Database - Feature #3930

allow session level control over the database(s) which are auto-connected

02/26/2019 01:24 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Igor Skornyakov	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Database - Bug #5440: finish DatabaseManager refactoring for impli...	**New**

History

#1 - 02/26/2019 01:41 PM - Greg Shah

To implement this, we must separate database auto-connect from server startup. Then we must allow auto-connect to be controlled at the session level. We need to be sure to support "partitioned" applications (this is my term for the work done in #2475). The idea here is that the semantics of CONNECT and DISCONNECT should not be associated with how the FWD server starts, but instead the semantics should be driven by the connection requirements of each application that is running in the server.

For more discussion see #3926.

#2 - 04/19/2021 03:37 PM - Igor Skornyakov

- Status changed from New to WIP

- Assignee set to Igor Skornyakov

#3 - 05/07/2021 11:27 AM - Igor Skornyakov

Greg Shah wrote:

> To implement this, we must separate database auto-connect from server startup. Then we must allow auto-connect to be controlled at the session level. We need to be sure to support "partitioned" applications (this is my term for the work done in #2475). The idea here is that the semantics of CONNECT and DISCONNECT should not be associated with how the FWD server starts, but instead the semantics should be driven by the connection requirements of each application that is running in the server.
>
> For more discussion see #3926.

From the discussion in #4175, I understood that in FWD the CONNECT does not mean establishing the JDBC connection to the relational database, but a connection to another instance of the FWD server.

Is the objective of this task means that we should have statically configured databases the JDBC connection to which will be postponed until explicit CONNECT from the "in-process" or remote FWD server? In particular (as the extreme use case), it should be possible to have a "database only" FWD server (like OE proserv) which will be used by other FWD servers for CONNECT, in particular, a CONNECT statement can be automatically executed at the "guest" server startup?
Thank you.

#4 - 05/07/2021 12:00 PM - Greg Shah

> I understood that in FWD the CONNECT does not mean establishing the JDBC connection to the relational database,

True.

> but a connection to another instance of the FWD server.

Not exactly. In the 4GL, the CONNECT statement is an actual connection to the database, though it can be done via shared memory instead of TCP/IP. In FWD, the CONNECT statement is the equivalent to establishing a logical database connection. That is why it is not directly mapped to JDBC.

How this is done depends on which FWD server is "authoritative" for the target database instance. The authoritative FWD server handles things (e.g. locking) which are controlled outside of the database server (e.g. PostgreSQL) itself. This means that we must communicate with the authoritative FWD server to obtain a lock but the rest of the communication can be done directly to the database server.

The most common case is when the converted appliction is running inside the same FWD server as is authoritative for the target database. In such a scenario, all the access is local and there is no special connection to a FWD server.

In the less common case of a target database whose authoritative FWD server is "remote" (i.e. is not the same server in which the converted code is running), this will indeed also require a connection to that remote FWD server. Even here the connection is virtual (we call it a "virtual session"). If the 2 FWD servers already have an active network connection between them, only a new virtual session is established. Only if this is the first virtual session between 2 FWD servers is there a new network connection established.

Your work is about the logical database session, which is different.

#5 - 05/07/2021 12:06 PM - Igor Skornyakov

Greg Shah wrote:

> Your work is about the logical database session, which is different.

Sorry, what is a *logical database session*? Is it something we do not have now?
Thank you.

#6 - 05/07/2021 12:20 PM - Greg Shah

It is the same thing we have now. I'm just referring to the **concept** that it is a logical and not physical session.

#7 - 05/07/2021 12:34 PM - Igor Skornyakov

Greg Shah wrote:

> It is the same thing we have now. I'm just referring to the **concept** that it is a logical and not physical session.

I see. Where can I read about "multi-application partitioning"?
Thank you.

#8 - 05/07/2021 12:49 PM - Eric Faulhaber

Any work done so far on this concept is in #2475. There is no other documentation per se.

The requirements which led to this issue are described in #3926.

#9 - 05/07/2021 01:01 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Any work done so far on this concept is in #2475. There is no other documentation per se.
>
> The requirements which led to this issue are described in #3926.

Thank you. Reading

#10 - 05/07/2021 01:40 PM - Igor Skornyakov

It is difficult to follow the discussion in #2475. It seems that it is mostly about conversion.

Here is my understanding of the situation and corresponding questions:

1. It is possible to deploy more than one application in a single FWD server (is an example/documentation of a corresponding configuration available or where in the code should I look to understand how it works?)
2. If an FWD server hosts more than one application there should be an additional layer that is an "owner" of the databases the applications work with and a connection from the application to its database(s) should be not the same as establishing JDBC connection from the FWD server process at its startup, In particular, this layer should manage locks (is it what is this task about)?
   Thank you.

#11 - 05/07/2021 05:59 PM - Eric Faulhaber

Let's take a step back and focus on the actual goal of this task, which is not so much about partitioned applications (though it will help enable such situations).

At present, we have a situation where certain databases are configured as "auto-connect" and others are not. Essentially, what this means is that a FWD database that is configured as auto-connect can be referenced by converted business logic without an explicit 4GL CONNECT statement, while a FWD database not so configured requires an explicit CONNECT statement before the database can be referenced in business logic. The problem is that the notion of auto-connect in the 4GL is not scoped to a database, it is scoped to a user session. Back in the early days of FWD, this distinction was not made, and I implemented it such that a database was either auto-connect for all sessions or for none.

The choice whether or not a database is auto-connect needs to be scoped to each session. In the directory today, this setting is configured under the ../database/<database name>/p2j/load_at_startup path. If true, the DatabaseManager "registers" that database once, during server startup, such that any converted code running in any user session in that FWD server can reference and use that database without issuing an explicit CONNECT statement first. If false, converted code can only reference and use that database if an explicit CONNECT statement has been executed first.

The problem with the current approach is that many applications are written such that it is assumed that different types of sessions need a different set of databases to be auto-connected (or none at all). Having a database connected (or not) server-wide does not provide enough flexibility, and business logic which assumes a database is (or is not) auto-connected will not run correctly if this server-wide setting does not match that program's assumptions. Thus, we need to:

- move the directory configuration for auto-connect from a database-oriented location (i.e., ../database/<database name>/p2j/load_at_startup) to a session-oriented location (e.g., /server/default/runtime/default/...);
  - this must support auto-connecting to both local and remote databases;
- rework the runtime to not load/register a database at server startup, but rather at session startup (if auto-connected) or not at all (assuming explicit CONNECT statements will be executed by the business logic as needed);
  - this essentially means auto-connect will execute an implicit CONNECT early in the session (before any converted code runs) and execute an implicit DISCONNECT when the session ends;
- possibly rework the database registration code a bit to avoid database registration/deregistration "thrashing" in cases of sessions connecting and disconnecting;
- fix the current reference counting mechanism used for database registration in DatabaseManager, which I think does not clean up properly in cases where sessions which executed an explicit CONNECT end without processing an explicit DISCONNECT.

Instead of configuring a simple TRUE/FALSE for auto-connect as we do today with the load_at_startup configuration item, we would want a slightly more advanced configuration item, which would support session-specific auto-connect specifications for both local and remote databases. This could be a container under a /server/default/runtime/default/ (or other, session-specific, override) path. Something like:

```
<node class="container" name="auto-connect">
  <node class="strings" name="connect-parameters">
     <node-attribute name="value" value="local1"/>
     <node-attribute name="value" value="local2"/>
     <node-attribute name="value" value="remote1 -H 10.45.116.30 -S rptsrv"/>
  </node>
</node>
```

The connect-parameters values would each be the equivalent of the parameters which would be passed at the 4GL command line or with an explicit CONNECT statement. The local1, local2, remote1 example values refer to physical database names, each of which would have a ../database/<database name>/ path in this directory. The rptsrv example would need to be configured in the port_services section of the directory (or the port number of a FWD server running on 10.45.116.30 could be specified instead). The parameters can be any parameter supported by the ConnectionManager.connect method.

Greg, if I've left anything out or you had something different in mind, please feel free to comment.

#12 - 05/08/2021 09:45 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Let's take a step back and focus on the actual goal of this task, which is not so much about partitioned applications (though it will help enable such situations).

Eric, thank you very much for the detailed explanations.

However, I'm still confused about the notion of "session-based auto-connect". From your description, it is unclear (for me) how to figure which databases should be auto-connected for a particular session.
Please note also the auto-connect on a session start can take a few seconds since we have to populate the meta-database (if I understand correctly).

#13 - 05/08/2021 02:08 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric, thank you very much for the detailed explanations.

You're welcome.

> However, I'm still confused about the notion of "session-based auto-connect". From your description, it is unclear (for me) how to figure which databases should be auto-connected for a particular session.

It is up to the person who configures the directory for a given project to make this determination. Anything configured in the /server/default/runtime/default/ path provides the default behavior. If different sessions need to override this behavior with a different list of auto-connects, an administrator would place their specific configurations for other sessions in more specific paths in the directory. The directory reading algorithm searches through the directory from most to least specific, and chooses the appropriate settings for the current session.

Let me simplify my proposed configuration from my previous post. Consider this entry in the directory:

```
<server>
  <default>
    <runtime>
       <default>
         ...
         <node class="strings" name="auto-connect">
           <node-attribute name="value" value="local1"/>
           <node-attribute name="value" value="local2"/>
           <node-attribute name="value" value="remote1 -H 10.45.116.30 -S rptsrv"/>
         </node>
         ...
```

To read this, you would do something like this in the code:

```
private static final String CFG_AUTO_CONNECT = "auto-connect";

...

Directory dir = DirectoryManager.getInstance();
List<String> connectStrings = dir.getStrings(Directory.ID_RELATIVE, CFG_AUTO_CONNECT);
```

Now, connectStrings contains the appropriate list of CONNECT strings for the current session (in this case, the default one from /server/default/runtime/default/, since we did not specify an auto-connect list of strings at a more specific path in the directory). The DirectoryManager will take care of reading the configuration from the right location. To understand the search algorithm, see the javadoc for Utils.getDirectoryNodeWorker, which is the worker method which all the more specialized directory access convenience methods eventually use.

> Please note also the auto-connect on a session start can take a few seconds since we have to populate the meta-database (if I understand correctly).

For the first session that registers a database, yes, there is this and other expensive setup that needs to take place. But each session is not getting its own copy of this information, it is still shared. So, it only needs to be done for the first session to connect. This is an issue we have already today with explicit CONNECT statements for databases which have not previously been registered at server startup or by another session previously executing a CONNECT.

Note that currently, we have a reference count mechanism, so that the last session to explicitly DISCONNECT from a database causes the runtime to go through a de-registration process for that database. That code does not get exercised much, and I'm concerned it may be broken in several ways:

- The reference count is probably not maintained correctly, because while it is a best practice to issue a DISCONNECT for every CONNECT, this probably does not always happen in real 4GL code. IIRC, we only decrement the reference count on DISCONNECT (not on regular session cleanup), so if the original code does not explicitly DISCONNECT, our reference count will be too high.
- The de-registration code is old and probably has not been maintained in sync with newer changes to the database registration code. So, there probably are data structures to which we add information on registration, which are not cleaned up on de-registration.

