Database - Feature #2143

Feature # 1656 (Closed): add Microsoft SQL Server support

prototype IKVM solution for Java UDFs compiled to CIL assemblies and loaded in SQL Server

05/02/2013 02:14 PM - Eric Faulhaber

Status:	Closed	**Start date:**	08/26/2013
Priority:	Normal	**Due date:**	09/05/2013
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	60.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Blocks Database - Feature #2148: enhance prototype to compile p2jpl.jar file ...	**Closed**	**09/06/2013**	**09/27/2013**

History

#1 - 05/02/2013 02:20 PM - Eric Faulhaber

- Estimated time set to 284.00

This will provide a partial analog (i.e., the subset we need) to PL/Java on SQL Server. The features we need are:

- the ability to load one or more JVMs within the SQL Server database backend process(es);
- a way to register user defined functions implemented in Java within a SQL Server database, possibly dynamically (as opposed to statically loading a jar file into a database as a manual step);
- the ability to back those functions with JNI (or a .NET bridge), so that they are executed in a JVM and the result returned to SQL Server;
- a minimal type mapping between SQL and Java data types, which will allow us to use our existing UDF definitions (i.e., in the com.goldencode.p2j.persist.pl package).

The preferred approach is to use SQL Server C bindings with JNI, but we should investigate the pros and cons of using a .NET/Java bridge.

#2 - 05/02/2013 02:20 PM - Eric Faulhaber

- Estimated time changed from 284.00 to 288.00

#3 - 05/02/2013 10:14 PM - Eric Faulhaber

- Subject changed from native UDF support for Java in SQL Server to prototype IKVM solution for Java UDFs compiled to CIL assemblies and loaded in SQL Server

- Due date set to 09/05/2013

- Assignee changed from Eugenie Lyzenko to Eric Faulhaber

- Start date set to 08/26/2013

- Estimated time changed from 288.00 to 60.00

Preliminary research indicates that using IKVM is feasible and probably represents the simplest solution. Turns out SQL Server does not support C language bindings for UDFs, but it does support .NET.

The scope of this prototype is as a proof of concept:

- writing a simple Java implementation of a UDF;
- compiling it to CIL with IKVMC;
- getting it to loading into SQL Server;
- successfully executing it in a SELECT statement.

#4 - 02/19/2014 02:54 PM - Ovidiu Maxiniuc

Progress so far:

- created a small test java class with 2 operators
- compiled the java source code and built the jar
- the jar was successfully converted to dll (the converter/compiler is smart enough to generate a .exe if a class has a main method that was declared as the entry point of the application)
- I attempted to convert the whole p2jpl.jar. Operation finished and wrote an output exe file, but there was tons of warnings (because of the jar dependencies).
- in the case of .exe, the application refused to be executed until I added two support libraries from ikvm (Core & Runtime). I guess this is ok and it also proved the conversion was successful.
- I created a small Sql server project in VS2013 and created a "CLR Scalar Function". For the moment the input/return types are those from c++, need to investigate if other java-specific datatypes can be supported (there is a support for adding new datatypes). The function declared in sql should refer the previously compiled dll.
- I added the dll to project and the two dependencies. They were resolved just fine, I was able to see the Operators class inside the assembly along with the two methods I declared. The java.lang.long datatypes were resolved using the additional ddl s from IKVM.
- Unfortunately, I got stuck here, the IDE refuses to associate the EXTERNAL NAME with the assembly I added to references. There is no auto-complete feature nor any other "VISUAL" help.

I will continue my investigations.

#5 - 02/19/2014 04:00 PM - Eric Faulhaber

- Status changed from New to WIP

- Assignee changed from Eric Faulhaber to Ovidiu Maxiniuc

Sounds like you're making good headway, Ovidiu.

> I attempted to convert the whole p2jpl.jar. Operation finished and wrote an output exe file, but there was tons of warnings (because of the jar dependencies).

What dependencies are those? This jar should stand alone.

#6 - 02/20/2014 01:50 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Sounds like you're making good headway, Ovidiu.

This was the easy part.. the time-consuming part is to put all pieces to work together and fix all errors that keep popping out.

> I attempted to convert the whole p2jpl.jar. Operation finished and wrote an output exe file, but there was tons of warnings (because of the jar dependencies).

> What dependencies are those? This jar should stand alone.

Well, this might be an issue. Currently the content of this jar refers a lot of other libraries, APIs not directly implemented by IKVM or p2j classes that are not included inside the:
antlr, dom4j, apache.commons, apache.logging, apache.axiom, javax.wsdl, com.goldencode.p2j.net, com.goldencode.p2j.admin, com.goldencode.p2j.ui.
Probably in the last updates there was some imports added in some places that make those dependencies ?
Here are the first few warnings:

```
warning IKVMC0105: Unable to compile class "com.goldencode.expr.ExpressionLexer"
    (missing class "antlr.CharScanner")
warning IKVMC0105: Unable to compile class "com.goldencode.expr.ExpressionParser"
    (missing class "antlr.LLkParser")
warning IKVMC0105: Unable to compile class "com.goldencode.expr.ExtraAst"
    (missing class "antlr.CommonAST")
warning IKVMC0105: Unable to compile class "com.goldencode.p2j.security.DatabaseAuthenticationHook$1"
    (missing class "com.goldencode.p2j.net.SessionListener")
warning IKVMC0105: Unable to compile class "com.goldencode.p2j.ui.ComponentConfig"
    (missing class "com.goldencode.p2j.ui.client.widget.WidgetConfig")
warning IKVMC0105: Unable to compile class "com.goldencode.p2j.ui.BaseConfig"
    (missing class "com.goldencode.p2j.ui.ComponentConfig")
warning IKVMC0105: Unable to compile class "com.goldencode.p2j.ui.BrowseColumnConfig"
    (missing class "com.goldencode.p2j.ui.ComponentConfig")
warning IKVMC0105: Unable to compile class "com.goldencode.p2j.ui.BrowseConfig"
    (missing class "com.goldencode.p2j.ui.BaseConfig")
```

