# Database - Feature #4175

## add support for -1 (single user) option for CONNECT statement

08/08/2019 10:17 AM - Eric Faulhaber

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

**Description**

**Related issues:**

| | |
|---|---|
| Related to Database - Feature #3813: misc DB features part deux | **Closed** |
| Related to Runtime Infrastructure - Feature #5279: add support for an optiona... | **New** |

### History

#### #1 - 08/08/2019 10:19 AM - Eric Faulhaber

Single user mode of the CONNECT statement is not currently supported. We'll need a test case to understand exactly how this works, but if the idea is to disallow any other connection to the database, once the first connection is made with this option, we can support this via ConnectionManager. This will only work for connections made through FWD. External connections to a standalone, mult-user database server will not be affected.

#### #2 - 01/24/2020 08:11 AM - Greg Shah

*- Related to Feature #3813: misc DB features part deux added*

#### #3 - 01/24/2020 08:11 AM - Greg Shah

*- Status changed from New to WIP*

*- Assignee set to Igor Skornyakov*

#### #4 - 04/24/2020 11:44 AM - Greg Shah

What is the status of this task?

#### #5 - 04/24/2020 01:06 PM - Igor Skornyakov

I was not working in this task/ As far as I remember, I was reassigned to another one.

#### #6 - 04/20/2021 12:45 PM - Igor Skornyakov

It seems that CONNECT -1 is indeed about a single **user**, not a single **session**.
The following test

```
DISCONNECT dblias1 NO-ERROR.
DISCONNECT dblias2 NO-ERROR.
CONNECT -1 -db dbname -H host -S port -N tcp -ld dblias1.
CONNECT -1 -db dbname -H host -S port -N tcp -ld dblias2.

DEF VAR p AS CHAR NO-UNDO.

p = DBPARAM('dbalias1').
MESSAGE p.

p = DBPARAM('dbalias2').
MESSAGE p.
```

runs w/o problem even in multiple sessions and DBPARAM() return expected values.

#### #7 - 04/20/2021 01:49 PM - Igor Skornyakov

In addition. For the table database, the "-1" option is mandatory for CONNECT and only a single session can connect to such database.

#### #8 - 04/20/2021 03:40 PM - Igor Skornyakov

I've managed to add two users to the test database and repeated the test described before but using different users for different connections. Again, everything works fine even in two different sessions.

At this moment I've run out of ideas. I cannot imagine any reasonable semantics for the "-1" CONNECT option apart from the trivial "tautological" case of the file database.

#### #9 - 04/20/2021 03:56 PM - Igor Skornyakov

Igor Skornyakov wrote:

> At this moment I've run out of ideas. I cannot imagine any reasonable semantics for the "-1" CONNECT option apart from the trivial "tautological" case of the file database.

I understand that some of our customers want to have support for this option. Maybe it makes sense to ask them what semantics they expect?

#### #10 - 04/20/2021 04:30 PM - Greg Shah

> Again, everything works fine even in two different sessions.

Please explain how you run different sessions. The testcase from #4175-6 has no explicit pause, so it seems like if you run this it will only pause when the external procedure exits ("Procedure complete. Press space bar to continue."). That implicit pause may be after any disconnection has occurred so it may not be a correct test.

I would put a pause. after the CONNECT statements and before the MESSAGE statements. Then I would run 2 separate mpro processes to execute the program, leaving both at the pause at the same time (if possible).

If you have done that and it still allows both to work, then I wonder if this is really about how the database server is started. If proserve is used to start the database, this is by nature a multi-user database server. Perhaps there is no way to lock it down to a single user/session once the database is opened by proserve. Instead, if you use pro or _progress -1 (ChUI mode) or prowin.exe -1 to access a database that is not already open, then it will be opened privately by that process. In other words, it won't be open in multi-user mode. On the other hand, using mpro, _progress or prowin.exe by default will try to connect to an existing multi-user database server.

If this is the case, then it may mean that CONNECT -1 will only work with a database that is not already open as a multi-user database server. So you could retry the test I mention above but make sure there is no multi-user database server already running.

#### #11 - 04/20/2021 04:40 PM - Igor Skornyakov

Greg Shah wrote:

> Please explain how you run different sessions. The testcase from #4175-6 has no explicit pause, so it seems like if you run this it will only pause when the external procedure exits ("Procedure complete. Press space bar to continue."). That implicit pause may be after any disconnection has occurred so it may not be a correct test.
>
> I would put a pause. after the CONNECT statements and before the MESSAGE statements. Then I would run 2 separate mpro processes to execute the program, leaving both at the pause at the same time (if possible).

I've opened two Procedure Editor windows and run the test in both. After the end I ee (using Database Administrator) two connected databases in both. Please note also that in both sessions I connect twice to the same database with different logical names and users.

> If you have done that and it still allows both to work, then I wonder if this is really about how the database server is started. If proserve is used to start the database, this is by nature a multi-user database server. Perhaps there is no way to lock it down to a single user/session once the database is opened by proserve. Instead, if you use pro or _progress -1 (ChUI mode) or prowin.exe -1 to access a database that is not already open, then it will be opened privately by that process. In other words, it won't be open in multi-user mode. On the other hand, using mpro, _progress or prowin.exe by default will try to connect to an existing multi-user database server.
>
> If this is the case, then it may mean that CONNECT -1 will only work with a database that is not already open as a multi-user database server. So you could retry the test I mention above but make sure there is no multi-user database server already running.

The database server was pre-started. I do not understand your suggestion about a test without a pre-started server. To what the test will connect? As I wrote before if we connect to the database specifying file then "-1" is more to less a tautology - 4GL requires it and does not allow multiple connections to the file database.