Note my comment from earlier:

> possibly rework the database registration code a bit to avoid database registration/deregistration "thrashing" in cases of sessions connecting and disconnecting;

What I meant by this is that a good bit of setup is needed when a database is first registered (like the metadata work you noted). If we have a situation where the reference count for sessions connected to a database (in the 4GL sense) reaches 0 frequently, we don't want to tear all this setup down, only to have to set it all up again the next time a new session is connected and the reference count goes back to 1. I am wondering if we should remove the de-registration altogether and just leave all the setup in place for the server's lifetime. This should be fine in the majority of cases, but it might present a problem if an application has a lot of transient connections to a lot of databases.

Anyway, let's get the auto-connect issue fixed first, and then we can figure out the best approach, if any, for de-registration.

#14 - 05/08/2021 02:26 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
> It is up to the person who configures the directory for a given project to make this determination. Anything configured in the /server/default/runtime/default/ path provides the default behavior. If different sessions need to override this behavior with a different list of auto-connects, an administrator would place their specific configurations for other sessions in more specific paths in the directory. The directory reading algorithm searches through the directory from most to least specific, and chooses the appropriate settings for the current session.

My question was about the session attribute the selection of the connection profile should be based on.

> Anyway, let's get the auto-connect issue fixed first, and then we can figure out the best approach, if any, for de-registration.

I see, thank you.

#15 - 05/08/2021 06:06 PM - Eric Faulhaber

Igor Skornyakov wrote:

> My question was about the session attribute the selection of the connection profile should be based on.

I'm sorry, I don't understand what you mean.

#16 - 05/09/2021 03:56 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > My question was about the session attribute the selection of the connection profile should be based on.
>
> I'm sorry, I don't understand what you mean.

We're talking that different sessions can be auto-connected to different databases. But in your examples, we have only one configuration. I do not think that "different sessions" mean "different users". I think it is more likely that it should be like different rights based on the connection certificate. Is it correct?
Thank you.

#17 - 05/10/2021 07:54 AM - Greg Shah

> But in your examples, we have only one configuration.

His example is the equivalent of the current approach which sets the global flag at the database level. By using the /server/default/runtime/default/ path one sets the default for all sessions which don't have a more specific override.

> I do not think that "different sessions" mean "different users". I think it is more likely that it should be like different rights based on the connection certificate. Is it correct?

It depends. If the FWD client is started using a certificate, then the certificate can be used to match the correct FWD account associated with the security context for the session. Certs are mostly used for process accounts (e.g. batch, appserver agents) and server accounts. Interactive FWD clients usually use some kind of application-level (4GL) password security instead of certificates.

Don't get hung up on this aspect of things. The helper methods for lookup in the directory will already handle the search algorithm to find the most specific auto-connect section which applies to the account of the session. The configuration can be written in multiple places in the directory, leading to the result which is being able to have different auto-connect (or no auto-connect) settings for different accounts/sessions (since every session has an account associated).

#18 - 05/10/2021 08:02 AM - Igor Skornyakov

Greg Shah wrote:

> Don't get hung up on this aspect of things. The helper methods for lookup in the directory will already handle the search algorithm to find the most specific auto-connect section which applies to the account of the session. The configuration can be written in multiple places in the directory, leading to the result which is being able to have different auto-connect (or no auto-connect) settings for different accounts/sessions (since every session has an account associated).

OK. Thank you.
BTW: the planned changes will affect the core functionality. Should I create a task branch for them?
Thank you.

#19 - 05/10/2021 08:08 AM - Greg Shah

No. In my opinion, this is not a very risky change. If you disagree, please help us understand what parts are risky.

#20 - 05/10/2021 08:28 AM - Igor Skornyakov

Greg Shah wrote:

> No. In my opinion, this is not a very risky change. If you disagree, please help us understand what parts are risky.

Well, if I understand #3930-11 correctly the database configuration in the directory.xml is supposed to be changed in a non-backward-compatible way. Or I should check the "new" configuration first and then revert to the old-style configuration (or vice-versa)?

#21 - 05/10/2021 09:15 AM - Greg Shah

Igor Skornyakov wrote:

> Greg Shah wrote:
>
> > No. In my opinion, this is not a very risky change. If you disagree, please help us understand what parts are risky.
>
> Well, if I understand #3930-11 correctly the database configuration in the directory.xml is supposed to be changed in a non-backward-compatible way. Or I should check the "new" configuration first and then revert to the old-style configuration (or vice-versa)?

Good idea. Please do this. Once we have all project configurations switched to the new approach we will remove the code that reads the old configuration.

#22 - 05/10/2021 02:18 PM - Igor Skornyakov

As I can see in the OpenEdge documentation (https://docs.progress.com/bundle/openedge-programmimg-interfaces-117/page/Auto-connect.html) the "auto-connect" has a very special meaning in 4GL:

```
The auto-connect feature uses information stored in one database to connect to a second database at runtime. T
he database that contains the connect information is the primary application database; it must be connected be
fore OpenEdge can execute the auto-connect.

OpenEdge executes the auto-connect when a precompiled procedure references data from the second database. It e
xecutes immediately prior to running the precompiled procedure. It does not work with procedures run from the
Editor or otherwise compiled on-the-fly.

If you use a CONNECT statement while you are connected to the primary database, OpenEdge merges the informatio
```

```
n in the CONNECT statement with the information in the primary database's auto-connect list. If there is a con
flict, the information in the CONNECT statement takes precedence. For more information, see the CONNECT Statem
ent reference entry in OpenEdge Development: ABL Reference.

For information on how to set up an auto-connect list in the primary database, see OpenEdge Deployment: Managi
ng ABL Applications.
```

This looks not exactly the same we've discussed here. Is this difference intentional?
Thank you.

#23 - 05/10/2021 02:44 PM - Eric Faulhaber

Igor Skornyakov wrote:

> This looks not exactly the same we've discussed here. Is this difference intentional?

I was not aware of this implementation choice, and sorry for the confusing language that we've used. Maybe we need to change the directory path name for this feature to something less confusing. Anyway, I think we can emulate this behavior with the directory configuration I've already proposed by just configuring any additional strings needed to auto-connect (in this newly discovered sense -- new for me, at least) to any database(s) which would be connected by the primary database.

Greg, do you think it is necessary to extend this proposed implementation to optionally attach "auto-connect" (in the new sense) statements to the database section of the directory?

#24 - 05/10/2021 03:22 PM - Greg Shah

At this time, we don't care about this "auto-connect" 4GL feature at all. It has nothing to do with our work and can be completely ignored.

Change the name of the directory node in the new configuration to make it less confusing.

#25 - 05/10/2021 03:27 PM - Eric Faulhaber

Greg Shah wrote:

> Change the name of the directory node in the new configuration to make it less confusing.

I propose database-connections.

#26 - 05/10/2021 03:30 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Greg Shah wrote:
>
> > Change the name of the directory node in the new configuration to make it less confusing.
>
> I propose database-connections.

Actually, my question was not about the name but if you're planning to implement auto-connect in the 4GL sense. This seems to be related to the scope of this task.

#27 - 05/10/2021 03:38 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Actually, my question was not about the name but if you're planning to implement auto-connect in the 4GL sense. This seems to be related to the scope of this task.

I thought Greg answered this in #3930-24. No, we are not.

#28 - 05/10/2021 03:40 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Actually, my question was not about the name but if you're planning to implement auto-connect in the 4GL sense. This seems to be related to the scope of this task.
>
> I thought Greg answered this in #3930-24. No, we are not.

I see, Thank you.

#29 - 05/11/2021 03:44 PM - Igor Skornyakov

It seems that at this moment a CONNECT statement support for the local databases is broken.
For example, if the hotel database is configured with load_at_startup=TRUE the CONNECT hotel correctly fails with "database" is already connected" error. However, if load_at_startup=FALSE it fails with exception

```
[05/11/2021 22:38:14 GMT+03:00] (com.goldencode.p2j.persist.Persistence:SEVERE) Error executing SQL statement:
  SELECT COUNT(*) FROM meta_user
com.goldencode.p2j.persist.PersistenceException: Failed to execute [SELECT COUNT(*) FROM meta_user]
Caused by: java.sql.SQLException: A problem occurred while trying to acquire a cached PreparedStatement in a b
ackground thread.
        at com.mchange.v2.c3p0.stmt.GooGooStatementCache.acquireStatement(GooGooStatementCache.java:571)
        at com.mchange.v2.c3p0.stmt.GooGooStatementCache.checkoutStatement(GooGooStatementCache.java:204)
        at com.mchange.v2.c3p0.impl.NewPooledConnection.checkoutStatement(NewPooledConnection.java:321)
        at com.mchange.v2.c3p0.impl.NewProxyConnection.prepareStatement(NewProxyConnection.java:553)
        at com.goldencode.p2j.persist.Persistence.getSingleSQLResult(Persistence.java:2230)
        at com.goldencode.p2j.persist.ConnectionManager.getAuthLevel(ConnectionManager.java:2064)
        at com.goldencode.p2j.util.MetadataSecurityOps.getAuthLevel(MetadataSecurityOps.java:231)
        at com.goldencode.p2j.util.SecurityOps.getDefaultUserid(SecurityOps.java:394)
        at com.goldencode.p2j.persist.ConnectionManager.putConnected(ConnectionManager.java:4047)
        at com.goldencode.p2j.persist.ConnectionManager.makeConnection(ConnectionManager.java:3551)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:3357)
        at com.goldencode.p2j.persist.ConnectionManager.connectDbImpl(ConnectionManager.java:3030)
        at com.goldencode.p2j.persist.ConnectionManager.connect_(ConnectionManager.java:668)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:571)
        at com.goldencode.testcases.meta.Connect.lambda$execute$0(Connect.java:25)
        at com.goldencode.p2j.util.Block.body(Block.java:605)
        at com.goldencode.p2j.util.BlockManager.processBody(BlockManager.java:8588)
        at com.goldencode.p2j.util.BlockManager.topLevelBlock(BlockManager.java:8257)
        at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:525)
        at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:496)
        at com.goldencode.testcases.meta.Connect.execute(Connect.java:23)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at com.goldencode.p2j.util.Utils.invoke(Utils.java:1558)
        at com.goldencode.p2j.main.StandardServer$MainInvoker.execute(StandardServer.java:2227)
        at com.goldencode.p2j.main.StandardServer.invoke(StandardServer.java:1663)
        at com.goldencode.p2j.main.StandardServer.standardEntry(StandardServer.java:582)
        at com.goldencode.p2j.main.StandardServerMethodAccess.invoke(Unknown Source)
        at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:156)
        at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:783)
        at com.goldencode.p2j.net.Conversation.block(Conversation.java:422)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:232)
        at java.lang.Thread.run(Thread.java:748)
Caused by: org.h2.jdbc.JdbcSQLSyntaxErrorException: Table "META_USER" not found; SQL statement:
```

Should I fix this in the scope of this task?
Thank you.

#30 - 05/11/2021 04:05 PM - Ovidiu Maxiniuc

Igor,
Does the database already exist? Because if there is no physical file at the moment the connection is made, H2 will automatically create a new empty one.
Is the database you want to connect already populated? Does META_USER table exist in your database?