#7 - 02/21/2014 03:46 PM - Ovidiu Maxiniuc

- File sqlqueries-conc1.sql added

- File Functions.cs added

- File Functions2.cs added

- File Operators.java added

The SQL Server can execute 'external' code in two ways:
1. **as 'Extended Stored Procedures'**. This is the deprecated solution and, even if it is yet available in SQL Server 2012, Microsoft intends to remove this feature in a future version of SQL Server. Developpers are adviced not to use this feature in new development work, and modify applications that currently use this feature as soon as possible.
The extended procedures are also Windows dlls but they need to use a special API in order to be successfully loaded and executed by the Server.
I managed to add (at least it was visible in management console) the test dll converted by ikvm using

```
   EXEC  sp_addextendedproc 'P2J_Add', 'c:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\MSSQL\Binn\p2
j_operators.dll';
```

However, I was not able to execute anything using this feature because of multiple reasons: * converted dlld most likely did not implement the correct API * the methods from java are mapped to methods from CLR objects. The syntax does not allow it
select Operators.P2J_Add(2, 2) * this concept is built for running trully natibe C/C++ code
I only investigated this as a temporary alternative.

2. **the CLR Integration Concept**. This enables to write stored procedures, triggers, user-defined types, user-defined functions, user-defined aggregates, and streaming table-valued functions, using any .NET Framework language that is compiled to CLI.

I implemented a small test class (c# with sample test funtions simmilar to those needed by p2jpl). I was able to install the assembly into database and perform some basic selects using the UDFs. With this opportunity I discoved that there is a rather strong-typed environment and the names of the methods/functions cannot be overloaded. I believe that psql and h2 also have the same requirements, but the renaming (indexing of the overloaded methods) happens transparently, when installing the cod.
The CLR-SQL type mapping can be found here: [[http://msdn.microsoft.com/en-us/library/bb386947(v=vs.110).aspx]]
We need to check the additional mapping from/to java to CLR.
As mentioned in a previous note the conversion of java code executed without problems. The generated dll has (at least for the simple testcase) two dependencies: IKVM.Runtime.dll and IKVM.OpenJDK.Core.dll. Before adding to database the converted assembly that contain our functions, the dependecies must be added, one at a time. Here is the big problem. The two dlls above mentioned, are mutual dependant. This is rather strange in dll world but possible. The IKVM does a dirty little trick for compiling them, using a two stage compile (this can be found in the chapter 'Additional Information' in the HOWTO document that came with the ikvm sources).
I tried to add the two dlls in a single batch by grouping them using GO sql statement but this was useless.
Other important thing to note is that execution of CLR code must be manually configured for the current database using:

```
   -- enabling execution oc CLR code on SQL server:
   EXEC sp_configure 'clr enabled';       -- just display the current status, default is 0 (false)
   EXEC sp_configure 'clr enabled' , '1';
   RECONFIGURE;
```

At this moment I try to build the base dlls as a single compile unit so it will be successfully loaded into SQL server.

#8 - 02/24/2014 01:08 PM - Ovidiu Maxiniuc

Finally, I have managed to:

- install the IKVM assemblies into SQL Server
- installed the assembly converted from java
- created two basic functions (add(long, long) and concat(String, String))
- executed a select that calls the two functions that in turn will execute the code converted from java

I believe this is the target for 1st step of this task. Be back soon with details.

#9 - 02/24/2014 02:00 PM - Eric Faulhaber

This is great news! When you are ready, please post an update and detailed instructions about:

- the specific software versions you have used for this POC (Windows, compiler, .NET framework, SQL Server, IKVM, etc.), and (with respect to open source software or free developer editions) how/where to get them;
- compiling/building the assembly from the Java source code;
- installation into a SQL Server database instance.

#10 - 02/24/2014 03:32 PM - Ovidiu Maxiniuc

- File Operators.java added

- File sqlqueries-conc1.sql added

Eric Faulhaber wrote:

> This is great news! When you are ready, please post an update and detailed instructions about:
>
> - the specific software versions you have used for this POC (Windows, compiler, .NET framework, SQL Server, IKVM, etc.), and (with respect to open source software or free developer editions) how/where to get them;
> - compiling/building the assembly from the Java source code;
> - installation into a SQL Server database instance.

Software configuration

Here is the detailed software configuration (usually they are the latest versions of the free 'distribution' for the downloadable binaries/sources but for some tasks, when special versions are required they are documented):

- **Microsoft Windows 7 Home Premium**, X86-based (32 bit), 6.1.7601 Service Pack 1 Build 7601
- **Microsoft Visual Studio Express 2013 for Windows Desktop** (version 12.0.21005.1 REL), running on .NET framework 4.5.50938 (not really needed for our purpose, just for testing of some ideas) [[http://www.visualstudio.com/downloads/download-visual-studio-vs#d-express-windows-desktop]]
- **SQL Server Express 2012** (version 2011.110.2100.60) [[http://www.microsoft.com/en-us/download/details.aspx?id=29062]] This version will not be used on production, of course.
- **Microsoft SQL Server Management Studio** (version 11.0.2100.60), running on .NET Framework 4.0.30319.18408, downloaded with SQL Server Express
- **Oracle Java** 7 Update 51 [[http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html]]
- **.NET Reflector** (version 8.3.3.115) (a trial-mode tool used for inspection of generated assemblies, not directly needed for our purpose, just mentioned it here if ever needed again) [[http://www.red-gate.com/dynamic/products/dotnet-development/reflector/download]]
- **IKVM** (version 7.2.4630.5) [[http://sourceforge.net/projects/ikvm/files/ikvm/7.2.4630.5/]]
- **NAnt** A .NET Build Tool (only needed to re-compile IKVM, must be version 0.85, the latest, 0.92 won't work!) [[http://sourceforge.net/projects/nant/files/nant/0.85/]]
- **OpenJDK 7u6 b24** sources (only needed to re-compile IKVM) [[http://sourceforge.net/projects/ikvm/files/ikvm/7.2.4630.5/]]
- **ICSharpCode.SharpZipLib.dll** (only needed to re-compile IKVM) [[http://www.icsharpcode.net/opensource/sharpziplib/]]

