Database - Feature #3757

add metadata support

10/21/2018 02:39 PM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Feature #3814: more schema metadata	**Closed**

History

#1 - 10/21/2018 02:39 PM - Greg Shah

Add support for the following fields:

```
_database-feature._dbfeature_enabled
_database-feature._dbfeature-id
_connect._connect-type
_filelist._filelist-name
_connect._connect-disconnect
_connect._connect-transid
_connect._connect-server
_startup._startup-locktable
_lock._lock-chain
_connect._connect-batch
```

#2 - 11/26/2018 01:48 AM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

Ovidiu, as you have down time from other tasks, please investigate and document the meanings of the above metadata tables/fields. We have partial implementations of the _connect and _lock tables, so we generally know what these do, though not necessarily how each of the new fields are used.

#3 - 11/26/2018 12:31 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

_filelist._filelist-name, _filelist._FileList-BlkSize and _filelist._fileList-id

This _meta table allow the 4GL programmer to get the database full filename. There may be multiple records for each database. The FIRST record seems to always be the 'main' .db file, like: c:\Users\Administrator\Documents\p2j_tests\dbs\p2j_test.db. The NEXT records represent the additional

data files (database areas ?), having same name, with a possible numeric suffix (_7, _12, etc) and different extensions: .b1, .d1, .b2, .d2, etc.
Additional info for the table:
https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm%2Fdatabase-file-status-(-filelist).html%23.

_lock._lock-chain

As described in https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm%2Flock-table-status-(-lock).html%23, this is the 'Chain number'.
I tried to do some investigations on this _meta table. The name suggests that we can find here references to all active locks. However, my queries returned 8192 records (as declared in _startup-locktable, see below), but all they contain are unknowns (?) for each field of the records, although the test procedure intentionally locked some records in all modes. I assume something went wrong with my testcase.

_startup._startup-locktable

This is the maximum size of locking table specified startup. This is the value specified by -L command-line parameter.

_database-feature._dbfeature_enabled and _database-feature._dbfeature-id

The _database-feature table is used by 4GL programmer to obtain the status of some database features. The known list is:

Id	Name	Observations
1	OpenEdge Replication	TBA
2	Failover Clusters	
3	DataXted Remote Edition	
4	Reserved	
5	Large Files	
6	Database Auditing	
7	JTA	
8	After Image Mangement/Archiver	
9	64 Bit DBKEYS	With FWD, rowid s are always 64 bit.
10	Large Keys (indexes)	In OpenEdge Release 10.1B the index key size was increased to 2000 bytes including storage overhead. This effected both the assignment of index key values and sorting using the BY option in ABL queries. Note that the above only applies to databases whose block size is above 2KB and of course temp-tables, whose block size is 4KB by default. Databases with 1K and 2K block sizes adhere to the previous index entry size of approximately 200 characters. **This key is set to "1" when the index key size is increased (only 1970 in fact), and "0" when the index key size is limited to 192 bytes.**
11	64 Bit Sequences	In all dialects used by FWD, sequences are 64 bit.
12	DataXted Integration Edition	TBA
13	Encryption	
14	Multi-tenancy	
15	Concurrent JTA and Replication	
16	Reserved	
17	MT Index Rebuild	
18	MT Data Move	
19	Roll Forward Restricted Mode	
20	TP Index Rebuild	
21	Table Partitioning	
22	Read-only Partitions	
23	New VST Tables	
24	Partition Move	
25	Partition Copy	

26	Authentication Gateway
27	Change Data Capture

Probably, for the beginning, it's best to find the list of the features an application is interested and implement support for those. Some of these might not be applicable, some of then are constants (like 64 Bit DBKEYS, Large Keys or 64 Bit Sequences) in FWD implementation.
The article that describes the table, does not describe the records:
https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm/database-features-(-database-feature).html.

_connect._connect-type, _connect._connect-disconnect, _connect._connect-transid, _connect._connect-server and _connect._connect-batch

This table allow the programmer to be aware of all database connection at a given moment. The table is described here:
https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dmadm%2Fdatabase-connection-(-connect).html%23.

