GemStone®

GemConnect Release Notes

Version 2.0

January 2007

GemStone™

INTELLECTUAL PROPERTY OWNERSHIP

This documentation is furnished for informational use only and is subject to change without notice. GemStone Systems, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation.

This documentation, or any part of it, may not be reproduced, displayed, photocopied, transmitted, or otherwise copied in any form or by any means now known or later developed, such as electronic, optical, or mechanical means, without express written authorization from GemStone Systems, Inc.

Warning: This computer program and its documentation are protected by copyright law and international treaties. Any unauthorized copying or distribution of this program, its documentation, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted under the maximum extent possible under the law.

The software installed in accordance with this documentation is copyrighted and licensed by GemStone Systems, Inc. under separate license agreement. This software may only be used pursuant to the terms and conditions of such license agreement. Any other use may be a violation of law.

Use, duplication, or disclosure by the Government is subject to restrictions set forth in the Commercial Software - Restricted Rights clause at 52.227-19 of the Federal Acquisitions Regulations (48 CFR 52.227-19) except that the government agency shall not have the right to disclose this software to support service contractors or their subcontractors without the prior written consent of GemStone Systems, Inc.

This software is provided by GemStone Systems, Inc. and contributors “as is” and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall GemStone Systems, Inc. or any contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

COPYRIGHTS

PATENTS

GemStone is covered by U.S. Patent Number 6,256,637 “Transactional virtual machine architecture”, Patent Number 6,360,219 “Object queues with concurrent updating”, and Patent Number 6,567,905 “Generational Garbage Collector”. GemStone may also be covered by one or more pending United States patent applications.

TRADEMARKS

GEMSTONE™, **GemBuilder**, and the GemStone logo are trademarks or registered trademarks of GemStone Systems, Inc. in the United States and other countries.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Oracle is a registered trademarks of Oracle Corporation. **Oracle9i** is a trademark of Oracle Corporation.

Sun, **Sun Microsystems** and **Solaris** are trademarks or registered trademarks of Sun Microsystems, Inc. All **SPARC** trademarks, including **SPARCstation**, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation is licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Linux is a registered trademark of Linus Torvalds and others.

Red Hat and all **Red Hat**-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

AIX and **POWER4** are trademarks or registered trademarks of International Business Machines Corporation.

Microsoft, **MS**, **Windows, Windows XP** and **Windows 2000** are registered trademarks of Microsoft Corporation in the United States and other countries.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective owners. Specifications are subject to change without notice. All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized. GemStone cannot attest to the accuracy of this information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

GemStone Systems, Inc.
1260 NW Waterhouse Avenue, Suite 200
Beaverton, OR 97006

Preface

About the Documentation

These release notes describe the changes in the GemConnect version 2.0 release. We recommend that everyone using GemConnect read these release notes before beginning installation or development. These release notes are also available on the GemStone customer support website, as described in the next section.

For information on installing or upgrading to this version of GemConnect, please refer to the *GemConnect Installation Guide*.

Typographical Conventions

This document uses the following typographical conventions:

- Commands issued at a command prompt are shown in **bold** typeface. For example:

  copydbf

- Smalltalk methods, GemStone environment variables, file names and paths, listings, and prompts are shown in `monospace` typeface. For example:

  ```
  markForCollection
  ```

- Place holders that are meant to be replaced with real values are shown in *italic* typeface. For example:

  StoneName.conf

Technical Support

GemStone provides several sources for product information and support. The product-specific manuals and online help provide extensive documentation, and should always be your first source of information. GemStone Technical Support engineers will refer you to these documents when applicable.

GemStone Web Site: http://support.gemstone.com

GemStone's Technical Support website provides a variety of resources to help you use GemStone products. Use of this site requires an account, but registration is free of charge. To get an account, just complete the Registration Form, found in the same location. You'll be able to access the site as soon as you submit the web form.