Building the assembly from the Java source code

This is the easy step. The java source code must be first compiled and packed in a jar archive. I used the latest JDK (set in PATH) for these task (no special switches here):

```
javac Operators.java
jar cfv p2j_operators.jar Operators.class
```

The second step is converting the jar to dll assembly. If there is a main() method in any compiled classes in the jar, the result of the conversion will be an exe. This is not important, the SQL server will handle both, equally. At least for the very simple test-case the conversion command is straightforward:

```
ikvmc p2j_operators.jar
```

A new file p2j_operators.exe will be created. If the main() method has some test-cases these can be run at this moment to verify the conversion (eventually, some of the IKVM dlls must be made available, or else the application will crash).

Installation into a SQL Server database instance.

Before installation of the new code, there are a few steps that must be performed:

- Enabling execution of CLR code on SQL server:

  ```
  EXEC sp_configure 'clr enabled';
  EXEC sp_configure 'clr enabled' , '1';
  RECONFIGURE;
  ```

  Of course, the first line will print the status of the 'clr enabled' flag. If already set to 1, no need to run the next two lines.
- Set

  ```
  ALTER DATABASE p2j_test SET TRUSTWORTHY ON
  ```

  This is needed for installing IKVM assemblies. However, I need to double-check if this is really needed on a clean database. I will update this note when I know for sure.
- The .NET framework or SQL Server seem to gave a glitch, the following files Accessibility.dll, System.Drawing.dll, System.Runtime.Serialization.Formatters.Soap.dll and System.Windows.Forms.dll from \Windows\Microsoft.NET\Framework\v4.0.30319 should be copied to ikvm\bin directory.
- Time to install the IKVM assemblies (will install all dependencies recursivelly)

  ```
  CREATE ASSEMBLY ikvm_runtime
  FROM 'c:\Program Files\ikvm-7.2.4630.5\bin\IKVM.Runtime.dll'
  WITH PERMISSION_SET = UNSAFE

  CREATE ASSEMBLY ikvm_openjdk_jdbc
  FROM 'c:\Program Files\ikvm-7.2.4630.5\bin\IKVM.OpenJDK.Jdbc.dll'
  WITH PERMISSION_SET = UNSAFE
  ```

  The second assembly tree is needed if classes form java.sql.* are involved (like Date and Timestamp).
- Now we can finally insert into database out assembly:

  ```
  CREATE ASSEMBLY [p2j_operators]
  AUTHORIZATION [dbo]
  FROM 'c:\Work\1stSqlProject\P2JMsSql1\java\p2j_operators.dll'
  WITH PERMISSION_SET = SAFE
  ```

- In order to be used by SQL queries, we need to define SQL functions based on the 'external' functions imported previously:

  ```
  CREATE FUNCTION [dbo].[P2J_Add32bitScalars]
  (@param1 INT, @param2 INT)
  RETURNS INT
  AS EXTERNAL NAME p2j_operators.Operators.P2J_Add32bitScalars
  ```

  The SQL functions cannot be overloaded and the exact type mapping must be used. The sql name does not need to be the same with the one

from java/CLR.
- At this moment, hoping there were no errors, we can use our functions:

```
select [dbo].[P2J_Add32bitScalars](-10, 20) as 'int';
```

I have attached the updates Operators source and a SQL script with commands/queries used. Note that multiple SQL function definitions cannot appear in a single BATCH, they need to be executes one at a time or separated by GO statements.

#11 - 02/25/2014 02:25 PM - Ovidiu Maxiniuc

My previous note was written rather quickly based on the notes I took while doing investigations and some actions were not justified or I was not sure if they are really needed.

The TRUSTWORTHY database property must be set to on in order to create the assembly from System.Drawing.dll.

> *CREATE ASSEMBLY for assembly 'System.Drawing' failed because assembly 'System.Drawing' is not authorized for PERMISSION_SET = UNSAFE. The assembly is authorized when either of the following is true: the database owner (DBO) has UNSAFE ASSEMBLY permission and the database has the TRUSTWORTHY database property on; or the assembly is signed with a certificate or an asymmetric key that has a corresponding login with UNSAFE ASSEMBLY permission.*

The following files Accessibility.dll, System.Drawing.dll, System.Runtime.Serialization.Formatters.Soap.dll and System.Windows.Forms.dll from \Windows\Microsoft.NET\Framework\v4.0.30319 should be copied to ikvm\bin directory. The CLR/SQL Server is unable to resolve / find them automatically

> *Assembly 'IKVM.Runtime' references assembly 'system.drawing, version=2.0.0.0, culture=neutral, publickeytoken=b03f5f7f11d50a3a.', which is not present in the current database. SQL Server attempted to locate and automatically load the referenced assembly from the same location where referring assembly came from, but that operation has failed (reason: 2(The system cannot find the file specified.)). Please load the referenced assembly into the current database and retry your request.*

The clr enabled must be set to 1. No .NET code is executed otherwise. This is a SQL-Server level flag, that affects all databases from the current instance. Here is the error when attempting to execute CLR code when set to 0:

> *Execution of user code in the .NET Framework is disabled. Enable "clr enabled" configuration option.*

Additional warnings like the following will be printed:

> *Warning: The Microsoft .NET Framework assembly 'system.drawing, version=4.0.0.0, culture=neutral, publickeytoken=b03f5f7f11d50a3a, processorarchitecture=msil.' you are registering is not fully tested in the SQL Server hosted environment and is not supported. In the future, if you upgrade or service this assembly or the .NET Framework, your CLR integration routine may stop working. Please refer SQL Server Books Online for more details.*

The IKVM.Runtime.dll assembly fail to install with PERMISSION_SET = SAFE because some dependency assemblies use explicit synchronization and this is not allowed in safe assemblies.