Not very clear all of these records do:

Name	Description	Observations
_Connect-Disconnect	Displays a disconnect flag.	Found value 0. Irrelevant
_Connect-TransId	Displays the current transaction ID.	It seems that transactions have ID. Or at least a counter. Not aware if we have such support or what could be the exact use of it.
_Connect-Server	Identifies the server if the connection is remote.	This is an INTEGER value. Might be a boolean value (?/0/1) or a key to another meta table.
_Connect-Type	Displays the connection type: SELF, REMC, BROK, SERV, or BTCH.	
_Connect-Batch	Displays batch users status.	This is a CHAR field describing BATCH connections. For standard interactive connection the value is "No". Assuming it is "Yes" for BATCH connection. Anyway, the very same information can be obtained from _Connect-Type field, above.

#4 - 12/07/2018 02:50 PM - Ovidiu Maxiniuc

I updated the previous entries and I see that there are a few more metadata tables/fields that are accessed: _DbStatus._DbStatus-TotalBlks, _DbStatus_DbStatus-EmptyBlks, _DbStatus_DbStatus-FreeBlks, _DbStatus_DbStatus-RMFreeBlks.

I think they were missing from the initial scan because they are part of dynamic queries or accessed using derefrence operator (::), in which cases they are just strings not identifiers.

I will update this list as I notice other ones.

#5 - 12/07/2018 02:59 PM - Eric Faulhaber

Ovidiu, please implement the metadata tables/fields in this task. The priority right now (must be done by end of next week) is that conversion works (i.e., code which references these database resources will convert and compile properly). Unless it's trivial to implement as you go, runtime support will be added later.

Where a metadata field looks like something very specific to the Progress database implementation (e.g., all the fields you mentioned in your last post), it is probably being used in a support tool which would be a candidate for exclusion from conversion. PostgreSQL-specific DBA tools will be used with the new environment. Please review the context of these metadata uses and report the purpose of the programs in which they are referenced, to the extent you can determine this from the source code. We will discuss with the customer, if a program looks like an admin tool which should be excluded.

#6 - 12/07/2018 03:20 PM - Eric Faulhaber

Eric Faulhaber wrote:

> Please review the context of these metadata uses and report the purpose of the programs in which they are referenced, to the extent you can determine this from the source code.

To be clear, I am only interested in the subset of programs that look like admin tools. I don't need the purpose of programs which are part of the normal runtime use of the application.

#7 - 12/08/2018 11:30 AM - Constantin Asofiei

I have files failing with these metadata fields, and they are not in the list for this task:

```
mtc._Connect._connect-Usr
mtc._myConnection._MyConn-UserId
```

#8 - 12/08/2018 12:57 PM - Eric Faulhaber

Constantin Asofiei wrote:

> I have files failing with these metadata fields, and they are not in the list for this task:
> [...]

These fields are not in this task because they already are supported, though the support is implemented in the runtime. The only requirement for conversion should be that they are present in standard.df (and in the version of standard.df we are using for the POC, they are). So, there is some unexpected problem. What are the failures, exactly?

#9 - 12/08/2018 01:02 PM - Constantin Asofiei

Eric Faulhaber wrote:

> What are the failures, exactly?

The logs are in #3750-29

#10 - 12/08/2018 01:13 PM - Eric Faulhaber

I've added _connect and _myconnection to the standard section of p2j.cfg.xml in rev 17 of the customer application project. This will ensure these tables are included in the m0 conversion phase.

#11 - 01/02/2019 11:33 AM - Greg Shah

What runtime work is needed to complete this task?

#12 - 01/03/2019 03:07 PM - Eric Faulhaber

Greg Shah wrote:

> What runtime work is needed to complete this task?

I think the _database-feature and _startup._startup-locktable changes can be handled via metaschema.xml changes. The latter field is an implementation specific field; we don't place any specific limit to the number of locks in FWD. I think we can put the default initial value for the field into records of this table. The question for both tables is how many records of each table are there?

Based on Ovidiu's research in #3757-3, I think supporting the _filelist metadata table in FWD makes no sense and, in fact, I'm not sure how we would come up with a useful implementation. Every field in this table represents some information that is very specific to the Progress database implementation. There is no useful analog to this information in the FWD persistence implementation. The first 2 uses in the current project seem to be utility uses, which need to be replaced by other admin tools. The last may need to be replaced or edited to devise a new way to perform a similar operation.