The following types of information are provided at this web site:

Help Request allows designated support contacts to submit new requests for technical assistance and to review or update previous requests.

Documentation for GemConnect is provided in PDF format. This is the same documentation that is included with your GemConnect product.

Release Notes and **Install Guides** for your product software are provided in PDF format in the Documentation section.

Downloads and **Patches** provide code fixes and enhancements that have been developed after product release. Most code fixes and enhancements listed on the GemStone Web site are available for direct downloading.

Bugnotes, in the Learning Center section, identify performance issues or error conditions that you may encounter when using a GemStone product. A bugnote describes the cause of the condition, and, when possible, provides an alternative means of accomplishing the task. In addition, bugnotes identify whether or not a fix is available, either by upgrading to another version of the product, or by applying a patch. Bugnotes are updated regularly.

TechTips, also in the Learning Center section, provide information and instructions for topics that usually relate to more effective or efficient use of GemStone products. Some Tips may contain code that can be downloaded for use at your site.

Community provides customer forums for discussion of GemStone product issues.

Technical information on the GemStone Web site is reviewed and updated regularly. We recommend that you check this site on a regular basis to obtain the latest technical information for GemStone products. We also welcome suggestions and ideas for improving and expanding our site to better serve you.

You may need to contact Technical Support directly for the following reasons:

- Your technical question is not answered in the documentation.
- You receive an error message that directs you to contact GemStone Technical Support.
- You want to report a bug.
- You want to submit a feature request.

Questions concerning product availability, pricing, keyfiles, or future features should be directed to your GemStone account manager.

When contacting GemStone Technical Support, please be prepared to provide the following information:

- Your name, company name, and GemStone/S license number
- The GemStone product and version you are using
- The hardware platform and operating system you are using
- A description of the problem or request

- Exact error message(s) received, if any

Your GemStone support agreement may identify specific individuals who are responsible for submitting all support requests to GemStone. If so, please submit your information through those individuals. All responses will be sent to authorized contacts only.

For non-emergency requests, the support website is the preferred way to contact Technical Support. Only designated support contacts may submit help requests via the support website. If you are a designated support contact for your company, or the designated contacts have changed, please contact us to update the appropriate user accounts.

Email: support@gemstone.com

Telephone: (800) 243-4772 or (503) 533-3503

Requests for technical assistance may also be submitted by email or by telephone. We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system that is non-operational. In these cases, please also submit your request via the web or email, including pertinent details such error messages and relevant log files.

If you are reporting an emergency by telephone, select the option to transfer your call to the technical support administrator, who will take down your customer information and immediately contact an engineer.

Non-emergency requests received by telephone will be placed in the normal support queue for evaluation and response.

24x7 Emergency Technical Support

GemStone offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, if they encounter problems that cause their production application to go down, or that have the potential to bring their production application down. For more details, contact your GemStone account manager.

Training and Consulting

Consulting and training for all GemStone products are available through GemStone's Professional Services organization.

- Training courses are offered periodically at GemStone's offices in Beaverton, Oregon, or you can arrange for onsite training at your desired location.
- Customized consulting services can help you make the best use of GemStone products in your business environment.

Contact your GemStone account representative for more details or to obtain consulting services.

Contents

Chapter 1. GemConnect 2.0 Release Notes

Chapter 1

GemConnect 2.0 Release Notes

GemConnect version 2.0 is a new release of the GemConnect product, including significant new features, adding performance improvements, and adding support for GemStone/S 64 Bit, the 64-Bit server product. This version of GemConnect supports both GemStone/S 6.x and GemStone/S 64 Bit versions 1.x and 2.x.

These release notes provide details of the changes in this release. Please take time to read through them before installing the product, to acquaint yourself with the changes. This release also provides updated documentation.

To install GemConnect version 2.0, follow the instructions in the *GemConnect Installation Guide*.