> *CREATE ASSEMBLY failed because method "getAttribute" on type "java.text.AttributedString" in safe assembly "IKVM.OpenJDK.Text" has a synchronized attribute. Explicit synchronization is not allowed in safe assemblies.*

The IKVM.OpenJDK.Jdbc.dll assembly fail to install with PERMISSION_SET = SAFE failed because some 'initialize' methods is storing to a static

field. Storing to a static field is not allowed in safe assemblies.

> *CREATE ASSEMBLY failed because method 'initialize' on type 'java.sql.DriverManager' in safe assembly 'IKVM.OpenJDK.Jdbc' is storing to a static field. Storing to a static field is not allowed in safe assemblies.*

In normal developing process, newer versions of the initial source code are usually needed to be loaded for testing. To do that, the assembly have to be unloaded in reverse order: first ALL defined functions must be drop-ped, then the assembly itself. After the new assembly is compiled, it must be added to database then the functions redefined. There is no need to reload the IKVM assemblies unless they need to be recompiled.

One thing that I investigated today and **I could not find a satisfactory answer** is the use of java objects. In my tests I was only able to define UDFs using scalar datatypes that fully match in java, .NET and SQL:

java	CLR (CLR Object)	SQL
boolean	bool (Boolean)	BIT
int	int (Int32)	INT
long	long (Int64)	BIGINT
String	string (String)	NVARCHAR(*n*), *n* must be apriori known (like defining a SQL column)

The java objects (others than String) (Boolean, Integer, Long, BigDecimal, java.sql.Date, java.sql.Timestamp) are translated by IKVM to their counterpart converted implementation from OpenJDK. Practically they are just other objects unrelated to the classes available in .NET so the SQL Server is unable to map them with SQL datatypes.

The solution could be to define UDT (User Defined Types) for each of the above classes. For this we need to add special attribute to the assembly that holds our datatypes (IKVM.OpenJDK.Core):

```
ALTER ASSEMBLY [IKVM.OpenJDK.Core] WITH VISIBILITY = ON
```

Then we can create the UDT in the database:

```
CREATE TYPE JBoolean EXTERNAL NAME [IKVM.OpenJDK.Core].[java.lang.Boolean]
```

Unfortunately, this statement will fail:

> *CREATE TYPE failed because type "java.lang.Boolean" does not conform to the UDT specification: missing custom attribute "Microsoft.SqlServer.Server.SqlUserDefinedTypeAttribute".*

This looks normal, the SQL server need to have some interface for handling these new defined objects. For it (and CLR generally), the [IKVM.OpenJDK.Core].[java.lang.Boolean] is visible as

```
public sealed class Boolean : Object, Serializable.__Interface, Comparable, ISerializable
```

as converted from OpenJDK by IKVM.
I do not know at this moment if recompiling IKVM assemblies would be of any help because IKVM processes the already-compiled classes not java source code.

#12 - 02/25/2014 05:14 PM - Eric Faulhaber

Ovidiu, nice work so far.

Please continue investigating the data type issue. Can we create a very thin "shim" layer that maps some very basic UDTs (which conform to the required UDT specification) to the Java types we need? If so, this would suggest we eventually could map these UDTs directly to our P2J wrapper types.

In addition, we need to expand the scope of this POC with 2 more items, which should have been in the original specification:

- We need to be able to invoke the UDFs from HQL. The syntax in the SQL sample script seems a bit unusual. Please investigate how we might deal with this from HQL. The HQL we emit into converted code must be uniform, regardless of the back-end database in use. If Hibernate has a way to deal with this through the dialect system, great. If we have to massage the HQL into something dialect-specific at runtime, we can do this with the HQLPreprocessor. If that's not possible, we may have to tweak the way Hibernate generates SQL Server-friendly SQL from our homogeneous HQL. For now, we just want to know how these UDFs could be invoked from HQL, given that odd SQL syntax.
- So far, for any Windows development, we have been able to avoid setting up a "true" Windows development environment by taking advantage of open source projects like mingw (allows native Windows C/C++ development on Linux without a full Cygwin installation or Visual Studio development environments). This task goes a level beyond that, in that it uses C#/.NET. But there is the Mono project for .NET development. Once everything is working to your satisfaction in the environment you described above, please investigate whether you can get it to work with that environment (or another, if something else exists). Let us know if you think doing that will entail an unusual amount of work.

#13 - 02/26/2014 01:54 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, nice work so far.

Thanks.

> Please continue investigating the data type issue. Can we create a very thin "shim" layer that maps some very basic UDTs (which conform to the required UDT specification) to the Java types we need? If so, this would suggest we eventually could map these UDTs directly to our P2J wrapper types.

I am investigating at this moment how such class layer can be used. Yes, it's logically to write these UDT adapters directly to our P2J wrapper types at least for two reasons: we have the code and we can do adjustments to it (at least much more easier than those from OpenJDK) and second, the data in tables are serializations of P2J data so it does not make sense to round-trip using Java types (for both input and output values).

> In addition, we need to expand the scope of this POC with 2 more items, which should have been in the original specification:
>
> - We need to be able to invoke the UDFs from HQL. The syntax in the SQL sample script seems a bit unusual. Please investigate how we might deal with this from HQL. The HQL we emit into converted code must be uniform, regardless of the back-end database in use. If Hibernate has a way to deal with this through the dialect system, great. If we have to massage the HQL into something dialect-specific at runtime, we can do this with the HQLPreprocessor. If that's not possible, we may have to tweak the way Hibernate generates SQL Server-friendly SQL from our homogeneous HQL. For now, we just want to know how these UDFs could be invoked from HQL, given that odd SQL syntax.

The [and] can be safely removed (I believe) because they just let the SQL server know the content is an identifier and not a reserved keyword. I expect P2J won't have such conflicts. On the other hand, the dbo. seems to be mandatory for UDFs, the database names, table names and field names can be specified directly, but I could not call UDFs without the schema prefix. However, I believe that the HQLPreprocessor can fix these

special calls by prefixing with "dbo.".