#### #12 - 04/20/2021 04:51 PM - Igor Skornyakov

Greg Shah wrote:

> I would put a pause. after the CONNECT statements and before the MESSAGE statements. Then I would run 2 separate mpro processes to execute the program, leaving both at the pause at the same time (if possible).

With PAUSE result is the same.

#### #13 - 04/20/2021 05:22 PM - Greg Shah

> After the end I ee (using Database Administrator) two connected databases in both.

What is "ee"?

> The database server was pre-started. I do not understand your suggestion about a test without a pre-started server. To what the test will connect?

If you specify the physical database in the connect statement and use -1, I suspect that the database will be loaded inside the 4GL client process.

> As I wrote before if we connect to the database specifying file then "-1" is more to less a tautology - 4GL requires it and does not allow multiple connections to the file database.

I'm not sure what you mean by "specifying file". You may be talking about the same thing as I am saying.

If you have the physical database files for p2j_test in the current directory and there is no multi-user database server that has p2j_test open, then I think you can get a single session mode in 2 ways:

- Run pro -db p2j_test .... I think the -1 is optional here. If the multi-user database already has p2j_test open, I think this will error out (the pro session will not startup properly). Assuming the database is not already open, I think this creates a private instance of the database inside this process.
- Run a procedure using any of the 4GL clients (pro, mpro, prowin.exe...) and inside that procedure issue a CONNECT -1 -d p2j_test .... I think that if the database is not already open, then this is the same as specifying -db p2j_test on the command line, that it will create a private instance of the database inside this process. It seems (from your testing) that if the database is already open using a multi-user database server (proserve), then it will silently connect without errors.

I doubt that the user has anything to do with this. I think this is about a private (in-process) instance of the database vs a multi-user database server instance.

**#14 - 04/20/2021 05:44 PM - Igor Skornyakov**

Greg Shah wrote:

> After the end I ee (using Database Administrator) two connected databases in both.

What is "ee"?

Sorry. It stands for "see".

> The database server was pre-started. I do not understand your suggestion about a test without a pre-started server. To what the test will connect?

If you specify the physical database in the connect statement and use -1, I suspect that the database will be loaded inside the 4GL client process.

> As I wrote before if we connect to the database specifying file then "-1" is more to less a tautology - 4GL requires it and does not allow multiple connections to the file database.

I'm not sure what you mean by "specifying file". You may be talking about the same thing as I am saying.

Yes, it looks like that. Sorry for using non-standard wording.

If you have the physical database files for p2j_test in the current directory and there is no multi-user database server that has p2j_test open, then I think you can get a single session mode in 2 ways:

- Run pro -db p2j_test .... I think the -1 is optional here. If the multi-user database already has p2j_test open, I think this will error out (the pro session will not startup properly). Assuming the database is not already open, I think this creates a private instance of the database inside this process.
- Run a procedure using any of the 4GL clients (pro, mpro, prowin.exe...) and inside that procedure issue a CONNECT -1 -d p2j_test .... I think that if the database is not already open, then this is the same as specifying -db p2j_test on the command line, that it will create a private instance of the database inside this process. It seems (from your testing) that if the database is already open using a multi-user database server (proserve), then it will silently connect without errors.

I doubt that the user has anything to do with this. I think this is about a private (in-process) instance of the database vs a multi-user database server instance.

OK. I will test the scenarios you've described tomorrow. However, I do not understand how the FWD equivalent should look like.

#### #15 - 04/20/2021 06:37 PM - Greg Shah

If I'm correct, then FWD is like the 2nd scenario.  We don't have the command line case.  Instead, we have the CONNECT case.  If -1 is present, then should act like the 4GL:

- If the database has no other sessions connected, then we connect but lock out anyone else from connecting until we are disconnected explicitly or our session exits.
- If the database has other sessions connected, we allow the connection but we silently ignore the -1.

This assumes my guesses are right.  Anyway, it seems pretty straightforward to implement this kind of behavior.

#### #16 - 04/21/2021 04:23 AM - Igor Skornyakov

Greg Shah wrote:

> If I'm correct, then FWD is like the 2nd scenario.  We don't have the command line case.  Instead, we have the CONNECT case.  If -1 is present, then should act like the 4GL:
>
> - If the database has no other sessions connected, then we connect but lock out anyone else from connecting until we are disconnected explicitly or our session exits.
> - If the database has other sessions connected, we allow the connection but we silently ignore the -1.
>
> This assumes my guesses are right.  Anyway, it seems pretty straightforward to implement this kind of behavior.

Indeed, this is easy to implement. However, I do not understand how to test this scenario with 4GL. Only one session can be connected to p2_test regardless of how it is done - in the command line or using CONNECT.

#### #17 - 04/21/2021 05:26 AM - Greg Shah

> Only one session can be connected to p2_test regardless of how it is done - in the command line or using CONNECT.

Are you talking about the case where there are no multi-user database servers started for this database?

When you start a proserve multi-user database server for p2j_test, what are the results of the same tests?

**#18 - 04/21/2021 05:36 AM - Igor Skornyakov**

Greg Shah wrote:

> When you start a proserve multi-user database server for p2j_test, what are the results of the same tests?

I've not used proserve  before and my first attempts failed. So I've used an existing multi-user database. Does it make difference?
Thank you.

**#19 - 04/21/2021 05:46 AM - Greg Shah**

> Does it make difference?

Probably not.

Do you see anything inconsistent with my "guesses"?  Can you implement this now?