The new _connect table fields will need to be integrated into the ConnectTableUpdater class. This may require some new runtime infrastructure (e.g., _connect._connect-transid, _connect._connect-batch, _connect._connect-server, etc.).

The _lock._lock-chain field would need to be integrated into the LockTableUpdater class. However, as this information is very specific to the Progress

database lock implementation and has no analog in the FWD persistence implementation, there is nothing useful we can store in this field. So, we probably will just store the default initial value. The uses of this information by nature would be DBA-type utilities, which should be replaced with tools specific to the new backing database(s).

#13 - 01/07/2019 04:05 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> I think the _database-feature and _startup._startup-locktable changes can be handled via metaschema.xml changes. The latter field is an implementation specific field; we don't place any specific limit to the number of locks in FWD. I think we can put the default initial value for the field into records of this table. The question for both tables is how many records of each table are there?

The _database-feature has the runtime support in MetadataManager.populateDatabaseFeatures in 3750a. In fact the single feature that is checked is the support for large index keys. This depends on the backing database (as you know MS SQL Server has a limit here) so the response to the question is returned after checking the P2JDialect.getIndexLengthLimit(). I gathered the other values and added them with their default values or the values appropriate to FWD (which are static can could have been be added via metadata.xml). I think that I covered all known records for this table. OTOH, the _startup is a *deprecated* VST in newer ABL. As documentation states, starting from OE 11.5, this VST is no longer updated. The recommendation for the ABL programmer is to use _DbParams VST. I think We should implement the latter one and use delegation when returning _startup values. Normally, there is only one record in _startup table but the number of records in _DbParams is variable: one for each legal database parameter (100+ for OE 11.6).

A common feature of these VST is that the value is READ-ONLY: after loading a record in the buffer the ABL programmer can change the fields, but there is not flush, as far as I can see. As result, when the changed record is re-retrieved after an apparent commit the old value is obtained, again. Note that no error or warning is issued whatsoever. I think FWD does not support this, yet.

> Based on Ovidiu's research in #3757-3, I think supporting the _filelist metadata table in FWD makes no sense and, in fact, I'm not sure how we would come up with a useful implementation. Every field in this table represents some information that is very specific to the Progress database implementation. There is no useful analog to this information in the FWD persistence implementation. The first 2 uses in the current project seem to be utility uses, which need to be replaced by other admin tools. The last may need to be replaced or edited to devise a new way to perform a similar operation.

In customer's project they are using this table for locating the database on disk. Beside presenting this information to user and other text data read directly from the associated log files (.lg) the application can obtain the physical database name to be used with external tools or create a backup of the database after making sure the database is disconnected/offline.

> The new _connect table fields will need to be integrated into the ConnectTableUpdater class. This may require some new runtime infrastructure (e.g., _connect._connect-transid, _connect._connect-batch, _connect._connect-server, etc.).

I haven't tested this yet, but I think this is READ-ONLY, too. Otherwise it would be very easy to disconnect a user: just drop its record from this table. The application uses

> The _lock._lock-chain field would need to be integrated into the LockTableUpdater class. However, as this information is very specific to the Progress database lock implementation and has no analog in the FWD persistence implementation, there is nothing useful we can store in this field. So, we probably will just store the default initial value. The uses of this information by nature would be DBA-type utilities, which should be replaced with tools specific to the new backing database(s).

Yes, I think this can be done. The RecordLockEvent received by LockTableUpdater seems to have enough info. However, I see a problem here: LockTableUpdater uses an AssociatedThread to process events in the background. This means the data it processes is grouped by context. In ABL this VDT collects information globally, from the whole database.

#14 - 01/08/2019 12:34 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Yes, I think this can be done. The RecordLockEvent received by LockTableUpdater seems to have enough info. However, I see a problem here: LockTableUpdater uses an AssociatedThread to process events in the background. This means the data it processes is grouped by context. In ABL this VDT collects information globally, from the whole database.