Overview

Changes in GemConnect version 2.0 include:

- Use of the more recent (8.0 and later) Oracle OCI API protocol. This improved API is used to allow support for some of the new features; it also required minor changes to GemConnect error handling.
- Ability to read UTF8 and UTF16 Oracle data.
- Added ability to use variable binding in SQL statements.
- Added ability to re-use prepared SQL statements.
- Mechanisms added to support Oracle Array processing.

Support for some of these new features is via a new GemConnect Class, **`GsRdbWriteStream`**. This class provides a write stream object that buffers the data, and uses Oracle Array processing to execute batches of Oracle SQL in a single round trip. This provides the ability to perform SQL DELETE, INSERT, and UPDATE statements using variable binding and prepared SQL statements.

To be able to support the new Oracle OCI API plus the new `GsRdbWriteStream` functionality, a number of minor changes were made to existing GemConnect classes.

New Features and Changes

Support for accessing Oracle UTF8/UTF16 data.

GemConnect can now automatically map character data from data stored in Oracle as UTF8 or UTLF16 to GemStone String and DoubleByteString objects.

To support this, a new instance variable, **`charConversion`**, has been added to the GsOracleConnection class, and associated accessor methods charConversion and charConversion:. This variable can be set to one of the following values:

Table 1 charConversion valid values

nil	no conversion (character data mapped directly into / from GemStone).
#UTF8	character data converted from UTF8 into GS Strings or DoubleByteStrings
#UTF16	character data converted from UTF16 into GS Strings or DoubleByteStrings

A new error, #typeConversionError, has been added, which is generated if there is a problem with conversion:

typeConversionError - Cannot convert GS Object for Oracle column, connection: <conn> stream: <stream> details: <details array>

Note that you need to set the environment $NLS_LANG to the appropriate UTF character set when using the charConversion feature. For example:

C shell:

```
% setenv NLS_LANG=AMERICAN_AMERICA.UTF8
```

Bourne shell or Korn shell:

```
$ NLS_LANG=AMERICAN_AMERICA.UTF8
$ export NLS_LANG
```

Added class GsRdbWriteStream

`GsRdbWriteStream` is a subclass of `GsRdbReadStream`, and as the name implies, is a write stream with behavior the inverse of the `GsRdbReadStream`. New methods on class `GsOracleConnection` (described below) similar to the original #openCursorOn: methods are used to create a `GsRdbWriteStream` instance that performs either an Oracle INSERT, DELETE, or UPDATE operation. The `GsRdbWriteStream` instance includes a buffer for holding entries waiting to be written to the Oracle database, sized according to the connection batchSize at the time the stream is created. An entry is added to the buffer by sending the message:

aGsRdbWriteStream `nextPut:` *tupleInstance*

or a collection of tupleInstances can be iteratively added by:

aGsRdbWriteStream `nextPutAll:` *collectionOfTupleInstances*

The contents of the buffer are flushed to the Oracle database when the buffers are full, when an Oracle commit is performed, or if manually specified by a `flush` command.

A `GsRdbWriteStream` can also specify a dependency list, which is a list of other streams that must be flushed before this one can be flushed. The methods are described in "Other GsRdbWriteStream Methods" on page 1-5.

You may also clear the buffer without writing the contents to Oracle, see the `clear` method described under"GsRdbWriteStream Major Methods" on page 1-5.

The methods for creating an instance of `GsRdbWriteStream` are similar to the `openCursorOn:` methods used for generated a `GsRdbReadStream`. Like `openCursorOn:`, the methods `rdbTableName`, `rdbColumnMapping`, and `rdbPrimaryKeyMaps` on the tupleClass are used to determine the mapping between the GemStone tupleClass and the Oracle DB table. And like `openCursorOn:`, there are variations that allow you to override the default table name, column mapping, or primary key mapping when useful.