#31 - 05/11/2021 04:09 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor,
> Does the database already exist? Because if there is no physical file at the moment the connection is made, H2 will automatically create a new empty one.
> Is the database you want to connect already populated? Does META_USER table exist in your database?

Ovidiu.
This is our hotel database (PG) which was added to the test project. It contains the META_USER table.

#32 - 05/11/2021 04:15 PM - Ovidiu Maxiniuc

I was looking at the last line of exception you posted. It says: org.h2.jdbc.JdbcSQLSyntaxErrorException.
Yet, H2 is the default database for that project, not PostgreSQL. I assume you changed the persistence option at config time.

#33 - 05/11/2021 04:16 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I was looking at the last line of exception you posted. It says: org.h2.jdbc.JdbcSQLSyntaxErrorException.
> Yet, H2 is the default database for that project, not PostgreSQL. I assume you changed the persistence option at config time.

Yes, I did. And I've tested the Hotel app with this database before.

#34 - 05/11/2021 04:25 PM - Igor Skornyakov

Igor Skornyakov wrote:

> Ovidiu Maxiniuc wrote:
>
> > I was looking at the last line of exception you posted. It says: org.h2.jdbc.JdbcSQLSyntaxErrorException.
> > Yet, H2 is the default database for that project, not PostgreSQL. I assume you changed the persistence option at config time.

> Yes, I did. And I've tested the Hotel app with this database before.

Sorry, I've overlooked that it was indeed an H2 database but I've forgotten to put it in the correct place. Please disregard.

#35 - 05/11/2021 04:34 PM - Ovidiu Maxiniuc

These things happen. I did this myself and not even once.

Just as a remainder, although you are probably already aware of this: you cannot use the database directly in same external procedure where you connect it. All access to database must be performed from a second procedure called after the connect statement. For further details, please read the note at the beginning of the CONNECT statement paragraph in the ABL Reference manual.

#36 - 05/11/2021 04:37 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Just as a remainder, although you are probably already aware of this: you cannot use the database directly in same external procedure where you connect it. All access to database must be performed from a second procedure called after the connect statement. For further details, please read the note at the beginning of the CONNECT statement paragraph in the ABL Reference manual.

Yes, I've read it recently, thanks for reminding me. BTW: how do we support this restriction? It looks weird to me.
Thank you.

#37 - 05/11/2021 05:02 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> BTW: how do we support this restriction? It looks weird to me.

I do not think we do. FWD connects the databases 'instantly'.
OE, OTOH, needs the start of an external procedure to realize whether a new database was added. I do not know why is this constraint :(. This is how they implemented it.

#38 - 05/11/2021 05:06 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor Skornyakov wrote:
>
> > BTW: how do we support this restriction? It looks weird to me.
>
> I do not think we do. FWD connects the databases 'instantly'.
> OE, OTOH, needs the start of an external procedure to realize whether a new database was added. I do not know why is this constraint :(. This is how they implemented it.

Thank you! This is great that we do not support this!

#39 - 05/11/2021 05:21 PM - Igor Skornyakov

Eric/Greg.

At this moment we initialize the meta-database for all configured local databases regardless of the load_at_startup setting.
I think we should retain this. Otherwise, it is logical to destroy/re-initialize it after all auto-connected sessions are closed and a new one is started.

It seems that it will be just an unneeded delay.

What do you think?
Thank you.

#40 - 05/11/2021 05:30 PM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> Ovidiu Maxiniuc wrote:
>
> > Igor Skornyakov wrote:
> >
> > > BTW: how do we support this restriction? It looks weird to me.
>
> > I do not think we do. FWD connects the databases 'instantly'.
>
> Thank you! This is great that we do not support this!

Sorry, I am afraid I was not very clear here. What I intended to say is that FWD does not support this currently. This breaks compatibility with OE. We

should do.

#41 - 05/11/2021 05:59 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Sorry, I am afraid I was not very clear here. What I intended to say is that FWD does not support this currently. This breaks compatibility with OE. We should do.

Oh, I see. Do have any thoughts on how it can be done? It looks very tricky to me.

#42 - 05/11/2021 06:30 PM - Greg Shah

> BTW: how do we support this restriction? It looks weird to me.

> I do not think we do. FWD connects the databases 'instantly'.

It is true that we don't implement this today. The reason is simple: we assume that all converted code is compilable in the 4GL. In other words the 4GL source code can execute in the 4GL runtime environment. The application has long since been written so that no references to a newly connected database will occur within the same 4GL source file. If they tried to put these references in, the code would not compile in the 4GL. So: code that can compile will convert without ever having this issue.

> OE, OTOH, needs the start of an external procedure to realize whether a new database was added. I do not know why is this constraint :(. This is how they implemented it.

They implement it that way because they have hard coded references to database names/structure at compile-time. CONNECT is a runtime idea. You can't have a compile time/static database reference in a file that doesn't know it is connected until runtime.

#43 - 05/11/2021 06:57 PM - Greg Shah

Actually, we really do implement this same limit. Our parser will not properly parse database references unless the conversion has been configured to "connect" a specific list of databases (and optionally mappings of aliases to databases). This is the same concept.

#44 - 05/12/2021 04:23 AM - Igor Skornyakov

Greg Shah wrote:

> Actually, we really do implement this same limit. Our parser will not properly parse database references unless the conversion has been configured to "connect" a specific list of databases (and optionally mappings of aliases to databases). This is the same concept.

I see. Thank you.

#45 - 05/12/2021 01:24 PM - Igor Skornyakov

I'm trying to understand the logic around reference counting with DatabseManager.transientDatabases. It looks pretty complicated.
One thing that I cannot understand is why deregisterDatabase is called from the ConnectionManager.closeBufferScope() while I do not see registration associated with RecordBuffer-related events.

Eric,
Did you mean this reference counting in #3930-11?
For me, it seems more logical to keep the set of transient databases associated with a session as a session attribute and instead of reference counting to keep a set of sessions using the database in a Map<Database, Set<Session>> in the DatabaseManager. This will allow to force accurate cleanup on the session end and to validate the consistency on debugging. I do not think that additional overhead will be noticeable. However. this approach obviously can't be a replacement of the DatabseManager.transientDatabases ref. counters at least because of the deregistration on ConnectionManager.closeBufferScope() mentioned above.
What do you think?
Thank you.

#46 - 05/12/2021 02:21 PM - Eric Faulhaber

The concept of "transient" databases is old and is not really something I care to preserve. It was developed back when I was thinking that there were either permanently connected databases and databases that were connected transiently, via the CONNECT statement. This thinking is now obsolete, considering how this task proposes to change the connection implementation. The reference counting mechanism therefore is probably obsolete as well.

The goal back then was (and still is now) to minimize the amount of redundant work done to set up all the infrastructure needed for a particular database to be used. With the reference counting, I was trying to minimize the times we update these data structures for databases which are "connected" and "disconnected" (in the 4GL sense). I considered these "transient", but now it is how we will handle all databases. If you have a better way in mind to manage this than the somewhat clumsy reference counting I implemented long ago, let's explore it.

I think in most cases, there will be a small number of databases which are used by a lot of the sessions in a deployed application, with perhaps some other databases that are used by fewer sessions, or less frequently at least. We need to support both cases well, with minimal startup delay for any session. If we need to use a bit more memory to store information about less frequently used databases longer, that is ok, though I would like it if these were eventually cleaned up, if not in use at all for some period of time.

#47 - 05/12/2021 02:36 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The concept of "transient" databases is old and is not really something I care to preserve. It was developed back when I was thinking that there were either permanently connected databases and databases that were connected transiently, via the CONNECT statement. This thinking is now obsolete, considering how this task proposes to change the connection implementation. The reference counting mechanism therefore is probably obsolete as well.
>
> The goal back then was (and still is now) to minimize the amount of redundant work done to set up all the infrastructure needed for a particular database to be used. With the reference counting, I was trying to minimize the times we update these data structures for databases which are "connected" and "disconnected" (in the 4GL sense). I considered these "transient", but now it is how we will handle all databases. If you have a better way in mind to manage this than the somewhat clumsy reference counting I implemented long ago, let's explore it.
>
> I think in most cases, there will be a small number of databases which are used by a lot of the sessions in a deployed application, with perhaps some other databases that are used by fewer sessions, or less frequently at least. We need to support both cases well, with minimal startup delay for any session. If we need to use a bit more memory to store information about less frequently used databases longer, that is ok, though I would like it if these were eventually cleaned up, if not in use at all for some period of time.

I see, thank you. The set of non-permanent databases used by the session will be cleaned at the session close. This set will be small. The Map<Database, Set<Session>> in the DatabaseManager can be WeakHashMap.

Is it OK to remove DatabseManager.transientDatabase() and replace it with new data structures? What about deregisterDatabase call from ConnectionManager.closeBufferScope()?

Please also review my question in #3930-39.
Thank you.

#48 - 05/12/2021 03:15 PM - Eric Faulhaber

Igor Skornyakov wrote:

> At this moment we initialize the meta-database for all configured local databases regardless of the load_at_startup setting.
> I think we should retain this. Otherwise, it is logical to destroy/re-initialize it after all auto-connected sessions are closed and a new one is started.
>
> It seems that it will be just an unneeded delay.

I agree that we want to avoid startup delay for any session, and I think this plan works for most applications, where there is one or a small number of databases used by most sessions. I am concerned about the less common case where some number of databases are used less frequently. The footprint for the metadata database for each of these can be quite large. How would you propose we handle this potential situation? We haven't encountered this as a problem yet, but I don't doubt there is an application out there which will cause a problem in this regard.

We have a similar problem with the ConversionPool, which loads the P2O ASTs for all local databases, and these can be big memory consumers. I guess this particular issue is outside the scope of this task, but it comes to mind when considering your question about the metadata.

Since we are rewriting this logic now, I think we should plan for this eventuality. I tried to address this with the database registration reference counting, but that was not a great solution. I think we should write the logic to be able to clean up and reload the information needed by a particular database (including the metadata) if no session is using it, but to make a configurable delay before this cleanup occurs. It would default to -1 (never clean it up); 0 would mean clean it up immediately if no session is using it; a positive number would be the number of seconds to wait before cleaning it up if no session is using it. I eventually would like to allow the ConversionPool to leverage this feature for its AST instances, but again, that is beyond the scope of this task.

#49 - 05/12/2021 03:18 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Since we are rewriting this logic now, I think we should plan for this eventuality. I tried to address this with the database registration reference counting, but that was not a great solution. I think we should write the logic to be able to clean up and reload the information needed by a particular database (including the metadata) if no session is using it, but to make a configurable delay before this cleanup occurs. It would default to -1 (never clean it up); 0 would mean clean it up immediately if no session is using it; a positive number would be the number of seconds to wait before cleaning it up if no session is using it.

I like this idea. Will implement this logic.

#50 - 05/14/2021 06:13 PM - Igor Skornyakov

There is a lot of commented code in the DatabaseManager. This makes reading and understanding the code difficult. Do we really need these fragments or they can be removed?
Thank you.