- So far, for any Windows development, we have been able to avoid setting up a "true" Windows development environment by taking advantage of open source projects like mingw (allows native Windows C/C++ development on Linux without a full Cygwin installation or Visual Studio development environments). This task goes a level beyond that, in that it uses C#/.NET. But there is the Mono project for .NET development. Once everything is working to your satisfaction in the environment you described above, please investigate whether you can get it to work with that environment (or another, if something else exists). Let us know if you think doing that will entail an unusual amount of work.

I am not sure I understand this right. From my POV, the problem is the SQL Server that requires Windows NT platform. The IKVM is advertised to work with Mono (I did not had the chance, yet, but I will check soon the cross-conversion). Depending on the additional tools from previous point, I believe the development can be done entirely on Linux. Only the deployment and testing need the Windows platform as I am not sure SQL Server will run on Wine.

#14 - 02/26/2014 02:33 PM - Greg Shah

> Only the deployment and testing need the Windows platform as I am not sure SQL Server will run on Wine.

No problem. We have no intention of trying to run SQLServer on Linux/UNIX.

Our objective is to try to get all development and even the builds (of the DLLs/assemblies) done using open source tools. It is OK to have to build (compile/link) on Windows as well, BUT if possible we want to be able to do this without a dependency on MS Visual Studio and Microsoft's other .NET tooling.

All the other Windows code so far has been handled using MINGW, but that is not .NET.

#15 - 02/27/2014 03:26 PM - Ovidiu Maxiniuc

- File P2JLogical.cs added

- File P2JWrappers.sql added

- File IkvmLogical.cs added

My idea is to implement the requested interface for SQL server and defer the processing to a component. The same pattern is used for both java/ikvm and p2j classes.

- using the java/ikvm backing classes allows to use the existing Functions/Operators static functions in SQL

- using p2j classes directly have no obvious advantage, need to study this further on.

I have created a C# class that will be mapped to SQL BIT that wraps the P2J com.goldencode.p2j.util.logical (attached). The only communication with SQL clients is via string format through two methods: ToString() and Parse(SqlString s). NULL values are handled in dedicated code. The persistence to SQL database can be left to SERVER "native" (probably the entire objects are serialized) and also can be manually serialized using Write(System.IO.BinaryWriter w) and Read(System.IO.BinaryReader r) - similar to our serialization - in a more optimized manner. Unfortunately, as far as I understand, there is no possibility to do implicit casting to and from native datatypes. The only solution is via the string format as shortly described above.

Shortly, to install the assembly, the following steps should be followed:

- compile p2jpl.jar to an library (dll or exe) and install it as an assembly in database in SQL (as in one of my previous posts). Because of its content, it can only be created with PERMISSION_SET = UNSAFE.
- compile the P2JLogical.cs (and all other similar C# classes) into a P2JWrapper.dll. Install it also as an assembly.
- now we can create our UDTs:

```
CREATE TYPE P2JLogical EXTERNAL NAME [p2j_wrapper].[P2JLogical]
```

  We will have to repeat this for each P2J type.
- Now create the table directly with the new UDTs:

```
CREATE TABLE [p2j_wrappers] (
    [id] [bigint] IDENTITY(21764, 17) NOT NULL,
    [p2j_logical] [P2JLogical] NULL,
    [sql_logical] [BIT] NULL
) ON [PRIMARY]
```

- to populate the table we need to pass the values as strings, except for NULL s that will be used as such.

```
INSERT INTO [p2j_wrappers] VALUES ('true', 1)
INSERT INTO [p2j_wrappers] VALUES ('1', 1)
-- INSERT INTO [p2j_wrappers] VALUES (1, 1) <-- fails, Operand type clash: int is incompatible with P2JLog
ical
INSERT INTO [p2j_wrappers] VALUES ('false', 0)
INSERT INTO [p2j_wrappers] VALUES ('0', 0)
-- INSERT INTO [p2j_wrappers] VALUES (0, 0) <-- fails, Operand type clash: int is incompatible with P2JLog
ical
INSERT INTO [p2j_wrappers] VALUES (NULL, null)
```

- finally we can query the data:

```
SELECT * FROM [p2j_wrappers]
SELECT [p2j_logical], [sql_logical] FROM [p2j_wrappers]
```

Note that SQL Server allows that CLR methods to be applied directly to UDT:

```
SELECT [p2j_logical].P2J_And('falsE'),
       [p2j_logical].P2J_And(' true')
   FROM [p2j_wrappers]
```

I wonder if this could come in handy to use as an alternative to Functions / Operations: each method of these classes to be replaced by such a method of the first argument and drop the UDF at all. So instead of writing SELECT p2j_and(a, b) FROM table1 we will have SELECT a.and(b) FROM table1. I think this will avoid the method overloading mentioned in a previous post. But this is just an idea, must test is HQL can support it for a dialect.

An issue I found are SQL statements like:

```
SELECT * FROM [p2j_wrappers] WHERE [p2j_logical]
```

The evident solution is to rewrite it as:

```
SELECT * FROM [p2j_wrappers] WHERE [p2j_logical] = '1'
```

But this looks bad. I intended to find a solution and added a method that returns the values as a bool (this is the mapped type of SQL BIT type). This way the query becomes:

```
SELECT * FROM [p2j_wrappers] WHERE [p2j_logical].value()
```

This won't work. Later I discovered that the 'native' case also does not:

```
SELECT * FROM [p2j_wrappers] WHERE [sql_logical]
```

Instead the value() can be used to extract the native value from UDT:

```
SELECT [p2j_logical].value(), [p2j_logical].P2J_And('true').value() FROM [p2j_wrappers]
```

#16 - 03/04/2014 03:50 PM - Ovidiu Maxiniuc

- File note16.txt added

- File TestApp2.java added

During the last days I tried to find a solution for compiling / converting previously attached files in linux using ikvm and mono cross-compiling.