**#20 - 04/21/2021 05:53 AM - Igor Skornyakov**

Greg Shah wrote:

> Does it make difference?
>
> Probably not.
>
> Do you see anything inconsistent with my "guesses"?  Can you implement this now?

Your guesses look logical to me. I can implement it now.

BTW: I've managed to start proserve using proserve p2j_test.db -S 2098 -H hostname -B 10000 but I cannot connect to the database - errno 0. (1432).

Should I suspend my attempts to deal with this and start implementing the logic you've suggested?
Thank you.

#### #21 - 04/21/2021 05:55 AM - Greg Shah

What is the "f" in "fp2j_test.db"?

#### #22 - 04/21/2021 05:56 AM - Igor Skornyakov

Greg Shah wrote:

> What is the "f" in "fp2j_test.db"?

Sorry, it is a misprint. Should be just p2j_test.db

#### #23 - 04/21/2021 06:06 AM - Greg Shah

Running the test with an existing mutli-user database server should be sufficient.  Go ahead with the implementation.

#### #24 - 04/21/2021 06:30 AM - Igor Skornyakov

Greg Shah wrote:

> Running the test with an existing mutli-user database server should be sufficient.  Go ahead with the implementation.

OK. Thank you.

BTW: I was able to connect to the database started with proserve as mpro p2jtest, but fail to do this using CONNECT -1 -db p2j_test -H 127.0.0.1 -S 2826 -N tcp -ld p2j_test1 -U <uid> -P <pwd>.

The error is:

```
Disconnected from server because database name was incorrect. (437)
                  ** Server rejected login. (700)
```

#### #25 - 04/21/2021 06:36 AM - Greg Shah

That seems inconsistent.  How is the CONNECT different from when you tested with the existing multi-user database server (not p2j_test).

#### #26 - 04/21/2021 06:50 AM - Igor Skornyakov

Well, if I use CONNECT -1 -db p2j_test -ld p2j_test1 -U <uid> -P <pwd>. while another mpro p2j_test is running in another window I get

```
** The database p2j_test is in use in multi-user mode. (276)
```

However, I get the same when I'm trying to do this starting mpro without arguments right after starting the broker.

It looks that for FWD this is the only way to handle CONNECT -1 in a way that is compatible with 4GL.

In my understanding, the "-1" option is just for forcing the user to specify explicitly to what type of database he/she is going to connect. It is required for a physical ("file") database and is not accepted for a connection to a server. There is no way to enforce a single-user mode when connecting to a server and, as far as I understand, this is the only type of connection supported by FWD.

#### #27 - 04/21/2021 06:55 AM - Igor Skornyakov

Igor Skornyakov wrote:

> Well, if I use CONNECT -1 -db p2j_test -ld p2j_test1 -U <uid> -P <pwd>. while another mpro p2j_test is running in another window I get [...]
>
> However, I get the same when I'm trying to do this starting mpro without arguments right after starting the broker.
>
> It looks that for FWD this is the only way to handle CONNECT -1 in a way that is compatible with 4GL.
>
> In my understanding, the "-1" option is just for forcing the user to specify explicitly to what type of database he/she is going to connect. It is required for a physical ("file") database and is not accepted for a connection to a server. There is no way to enforce a single-user mode when connecting to a server and, as far as I understand, this is the only type of connection supported by FWD.

As you've mentioned when I'm trying to connect to a server (not proserve) the "-1" option is just ignored. May be this is more correct for FWD?

#### #28 - 04/21/2021 07:03 AM - Igor Skornyakov

Sorry, I've started mpro from the directory where the p2_test.db file was located. I cannot connect to the broker when starting from another directory.

#### #29 - 04/21/2021 07:10 AM - Greg Shah

Customers use this to ensure that no other processing is occurring for a database while they are doing some kind of critical update/maintenance. Making this do nothing is not acceptable.

Previously you stated that you could connect 2 sessions using CONNECT -1 ... when there was an existing multi-user database server. Please help me understand why you cannot recreate this scenario now.

Please work to figure out the exact rules for this -1 mode. Document them here. We can't just ignore this. I've already explained that when no other sessions are connected to a database, then in FWD we can easily make the -1 mode disallow other connections while the -1 connection exists. It

seems the question is how to handle the case where we try to CONNECT -1... while there is already a multi-user database server that has the database open.

#### #30 - 04/21/2021 07:31 AM - Igor Skornyakov

Greg Shah wrote:

> Customers use this to ensure that no other processing is occurring for a database while they are doing some kind of critical update/maintenance. Making this do nothing is not acceptable.
>
> Previously you stated that you could connect 2 sessions using CONNECT -1 ... when there was an existing multi-user database server. Please help me understand why you cannot recreate this scenario now.
>
> Please work to figure out the exact rules for this -1 mode. Document them here. We can't just ignore this. I've already explained that when no other sessions are connected to a database, then in FWD we can easily make the -1 mode disallow other connections while the -1 connection exists. It seems the question is how to handle the case where we try to CONNECT -1... while there is already a multi-user database server that has the database open.

Sorry, I do too many stupid mistakes.

The final results are:

1. Start the broker using proserve p2j_test.db -S 2098 -H localhost -B 10000
2. Start the first mpro and execute CONNECT -1 -db p2j_test -N tcp -H localhost -S 2098 -ld p2j_test1 -U ... -P ....
3. Start the second mpro and execute CONNECT -db p2j_test -N tcp -H localhost -S 2098 -ld p2j_test1 -U ... -P ....
   Both connections were established.

1. Restart the broker
2. Execute the same command but in a different order (first w/o '-1', second with it)
   Both connections were established.