#51 - 05/15/2021 06:19 PM - Igor Skornyakov

When I'm trying to connect to a database in the session start (in my case it was a connection from the application server) I get an exception:

```
[05/16/2021 01:07:31 GMT+03:00] (LogicalTerminal:WARNING) No client parameters are set: calling the client-sid
e to determine the driver type is expensive!
[05/16/2021 01:07:31 GMT+03:00] (ErrorManager:SEVERE) {0000000C:0000001D:k32vu5ru93h3k19l} stack trace follows
com.goldencode.p2j.persist.PersistenceException: Error populating metadata database local/custdb/meta
Caused by: java.lang.RuntimeException: Unresolvable remote export public abstract boolean com.goldencode.p2j.u
i.ClientExports.isChui().
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.obtainRoutingKey(RemoteObject.java:1580)
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.invokeCore(RemoteObject.java:1464)
        at com.goldencode.p2j.net.InvocationStub.invoke(InvocationStub.java:145)
        at com.sun.proxy.$Proxy10.isChui(Unknown Source)
        at com.goldencode.p2j.ui.LogicalTerminal.isChui(LogicalTerminal.java:9264)
        at com.goldencode.p2j.ui.LogicalTerminal.init(LogicalTerminal.java:1411)
        at com.goldencode.p2j.security.ContextLocal.initializeValue(ContextLocal.java:655)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:494)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.ui.LogicalTerminal.locate(LogicalTerminal.java:11772)
        at com.goldencode.p2j.ui.LogicalTerminal.lambda$initialize$7(LogicalTerminal.java:13048)
```

```
        at com.goldencode.p2j.util.TransactionManager$ContextContainer.obtain(TransactionManager.java:10525)
        at com.goldencode.p2j.util.TransactionManager.getTransactionHelper(TransactionManager.java:1726)
        at com.goldencode.p2j.util.ProcedureManager$WorkArea.init(ProcedureManager.java:4436)
        at com.goldencode.p2j.security.ContextLocal.initializeValue(ContextLocal.java:655)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:494)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.util.ProcedureManager.getProcedureHelper(ProcedureManager.java:1026)
        at com.goldencode.p2j.persist.BufferManager.<init>(BufferManager.java:644)
        at com.goldencode.p2j.persist.BufferManager.<init>(BufferManager.java:514)
        at com.goldencode.p2j.persist.BufferManager$1.initialValue(BufferManager.java:528)
        at com.goldencode.p2j.persist.BufferManager$1.initialValue(BufferManager.java:526)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:492)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.persist.BufferManager.get(BufferManager.java:718)
        at com.goldencode.p2j.persist.Persistence$Context.<init>(Persistence.java:3979)
        at com.goldencode.p2j.persist.Persistence$Context.<init>(Persistence.java:3973)
        at com.goldencode.p2j.persist.Persistence$1.initialValue(Persistence.java:698)
        at com.goldencode.p2j.persist.Persistence$1.initialValue(Persistence.java:695)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:492)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.persist.Persistence.beginTransaction(Persistence.java:3026)
        at com.goldencode.p2j.persist.Persistence.beginTransaction(Persistence.java:2766)
        at com.goldencode.p2j.persist.meta.MetadataManager.populateDatabase(MetadataManager.java:650)
        at com.goldencode.p2j.persist.DatabaseManager.initMetaDb(DatabaseManager.java:1298)
        at com.goldencode.p2j.persist.DatabaseManager.activateMetaDb(DatabaseManager.java:1219)
        at com.goldencode.p2j.persist.DatabaseManager.lambda$activate$2(DatabaseManager.java:1259)
        at com.goldencode.p2j.persist.DatabaseManager.activateAndRegister(DatabaseManager.java:1281)
        at com.goldencode.p2j.persist.DatabaseManager.activate(DatabaseManager.java:1258)
        at com.goldencode.p2j.persist.ConnectionManager.makeConnection(ConnectionManager.java:3544)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:3357)
        at com.goldencode.p2j.persist.ConnectionManager.connectDbImpl(ConnectionManager.java:3030)
        at com.goldencode.p2j.persist.ConnectionManager.connect_(ConnectionManager.java:668)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:571)
        at com.goldencode.p2j.persist.DatabaseManager$1.initialize(DatabaseManager.java:446)
        at com.goldencode.p2j.net.BaseSession.sendInitializationEvent(BaseSession.java:577)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:225)
        at java.lang.Thread.run(Thread.java:748)
```

Initially, it was an NPE in the ProcedureManager.processResource and I've added a check for null at line 703.

For me, it looks strange that we're calling LogicalTerminal at this place.
Any suggestions?
Thank you.

#52 - 05/19/2021 12:06 PM - Eric Faulhaber

Igor Skornyakov wrote:

> There is a lot of commented code in the DatabaseManager. This makes reading and understanding the code difficult. Do we really need these fragments or they can be removed?
> Thank you.

These were left over from the shift from Hibernate. I think at this point, they can be removed.

#53 - 05/19/2021 12:08 PM - Eric Faulhaber

Igor Skornyakov wrote:

> When I'm trying to connect to a database in the session start (in my case it was a connection from the application server) I get an exception:
> [...]
>
> Initially, it was an NPE in the ProcedureManager.processResource and I've added a check for null at line 703.
>
> For me, it looks strange that we're calling LogicalTerminal at this place.
> Any suggestions?

This does look strange, and unfortunately I cannot offer much insight as to why LogicalTerminal would be involved. It is not clear why this is happening now and was not before.

#54 - 05/19/2021 12:15 PM - Igor Skornyakov

Eric Faulhaber wrote:

> These were left over from the shift from Hibernate. I think at this point, they can be removed.

Thank you.

#55 - 05/19/2021 12:15 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > When I'm trying to connect to a database in the session start (in my case it was a connection from the application server) I get an exception:
> > [...]
> >
> > Initially, it was an NPE in the ProcedureManager.processResource and I've added a check for null at line 703.
> >
> > For me, it looks strange that we're calling LogicalTerminal at this place.
> > Any suggestions?
>
> This does look strange, and unfortunately I cannot offer much insight as to why LogicalTerminal would be involved. It is not clear why this is happening now and was not before.

I see, thank you. Analyzing.

#56 - 05/19/2021 12:23 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> > These were left over from the shift from Hibernate. I think at this point, they can be removed.
>
> Thank you.

I should get a second opinion from Ovidiu...

Ovidiu, we do have some TODOs like "implement without Hibernate". I THINK by now we have addressed these, or at least we have found that they are no longer necessary. Do you have any concerns dropping this commented code?

#57 - 05/19/2021 01:00 PM - Ovidiu Maxiniuc

I agree. The disabled code related to Hibernate can be dropped at this moment.

#58 - 05/19/2021 09:49 PM - Eric Faulhaber

From Igor via email:

At this moment I've implemented most of the functionality except timed meta-database destroy. It took a significant refactoring of the DatabaseManager. The CONNECT to a local database from the application works OK, but I've encountered an unexpected problem described in #3930-51 on the session start. The timed destroy and rebuild of the meta database should not take more than a day.

#59 - 05/20/2021 11:05 AM - Igor Skornyakov

I've implemented the session-local databases' activation in the SessionListener.initialize() method of the SessionListener instance defined in the DatabaseManager.
It looks that it is the wrong place since the Persistence.context is not yet initialized. If I set the database-connections node from default to the appserver_process I again get an exception that is different from the one described in #3930-51 (again, I have to add a null check to the ProcedureManager to avoid NPE).

```
com.goldencode.p2j.persist.PersistenceException: Error populating metadata database local/custdb/meta
Caused by: java.lang.RuntimeException: Unresolvable remote export public abstract boolean com.goldencode.p2j.u
i.ClientExports.isChui().
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.obtainRoutingKey(RemoteObject.java:1580)
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.invokeCore(RemoteObject.java:1464)
        at com.goldencode.p2j.net.InvocationStub.invoke(InvocationStub.java:145)
        at com.sun.proxy.$Proxy10.isChui(Unknown Source)
        at com.goldencode.p2j.ui.LogicalTerminal.isChui(LogicalTerminal.java:9273)
        at com.goldencode.p2j.ui.LogicalTerminal.init(LogicalTerminal.java:1412)
        at com.goldencode.p2j.security.ContextLocal.initializeValue(ContextLocal.java:655)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:494)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.ui.LogicalTerminal.locate(LogicalTerminal.java:11781)
        at com.goldencode.p2j.ui.LogicalTerminal.lambda$initialize$7(LogicalTerminal.java:13057)
        at com.goldencode.p2j.util.TransactionManager$ContextContainer.obtain(TransactionManager.java:10525)
        at com.goldencode.p2j.util.TransactionManager.getTransactionHelper(TransactionManager.java:1726)
        at com.goldencode.p2j.util.ProcedureManager$WorkArea.init(ProcedureManager.java:4436)
        at com.goldencode.p2j.security.ContextLocal.initializeValue(ContextLocal.java:655)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:494)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.util.ProcedureManager.getProcedureHelper(ProcedureManager.java:1026)
        at com.goldencode.p2j.persist.BufferManager.<init>(BufferManager.java:644)
        at com.goldencode.p2j.persist.BufferManager.<init>(BufferManager.java:514)
        at com.goldencode.p2j.persist.BufferManager$1.initialValue(BufferManager.java:528)
        at com.goldencode.p2j.persist.BufferManager$1.initialValue(BufferManager.java:526)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:492)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.persist.BufferManager.get(BufferManager.java:718)
        at com.goldencode.p2j.persist.Persistence$Context.<init>(Persistence.java:3979)
        at com.goldencode.p2j.persist.Persistence$Context.<init>(Persistence.java:3973)
        at com.goldencode.p2j.persist.Persistence$1.initialValue(Persistence.java:698)
        at com.goldencode.p2j.persist.Persistence$1.initialValue(Persistence.java:695)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:492)
        at com.goldencode.p2j.security.ContextLocal.get(ContextLocal.java:430)
        at com.goldencode.p2j.persist.Persistence.beginTransaction(Persistence.java:3026)
        at com.goldencode.p2j.persist.Persistence.beginTransaction(Persistence.java:2766)
        at com.goldencode.p2j.persist.meta.MetadataManager.populateDatabase(MetadataManager.java:650)
        at com.goldencode.p2j.persist.DatabaseManager.initMetaDb(DatabaseManager.java:1298)
        at com.goldencode.p2j.persist.DatabaseManager.activateMetaDb(DatabaseManager.java:1219)
        at com.goldencode.p2j.persist.DatabaseManager.lambda$activate$2(DatabaseManager.java:1259)
        at com.goldencode.p2j.persist.DatabaseManager.activateAndRegister(DatabaseManager.java:1281)
        at com.goldencode.p2j.persist.DatabaseManager.activate(DatabaseManager.java:1258)
        at com.goldencode.p2j.persist.ConnectionManager.makeConnection(ConnectionManager.java:3544)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:3357)
        at com.goldencode.p2j.persist.ConnectionManager.connectDbImpl(ConnectionManager.java:3030)
        at com.goldencode.p2j.persist.ConnectionManager.connect_(ConnectionManager.java:668)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:571)
        at com.goldencode.p2j.persist.DatabaseManager$1.initialize(DatabaseManager.java:446)
```

```
        at com.goldencode.p2j.net.BaseSession.sendInitializationEvent(BaseSession.java:577)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:225)
        at java.lang.Thread.run(Thread.java:748)
```

From the other side, I do not see another place to retrieve the correct values via the DirectoryManager.getInstance().getStrings(Directory.ID_RELATIVE, CFG_AUTO_CONNECT) call.