There are separate methods for handling INSERT, DELETE, and UPDATE SQL requests. These are the added GsOracleConnection methods:

`openInsertCursorOn:` *tupleClass*
`openInsertCursorOn:` *tupleClass* `columnMapping:` *colMap*
`openInsertCursorOn:` *tupleClass* `columnMapping:` *colMap* `tableName:` *name*

`openDeleteCursorOn:` *tupleClass*
`openDeleteCursorOn:` *tupleClass* `keyMapping:` *keyMap*
`openDeleteCursorOn:` *tupleClass* `keyMapping:` *keyMap* `tableName:` *name*

`openUpdateCursorOn:` *tupleClass*
`openUpdateCursorOn:` *tupleClass* `columnMapping:` *colMap*
`keyMapping:` *keyMap*
`openUpdateCursorOn:` *tupleClass* `columnMapping:` *colMap*
`keyMapping:` *keyMap* `tableName:` *name*

Entries in the columnMaps and keyMaps must be ordered in the same sequence in which they appear in the Oracle table. So for example, if you have an Oracle table with rows in this order:

```
('ID', 'LAST_NAME', 'FIRST_NAME', 'ADDRESS')
```

Your column map information must order them like:

```
#((ID id id id:)
(LAST_NAME lastName lastName lastName:)
(FIRST_NAME firstName firstName firstName:)
(ADDRESS address address address:))
```

In the case of `openUpdateCursorOn:*`, all the entries in the *keyMap* must also be present in the *colMap*.

You can also supply a tupleClass of Array, in which case you only supply abbreviated column map information with two elements:

```
#((ID id)(LAST_NAME lastName)(FIRST_NAME firstName)
(ADDRESS address))
```

When you pass an Array to `nextPut:`, the elements in the array are mapped first-to-last as specified by the column mapping.

Examples:

To insert data for objects contained in TupleInstanceCollection:

```
writeStream := conn openInsertCursorOn: TupleClass.
writeStream nextPutAll: TupleInstanceCollection.
conn commitTransaction.  "This also makes sure that writeStream
is flushed"
```

To delete data for objects contained in TupleInstanceCollection:

```
writeStream := conn openDeleteCursorOn: TupleClass.
writeStream nextPutAll: TupleInstanceCollection.
conn commitTransaction.  "This also makes sure that writeStream
is flushed"
```

To delete data using arrays and just the key “ID” for ID = 1001, 1002, 1003:

```
writeStream := conn
   openDeleteCursorOn: Array
   keyMapping: #((ID id)).
writestream nextPut: #(1001).
writestream nextPut: #(1002).
writestream nextPut: #(1003).
conn commitTransaction.
```

or alternatively:

```
writeStream := conn
   openDeleteCursorOn: Array
   keyMapping: #((ID id)).
writestream nextPutAll: #((1001)(1002)(1003)).
conn commitTransaction.
```

To update the column “ADDRESS”, using “ID” as the key, for a bunch of TupleInstances in TupleInstanceCollection:

```
writeStream := conn
   openUpdateCursorOn: TupleClass
   columnMapping: #((ID id id id:)(ADDRESS address address
address:))
   keyMapping: #((ID id id id:)).
writeStream nextPutAll: TupleInstanceCollection.
conn commitTransaction.  "This also makes sure that writeStream
is flushed"
```

Here is a variation for UPDATE that uses Arrays:

```
writeStream := conn
   openUpdateCursorOn: Array
   columnMapping: #((ID id)(ADDRESS address))
   keyMapping: #((ID id id:)).
writeStream nextPut: #(101 '123 West 12th Ave').
writeStream nextPut: #(102 '707 North Elm St').
conn commitTransaction.  "This also makes sure that writeStream
is flushed"
```

Note that for `openUpdateCursorOn:*`, the columns specified in the keyMap must also be included in the columnMap.

GsRdbWriteStream Major Methods

The following are the major methods used to manipulate write streams:

`nextPut:` *tupleInstance*
: Write the contents of *tupleInstance* to the stream, as defined by the various mapping specifications.