1. Restart the broker
2. Connect to p2j_test from two mpro sessions with '-1' option but different users.
   Both connections were established.

I came to the conclusion that for multi-user databases the '-1' option has no effect.

#### #31 - 04/21/2021 07:35 AM - Greg Shah

Good.  This is consistent with my previous suggestions.  Please go ahead with the implementation.

#### #32 - 04/21/2021 07:40 AM - Igor Skornyakov

Greg Shah wrote:

> Good.  This is consistent with my previous suggestions.  Please go ahead with the implementation.

OK. Thank you.

#### #33 - 04/21/2021 07:49 AM - Igor Skornyakov

Greg Shah wrote:

> Good.  This is consistent with my previous suggestions.  Please go ahead with the implementation.

Sorry, Greg. Your suggestion implies that in some cases the "-1" option **does** have an effect (if it was used in a first connection after the server start).
However, my tests do not approve this - according to the results it **never** has any effect and should just be ignored as we do now.
Have I misunderstood what you mean?
Thank you.

#### #34 - 04/21/2021 08:27 AM - Greg Shah

I think you misunderstand.  Do NOT assume that we only act as if FWD was a multi-user database server.  As I explained in #4175-29, we have real customer code that requires this behavior.

> Your suggestion implies that in some cases the "-1" option does have an effect (if it was used in a first connection after the server start).

Close, but not exactly.  It has nothing to do with the server start.  We should only care if there are already connected sessions or not.

If there are no existing connections, then when we CONNECT -1 ... we should act as if the database is private to that session and no other sessions will be allowed.  It doesn't matter that we are running in a multi-user server.  When the -1 session disconnects from that database or that session ends, then the database would be open for new connections.  A subsequent session might try to CONNECT -1 ... to that database and it would also be able to do so in -1 mode if there are still no other connections.

As your testing shows, we would silently ignore -1 connection if there are already other sessions connected.

> However, my tests do not approve this - according to the results it never has any effect and should just be ignored as we do now.

It is my understanding that this is not correct.  If there IS NO multi-user database server involved in the 4GL, then the CONNECT -1 does open the database inside the local process, making it private and disallowing any other session from using it.  Isn't that the case?

We will support this case as I noted above.  Forget that FWD is multi-user and focus on meeting the customer requirement.

#### #35 - 04/21/2021 08:34 AM - Igor Skornyakov

Greg Shah wrote:

> We will support this case as I noted above.  Forget that FWD is multi-user and focus on meeting the customer requirement.

Got it. Thank you.

#### #36 - 04/21/2021 01:50 PM - Igor Skornyakov

I'm trying to test my support for CONNECT -1.
The test program is:

```
CONNECT -db SOME_DATABASE -N tcp -H localhost -S 5433 -ld hotel -U fwd-admin -P admin.
PAUSE.
```

The application fails with the following exception:

```
com.goldencode.p2j.util.ErrorConditionException: com.goldencode.p2j.persist.PersistenceException: javax.net.ss
l.SSLHandshakeException: Conversation [0000000E:bogus]:Handshake timed out.
Caused by: com.goldencode.p2j.persist.PersistenceException: javax.net.ssl.SSLHandshakeException: Conversation
[0000000E:bogus]:Handshake timed out.
Caused by: javax.net.ssl.SSLHandshakeException: Conversation [0000000E:bogus]:Handshake timed out.
        at com.goldencode.p2j.net.NIOSslSocket.<init>(NIOSslSocket.java:198)
        at com.goldencode.p2j.net.NetSocket$1.create(NetSocket.java:149)
        at com.goldencode.p2j.net.SessionManager.connect(SessionManager.java:1215)
        at com.goldencode.p2j.net.SessionManager.createQueue(SessionManager.java:1124)
        at com.goldencode.p2j.net.RouterSessionManager.connectVirtual(RouterSessionManager.java:516)
        at com.goldencode.p2j.net.RouterSessionManager.connectVirtual(RouterSessionManager.java:448)
        at com.goldencode.p2j.net.SessionManager.connectVirtual(SessionManager.java:726)
        at com.goldencode.p2j.persist.ConnectionManager.makeConnection(ConnectionManager.java:3558)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:3375)
        at com.goldencode.p2j.persist.ConnectionManager.connectDbImpl(ConnectionManager.java:3028)
        at com.goldencode.p2j.persist.ConnectionManager.connect_(ConnectionManager.java:676)
        at com.goldencode.p2j.persist.ConnectionManager.connect(ConnectionManager.java:579)
        at com.goldencode.testcases.ias.ConnectS.lambda$0(ConnectS.java:27)
```

It doesn't look like an attempt to connect to a database.
What I'm doing wrong?
Thank you.

#### #37 - 04/21/2021 02:01 PM - Greg Shah

I don't know exactly, but the ConnectionManager is definitely trying to make a new connection to some remote FWD server.  That fails and dies badly.

Something about your connection parameters leads to this result.  I think you have to debug it unless Eric can explain the situation.

#### #38 - 04/21/2021 02:05 PM - Igor Skornyakov

Greg Shah wrote:

> I don't know exactly, but the ConnectionManager is definitely trying to make a new connection to some remote FWD server.  That fails and dies badly.
>
> Something about your connection parameters leads to this result.  I think you have to debug it unless Eric can explain the situation.

With a debugger, I see that FWD always tries to do this if the connection is not to a local database (port is specified). This is in fact logical because otherwise, we need to have at least something like JDBC URL template(s) in the configuration ( single template or a template per TCP endpoint).

#### #39 - 04/21/2021 02:13 PM - Igor Skornyakov

Actually, I do not understand how the current support for CONNECT is supposed to work. Is it documented?
Thank you.