Is it possible to temporarily switch to the bootstrap SecurytyContext?

Thank you.

#60 - 05/20/2021 12:21 PM - Eric Faulhaber

It is hard to know what is going on without seeing the changes you've made. If they are mostly isolated to DatabaseManager and ConnectionManager, please post a patch of these two files so I can review. If they are broader, please create a branch. Although I wanted to avoid a separate branch, we can't commit your current changes, since clearly they will cause problems in their current state.

#61 - 05/20/2021 12:28 PM - Igor Skornyakov

- File DatabaseManager.diff added

- File ConnectionManager.diff added

- File Session.diff added

- File RouterSessionManager.diff added

Eric Faulhaber wrote:

> It is hard to know what is going on without seeing the changes you've made. If they are mostly isolated to DatabaseManager and ConnectionManager, please post a patch of these two files so I can review. If they are broader, please create a branch. Although I wanted to avoid a separate branch, we can't commit your current changes, since clearly they will cause problems in their current state.

Please find the diff files for the most important changes. Sorry, no Javadocs yet.
Thank you,
Igor.

#62 - 05/20/2021 12:42 PM - Greg Shah

There should not be any changes in the com.goldencode.p2j.net package. It is not appropriate to have database-specific behavior in that package.

#63 - 05/20/2021 12:43 PM - Igor Skornyakov

Greg Shah wrote:

> There should not be any changes in the com.goldencode.p2j.net package. It is not appropriate to have database-specific behavior in that package.

I see. Will be re-worked. Thank you.

#64 - 05/20/2021 12:54 PM - Igor Skornyakov

- File BaseSession.diff added

Greg Shah wrote:

> There should not be any changes in the com.goldencode.p2j.net package. It is not appropriate to have database-specific behavior in that package.

I'm sorry. How can we avoid any changes at least in the RouterSessionManager if the databases are supposed to be connected on the network session start? Or I misunderstand something?
The changes in the BaseSession (see attachment) are general-purpose to make it possible to use Session instances as keys in maps. Is it acceptable?
Thank you.

#65 - 05/20/2021 01:10 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Greg Shah wrote:
>
> > There should not be any changes in the com.goldencode.p2j.net package. It is not appropriate to have database-specific behavior in that package.
>
> I'm sorry. How can we avoid any changes at least in the RouterSessionManager if the databases are supposed to be connected on the network session start? Or I misunderstand something?

I think you were on the right track with using the SessionListener interface. If we find bugs with that approach, then we will need to fix them. But the point is that the net package provides low-level network services that are not specific to any other module. If we find ourselves needing to add import statements for modules which are supposed to be clients of these services, then we have a separation of concerns issue.

#66 - 05/20/2021 01:11 PM - Igor Skornyakov

Eric Faulhaber wrote:

> I think you were on the right track with using the SessionListener interface. If we find bugs with that approach, then we will need to fix them. But the point is that the net package provides low-level network services that are not specific to any other module. If we find ourselves needing to add import statements for modules which are supposed to be clients of these services, then we have a separation of concerns issue.

I see. Thank you.

#67 - 05/20/2021 01:18 PM - Eric Faulhaber

The way I would approach this would be to:

1. Take a working, test configuration which currently connects a database on startup.
2. Write and convert the simplest possible 4GL test case (program 1) which uses that database. This should work with no code changes at all.
3. Change the configuration to set load_at_startup from true to false.
4. Write and convert a simple 4GL test case (program 2) which:
    - connects to that database using an explicit CONNECT statement;
    - runs program 1;
    - disconnects from that database using an explicit DISCONNECT statement;
    - so far, we still have made no changes to the existing FWD code base; all this existing functionality should work.
5. Add the proposed changes to the directory to use the same CONNECT statement string in the new database-connections section of the directory.

Now, the goal would be run program 1 without program 2, doing the equivalent of the CONNECT/DISCONNECT statements in an implementation of a SessionListener interface, using the information in the database-connections configuration. I think the first pass should use the ConnectionManager.connect method with as little change as possible. If that works, the next step would be to refactor the DatabaseManager.registerDatabase and related code to drop the load_at_startup logic and to use a better mechanism than the current reference counting.

I'm sorry if this covers ground you've already covered, but hopefully there's something helpful there.

#68 - 05/20/2021 01:18 PM - Greg Shah

> How can we avoid any changes at least in the RouterSessionManager if the databases are supposed to be connected on the network session start? Or I misunderstand something?

com.goldencode.p2j.net, including RouterSessionManager is a generic secure networking transport. It is not specific to FWD and it should have nothing related to the 4GL compatibility.

Instead of looking at the session level stuff (which is generic), please look at the places in the code where we execute 4GL converted code. For interactive sessions and batch sessions, this is in StandardServer.standardEntry().

Constantin: Is there a different entry point for appserver agents?

> The changes in the BaseSession (see attachment) are general-purpose to make it possible to use Session instances as keys in maps. Is it acceptable?

I don't object to the non-database changes in BaseSession. But I wonder if we need such a thing for this case. Can't we just store a list of connected databases in a ContextLocal and then we could use the normal ContextLocal.cleanup() method to do the implicit disconnect.

#69 - 05/20/2021 01:21 PM - Eric Faulhaber

Greg Shah wrote:

> Instead of looking at the session level stuff (which is generic), please look at the places in the code where we execute 4GL converted code. For interactive sessions and batch sessions, this is in StandardServer.standardEntry().

Igor, I would follow Greg's advice here instead of my earlier comments on the SessionListener use. I haven't done much work in that specific area of the code, so he would know better.

#70 - 05/20/2021 01:24 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: Is there a different entry point for appserver agents?

In case of appserver Agents, StandardServer.standardEntry calls AppServerManager.startAppServer() which ends up starting the agent via Agent.listen(). If you want to execute code **before** the agent executes its startup procedure and started listening for tasks, do it in Agent.prepare - this ensures that if the Agent is reset, the state will be restored.

#71 - 05/20/2021 01:26 PM - Igor Skornyakov

Greg Shah wrote:

> I don't object to the non-database changes in BaseSession. But I wonder if we need such a thing for this case. Can't we just store a list of connected databases in a ContextLocal and then we could use the normal ContextLocal.cleanup() method to do the implicit disconnect.

The idea was to replace the existing logic based on the reference counters with another one that is easier to debug. For this purpose, I suggest keeping the set of sessions using a particular non-permanent database. But for this, I need a correct hashCode/equals implementation for Session.

#72 - 05/20/2021 01:27 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, I would follow Greg's advice here instead of my earlier comments on the SessionListener use. I haven't done much work in that specific area of the code, so he would know better.

Got it. Thank you!

#73 - 05/20/2021 01:29 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > Constantin: Is there a different entry point for appserver agents?
>
> In case of appserver Agents, StandardServer.standardEntry calls AppServerManager.startAppServer() which ends up starting the agent via Agent.listen(). If you want to execute code **before** the agent executes its startup procedure and started listening for tasks, do it in Agent.prepare - this ensures that if the Agent is reset, the state will be restored.

Thank you, Constantin.
Eric: is it correct that the entry point mentioned by Constantin is the only one where the non-permanent databases should be auto-connected?
Thank you.

#74 - 05/20/2021 01:48 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric: is it correct that the entry point mentioned by Constantin is the only one where the non-permanent databases should be auto-connected?

What do you mean by "non-permanent" databases in this context?

#75 - 05/20/2021 01:51 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Eric: is it correct that the entry point mentioned by Constantin is the only one where the non-permanent databases should be auto-connected?
>
> What do you mean by "non-permanent" databases in this context?

I mean databases specified in the "database-connections" node(s).

#76 - 05/20/2021 01:57 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> > Igor Skornyakov wrote:
> >
> > > Eric: is it correct that the entry point mentioned by Constantin is the only one where the non-permanent databases should be auto-connected?
> >
> > What do you mean by "non-permanent" databases in this context?
>
> I mean databases specified in the "database-connections" node(s).

OK, I was confused, because this implies there are "permanent" databases, but I don't know what those would be. All databases which have to be used by an application must be configured in some version of the database-connections nodes, whether as the default for all accounts, or in more specific accounts. I would not use the term "non-permanent" databases in the code or documentation, as I think this will cause others confusion as well.

But to your question: Constantin was answering in the context of appserver agents. For non-appserver configurations, please use the entry point Greg mentioned.

#77 - 05/20/2021 01:59 PM - Igor Skornyakov

Eric Faulhaber wrote:

> But to your question: Constantin was answering in the context of appserver agents. For non-appserver configurations, please use the entry point Greg mentioned.

I see. Thank you.

#78 - 05/20/2021 02:00 PM - Eric Faulhaber

Eric Faulhaber wrote:

> But to your question: Constantin was answering in the context of appserver agents. For non-appserver configurations, please use the entry point Greg mentioned.

To clarify, the connections must happen in both places.

Constantin, where would the disconnect occur? I don't think ContextLocal.cleanup would work for appserver agents, is that right?

#79 - 05/20/2021 02:00 PM - Greg Shah

Constantin: How do we avoid the StandardServer.standardEntry() connection logic when it is called for an appserver agent?

#80 - 05/20/2021 02:17 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Constantin, where would the disconnect occur? I don't think ContextLocal.cleanup would work for appserver agents, is that right?

The Agent's context can be reset explicitly, or is 'reset' implicitly when the Agent gets shut down - both use the normal ContextLocal.cleanup approach. See Agent$ResetContextCommand for explicit reset.

#81 - 05/20/2021 02:19 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: How do we avoid the StandardServer.standardEntry() connection logic when it is called for an appserver agent?

No special logic needed, just do it after this code:

```
public static boolean standardEntry()
{
   ClientParameters params = localParams.get();
   // this is an app server and needs its own management
   if (AppServerManager.startAppServer())
   {
      return false;
   }
```

If a FWD client is determined to be an appserver Agent, then standardEntry will not execute the 'normal client' code.

#82 - 05/20/2021 05:29 PM - Igor Skornyakov

Initialization of the transient auto-connected databases in StandardServer.standardEntry() (before startupProc processing and Agent.prepare() (before AppServerManager.registerAgent) works.
Will make more tests tomorrow.

#83 - 05/21/2021 02:01 AM - Constantin Asofiei

Igor, as I mentioned - Agents can reset their context (for State-reset mode) - in this case, I think these databases will be disconnected via ContextLocal.cleanup. So they need to be reconnected in Agent.prepare().

#84 - 05/21/2021 03:04 AM - Igor Skornyakov

Constantin Asofiei wrote:

> Igor, as I mentioned - Agents can reset their context (for State-reset mode) - in this case, I think these databases will be disconnected via ContextLocal.cleanup. So they need to be reconnected in Agent.prepare().

Constantin.
At this moment I do not keep the auto-connected databases in the ContextLocal. Anyway, how can I simulate agent reset in the test?
Thank you.