If lock event information needs to be collected that is specific to the locking context, it should be done when creating the lock event, before it is pushed onto the queue that later gets processed by the AssociatedThread. Ultimately, we need to rework this mechanism entirely, so we are not executing SQL for every lock event (it is way too resource-intensive). But that is not in scope for this task.

#15 - 01/11/2019 09:40 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Eric Faulhaber wrote:
>
> > I think the _database-feature and _startup._startup-locktable changes can be handled via metaschema.xml changes. The latter field is an implementation specific field; we don't place any specific limit to the number of locks in FWD. I think we can put the default initial value for the field into records of this table. The question for both tables is how many records of each table are there?
>
> The _database-feature has the runtime support in MetadataManager.populateDatabaseFeatures in 3750a. In fact the single feature that is checked is the support for large index keys. This depends on the backing database (as you know MS SQL Server has a limit here) so the response to the question is returned after checking the P2JDialect.getIndexLengthLimit(). I gathered the other values and added them with their default values or the values appropriate to FWD (which are static can could have been be added via metadata.xml). I think that I covered all known records for this table.

The current implementation causes a regression, in that the server will fail to start for any project which does not use the _database-feature table. This is due to this bit of code in MetadataManager.populateDatabaseFeatures:

```
      Helper h = helpers.get("DatabaseFeature");
      if (h == null)
      {
         throw new TypeNotPresentException("DatabaseFeature", null);
      }
```

This should not be a fatal error. Most applications we've seen do not use _database-feature, so the condition h == null should just result in skipping the code which populates this table. Are there other, new dependencies downstream that require this to be a fatal error? If so, these must be removed.

#16 - 01/11/2019 10:39 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> The current implementation causes a regression, in that the server will fail to start for any project which does not use the _database-feature table. [...] This should not be a fatal error. Most applications we've seen do not use _database-feature, so the condition h == null should just result in skipping the code which populates this table. Are there other, new dependencies downstream that require this to be a fatal error? If so, these must be removed.

Indeed. if the meta table is not available at conversion time but it is used at runtime then is a configuration issue.
I fixed the regression in revision 11449/3750a. There should not be any other dependencies. I did a quick test and the server acted normally.

#17 - 01/14/2019 08:31 AM - Ovidiu Maxiniuc

At the end of last week, I found the runtime support for following VST is currently missing from FWD:

- _ActIndex - offers a view on index activity: the number of entry finds, creates, and deletes operations, the number of locked entries removed, and the numbers of split and free blocks. All of these seem to be needed in customer code;
- _ActLock - gives a *'snapshot'* on lock table activity (the number of share, exclusive, upgrade, Rec Get, and redundant requests; the number of exclusive, Rec Get, share, and upgrade grants; the number of exclusive, Rec Get, share, and upgrade waits; the number of downgrades, transactions committed, cancelled requests, and database up time;
- _ActRecord - offers information about record activity, including the number of bytes created, deleted, read, and updated; the number of fragments created, deleted, read, and updated; the number of records created, deleted, read, and updated; the number of transactions committed; and database up time;
- _ActServer - server activity information, including the number of bytes sent and received, the number of messages sent and received, the number of queries received, the number of records sent and received, the number of query time slice switches, the number of transactions committed, and database up time;
- _ActSummary - gives general information about database activity, including the number of transactions committed and rolled back; the number of records read, updated, created, and deleted; the number of record locks and waits; the number of database reads and writes; before-image and after-image information; and buffer information.
- _Servers - use this VST to obtain the status of each OE server running on the system, including the server number, process ID, and type; the protocol used; the number of logins and current users; the maximum number of users; and the server's port number;
- _Trans - offers information such as transaction number, state, start time, duration, user number, coordinator name, and transaction;
- _UserIO - gives information about the database input/output operations, including user number and name and the number of accesses, reads, and writes.
- _File-trig - seems to be used for obtaining a list of triggers (I believe this is where ABL stores some information about the triggers which I was not aware of when I implemented db trigger support). I found no description in the online manuals for this VST.