#### #40 - 04/21/2021 03:18 PM - Eric Faulhaber

Igor, as with much of the FWD runtime currently, there is no formal documentation which describes the internal implementation of the converted CONNECT statement in FWD. However, please see Port Services Mappings. This section describes how to map ports to allow the -S parameter of the CONNECT statement to work.

Do not conflate the 4GL CONNECT statement with a JDBC connection. The implementation of the CONNECT statement in FWD connects the current FWD session to the authoritative FWD server for a particular database via a server-to-server "virtual network session". I will refer to that connected server as the "remote" server and that connected database as the "remote" database, even though it could be the same FWD server as the "local" server originating the connection. Once connected, the requesting, local server is given proxy objects for the various services it needs to access the database at the remote side (e.g., RemotePersistence, RemoteLockManager, etc.). Record buffers, queries, etc. from that point operate normally against the remote database.

All of the above being said, I am reviewing the current ConnectionManager code now, and it seems over time, the connect code has been changed (by you, I think). A connect_ method and some related methods have been added and the connectImpl method has been orphaned. So, I'm not sure my description above of the intended design is still accurate.

In any case, the remoteness introduced by the use of the host (-H) and port/service (-S) parameters is an unnecessary complication for this issue. Your CONNECT statement is trying to establish a virtual network session with a FWD server listening on port 5433, which does not exist, as this is most likely your PostgreSQL port. Please try your CONNECT statement without these two parameters. This should tell the ConnectionManager that we are connecting to the current/local FWD server, and should avoid the remote connection error.

#### #41 - 04/21/2021 03:30 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, as with much of the FWD runtime currently, there is no formal documentation which describes the internal implementation of the converted CONNECT statement in FWD. However, please see Port Services Mappings. This section describes how to map ports to allow the -S parameter of the CONNECT statement to work.
>
> Do not conflate the 4GL CONNECT statement with a JDBC connection. The implementation of the CONNECT statement in FWD connects the current FWD session to the authoritative FWD server for a particular database via a server-to-server "virtual network session". I will refer to that connected server as the "remote" server and that connected database as the "remote" database, even though it could be the same FWD server as the "local" server originating the connection. Once connected, the requesting, local server is given proxy objects for the various services it needs to access the database at the remote side (e.g., RemotePersistence, RemoteLockManager, etc.). Record buffers, queries, etc. from that point operate normally against the remote database.
>
> All of the above being said, I am reviewing the current ConnectionManager code now, and it seems over time, the connect code has been changed (by you, I think). A connect_ method and some related methods have been added and the connectImpl method has been orphaned. So, I'm not sure my description above of the intended design is still accurate.
>
> In any case, the remoteness introduced by the use of the host (-H) and port/service (-S) parameters is an unnecessary complication for this issue. Your CONNECT statement is trying to establish a virtual network session with a FWD server listening on port 5433, which does not exist, as this is most likely your PostgreSQL port. Please try your CONNECT statement without these two parameters. This should tell the ConnectionManager that we are connecting to the current/local FWD server, and should avoid the remote connection error.

Thank you, Eric.
My changes to the  ConnectionManager are about CONNECT parameters only.
However, I do not understand some details in your explanations.

1. Do you mean that in fact the host/port parameters of the CONNECT are not supported? Or it can be the endpoint of the remote FWD server?
2. I do not see now the dbname parameter of the CONNECT is used. What if the remote FWD server is connected to more than one database?
3. What about the semantics of the '-1' option? Should it be applied at the target FWD server (and in this case we need a way to inform it about this) or it is applicable to the "client" side? I have an impression that Greg was talking about "remote" semantics.
   **Greg** - is my understanding correct?

Thank you.

#### #42 - 04/21/2021 03:50 PM - Eric Faulhaber

Igor Skornyakov wrote:

> However, I do not understand some details in your explanations.

> Do you mean that in fact the host/port parameters of the CONNECT are not supported? Or it can be the endpoint of the remote FWD server?

They are supported, but if you are connecting to a truly remote database managed by a separate FWD server, you need to have that server up and running and listening for remote connections on the port specified. If you use a service name instead of a port number, you must configure the port services in the directory to map that service name to the appropriate port. Assuming the network the FWD servers are running on uses the same host names or IP addresses as the original system, the host names should already work.

> I do not see now the dbname parameter of the CONNECT is used. What if the remote FWD server is connected to more than one database?

The dbname name parameter corresponds with the physical database name configured in the remote FWD server's directory. See Database Instances.

> What about the semantics of the '-1' option? Should it be applied at the target FWD server (and in this case we need a way to inform it about this) or it is applicable to the "client" side? I have an impression that Greg was talking about "remote" semantics.
> **Greg** - is my understanding correct?

Greg, correct me here if I'm wrong, but I think the limitation needs to be applied whether the connection is remote or local. Since the remote connections are managed through proxies, assuming we implement this limitation in the right place, I think it will be transparent as to whether the connection is remote or local.

#### #43 - 04/21/2021 03:58 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The dbname name parameter corresponds with the physical database name configured in the remote FWD server's directory. See Database Instances.

Do you mean that database name and user credentials provided as CONNECT parameters are propagated to the remote server? I've not found where it is done. Can you please advise where I should look? I may need to propagate the "-1" option as well.
Thank you.

#### #44 - 04/21/2021 04:39 PM - Greg Shah

> Greg, correct me here if I'm wrong, but I think the limitation needs to be applied whether the connection is remote or local.

Yes, you are probably right that we should implement it for local and remote databases.  Previously, I was not considering that aspect.