#85 - 05/21/2021 06:51 AM - Igor Skornyakov

I'm confused with the following code in the ConnectionManager.disconnectImmediately():

```
      Session session = sessions.remove(database);
      if (session != null)
      {
         // Close the session only if is not used by the dirty share manager
         // for this database.
         if (session.isRunning() && !DirtyShareFactory.isUsedByManager(database, session))
         {
            session.terminate();
         }
      }
```

It looks that we decide if the session should be terminated based on the state of a particular database. What if there is more than one connected database?
Thank you.

#86 - 05/21/2021 12:40 PM - Eric Faulhaber

IIRC, for remote connections, we create one virtual session between server A and server B, regardless of how many user contexts from A actually use the A->B connection. The first user context from server A to connect establishes the virtual session to server B, and the last user context from server A to disconnect tears down the virtual session to server B.

At least, this is how it is supposed to work. I don't think we have done much testing of having multiple user contexts over a virtual session since the early days of this code being added, and even then I don't remember if we tested more than one. The dirty share manager session mapping was added later by Stanislav (but still some time ago); I don't recall what the need was behind that.

> What if there is more than one connected database?

As long as there is at least one connected database between server A and B, the virtual session must remain up. The way this code is now doesn't look quite right in that regard.

#87 - 05/21/2021 12:56 PM - Igor Skornyakov

Eric Faulhaber wrote:

> IIRC, for remote connections, we create one virtual session between server A and server B, regardless of how many user contexts from A actually use the A->B connection. The first user context from server A to connect establishes the virtual session to server B, and the last user context from server A to disconnect tears down the virtual session to server B.
>
> At least, this is how it is supposed to work. I don't think we have done much testing of having multiple user contexts over a virtual session since the early days of this code being added, and even then I don't remember if we tested more than one. The dirty share manager session mapping was added later by Stanislav (but still some time ago); I don't recall what the need was behind that.
>
> > What if there is more than one connected database?
>
> As long as there is at least one connected database between server A and B, the virtual session must remain up. The way this code is now doesn't look quite right in that regard.

As far as I understand the same logic is used for local databases as well. Should I fix it now?
Thank you.

#88 - 05/21/2021 01:02 PM - Eric Faulhaber

Igor Skornyakov wrote:

> As far as I understand the same logic is used for local databases as well. Should I fix it now?

Maybe I'm misunderstanding your question...a local database does not use a network session. The network sessions only come into play for remote database connections which need a virtual session between servers. So, to what logic are you referring?

#89 - 05/21/2021 01:05 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:

> As far as I understand the same logic is used for local databases as well. Should I fix it now?

> Maybe I'm misunderstanding your question...a local database does not use a network session. The network sessions only come into play for remote database connections which need a virtual session between servers. So, to what logic are you referring?

Well, maybe I was confused by the code which I see right before the one I've quoted before:

```
if (!database.isLocal())
{
   DatabaseManager.deregisterDatabase(database);
}
```

Sorry. Let me make some debugging.

#90 - 05/21/2021 01:12 PM - Eric Faulhaber

The current deregisterDatabase code is just about cleaning up local data structures when a database connection ends (whether it is local or remote is not considered at this level).

Note that in the snippet of code you posted from disconnectImmediately, the session.terminate() call is bypassed if session comes back null from the sessions.remove(database) call, which it should for local databases.

#91 - 05/21/2021 01:13 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The current deregisterDatabase code is just about cleaning up local data structures when a database connection ends (whether it is local or remote is not considered at this level).
>
> Note that in the snippet of code you posted from disconnectImmediately, the session.terminate() call is bypassed if session comes back null from the sessions.remove(database) call, which it should for local databases.

Oh, I see. Thank you!

#92 - 05/21/2021 01:33 PM - Eric Faulhaber

That being said, I'm not sure that code looks right now...what if we had two database keys mapped to the same Session object (i.e. we had connected databases 1 and 2 in server B from server A, using a single, virtual network session). As we disconnect database 1, the sessions.remove(database) call will get the Session object used for both database 1 and database 2, and terminate the virtual session from server A->B. But now the database 2 connection between A->B is dead as well.

Maybe Stanislav's additional logic about the dirty share manager check saves us? But it seems the database->session mapping at the ConnectionManager level is flawed, at least the way it is used in disconnectImmediately.

#93 - 05/21/2021 01:39 PM - Igor Skornyakov

Eric Faulhaber wrote:

> That being said, I'm not sure that code looks right now...what if we had two database keys mapped to the same Session object (i.e. we had connected databases 1 and 2 in server B from server A, using a single, virtual network session). As we disconnect database 1, the sessions.remove(database) call will get the Session object used for both database 1 and database 2, and terminate the virtual session from server A->B. But now the database 2 connection between A->B is dead as well.
>
> Maybe Stanislav's additional logic about the dirty share manager check saves us? But it seems the database->session mapping at the ConnectionManager level is flawed, at least the way it is used in disconnectImmediately.

I will try to test it after the main logic related to this task will be finished.

#94 - 05/25/2021 03:53 PM - Igor Skornyakov

The functionality is implemented and tested with the test app. Will test with the real customer app.

Committed to the 3930a (based on 3821c) revision 12450.
Please review.
Thank you.

#95 - 05/26/2021 04:59 AM - Igor Skornyakov

There is a regression caused by my changes in a large customer application. Analyzing.

#96 - 05/26/2021 07:23 AM - Igor Skornyakov

Igor Skornyakov wrote:

> There is a regression caused by my changes in a large customer application. Analyzing.

The regression fixed. The "smoke test" of the large customer app. passed.

Committed to 3930a/12451.

#97 - 05/26/2021 01:52 PM - Constantin Asofiei

Igor, does the auto-connect directory configuration support the 'project token' approach? Here I mean that the client can specify an application token via the client:project:token=prj config at its startup, and the i.e. auto-connect.prj will have priority over (and will be used instead of) the auto-connect node.

The main reason is this: you have a client application and a server application, and the client doesn't use the database at all - only the server application uses it. But both run in the same FWD server, and the client must not see any database connected, if you do a i.e. CONNECTED("p2j_test").

#98 - 05/26/2021 02:15 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Igor, does the auto-connect directory configuration support the 'project token' approach? Here I mean that the client can specify an application token via the client:project:token=prj config at its startup, and the i.e. auto-connect.prj will have priority over (and will be used instead of) the auto-connect node.
>
> The main reason is this: you have a client application and a server application, and the client doesn't use the database at all - only the server application uses it. But both run in the same FWD server, and the client must not see any database connected, if you do a i.e. CONNECTED("p2j_test").

Eric,
We have discussed it before (see e.g. #3930-13, #3930-17). I understood that it is sufficient to use the directory lookup you've described and the correct set will be retrieved automatically. Please correct me if this is not correct and additional efforts are required,

#99 - 05/26/2021 02:25 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Constantin Asofiei wrote:
>
> > Igor, does the auto-connect directory configuration support the 'project token' approach? Here I mean that the client can specify an application token via the client:project:token=prj config at its startup, and the i.e. auto-connect.prj will have priority over (and will be used instead of) the auto-connect node.
> >
> > The main reason is this: you have a client application and a server application, and the client doesn't use the database at all - only the server application uses it. But both run in the same FWD server, and the client must not see any database connected, if you do a i.e. CONNECTED("p2j_test").
>
> Eric,
> We have discussed it before (see e.g. #3930-13, #3930-17). I understood that it is sufficient to use the directory lookup you've described and the correct set will be retrieved automatically. Please correct me if this is not correct and additional efforts are required,

I had not considered this feature. Constantin, the implementation so far only reads from the directory. What needs to be done to implement this? Can you point to a good example in the runtime code? It seems like this should be integrated into the directory support code to get it "for free" when reading from the directory, with a way for the developer to allow/disallow a project token override.

#100 - 05/26/2021 02:28 PM - Constantin Asofiei

The token support is already in FWD, and the directory utility APIs already have this (at least the Utils.getDirectoryNode APIs).

Igor, please test it, this way:

- change directory.xml to add an empty auto-connect.prj node
- create a simple program which has a find first book. and convert it.
- run this program, but pass the client:project:token=prj at the client.sh command.

If everything works OK, then the program will show an error that the database is not connected.

#101 - 05/26/2021 02:30 PM - Greg Shah

> I had not considered this feature. Constantin, the implementation so far only reads from the directory. What needs to be done to implement this?

We considered this explicitly above. Nothing needs to be done. I think Igor is correct. Reading from the directory using the proper utility functions will provide this automatically.

#102 - 05/26/2021 02:38 PM - Igor Skornyakov

Constantin Asofiei wrote:

> The token support is already in FWD, and the directory utility APIs already have this (at least the Utils.getDirectoryNode APIs).
>
> Igor, please test it, this way:
>
> - change directory.xml to add an empty auto-connect.prj node
> - create a simple program which has a find first book. and convert it.
> - run this program, but pass the client:project:token=prj at the client.sh command.
>
> If everything works OK, then the program will show an error that the database is not connected.

Constantin. Where exactly I should add this node?
Thank you.

#103 - 05/26/2021 02:40 PM - Constantin Asofiei

Igor Skornyakov wrote:

> Constantin. Where exactly I should add this node?

As a sibling of the auto-connect node.

#104 - 05/26/2021 02:44 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Igor Skornyakov wrote:
>
> > Constantin. Where exactly I should add this node?
>
> As a sibling of the auto-connect node.

There is no such node in our standard directory.xml. At this moment I use "database-connections" node as Eric suggested (initially the name was "auto-connect" but it was changed.

#105 - 05/26/2021 02:47 PM - Constantin Asofiei

Sorry, I didn't notice that. Use database-connections.prj.

#106 - 05/26/2021 02:50 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Sorry, I didn't notice that. Use database-connections.prj.

I see, Thank you. At this moment I'm working on #4972 and prefer not to distract. I will test later.

#107 - 05/26/2021 03:12 PM - Eric Faulhaber

I am starting my code review. I will have more detailed feedback later, but on a quick first pass, I noticed a few problems:

- The concept of "permanent" and "transient" databases, and the use of the load_at_startup configuration option have been preserved. All this should go away. There is no longer any database which should be loaded/initialized at server startup, other than the temp-table database, which is implicitly available always (unless persistence is entirely disabled). Any initialization should arise from sessions needing to connect to a database using the new configuration.
- The implementation uses SessionListener for cleanup purposes and adds references to network sessions in the DatabaseManager, which I'd like to avoid. Such references should only be in the ConnectionMangaer, and then only for the purpose of remote database connections over server-to-server virtual sessions. We discussed using ContextLocal for the cleanup instead of SessionListener, which I think would obviate the need for this more complicated implementation. Did you find a problem with the ContextLocal approach?
- All new static variables added to the DatabaseManager are all-caps with underscores, instead of camelCased. All-caps with underscores is only for constants (or write-once/read-many variables, which essentially are referenced as constants). Even log was changed to LOG, which in fact I am trying to change in the other direction (older code uses LOG, but it should be log, as this is not a constant).
- The DMOSignatureHelper fix should be in 3821c, not in this branch (other than after a rebase).