Notes:

- the _Act VST offer statistical information on the activity of some idioms;
- apparently the information from _Act VSTs is collected and just displayed for some administration purposes. Most liekely, it might not be critical for the stability of the application;
- there are some duplicate information. The same data is present in one or many of these VSTs;
- the access to all all _Act VSTs are time-critical. I started implementing support for 2nd using meta table support but it's very likely it will have a serious performance hit. After second thought I wonder whether we could create some in-memory statistics and serve them directly. The problem is when the access is part of a more complex query.
- another issue is with R/O feature of the buffers. In my previous commit I tried to skip the normal flush for VSTs so any eventual change in the buffer to be lost. However, the caching of Hibernate will still make the the changed fields to 'survive' the un-flushed release operation. At the time I write these I realize that the CREATE/DELETE operations might also have some constraints;
- for some VST the values returned are irrelevant because of the backing architecture. These cases need to be studied individually how can we emulate them or return data that makes sense in FWD environment;
- the _File-trig could be populated at server startup, when the triggers are initialized.

#18 - 01/14/2019 12:28 PM - Greg Shah

Most (if not all) of these missing metadata items seem to be for admin/monitoring/debugging/profiling purposes. I don't see much here that has a legitimate business purpose.

We should discuss (in the customer project) which programs use these features and whether they are needed for the current deadline.

#20 - 01/15/2019 12:45 PM - Ovidiu Maxiniuc

- File om_upd20190115a.zip added

The attached code was delayed and dropped from working branch as _ActLock is not needed for the current deadline.

#21 - 05/06/2020 02:17 PM - Greg Shah

- Related to Feature #3814: more schema metadata added

#22 - 05/07/2020 04:59 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> The _database-feature has the runtime support in MetadataManager.populateDatabaseFeatures in 3750a.

I understand that MetadataManager.populateDatabaseFeatures (see #3757-13) was removed. What was the reason for this?

I'm asking because I'm implementing the(table-driven) approach based on MetadataManager.populate<MetaTable> instead of current MetadataManager.populateDatabase based on xxx.meta.xml because Greg suggestion to populate SCHEMA non-VST tables based on the actual state of the database schema. I understand that MetadataManager.populateDatabase and xxx.meta.xml should be obsolete now.
Is my understanding correct?
Thank you.

#23 - 05/07/2020 06:16 PM - Greg Shah

I had not planned for you to make everything "live" yet. For this current task, it is OK to extend the current static approach to cover the fields and tables needed.

As to your questions above, we will need input from Ovidiu/Eric.

#24 - 05/07/2020 06:24 PM - Igor Skornyakov

Greg Shah wrote:

> I had not planned for you to make everything "live" yet. For this current task, it is OK to extend the current static approach to cover the fields and tables needed.
>
> As to your questions above, we will need input from Ovidiu/Eric.

I have no plans to make "live" **everything**. Just schema tables that is in the scope of #3814. But the approach is extensible. One just has to define and implement populate<MetaTable> method.

#25 - 05/07/2020 07:17 PM - Ovidiu Maxiniuc

Igor,
I do not remember exactly why the _database-feature was removed. It seems to me that, since work on _ActLock was delayed, I extracted all metadata-related from the trunk, but somehow I failed to add it to om_upd20190115a.zip attached to this task. The MetadataManager.populateDatabaseFeatures method reached the trunk in r11298 but reverted in the r11299 because it was a "duplicated code from parallel branch". Use bzr diff -c11298 src/com/goldencode/p2j/persist/meta/MetadataManager.java to get a patch of that changes.

#26 - 05/08/2020 03:20 AM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor,
> I do not remember exactly why the _database-feature was removed. It seems to me that, since work on _ActLock was delayed, I extracted all metadata-related from the trunk, but somehow I failed to add it to om_upd20190115a.zip attached to this task. The MetadataManager.populateDatabaseFeatures method reached the trunk in r11298 but reverted in the r11299 because it was a "duplicated code from parallel branch". Use bzr diff -c11298 src/com/goldencode/p2j/persist/meta/MetadataManager.java to get a patch of that changes.

I see. Thank you, Ovidiu.

Files

om_upd20190115a.zip	10.5 KB	01/15/2019	Ovidiu Maxiniuc