#### #45 - 04/21/2021 04:47 PM - Igor Skornyakov

Greg Shah wrote:

> Yes, you are probably right that we should implement it for local and remote databases.  Previously, I was not considering that aspect.

I have two questions.

1. For the local connection - should the "-1" option be considered only for databases that are not "loaded at startup"?
2. For the remote connection - should this option be considered at the "client" or "server" side (or both)? If at the server side should be it be also applicable only for the databases that are not "loaded at startup"?
   Thank you.

#### #46 - 04/21/2021 04:50 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Do you mean that database name and user credentials provided as CONNECT parameters are propagated to the remote server? I've not found where it is done. Can you please advise where I should look? I may need to propagate the "-1" option as well.

The server-to-server connection is made once and IIRC the user is authenticated at that point. I am fuzzy on the user authentication details, as I did not implement that part.

From that point on, the DatabaseManager gets a remote database configuration from the DatabaseManager$DatabaseConfigFetcher for the particular physical database it is trying to access, and the requesting context creates proxies to a remote Persistence object (and several other supporting proxies) for that database. From that point on, access to the remote database is transparent from the point of view of the higher level persistence framework.

Look at the call hierarchy from DatabaseManager$DatabaseConfigFetcher.fetch method up through ConnectionManager.makeConnection and ConnectionManager.connect to see how the remote access is implemented. Note the call to PersistenceFactory.getInstance(Database), which will return either a concrete Persistence object for a local connection or a proxy RemotePersistence object for a remote connection.

**#47 - 04/21/2021 04:53 PM - Igor Skornyakov**

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Do you mean that database name and user credentials provided as CONNECT parameters are propagated to the remote server? I've not found where it is done. Can you please advise where I should look? I may need to propagate the "-1" option as well.
>
> The server-to-server connection is made once and IIRC the user is authenticated at that point. I am fuzzy on the user authentication details, as I did not implement that part.
>
> From that point on, the DatabaseManager gets a remote database configuration from the DatabaseManager$DatabaseConfigFetcher for the particular physical database it is trying to access, and the requesting context creates proxies to a remote Persistence object (and several other supporting proxies) for that database. From that point on, access to the remote database is transparent from the point of view of the higher level persistence framework.
>
> Look at the call hierarchy from DatabaseManager$DatabaseConfigFetcher.fetch method up through ConnectionManager.makeConnection and ConnectionManager.connect to see how the remote access is implemented. Note the call to PersistenceFactory.getInstance(Database), which will return either a concrete Persistence object for a local connection or a proxy RemotePersistence object for a remote connection.

Got it. Thank you.

**#48 - 04/21/2021 04:54 PM - Eric Faulhaber**

Igor Skornyakov wrote:

> For the local connection - should the "-1" option be considered only for databases that are not "loaded at startup"?

I think so, but it depends what the 4GL does. I have not tested this, but it makes sense to me that auto-connect databases would not be limited later.

> For the remote connection - should this option be considered at the "client" or "server" side (or both)? If at the server side should be it be also applicable only for the databases that are not "loaded at startup"?

I don't see how the client side makes sense. W.r.t. loaded at startup, see above.

**#49 - 04/21/2021 04:57 PM - Igor Skornyakov**

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > For the local connection - should the "-1" option be considered only for databases that are not "loaded at startup"?
>
> I think so, but it depends what the 4GL does. I have not tested this, but it makes sense to me that auto-connect databases would not be limited later.

I agree. And it looks me compatible with the semantics of the "-1" suggested by Greg.

> > For the remote connection - should this option be considered at the "client" or "server" side (or both)? If at the server side should be it be also applicable only for the databases that are not "loaded at startup"?
>
> I don't see how the client side makes sense. W.r.t. loaded at startup, see above.

This looks logical for me as well.
**Greg** - do you agree?
Thank you.

**#50 - 04/21/2021 05:24 PM - Greg Shah**

> For the local connection - should the "-1" option be considered only for databases that are not "loaded at startup"?

Please consider it for any database. If we are the first session created or the only session existing, then a CONNECT -1... should lock out other sessions for the duration of the connection. I think this is needed because the customer usage pattern is that this will be used for databases that are

"loaded at startup". Also, in #3930, "loaded at startup" will be less global. Just ignore the "loaded at startup" and honor -1 even if it is connecting to a database we are already connected to.

#### #51 - 04/21/2021 05:36 PM - Eric Faulhaber

If at all possible, we should not have differential logic to enforce this limitation for remote and local databases. I think we should be able to find a place to inject this logic, regardless of whether the database is remote or local. Possibly in PersistenceFactory.getInstance, based on a flag stored in Persistence?

#### #52 - 04/21/2021 05:43 PM - Igor Skornyakov

Greg Shah wrote:

> For the local connection - should the "-1" option be considered only for databases that are not "loaded at startup"?

> Please consider it for any database. If we are the first session created or the only session existing, then a CONNECT -1... should lock out other sessions for the duration of the connection. I think this is needed because the customer usage pattern is that this will be used for databases that are "loaded at startup". Also, in #3930, "loaded at startup" will be less global. Just ignore the "loaded at startup" and honor -1 even if it is connecting to a database we are already connected to.

I see, thank you. More considerations seem to be important.

- Should the '-1' connection acquire "global lock" only for the **very** first connection or just if no other connections are active?
- For a remote database - will such a lock be in effect on the client side or/and on the remote one? Please note that the connection can be first locally but not on the remote side.
- It is also possible that the database is already locked remotely. Should the external connection be rejected in this case?