#108 - 05/26/2021 03:16 PM - Igor Skornyakov

Eric Faulhaber wrote:

> I am starting my code review. I will have more detailed feedback later, but on a quick first pass, I noticed a few problems:
>
> - The concept of "permanent" and "transient" databases, and the use of the load_at_startup configuration option have been preserved. All this should go away. There is no longer any database which should be loaded/initialized at server startup, other than the temp-table database, which is implicitly available always (unless persistence is entirely disabled). Any initialization should arise from sessions needing to connect to a database using the new configuration.
> - The implementation uses SessionListener for cleanup purposes and adds references to network sessions in the DatabaseManager, which I'd like to avoid. Such references should only be in the ConnectionMangaer, and then only for the purpose of remote database connections over server-to-server virtual sessions. We discussed using ContextLocal for the cleanup instead of SessionListener, which I think would obviate the need for this more complicated implementation. Did you find a problem with the ContextLocal approach?
> - All new static variables added to the DatabaseManager are all-caps with underscores, instead of camelCased. All-caps with underscores is only for constants (or write-once/read-many variables, which essentially are referenced as constants). Even log was changed to LOG, which in fact I am trying to change in the other direction (older code uses LOG, but it should be log, as this is not a constant).
> - The DMOSignatureHelper fix should be in 3821c, not in this branch (other than after a rebase).

I see. Thank you. Will be addressed.

#109 - 05/26/2021 03:27 PM - Igor Skornyakov

Eric Faulhaber wrote:

- The implementation uses SessionListener for cleanup purposes and adds references to network sessions in the DatabaseManager, which I'd like to avoid. Such references should only be in the ConnectionMangaer, and then only for the purpose of remote database connections over server-to-server virtual sessions. We discussed using ContextLocal for the cleanup instead of SessionListener, which I think would obviate the need for this more complicated implementation. Did you find a problem with the ContextLocal approach?

I prefer to avoid reference counters as it is a fragile technique. With ContextLocal I will have no way to figure that the database is no longer in use and can be deactivated.

#110 - 05/26/2021 03:34 PM - Eric Faulhaber

Igor Skornyakov wrote:

> I prefer to avoid reference counters as it is a fragile technique. With ContextLocal I will have no way to figure that the database is no longer in use and can be deactivated.

Why not just maintain a set of session IDs (see SecurityManager.getSessionId())? Add on connect, remove on disconnect.

#111 - 05/26/2021 03:42 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > I prefer to avoid reference counters as it is a fragile technique. With ContextLocal I will have no way to figure that the database is no longer in use and can be deactivated.
>
> Why not just maintain a set of session IDs (see SecurityManager.getSessionId())? Add on connect, remove on disconnect.

OK, I will try to re-work. Thank you.

#112 - 05/26/2021 04:27 PM - Igor Skornyakov

Eric Faulhaber wrote:

- The concept of "permanent" and "transient" databases, and the use of the load_at_startup configuration option have been preserved. All this should go away. There is no longer any database which should be loaded/initialized at server startup, other than the temp-table database, which is implicitly available always (unless persistence is entirely disabled). Any initialization should arise from sessions needing to connect to a database using the new configuration.

Sorry Eric, but this was your suggestion (if I understood correctly #3930-11). Are you sure that you want to completely remove the "permanent" databases support right now? Apart from backward compatibility, this can slightly slow down the application startup. We can use load_at_startup=FALSE by default for now.
What do you think?

#113 - 05/26/2021 09:35 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> - The concept of "permanent" and "transient" databases, and the use of the load_at_startup configuration option have been preserved. All this should go away. There is no longer any database which should be loaded/initialized at server startup, other than the temp-table database, which is implicitly available always (unless persistence is entirely disabled). Any initialization should arise from sessions needing to connect to a database using the new configuration.
>
> Sorry Eric, but this was your suggestion (if I understood correctly #3930-11). Are you sure that you want to completely remove the "permanent" databases support right now? Apart from backward compatibility, this can slightly slow down the application startup. We can use load_at_startup=FALSE by default for now.
> What do you think?

I think there was a miscommunication or misunderstanding with #3930-11. I was trying to convey that the architecture behind load_at_startup was the obsolete way that had to be replaced. Later (#3930-20), you noted that making non-backward-compatible changes to the directory was risky and you suggested we check the "new" configuration first and then revert to the old-style configuration. Greg agreed. But I understood this to mean that this was only about reading the old configuration and applying load_at_startup to the new implementation, as if a connection string had been provided for that database under /server/default/runtime/default/database-connections/ (i.e., the default for everyone). And this was only meant to be a temporary "bridge", to allow all projects to modify their directories safely after this change is rolled out.

I've been trying to steer you away from the concept of "transient" and "[non-]permanent" databases in #3930-46, #3930-74, #3930-76. I don't think it makes sense to keep this distinction in the implementation. As our understanding of this has evolved, there is no longer a notion of a "permanent" database which must be connected for all users. All connections are effectively "transient". As Constantin points out with his questions about the project tokens, it is not appropriate for some sessions to have a default database connection (not just hypothetically, but even practically, in current projects).

> ...this can slightly slow down the application startup

Please explain your concern in this regard. What will slow the startup if we do not differentiate between "transient" and "permanent" databases? If anything, application startup should be faster if we are not initializing databases when the server starts. However, this is not a goal. It is more of a concern if individual sessions are slow to start, and perhaps you mean this.

I am continuing my review, but let's keep this discussion going...

#114 - 05/27/2021 03:33 AM - Igor Skornyakov

Constantin Asofiei wrote:

> The token support is already in FWD, and the directory utility APIs already have this (at least the Utils.getDirectoryNode APIs).
>
> Igor, please test it, this way:
>
> - change directory.xml to add an empty auto-connect.prj node
> - create a simple program which has a find first book. and convert it.
> - run this program, but pass the client:project:token=prj at the client.sh command.
>
> If everything works OK, then the program will show an error that the database is not connected.

I've performed the test as following:

- Added the following node to the directory.xml

```
<node class="strings" name="database-connections.prj">
   <node-attribute name="values" value="fwd"/>
</node>
```

- Executed a test which requires fwd database w/o client:project:token=prj. Test failed with "unknown database" error.
- Executed the same test with client:project:token=prj. Test passed.

#115 - 05/27/2021 03:41 AM - Igor Skornyakov

Eric Faulhaber wrote:

> I think there was a miscommunication or misunderstanding with #3930-11. I was trying to convey that the architecture behind load_at_startup was the obsolete way that had to be replaced. Later (#3930-20), you noted that making non-backward-compatible changes to the directory was risky and you suggested we check the "new" configuration first and then revert to the old-style configuration. Greg agreed. But I understood this to mean that this was only about reading the old configuration and applying load_at_startup to the new implementation, as if a connection string had been provided for that database under /server/default/runtime/default/database-connections/ (i.e., the default for everyone). And this was only meant to be a temporary "bridge", to allow all projects to modify their directories safely after this change is rolled out.

I see, sorry for the misunderstanding. Will re-work.

> I've been trying to steer you away from the concept of "transient" and "[non-]permanent" databases in #3930-46, #3930-74, #3930-76. I don't think it makes sense to keep this distinction in the implementation. As our understanding of this has evolved, there is no longer a notion of a "permanent" database which must be connected for all users. All connections are effectively "transient". As Constantin points out with his questions about the project tokens, it is not appropriate for some sessions to have a default database connection (not just hypothetically, but even practically, in current projects).

Actually, I've retained the notion of a "transient" database as one which was explicitly connected. As you can see, my changes are about local databases only. The support for remote ones remains the same as before (including use of reference counters)

> ...this can slightly slow down the application startup

> Please explain your concern in this regard. What will slow the startup if we do not differentiate between "transient" and "permanent" databases? If anything, application startup should be faster if we are not initializing databases when the server starts. However, this is not a goal. It is more of a concern if individual sessions are slow to start, and perhaps you mean this.

Yes, I mean possible delays in the start of individual sessions.

#117 - 05/28/2021 03:17 PM - Igor Skornyakov

Eric Faulhaber wrote:

- The concept of "permanent" and "transient" databases, and the use of the load_at_startup configuration option have been preserved. All this should go away. There is no longer any database which should be loaded/initialized at server startup, other than the temp-table database, which is implicitly available always (unless persistence is entirely disabled). Any initialization should arise from sessions needing to connect to a database using the new configuration.

Fixed is described in #3930-113. Temporary keep the list of databases configured as "load_at_startup" and prepend it to the session auto-connected databases' list. TODO comments are added.

- The implementation uses SessionListener for cleanup purposes and adds references to network sessions in the DatabaseManager, which I'd like to avoid. Such references should only be in the ConnectionMangaer, and then only for the purpose of remote database connections over server-to-server virtual sessions. We discussed using ContextLocal for the cleanup instead of SessionListener, which I think would obviate the need for this more complicated implementation. Did you find a problem with the ContextLocal approach?

Replaced com.goldencode.p2j.net.Session with sessionId in the DatabaseManager as discussed in #3930-110. However, I use a sessionId which was added to the BaseSession since the value returned by SecurityManager.getSessionId() seems to be not appropriate here (I do not see how it can be used in the SessionListeter.terminate())

- All new static variables added to the DatabaseManager are all-caps with underscores, instead of camelCased. All-caps with underscores is only for constants (or write-once/read-many variables, which essentially are referenced as constants). Even log was changed to LOG, which in fact I am trying to change in the other direction (older code uses LOG, but it should be log, as this is not a constant).

Fixed.

Committed to 3930a/12477.

- The DMOSignatureHelper fix should be in 3821c, not in this branch (other than after a rebase).

Committed to 3821c/12473.

Getting read of DatabaseMananger.transientDatabases and DatabaseManager.permanentConnections

#118 - 05/30/2021 03:14 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> - The implementation uses SessionListener for cleanup purposes and adds references to network sessions in the DatabaseManager, which I'd like to avoid. Such references should only be in the ConnectionMangaer, and then only for the purpose of remote database connections over server-to-server virtual sessions. We discussed using ContextLocal for the cleanup instead of SessionListener, which I think would obviate the need for this more complicated implementation. Did you find a problem with the ContextLocal approach?
>
> Replaced com.goldencode.p2j.net.Session with sessionId in the DatabaseManager as discussed in #3930-110. However, I use a sessionId which was added to the BaseSession since the value returned by SecurityManager.getSessionId() seems to be not appropriate here (I do not see how it can be used in the SessionListeter.terminate())

I am confused...why are you still using SessionListener and tying all the connection information managed in DatabaseManager to *network* sessions? This seems to be an unnecessary complication of the implementation. The high-level mapping of information by session ID seems redundant with the transparent segregation of information by user context which the ContextLocal class already performs as its primary purpose.

There already is a ContextLocal variable in DatabaseManager (i.e., private static final ContextLocal<Context> context) which seems appropriate for tracking the information needed to establish and clean up these database connections. Instead of SessionListener.terminate(), we should be using the DatabaseManager$Context.cleanup() method which already is there. I don't think SessionListener should be used at all. Please review notes #3930-68, #3930-69, and #3930-80 again.

I think the autoConnect() method belongs in DatabaseManager and any information we need to track by session/context should be in DatabaseManager$Context.

You mentioned in #3930-109:

> I prefer to avoid reference counters as it is a fragile technique. With ContextLocal I will have no way to figure that the database is no longer in use and can be deactivated.