Linux has in the repositories an old version of the IKVM: 0.46.0.1. The problem is that this will convert but only java code that was generated by JDK6 and previous.
On the ikvm.net there are some new version of IKVM 7.x. But they only compile for windows / .net. To use it in linux a little trick must be used, the conversion must be run from Mono.

The windows build of IKVM need to be downloaded and installed in file-system. I used /usr/bin/ikvm/bin for binaries and /usr/lib/ikvm/ for libraries.

Now we can convert the p2jpl:

```
mono /usr/bin/ikvm/bin/ikvmc.exe -target:library lib/p2jpl.jar -out:bin/p2jpl.dll -version:0.0.1.0
```

then build the SqlUserDefinedType adapter assembly and the small test application:

```
mcs -r:/usr/lib/ikvm/IKVM.OpenJDK.Core.dll -r:bin/p2jpl.dll -r:System.Data.dll -target:library -out:bin/P2JLog
ical.dll src/P2JLogical.cs
```

```
mcs -r:/usr/lib/ikvm/IKVM.OpenJDK.Core.dll -r:bin/P2JLogical.dll -r:System.Data.dll -target:exe -out:bin/TestA
pp1.exe src/TestApp1.cs
```

At this moment, running the test application on Linux with Mono, fails with following error:

```
Unhandled Exception: System.TypeLoadException: A type load exception has occurred.
  at com.goldencode.p2j.util.logical..ctor (System.String value) [0x00000] in <filename unknown>:0
  at P2JLogical.Parse (SqlString s) [0x00000] in <filename unknown>:0
  at TestApp1.Program.Main (System.String[] args) [0x00000] in <filename unknown>:0
[ERROR] FATAL UNHANDLED EXCEPTION: System.TypeLoadException: A type load exception has occurred.
  at com.goldencode.p2j.util.logical..ctor (System.String value) [0x00000] in <filename unknown>:0
  at P2JLogical.Parse (SqlString s) [0x00000] in <filename unknown>:0
  at TestApp1.Program.Main (System.String[] args) [0x00000] in <filename unknown>:0
```

Apparently, the logical(String) c'tor is not processed correctly. Googling this shows that a library issue may be the cause even Mono itself.

On Windows the application crashes with the following error:

```
Unhandled Exception: System.IO.FileNotFoundException: Could not load file or assembly 'System.Data, Version=4.
0.0.0, Cul
ture=neutral, PublicKeyToken=b77a5c561934e089' or one of its dependencies. The system cannot find the file spe
cified.
File name: 'System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
   at TestApp1.Program.Main(String[] args)
```

At first look the .NET assemblies are not found where they are supposed to be.

On SQL server, the assemblies will load in database, even the UDT and tables can be created with them.
However, when running a query that uses the new UDT the following error is printed:

```
Msg 6522, Level 16, State 1, Line 1
A .NET Framework error occurred during execution of user-defined routine or aggregate "P2JLogical":
System.Security.SecurityException: Request failed.
System.Security.SecurityException:
   at P2JLogical.Parse(SqlString s)
```

I am continuing with investigations.

At some time, I tried to compile with the old IKVM (using a trimmed version of p2jpl that compiled with javac -target 6) and the execution of the test application was successful.
I also tried to access directly the P2JPL in TestApp2. This was also successful.
I am missing something here...

#17 - 03/06/2014 01:32 PM - Ovidiu Maxiniuc

- File P2JInt64.cs added

- File TestApp1.cs added

- File AssemblyInfo.cs added

- File note18.txt added

I discovered the errors from my previous note. The generated assemblies are compiled against different .Net frameworks so the Framework refused to load and execute them. It's a strange thing but, using the same parameters the on Windows, the target is version is v2.0.50727, while on Linux with Mono, the default is v4.0.30319.

The .Net target version can be check on windows using CorFlags utility that can be found in a place similar to

```
c:\Program Files\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools\CorFlags.exe
```

On Mono/Linux I found

```
monop2 --runtime-version bin/p2jpl.dll
```

but it will print the global runtime (runtime version: 4.0.30319.1), not the version extracted from actual file.

Because of the raised security checks in .Net 4.0, SQL Server refuses to execute our code converted to CLR probably because of some native code exiting in FileChecker, LaunchManager, MemoryManager, Registry. The generated code targeting .NET 2.0 (as generated by default by IKVM on windows) does not have such restrictions so the assemblies are loaded successfully.
To obtain the converted code targeting the .Net 2.0 on Mono/Linux the default assembly references need to be ignored and manually specify the desired ones. The compile script must be adjusted as follows:

```
mono --runtime=v2.0.50727 /usr/bin/ikvm/bin/ikvmc.exe -target:library -nostdlib -r:/usr/lib/mono/2.0/mscorlib.
dll -r:/usr/lib/mono/2.0/System.dll -out:bin/p2jpl.dll -version:0.0.1.1 lib/p2jpl.jar

dmcs -r:/usr/lib/mono/2.0/cscompmgd.dll -r:System.Windows.Forms -r:/usr/lib/ikvm/IKVM.OpenJDK.Core.dll -r:bin/
p2jpl.dll -pkg:dotnet -target:library -out:bin/P2JWrappers.dll src/P2JLogical.cs src/P2JInt64.cs src/AssemblyI
nfo.cs

dmcs -r:/usr/lib/mono/2.0/cscompmgd.dll -r:/usr/lib/ikvm/IKVM.OpenJDK.Core.dll -r:bin/P2JWrappers.dll -pkg:dot
net -target:exe -out:bin/TestApp1.exe src/TestApp1.cs
```

For that, the following prerequisites are needed:

- make sure the mono-complete package is installed
- the IKVM (vs. 0.46.0.1) from Linux repository must be completely uninstalled
- the latest IKVM (vs. 7.2.4630.5) should be configured (I installed to the same locations as removed version: the executable binaries to /usr/bin/ikvm and the assemblies to /usr/lib/ikvm)
- the IKVM assemblies must be added to the Global Assembly Cache. The best way to do this is using a shell script (with root rights) like this:

```
for line in $(find /usr/lib/ikvm/ -iname '*.dll'); do
    gacutil -i "$line"
done
```

This will make the assemblies accessible for all later commands.

