into another table. UnityJDBC does not support transactions across databases. Support for other JDBC methods is also limited by the underlying support of the JDBC driver for each data source. UnityJDBC requires a JDK of 1.6 or higher.

Chapter 8. UnityJDBC Driver Internals Overview

UnityJDBC contains an embedded database engine to join the results produced by executing queries on other JDBCaccessible sources. It requires a JDBC driver for each source to be accessed. The UnityJDBC architecture is the result of years of research and development and has been published in numerous technical and research publications.

Embedded Relational Database Engine

Embedded in UnityJDBC is a relational database engine and associated operators of selection, projection, grouping, ordering, and join. You can build your own global query spanning data sources by combining these operators into an execution tree. In the distribution is a file called ExampleEngine.java which demonstrates how to use the engine to build an execution tree. Also in this file is an example on how you can have Unity parse but not execute a global query. UnityJDBC will return its global query and execution plan which you can later execute. This feature gives you the opportunity to modify the global execution plan before execution if desired. It also allows you to track the progress of a global query at the operator level.

Chapter 9. History and Planned Features

1. History of UnityJDBC

UnityJDBC is the product of over 10 years of research and development in database integration and virtualization. UnityJDBC was first released in 2006 and commercial support and development has been ongoing since 2011.

Release Version and Date	Major Features
UnityJDBC v1.0 - May 2006	Cross-database join support, match functions, full opti- mizer, query by-pass
UnityJDBC v2.0 - May 2007	Connection pools, DataSource connections, more functions
UnityJDBC v3.0 - May 2008	Native INSERT/UPDATE/DELETE, INSERT INTOS- ELECT across databases, PreparedStatements, user-defined functions
UnityJDBC v4.0 - August 2011	Database dialect translation, paging using LIMIT/OFFSET, single database subqueries, result caching
UnityJDBC v4.1 - June 2012	Integration with SQuirreL SQL and JasperReports, BLOB support
UnityJDBC v4.2 - January 2013	Memory-optimized query execution engine, EXPLAIN for query plans, subqueries in FROM clause, improved Source- Builder GUI
UnityJDBC v4.3 - June 2014	Full subqueries, EXCEPT/INTERSECT, implementation of PooledDataSource and ConnectionPoolDataSource, sup- port for MongoDB, Splunk, Cassandra, and ServiceNow.
UnityJDBC v4.4 - May 2020	Support for Tableau integration for Splunk and MongoDB, performance improvements, improved MongoDB nesting processing

Table 9.1. UnityJDBC Release History

2. Planned Features

The following features are planned in coming versions. Version 5.0 will be released in May 2021 with new versions released approximately every year. If you have any feature requests, please e-mail support@unityjdbc.com.

Table 9.2. Planned Features in Coming Versions

Ver- sion	Feature Description
5.0	Query optimization and performance improvements, improved SourceBuilder user interface

3. Feature List

The following table summarizes the features of UnityJDBC and the version where they were first introduced.

Ver- sion	Feature Description
1.0	Cross-database SQL queries for any JDBC source
1.0	Query by-pass
1.0	MERGE feature with MATCH functions
1.0	Embedded relational database engine
1.0	Source and schema file encryption
1.0	Support for Applet
1.0	Support for query results/databases larger than main memory
2.0	DataSource connections
2.0	Pooled connections
3.0	Prepared Statements
3.0	User-defined Functions
3.0	INSERT, DELETE, UPDATE on a single source
3.0	INSERT, DELETE, UPDATE across sources
3.0	INSERT INTO across sources
4.0	Paging using LIMIT/OFFSET
4.0	Query and ResultSet caching
4.0	Universal dialect and function translation (support for sources missing functions)
4.0	Single source subqueries
4.1	Support for BLOBs/CLOBs.
4.2	Multiple source subqueries in FROM clause
4.2	EXPLAIN command for query execution information
4.2	Ability to control tables extracted into data virtualization by inclusion/exclusion patterns
4.3	Full subquery support including correlated subqueries
4.3	INTERSECT/EXCEPT
4.3	MongoDB, Splunk, ServiceNow, and Cassandra suppprt
4.4	Tableau JDBC integration with support for Splunk and MongoDB
4.4	MongoDB nested query optimizations
4.4	Adddtional SQL functional and translation support
4.4	Enhanced support and performance for Splunk, MongoDB, and ServiceNow

4. Contacts and Support

Please contact support@unityjdbc.com if you encounter any bugs, issues, or have feature requests.

This document is Copyright 2020 by Unity Data Inc. All rights reserved.