`nextPutAll:` *tupleInstanceCollection*
: Write the contents of multiple tupleInstances in the *tupleInstanceCollection* to the stream, as defined by the various mapping specifications.

`flush`
: Flush the contents of the write stream buffer to the Oracle DB. Flushes happen automatically when the buffer is filled, and when you do:
```
conn commitTransaction
```

`clear`
: Clear out the contents of the write stream buffer without writing the contents to the Oracle DB. Clears happen automatically when you do:
```
conn rollbackTransaction
```

`free`
: Free up and release all memory and buffers held by this stream. Should always be done when you are finished using a particular write (or read) stream. Free also performs a flush, so you should clear the buffer first if you don't want the contents flushed to the Oracle DB.

Other GsRdbWriteStream Methods

Here are several other GsRdbWriteStream methods that may prove useful. Some of these are provided to be consistent with GsRdbReadStreams.

`position`
: Reports the incremental position for the stream, starting at 0 when the buffer is first created, and then incremented each time an entry is added to the stream. The position will decrement appropriately when a clear is done, or when a “conn rollbackTransaction” erases the effects of earlier write stream writes.

`atEnd`
: Reports if we're at the end of a stream. For write streams this is always true.

`setToEnd`
Sets the buffer pointer to the end of the stream. Since we're always at the end on a write stream, this is a no-op.

`dependencyList`
Returns the dependency list for this stream. This is either nil, or an IdentitySet of other GsRdbWriteStreams which must be flushed before this stream can be flushed.

`addDependency:` *<aGsRdbWriteStream>*
Adds this stream to the dependencyList.

`removeDependency:` *<aGsRdbWriteStream>*
Removes this stream from the dependencyList.

GsRdbReadStream Methods not available in GsRdbWriteStream

The following are methods defined in `GsRdbReadStream` that cannot be used in a `GsRdbWriteStream`:

```
next
next:
next:into:
nextResultSet
nextResultSetWith:
nextResultSetWith:columnMapping:
skip:
upToEnd:
upToEndInto:
```

Buffering Behavior

Because of the buffering behavior of `GsRdbWriteStream`, you need to be careful to avoid either loosing data, or writing data to Oracle that you did not intend. In particular, ensure that you use `flush` or `clear` whenever appropriate.

GemConnect assists in this by performing a `flush` on all `GsRdbWriteStream` instances associated with a connection when you commit a transaction:

```
conn commitTransaction
```

and performs a clear on all associated `GsRdbWriteStream` instances when you do a rollback:

```
conn rollbackTransaction
```

but if you manually perform an Oracle commit or rollback, you will need to perform these operations as well. For example:

```
writestream flush.
conn executeNoResults: 'commit'.
```

or:

```
writestream clear.
conn executeNoResults: 'rollback'.
```

The order that GsRdbWriteStreams are flushed during a commitTransaction is the reverse order of when they were created - newest streams are flushed first. You can override this order by using the stream's dependencyList. Any streams listed in the dependencyList will be flushed prior to the flush of the primary stream.

Another side effect of buffering is that when GemConnect finally does the write to Oracle, there may be multiple errors generated for different objects. Error information is now returned differently to accommodate multiple errors. See the description below for details. Recovering from these multiple errors may be tricky and require the application to remember internally information about these past operations so that they can be replayed after a flush error. In some cases it might be simpler to forego the performance improvement that buffering provides and to use a batchSize of 1. This will cause the Oracle write to occur immediately after each object is put into the writeStream via a nextPut: or nextPutAll: call.

Internal Methods

Here are some of the internal methods used to implement the functionality described above. You normally should not need to use these methods, although they may prove useful under special circumstances.