I am not suggesting reference counters and I don't see how using ContextLocal prevents proper deactivation logic. You seem unconvinced. Please explain your reasoning.

> I use a sessionId which was added to the BaseSession since the value returned by SecurityManager.getSessionId() seems to be not appropriate here (I do not see how it can be used in the SessionListeter.terminate())

I don't understand this either, considering I am trying to break all logical ties between this new feature and any network code/concepts. Please help me understand the problem. If there is something that cannot be stored in DatabaseManger$Context (because, for instance, we need to share the knowledge of how long it has been since the last context disconnected from a database), it seems *that* information would be mapped outside of DatabaseManger$Context, in a data structure accessible outside of any particular context, but using *context*-specific (not *network*-specific) keys.

#119 - 05/30/2021 03:28 PM - Igor Skornyakov

Eric Faulhaber wrote:

> I am not suggesting reference counters and I don't see how using ContextLocal prevents proper deactivation logic. You seem unconvinced. Please explain your reasoning.

My understanding is that the database is subject to deactivation when there are no active connections **at all**. This is a global property and I cannot understand how the ContextLocal data can be sufficient. After all, before my changes, the disconnection logic (for 'transient' databases) was based not on the ContextLocal but on global maps in the DatabaseManager.

#120 - 05/30/2021 04:50 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> > I am not suggesting reference counters and I don't see how using ContextLocal prevents proper deactivation logic. You seem unconvinced. Please explain your reasoning.
>
> My understanding is that the database is subject to deactivation when there are no active connections **at all**. This is a global property and I cannot understand how the ContextLocal data can be sufficient. After all, before my changes, the disconnection logic (for 'transient' databases) was based not on the ContextLocal but on global maps in the DatabaseManager.

It was based on both context-local data (Context.localTransients) and on global data (DatabaseManager.transientDatabases).

Your understanding about deactivation is correct, but I don't agree with the conclusion you've reached. As noted in the final paragraph of my previous post, I am not suggesting all data must or can be stored in ContextLocal. Context-local data should be stored there, and global data in global data structures. I do not see a problem using both, and using a global data structure which is more robust than a reference counter certainly is appropriate. But using SessionListener (and specifically SessionListener.terminate()), net.Session, and net.BaseSession creates a logical (and concrete implementation) tie to the network logic and network event model which is not appropriate. The "session" which is the subject of this issue refers to a user context, not a network session.

#121 - 05/30/2021 05:10 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Your understanding about deactivation is correct, but I don't agree with the conclusion you've reached. As noted in the final paragraph of my previous post, I am not suggesting all data must or can be stored in ContextLocal. Context-local data should be stored there, and global data in global data structures. I do not see a problem using both, and using a global data structure which is more robust than a reference counter certainly is appropriate. But using SessionListener (and specifically SessionListener.terminate()), net.Session, and net.BaseSession creates a logical (and concrete implementation) tie to the network logic and network event model which is not appropriate. The "session" which is the subject of this issue refers to a user context, not a network session.

I see. But the entry points for the activation described by Greg and Constantin are associated with the creation of a new network session. I realize that different network sessions can correspond to the same user session but this will be handled automatically since in this case the same set of databases will be retrieved from the directory.
Do you mean that I should make this more explicit by using ContextLocal? This will of course allow getting rid of the DatabaseManager.sessionListener and changes to the network Session/BaseSession.
Is this what you mean?
Thank you.

#122 - 05/30/2021 05:16 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Do you mean that I should make this more explicit by using ContextLocal? This will of course allow getting rid of the DatabaseManager.sessionListener and changes to the network Session/BaseSession.
> Is this what you mean?
> Thank you.

Yes, please.

#123 - 05/30/2021 05:20 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Do you mean that I should make this more explicit by using ContextLocal? This will of course allow getting rid of the DatabaseManager.sessionListener and changes to the network Session/BaseSession.
> > Is this what you mean?
> > Thank you.

> Yes, please.

I see. It will be done tomorrow - it is too late for this in Moscow now. This should be easy. Sorry for the misunderstanding.

#124 - 05/31/2021 05:59 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Do you mean that I should make this more explicit by using ContextLocal? This will of course allow getting rid of the DatabaseManager.sessionListener and changes to the network Session/BaseSession.
> > Is this what you mean?
> > Thank you.
>
> 12481.
> Yes, please.

Fixed in 3930a/12481.

#125 - 05/31/2021 06:29 AM - Eric Faulhaber

Code review 3930a/12481:

Thanks for making these changes.

Please move the autoConnect method to DatabaseManager (#3930-118) and remove the first line, which gets s Session instance that is no longer needed. DatabaseManager.autoConnect() should still be called from the same locations.

Somewhere in the javadoc, please explain in much more detail the concepts of [de-]registration and [de-]activation, and the design of the related logic, especially the mechanism which waits until all databases have been disconnected for some period of time before cleaning up, and what exactly that cleanup entails.

Please ensure the format of the code meets the coding standards.

Please smoke-test with a real project, if you haven't already. Have you tested remote database connection?

#126 - 05/31/2021 06:34 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Code review 3930a/12481:
> Please move the autoConnect method to DatabaseManager (#3930-118) and remove the first line, which gets s Session instance that is no longer needed. DatabaseManager.autoConnect() should still be called from the same locations.
>
> Somewhere in the javadoc, please explain in much more detail the concepts of [de-]registration and [de-]activation, and the design of the related logic, especially the mechanism which waits until all databases have been disconnected for some period of time before cleaning up, and what exactly that cleanup entails.
>
> Please ensure the format of the code meets the coding standards.

OK.

> Please smoke-test with a real project, if you haven't already. Have you tested remote database connection?

I have tested with a real project before but will re-test. I've not tested with the remote database (not sure how to do this), but my changes should not affect remote connections at least until I will get rid of the DatabaseManager.transientDatabases.

#127 - 05/31/2021 06:48 AM - Eric Faulhaber

We have covered a lot of ground...could you please review and report what is left to do before we can consider this task feature complete (including all TODOs)?

#128 - 05/31/2021 06:59 AM - Igor Skornyakov

Eric Faulhaber wrote:

> We have covered a lot of ground...could you please review and report what is left to do before we can consider this task feature complete (including all TODOs)?

Well, I understand that apart from getting rid of DatabaseManager.permanentConnections and DatabaseManager.transientDatabases (and testing with remote database and real app), everything is done. The only remaining thing is removing DatabaseManager.loadedOnStartupDBs and references to it after the customer apps. will change the configuration based on 'load_at_statup' (marked with TODO now).

#129 - 06/01/2021 05:14 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Please move the autoConnect method to DatabaseManager (#3930-118) and remove the first line, which gets s Session instance that is no longer needed. DatabaseManager.autoConnect() should still be called from the same locations.

Done.

Test of the large customer app. passed after replacing readLock with writeLock on activation.

Committed to 3930a/12483.

#130 - 06/02/2021 12:34 AM - Eric Faulhaber

The changes in 3930a revs 12482-3 look good. Please add the more detailed javadoc and clean up the code to coding standards as requested in #3930-125, then rebase again if needed and merge to 3821c.

#131 - 06/02/2021 05:13 AM - Igor Skornyakov

Merged to 3821c/12490.

#132 - 06/11/2021 03:33 PM - Eric Faulhaber

For the remaining cleanup, please note that the explicit connections really shouldn't be treated much differently than the implicit ones. We want the two to share as much of the logic as possible. We don't need a distinction between "registration" and "activation". The purpose of both is to make a database usable by business logic during the time it is connected, and then to clean up resources when they are no longer needed. The only difference is whether business logic explicitly connects/disconnects a database, or whether this is done implicitly by runtime configuration.

That being said, the distinction between remote and local connections must be maintained, as additional network resources are involved in a remote connection.

#133 - 06/11/2021 03:41 PM - Igor Skornyakov

Eric Faulhaber wrote:

> For the remaining cleanup, please note that the explicit connections really shouldn't be treated much differently than the implicit ones. We want the two to share as much of the logic as possible. We don't need a distinction between "registration" and "activation". The purpose of both is to make a database usable by business logic during the time it is connected, and then to clean up resources when they are no longer needed. The only difference is whether business logic explicitly connects/disconnects a database, or whether this is done implicitly by runtime configuration.
>
> That being said, the distinction between remote and local connections must be maintained, as additional network resources are involved in a remote connection.

I see. Thank you!

#134 - 06/13/2021 04:29 PM - Igor Skornyakov

Get rid of the notion of the permanent connection.

Committed to 3930a revision 12499.
The notion of the transient connection looks more complicated. I still do not completely understand the logic around reference counters.

#135 - 06/13/2021 07:42 PM - Eric Faulhaber

- Related to Bug #5440: finish DatabaseManager refactoring for implicit/explicit connections added

#136 - 06/13/2021 07:56 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Get rid of the notion of the permanent connection.
>
> Committed to 3930a revision 12499.

Please do not re-use branches once they have been merged. Please extract your changes and either apply them directly to 3821c (if they compile, are functional, and are safe), or into a patch file made against the most current revision of 3821c (if they are not functional or particularly risky). In the latter case, please attach the patch file to #5440.

> The notion of the transient connection looks more complicated. I still do not completely understand the logic around reference counters.

I don't understand. As we've discussed on many occasions, we don't need to preserve the reference counting logic. It is obsolete, I've agreed that it is fragile logic, that it probably is broken already, and that it needs to be removed. Its only purpose was to clean up after all connections to a database have been terminated, and you've already invented a parallel mechanism to accomplish this. So, this was only about using your parallel cleanup mechanism for *all* connections *instead of* the reference counting.

In any event, we have no more time to spend on this task. The original feature is implemented at this point, any internal cleanup/refactoring will have to be done in #5440, but not now. Please stop working on this task after managing your in-flight changes as noted above.

#137 - 06/13/2021 09:12 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

#138 - 06/14/2021 08:07 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Please do not re-use branches once they have been merged. Please extract your changes and either apply them directly to 3821c (if they compile, are functional, and are safe), or into a patch file made against the most current revision of 3821c (if they are not functional or particularly risky). In the latter case, please attach the patch file to #5440.

Committed to 3821c/12532.

> The notion of the transient connection looks more complicated. I still do not completely understand the logic around reference counters.

> I don't understand. As we've discussed on many occasions, we don't need to preserve the reference counting logic. It is obsolete, I've agreed that it is fragile logic, that it probably is broken already, and that it needs to be removed. Its only purpose was to clean up after all connections to a database have been terminated, and you've already invented a parallel mechanism to accomplish this. So, this was only about using your parallel cleanup mechanism for *all* connections *instead of* the reference counting.

The problem is that the logic depends on the reference counting. With my initial approach based on network sessions, there was no need for the reference counters at all - we just need to wait when there will be no network sessions using the database. However, with user sessions, there is no such simple way to figure that the database is not in use anymore.

Files

Session.diff	1004 Bytes	05/20/2021	Igor Skornyakov
RouterSessionManager.diff	1007 Bytes	05/20/2021	Igor Skornyakov
DatabaseManager.diff	17.1 KB	05/20/2021	Igor Skornyakov
ConnectionManager.diff	1.39 KB	05/20/2021	Igor Skornyakov
BaseSession.diff	2.86 KB	05/20/2021	Igor Skornyakov