One more issue arises: how to handle **no overloaded methods** constraint ?
More than one method, property or field was found with name 'P2J_Add' in class 'P2JInt64' in assembly 'P2JWrappers'. Overloaded methods, properties or fields are not supported.
Operators and Functions have an abundance of such methods. Should we use type decorations resolved in HQL, too?

#18 - 03/06/2014 01:43 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> One more issue arises: how to handle **no overloaded methods** constraint ?

We had to deal with this constraint for H2 as well. Please have a look at H2Helper.getFunctionAliases. Can we re-factor/re-use this approach or something similar?

#19 - 03/06/2014 02:00 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> We had to deal with this constraint for H2 as well. Please have a look at H2Helper.getFunctionAliases. Can we re-factor/re-use this approach or something similar?

Yes, probably yes. Thanks.
I remember that I saw SQL user defined functions with counter, but not the context. I will handle this issue and add it to POC.

#20 - 03/10/2014 04:49 PM - Ovidiu Maxiniuc

I have some issues with the current generated datatypes in for the tables ddl. My concerns regard the following:

- use of BIT for logical. The BIT in SQL server is very strange... You cannot say something like SELECT * FROM table WHERE bit_field. It must be rewritten to SELECT * FROM table WHERE bit_field = 1. The bit_field = 1 is of type bool, which is different than BIT.
- P4GL decimals are stored as numeric(38, n) with some restrictions to n. double precision has a greater precision and maps to double scalar type of CLR / java.
- for strings we are using varchar(8000). As far as I remember, the character datatype in OpenEdge is double-byte enabled (unicode). We should use nvarchar even the maximum size drops to 4000. In fact this is the mapping CLR and java type, too.

#21 - 03/11/2014 04:01 PM - Ovidiu Maxiniuc

I have this exception:

```
[03/11/2014 21:54:43 EET] (com.mchange.v2.resourcepool.BasicResourcePool$AcquireTask:WARNING) com.mchange.v2.r
esourcepool.BasicResourcePool$AcquireTask@50201a40 -- Acquisition Attempt Failed!!! Clearing pending acquires.
 While trying to acquire a needed new resource, we failed to succeed more than the maximum number of allowed a
cquisition attempts (30). Last acquisition attempt exception:
java.sql.SQLException: No suitable driver
        at java.sql.DriverManager.getDriver(DriverManager.java:289)
        at com.mchange.v2.c3p0.DriverManagerDataSource.driver(DriverManagerDataSource.java:224)
        at com.mchange.v2.c3p0.DriverManagerDataSource.getConnection(DriverManagerDataSource.java:135)
        at com.mchange.v2.c3p0.WrapperConnectionPoolDataSource.getPooledConnection(WrapperConnectionPoolDataSo
```

```
urce.java:182)
        at com.mchange.v2.c3p0.WrapperConnectionPoolDataSource.getPooledConnection(WrapperConnectionPoolDataSo
urce.java:171)
        at com.mchange.v2.c3p0.impl.C3P0PooledConnectionPool$1PooledConnectionResourcePoolManager.acquireResou
rce(C3P0PooledConnectionPool.java:137)
        at com.mchange.v2.resourcepool.BasicResourcePool.doAcquire(BasicResourcePool.java:1014)
        at com.mchange.v2.resourcepool.BasicResourcePool.access$800(BasicResourcePool.java:32)
        at com.mchange.v2.resourcepool.BasicResourcePool$AcquireTask.run(BasicResourcePool.java:1810)
        at com.mchange.v2.async.ThreadPoolAsynchronousRunner$PoolThread.run(ThreadPoolAsynchronousRunner.java:
547)
```

I am using the com.microsoft.sqlserver.jdbc.SQLServerDriver driver set in hibernate/connection/driver_class of my database.
The hibernate/connection/url is jdbc:jtds:sqlserver://192.168.0.11:1433/myTestDatabase.

Is SQLServerDriver not compatible with c3p0 ?

#22 - 03/12/2014 04:19 PM - Ovidiu Maxiniuc

- File om_upd20140312a.zip added

Hurray!
I have fixed all issues mentioned in my previous notes and successfully run some basic 4GL statements using that are converted in queries using p2jpl functions.

I am attaching an update with some issues that need to be fixed (however, not all of them are related to MS SQL server).

I will write tomorrow a summary of the solution and step by step configuration, because the thread became dirty with my notes regarding some dead-ends I reached during my investigations.

#23 - 03/12/2014 04:21 PM - Eric Faulhaber

That is excellent news, Ovidiu, well done!

#24 - 03/13/2014 03:48 PM - Ovidiu Maxiniuc

- File om_upd20140313a.zip added

The attached update pack contains some missing files from previous archive. Please review.
The C# source code were temporarily added to src/com/goldencode/p2j/persist/pl, however, I believe they should be moved to native/ or something more appropriate.

- **Prerequisites**
  The following tools should be installed and configured
  1. IKVM see note 17 above.
  2. full Mono and C# cross compiler (same note 17)

3. sqsh and freetds installed and configured with the Windows machine
4. a working directory from where all the following commands will be executed with following empty dirs: src / lib / bin

- **Build/convert the p2j project and copy the p2jpl.jar to lib folder:**
  In the the p2j.cfg.xml the database should declare sqlserver2012 as supported dialect:

```
<parameter name="ddl-dialects" value="h2,postgresql,sqlserver2012" />
```

  If all goes well, the ddl/schema_table_<dbname>_sqlserver2012.sql will begin with (a lot of) statements that create external UDFs from our assembly.

- **Convert the p2jpl.jar to an assembly:**

```
mono --runtime=v2.0.50727 /usr/bin/ikvm/bin/ikvmc.exe -target:library lib/p2jpl.jar -nostdlib -r:/usr/lib/
mono/2.0/mscorlib.dll -r:/usr/lib/mono/2.0/System.dll -out:bin/p2jpl.dll -version:0.0.1.1 2> p2jpl_conv.lo
g
```

  The p2jpl.dll will be generated in /bin folder. A rather big log file will be created in the current folder (p2jpl_conv.log) with lots of warnings and hopefully, no errors.