GsRdbConnection Class methods to generate SQL:

```
generateBindSQLDeleteForTable: tableName keys: keyNames

GsRdbConnection Class >> generateBindSQLInsertForTable: tableName
   columns: colNames

GsRdbConnection Class >> generateBindSQLUpdateForTable: tableName
   columns: colNames keys: keyNames

GsRdbConnection Class >> generateBindSQLSetClause: colNames

GsRdbConnection Class >> generateBindSQLValuesClause: colNames

gGsRdbConnection Class >> generateBindSQLWhereClause: keyNames
```

These methods generate the SQL statements used in the various forms of openInsertCursorOn:*, openDeleteCursorOn:*, and openUpdateCursorOn:*.

The major difference over the generateSQL* methods in earlier releases is that these new methods provide “bind” variables.

For example:

```
GsRdbConnection generateBindSQLInsertForTable: 'MYTABLE'
   columns: #(ID FIRST_NAME LAST_NAME ADDRESS)
```

would generate the SQL statement:

```
'INSERT INTO MYTABLE (ID FIRST_NAME LAST_NAME ADDRESS)
   VALUES (:ID :FIRST_NAME :LAST_NAME :ADDRESS)'
```

OtherGsRdbConnection Class method:

```
openWriteCursorOn: sqlString tupleClass: tupleClass
   columnMapping: colMap bindInfo: bindInfoArray
```

This low-level method is used in the various openInsertCursorOn:*, openDeleteCursorOn:*, and openUpdateCursorOn:* methods described earlier. *bindInfoArray* is an array where element 1 is the name of the relational database table and elements 2..n are the names of columns in the table to be bound to.

New Errors

There have been a number of new errors added to GemConnect. They include:

invalidTupleInstance - a tuple was passed to a write stream that is not the expected class, connection: <*conn*> stream: <*stream*> tuple: <*tuple*>
Arg1: <*conn*> a GsRdbConnection
Arg2: <*stream*>a GsRdbReadStream or GsRdbWriteSTream
Arg3: <*tuple*> the TupleClass associated with the read / write stream

invalidSql - Invalid SQL statement for cursored execution, connection: <*conn*> query: <*sql statement*>
Arg1: <*conn*> a GsRdbConnection
Arg2: <*sql statement*> a String containing an SQL statement

As mentioned earlier, it is now inappropriate to use #openCursorOn: with fully qualified non-SELECT SQL statements. Doing so will generate this error:

invalidTableName - the table <*tablename*> does not exist in this relational database, connection: <*conn*>
Arg1: a GsRdbConnection
Arg2: a String containing the name of an Oracle table

columnBindingError - could not bind all columns in columnMap with relational table, connection: <*conn*> query: <*sql statement*>
Arg1: <*conn*>a GsRdbConnection
Arg2: <*sql statement*>a String containing an SQL statement

flushError - problem writing to relational database during write stream flush, connection: <*conn*> stream: <*stream*> details: <*details*>
Arg1: <*conn*>a GsRdbConnection
Arg2: <*stream*>a GsRdbWriteStream
Arg3: <*details*> an Array of details on the Oracle error(s)

One or more Oracle errors occurred during the flush of the associated write stream. Note that because of buffering, the #nextPut: or #nextPutAll: operation that provided the object causing this error may have occurred some time earlier.

typeConversionError - Cannot convert GS Object for Oracle column, connection: <*conn*> stream: <*stream*> details: <*details*>
Arg1: a GsRdbConnection
Arg2: a GsRdbWriteStream
Arg3: an Array: details on the type conversion error:
1: The object containing data for this Oracle row
2: The selector applied to the object for this Oracle column
3. The Oracle column number being written
4: The object being written to this Oracle column

A GemStone object being converted for an Oracle column is incompatible with the Oracle column DataType.

internalError - GemConnect internal error, connection: <*conn*> stream: <*stream*> details: <*details*>
Arg1: <*conn*>a GsRdbConnection
Arg2: <*stream*> a GsRdbWriteStream
Arg3: <*details*> an Array: details on the Oracle error(s)

This is an unexpected logic error from within GemConnect that should not have occurred. Contact GemStone Technical Support with background information on this error plus the contents of the <details> array, especially the last entry containing the diagnostic string.