#### #53 - 04/21/2021 05:45 PM - Igor Skornyakov

Eric Faulhaber wrote:

> If at all possible, we should not have differential logic to enforce this limitation for remote and local databases. I think we should be able to find a place to inject this logic, regardless of whether the database is remote or local. Possibly in PersistenceFactory.getInstance, based on a flag stored in Persistence?

I understand your point. Please consider however the possible difference of which connection should be considered "first" on difererent sides in the case of the remote connection.

#### #54 - 04/21/2021 05:52 PM - Igor Skornyakov

I also think that full transparency can be achieved easier if the Database (and related objects such as Persistence) was an interface and for the remote connection, a corresponding proxy was created.

#### #55 - 04/21/2021 05:58 PM - Greg Shah

> just if no other connections are active?

This one.

#### #56 - 04/21/2021 06:00 PM - Igor Skornyakov

Greg Shah wrote:

> > just if no other connections are active?
>
> This one.

I see. Thanks.

#### #57 - 04/22/2021 03:02 AM - Eric Faulhaber

Igor Skornyakov wrote:

> I also think that full transparency can be achieved easier if the Database (and related objects such as Persistence) was an interface and for the remote connection, a corresponding proxy was created.

I don't understand what a proxy for a Database object would do. Database objects are meant primarily as informational objects and convenient hash keys for information that needs to be mapped by physical database. There are no services a Database object provides that could usefully be called by proxy to do some remote work.

I implemented RemotePersistence as a subclass of, rather than a proxy for Persistence, because most of the methods are meant to operate directly against the "remote" database, once we have the information necessary to establish a direct JDBC connection to it. I did not want the overhead of a proxied call for any of these methods. There are some services (such as locking) where we have to coordinate with the remote server, and these are done via proxy objects (not proxy in the reflection sense, but proxy nonetheless).

**#58 - 04/22/2021 03:11 AM - Igor Skornyakov**

Eric Faulhaber wrote:

> I don't understand what a proxy for a Database object would do. Database objects are meant primarily as informational objects and convenient hash keys for information that needs to be mapped by physical database. There are no services a Database object provides that could usefully be called by proxy to do some remote work.
>
> I implemented RemotePersistence as a subclass of, rather than a proxy for Persistence, because most of the methods are meant to operate directly against the "remote" database, once we have the information necessary to establish a direct JDBC connection to it. I did not want the overhead of a proxied call for any of these methods. There are some services (such as locking) where we have to coordinate with the remote server, and these are done via proxy objects (not proxy in the reflection sense, but proxy nonetheless).

I see. Well, subclassing is the solution of course. Thank you for the clarification.

**#59 - 04/22/2021 04:04 AM - Eric Faulhaber**

Here's a crazy idea which I probably should not be posting at my current level of fatigue, but here goes...

Can we not simply think of the single-user mode as a kind of database-scoped exclusive lock? If any other session (local or remote) already is using the database in multi-user mode, they essentially hold a share lock, and no session can acquire the single-user, "exclusive" lock. However, if there is no session using the database in multi-user mode, no-one is holding a "share" lock on the database, and the session requiring single-user mode can acquire the exclusive lock. At that point, no other session can acquire a share (or an exclusive) lock, until the single-user mode session is finished and releases the exclusive lock.

Using this approach allows us to use all our existing infrastructure, including the remote lock manager proxy. We would just need to come up with a unique lock identifier which would never be used as a table or record identifier. The lock acquisition would have to happen as soon as possible after the LockManager (remote or local) is available (see [Remote]Persistence.createLockManager, before the ConnectionManager connection is finalized. The database-scope lock, whether share or exclusive, would need to be acquired in NO-WAIT mode, and if a LockUnavailableException is thrown, we report the appropriate error and unwind any necessary ConnectionManager state (if any at that point).

Of course, the lock must be released upon disconnection (normal or abnormal).

TBH, I'm too tired at the moment to make sure this design is an exact analogy for the requirements as we've worked them out so far, but please give it some thought...

**#60 - 04/22/2021 04:10 AM - Igor Skornyakov**

Eric Faulhaber wrote:

> Here's a crazy idea which I probably should not be posting at my current level of fatigue, but here goes...
>
> Can we not simply think of the single-user mode as a kind of database-scoped exclusive lock? If any other session (local or remote) already is using the database in multi-user mode, they essentially hold a share lock, and no session can acquire the single-user, "exclusive" lock. However, if there is no session using the database in multi-user mode, no-one is holding a "share" lock on the database, and the session requiring single-user mode can acquire the exclusive lock. At that point, no other session can acquire a share (or an exclusive) lock, until the single-user mode session is finished and releases the exclusive lock.
>
> Using this approach allows us to use all our existing infrastructure, including the remote lock manager proxy. We would just need to come up with a unique lock identifier which would never be used as a table or record identifier. The lock acquisition would have to happen as soon as possible after the LockManager (remote or local) is available (see [Remote]Persistence.createLockManager, before the ConnectionManager connection is finalized. The database-scope lock, whether share or exclusive, would need to be acquired in NO-WAIT mode, and if a LockUnavailableException is thrown, we report the appropriate error and unwind any necessary ConnectionManager state (if any at that point).
>
> Of course, the lock must be released upon disconnection (normal or abnormal).
>
> TBH, I'm too tired at the moment to make sure this design is an exact analogy for the requirements as we've worked them out so far, but please give it some thought...

Please note that the "-1" option is supposed to be silently ignored if there is another "normal" active connection. I'm not sure that it matches the semantics of the lock.