- **Gather C# source files:**
  Copy *.cs files (AssemblyInfo.cs, Common.cs, Functions.cs, Operators.cs) from src/com/goldencode/p2j/persist/pl/ of the p2j project to src/ folder.

- **Cross-compile the C# sources:**

```
dmcs -r:/usr/lib/mono/2.0/cscompmgd.dll -r:System.Windows.Forms -r:/usr/lib/ikvm/IKVM.OpenJDK.Core.dll -r:
/usr/lib/ikvm/IKVM.OpenJDK.Jdbc.dll -r:bin/p2jpl.dll -pkg:dotnet -target:library -out:bin/p2j2clr.dll src/
*.cs &> p2j2clr_compile.log
```

  You can inspect the output using:

```
      monop2 -r:bin/p2j2clr.dll
      monop2 -r:bin/p2j2clr.dll Functions | less
      monop2 -r:bin/p2j2clr.dll Operators | less
```

- **Copy the output files**
  Copy generated files (dlls from bin/ folder) to Windows machine where the SQL Server is running (ex: in C:\Work\x-compiled\). Make sure the following files can also be found in the same location:
  IKVM Runtime support (about 22 files),
  Additional files from .NET: System.Drawing.dll, System.Windows.Forms.dll, Accessibility.dll, System.Runtime.Serialization.Formatters.Soap.dll.

- **Install our support binaries**:
  After the database (p2j_test) has been created connect with sqsh to SQL Server and execute:

```
USE p2j_test
go
ALTER DATABASE p2j_test SET TRUSTWORTHY ON
go
CREATE ASSEMBLY [IKVM.Runtime] FROM 'C:\Work\x-compiled\IKVM.Runtime.dll' WITH PERMISSION_SET = UNSAFE
go
CREATE ASSEMBLY [IKVM.OpenJDK.Jdbc] FROM 'C:\Work\x-compiled\IKVM.OpenJDK.Jdbc.dll' WITH PERMISSION_SET =
UNSAFE
go
CREATE ASSEMBLY p2jpl FROM 'C:\Work\x-compiled\p2jpl.dll' WITH PERMISSION_SET = UNSAFE
go
CREATE ASSEMBLY p2j2clr FROM 'C:\Work\x-compiled\p2j2clr.dll' WITH PERMISSION_SET = UNSAFE
go
```

  Note:
  the paths given in these sql statements are relative to Windows machine even if they are launched from linux console.
  Now you can install the rest of the database using generated ddl/schema_table_p2j_test_sqlserver2012.sql.

- **Install SQL Server driver:**

Download Microsoft JDBC Driver 4.0 for SQL Server from [[http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774]]. Add sqljdbc4.jar with the rest of the libraries.

- **Configure p2j runtime:**
  Use the following settings in directory.xml in the database/p2j_test/hibernate section:

```
<node class="container" name="hibernate">
    <node class="string" name="dialect">
        <node-attribute name="value" value="com.goldencode.p2j.persist.dialect.P2JSQLServer2012Dialect"/>
    </node>
    <node class="container" name="connection">
        <node class="string" name="driver_class">
            <node-attribute name="value" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
        </node>
        <node class="string" name="url">
            <node-attribute name="value" value="jdbc:sqlserver://192.168.0.11:1433;databaseName=p2j_test"/
>
        </node>
        <node class="string" name="username">
            <node-attribute name="value" value="p2j"/>
        </node>
        <node class="string" name="password">
            <node-attribute name="value" value="p2j"/>
        </node>
   ....
```

  Use the appropriate values for server name/ip, server port, user and password.

- **Launch the P2J application**

#25 - 03/14/2014 03:48 PM - Ovidiu Maxiniuc

- File om_upd20140314a.zip added

- File p2j2clr-dependencies.png added

Changes in update zip:

- removed Operators.java (it was identical to bzr version)
- added full implementation of Operators.cs

#26 - 03/28/2014 12:37 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

#27 - 11/16/2016 12:07 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

Operators.java	431 Bytes	02/21/2014	Ovidiu Maxiniuc
Functions.cs	344 Bytes	02/21/2014	Ovidiu Maxiniuc
Functions2.cs	377 Bytes	02/21/2014	Ovidiu Maxiniuc
sqlqueries-conc1.sql	10.8 KB	02/21/2014	Ovidiu Maxiniuc
Operators.java	2.12 KB	02/24/2014	Ovidiu Maxiniuc
sqlqueries-conc1.sql	2.76 KB	02/24/2014	Ovidiu Maxiniuc
P2JLogical.cs	3.5 KB	02/27/2014	Ovidiu Maxiniuc
P2JWrappers.sql	1.28 KB	02/27/2014	Ovidiu Maxiniuc
lkvmLogical.cs	3.17 KB	02/27/2014	Ovidiu Maxiniuc
note16.txt	9.03 KB	03/04/2014	Ovidiu Maxiniuc
TestApp2.java	2.12 KB	03/04/2014	Ovidiu Maxiniuc
TestApp1.cs	669 Bytes	03/06/2014	Ovidiu Maxiniuc
AssemblyInfo.cs	1.35 KB	03/06/2014	Ovidiu Maxiniuc
P2JInt64.cs	3.56 KB	03/06/2014	Ovidiu Maxiniuc
note18.txt	5.8 KB	03/06/2014	Ovidiu Maxiniuc
om_upd20140312a.zip	112 KB	03/12/2014	Ovidiu Maxiniuc
om_upd20140313a.zip	129 KB	03/13/2014	Ovidiu Maxiniuc
om_upd20140314a.zip	126 KB	03/14/2014	Ovidiu Maxiniuc
p2j2clr-dependencies.png	38.8 KB	03/14/2014	Ovidiu Maxiniuc