Changes to existing code

GsRdbReadStream new instance variable sql

The existing class `GsRdbReadStream` has a new instance variable "sql" which holds the SQL string being prepared / executed in the Oracle OCI. The accessor method #sql returns the string. You should never attempt to set this value (a private method #_sql: is used internally when appropriate).

GsOracleConnection new instance variable batchSize

The existing class `GsOracleConnection` has a new instance variable "batchSize" with the usual accessing / updating methods (batchSize / batchSize:). This allows you to specify how many rows are batched before performing an Oracle call using Oracle Array Processing. Once specified, this value of batchSize is used for all subsequently generated read- and write-streams. The default setting for this instance variable is 20.

Note that batch size is also used in openCursorOn: calls for SELECT statements. A single Oracle call is done retrieving a batch of rows for the SELECT, and then subsequent #next calls to the read stream will retrieve entries from the buffer. This batching of SELECT statements is not a new feature -- it existed in earlier versions of GemConnect but was hardwired to a batchSize of 20. You now have control over the batchSize.

In the following examples, "conn" refers to a connected instance of `GsOracleConnection`.

Example:

```
conn batchSize "Retrieves the current setting of batchSize"

conn batchSize: 20  "Set the batchSize to 20"
```

Note that using a batchSize greater than 1 will delay the return of any Oracle errors until the entire batch is processed, and that there may be multiple errors returned at the same time. This may have some impact on application design.

Changes to #openCursorOn:

In order to be able to "map" the new functionality over the existing design, the original #openCursorOn: method (and those that call it, such as #execute:) are now more sensitive to how they are called. In the original design, it was acceptable to use openCursorOn: or execute: on insert / delete / update calls, such as:

```
conn execute: 'delete from Table where key = 123'.
```

With the new design, this will raise an error. You should now use instead:

```
conn executeNoReturn: 'delete from Table where key = 123'.
```

or

```
conn executeReturnRowsAffected: 'delete from Table where key =
123'.
```

GsOracleConnection Class >> messages

The format of information on an Oracle error returned by this method has changed. This was required because of differences in the new Oracle OCI API, plus the need to return multiple Oracle errors in a single batch.

The new format is an array:

1 - Oracle Call Interface (OCI) return code (usually -1)

2 - Number of errors (usually 1, unless there are multiple errors on a writeStream flush)

3 - Oracle error code

4 - Oracle error message

5 - GemStone object (the object written to a writeStream that generated this error)

6..N - Repeat 3-5 for each additional error

last - Internal diagnostic message

The last entry is a string containing internal diagnostic information that will help GemStone Technical Support track down logic errors in the case of unexpected failures.

Modified Errors

A number of GemConnect errors have had some slight changes made to the arguments they return, or in the messages they display. These include:

noChangeNotification - object change notification is not supported on this version of GemStone.

This error first appeared in GemConnect 1.1.5, but has been modified slightly, since GemStone/S 64 Bit versions 2.0 and greater will now support object change notification (versions prior to 2.0 do not).

A number of GemConnect errors have a different “details” array that now returns the same information as the `GsOracleConnection Class >> messages` method documented above. These include:

oracleError - an unexpected error was encountered during Oracle processing, connection: <*conn*> stream: <*stream*> details: <*details*>

Arg1: <*conn*> a GsRdbConnection

Arg2: <*stream*> a GsRdbWriteStream

Arg3: <*details*> an Array of details on the Oracle error(s)

queryError - an error was encountered while performing a relational query, connection: <*conn*> query: <*stream*> details: <*details*>

Arg1: <*conn*> a GsRdbConnection
Arg2: <*stream*> a GsRdbWriteStream
Arg3: <*details*> an Array of details on the Oracle error(s)