**#61 - 04/22/2021 05:07 AM - Eric Faulhaber**

Igor Skornyakov wrote:

> Please note that the "-1" option is supposed to be silently ignored if there is another "normal" active connection. I'm not sure that it matches the semantics of the lock.

That can be emulated by silently downgrading to a share lock if an exclusive lock attempt fails, instead of reporting an error. A failure to acquire a share lock OTOH means some other session holds an exclusive lock and thus is running in single user mode.

**#62 - 04/22/2021 06:32 AM - Igor Skornyakov**

Eric Faulhaber wrote:

Igor Skornyakov wrote:

> Please note that the "-1" option is supposed to be silently ignored if there is another "normal" active connection. I'm not sure that it matches the semantics of the lock.

That can be emulated by silently downgrading to a share lock if an exclusive lock attempt fails, instead of reporting an error. A failure to acquire a share lock OTOH means some other session holds an exclusive lock and thus is running in single user mode.

Well, this sounds like a really nice solution. Thinking about this. Thank you!

#### #63 - 04/22/2021 07:13 AM - Greg Shah

I like it.  It is conceptually correct.

#### #64 - 04/23/2021 04:00 AM - Greg Shah

I've been thinking about this task some more.  The purpose of this is to enable maintenance programs to be supported.  The thing that I don't like is that there is no guarantee that your maintenance program will run when other programs are not running or trying to run.  If another program "gets in first", then the maintenance program will think that it can run (it will connect) but it won't be safe.  If the 4GL behavior did not allow connection if the multi-user database server was active, then this would be safer.

I do wonder if this behavior was the result of including the TCP connectivity parameters in the same connect string as the -1.  If the session just tried to open up the database privately when the multi-user database was already controlling the database files, then I suspect that the -1 connection would have failed.

Regardless of that, I think we need to take a different approach.  Although our primary objective is to implement compatability, in this case I think we may want to avoid the implementation.  The result was getting too complex for something that won't even be a fully safe solution.

Igor: For this task, just implement a log entry (level WARNING) when the -1 option is passed in a CONNECT statement.  Otherwise, ignore this option.  The text should be "CONNECT '-1' option ignored in '%s' option string." where the %s is the full CONNECT option string.

I will add another task for a optional "maintenance program" which (if it is configured) is executed when the FWD server starts such that no other sessions are allowed to start up until this program completes.

#### #65 - 04/23/2021 04:01 AM - Greg Shah

*- Related to Feature #5279: add support for an optional maintenance program that executes before any other sessions are allowed to run added*

#### #66 - 04/23/2021 04:48 AM - Igor Skornyakov

Greg Shah wrote:

> I've been thinking about this task some more.  The purpose of this is to enable maintenance programs to be supported.  The thing that I don't like is that there is no guarantee that your maintenance program will run when other programs are not running or trying to run.  If another program "gets in first", then the maintenance program will think that it can run (it will connect) but it won't be safe.  If the 4GL behavior did not allow connection if the multi-user database server was active, then this would be safer.

I think that with the lock-based approach suggested by Eric, this can be resolved just by not ignoring the error in the attempt to acquire the EXCLUSIVE-LOCK and throwing error 276. This is what 4GL does if one tries to connect to a database by its filename when this database is managed by the broker. According to my tests, connecting to a database as a file is the only way to work with it in a single-user mode.

> I do wonder if this behavior was the result of including the TCP connectivity parameters in the same connect string as the -1.  If the session just tried to open up the database privately when the multi-user database was already controlling the database files, then I suspect that the -1 connection would have failed.
>
> Regardless of that, I think we need to take a different approach.  Although our primary objective is to implement compatability, in this case I think we may want to avoid the implementation.  The result was getting too complex for something that won't even be a fully safe solution.
>
> Igor: For this task, just implement a log entry (level WARNING) when the -1 option is passed in a CONNECT statement.  Otherwise, ignore this option.  The text should be "CONNECT '-1' option ignored in '%s' option string." where the %s is the full CONNECT option string.
>
> I will add another task for a optional "maintenance program" which (if it is configured) is executed when the FWD server starts such that no other sessions are allowed to start up until this program completes.

OK. Thank you.

#### #67 - 04/23/2021 04:59 AM - Greg Shah

If we decide to implement this in the future, we will implement with the error 276 as you note.  I think that is much better.

Since there is still the "race condition" with other sessions, I still think it is better to pursue #5279.  Go ahead with the warning log entry for now.

#### #68 - 04/23/2021 05:02 AM - Igor Skornyakov

Greg Shah wrote:

> If we decide to implement this in the future, we will implement with the error 276 as you note.  I think that is much better.
>
> Since there is still the "race condition" with other sessions, I still think it is better to pursue #5279.  Go ahead with the warning log entry for now.

OK.

#### #69 - 04/23/2021 05:53 AM - Igor Skornyakov

Since the CONNECT statement can contain information about multiple databases I've added the log message right where the lock should be acquired.

Committed to 3821c/12313.

#### #70 - 04/26/2021 08:21 AM - Eric Faulhaber

Code review 3821c/12313:

Warning added, functionally looks fine, but please follow coding standards when formatting (e.g., breaking long lines), so FWD source looks consistent across the project.

#### #71 - 04/26/2021 09:04 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Warning added, functionally looks fine, but please follow coding standards when formatting (e.g., breaking long lines), so FWD source looks consistent across the project.

Sorry. Fixed in 3821c/12331.

#### #72 - 04/26/2021 05:06 PM - Eric Faulhaber

*- Status changed from WIP to Closed*

*- % Done changed from 0 to 100*

I did some additional code cleanup in 3821c/12333. Considering #5279, this issue can be closed.