



GQLx
Language Reference

Notice of Copyright

This document and its contents are copyright © Twyst Software Engineering cc 2009.

Notice of Confidentiality

This document and its contents are confidential and may not be disclosed without express permission from Twyst Software Engineering cc.

Document Information

<i>Document Number</i>	TWYST-GQLx-001	
<i>Document Description</i>	GQLx Language Reference	

Revision History

<i>Revision</i>	<i>Date</i>	<i>Author</i>	<i>Description</i>	<i>Approved</i>
Rev 0.00	6 March 2009	Tony Seebregts	Initial draft.	



TABLE OF CONTENTS

1. Introduction.....4

 1.1 Scope.....4

 1.2 Terms and Abbreviations.....4

2. Overview.....4

3. Getting started.....4

 3.1 Requirements.....4

 3.2 Installation.....4

 3.3 Javascript Console.....5

 3.4 Java GUI.....5

 3.5 Java Console.....6

 3.6 A Note on Spelling.....7

4. GQLx Language Reference.....8

 4.1 OPEN.....8

 4.2 CLOSE.....9

 4.3 SELECT.....9

 4.4 SELECT..JOIN.....10

 4.5 INSERT.....11

 4.6 INSERT..SELECT.....12

 4.7 UPDATE.....13

 4.8 DELETE.....14

 4.9 SET.....14

 4.10 SHOW.....16

5. Reserved Words.....17

6. Limitations.....17

 6.1 Duplicate column names.....17

 6.2 Cell Formula's.....18

 6.3 JOIN's.....18

7. Coming Soon.....18

GQLx

Language Reference

1. INTRODUCTION

GQLx is a SQL-like language for accessing and manipulating the contents of Google Documents spreadsheets. This document describes the language and the functionality supported by the available GQLx implementations:

- Java GUI application
- Java console application
- online Javascript console
- embedded GQLx engine

The above implementations mostly share the same functionality – the small differences are compromises required by the different environments and are documented where relevant.

1.1 Scope

1.2 Terms and Abbreviations

- SQL - Structured Query Language
- API - Application Program Interface
- JRE - Java Runtime Engine

2. OVERVIEW

Google provides two 'views' of their Google Documents spreadsheets:

- a web application for use within standard web browsers
- a GData programmatic API with accompanying libraries

and for most purposes this is more than sufficient.

However, occasionally it is necessary to do more complex operations on a spreadsheet - for example:

- combining values from different worksheets
- viewing only a subset of the data in a worksheet
- updating values only for certain rows
- removing only rows that meet a set of criteria
- inserting rows from another data source

all of which require either a set of complex manual operations and cut-and-pastes between different worksheets and applications - or else a fair bit of programming. It is these operations that GQLx is intended to simplify.

3. GETTING STARTED

3.1 Requirements

The Java GUI and console applications require an installed Java Runtime Engine (JRE 1.6.10 or later).

The JRE is available at no cost from the Sun Microsystems Java website:

<http://java.com/en/download>

3.2 Installation

Installing the Java GUI and console applications is as simple as downloading the relevant binary JAR files from the GQLx website and placing them in a directory of your choice.

The applications are shipped as standalone JAR's and does not require any other files or directories (this may change out of necessity in future releases).

Likewise, uninstalling either of the applications is merely a case of deleting the JAR file.

3.3 Javascript Console

The Javascript console is hosted on the GQLx website – it is accessible from the `CONSOLE` page on

`http://gqlx.twyst.co.za`

The console page requires some reasonably large scripts (including Google's `gdata.js`) and may take some time to load initially.

Logging In

On first use (or if you have previously logged out) you will have to authorise access to your spreadsheets by clicking on the `LOGIN` button. This will take you to a Google login page and on completion of sign in and authorisation, return you to the GQLx web console.

The whole authorisation process can occasionally be a bit fraught and sometimes it is necessary to sign out of any active GMail or Google Doc's sessions to get it to work correctly.

Having authorised access once you do not need to repeat the process until you either log out again or the authorisation expires (which can take up to a month).

Logging Out

The authorisation token (not your Google user ID and password) is stored in a local browser cookie – if you are concerned about privacy or unauthorised access to your spreadsheets in your absence then clicking on the `LOGOUT` button will clear the GQLx cookie and authorisation token.

Executing GQLx scripts

The Javascript console displays a 'script' area where you can either type (or paste scripts from another text editor).

Clicking the `EXECUTE` button will execute either the entire contents of the script area – or if an area is highlighted, then just the statements contained in the highlighted text.

Notes

1. In its current incarnation, the Javascript console is more a technology demonstration than a fully fledged web application.

3.4 Java GUI

The Java GUI is shipped as a standalone JAR file (`gqlx-gui.jar`) and can be downloaded from the GQLx website downloads section.

Installation

Installing the application is as simple as downloading the JAR file and moving it to a directory of your choice. It does not require additional files so it is quite safe to leave lying around to hand on your desktop.

Invoking the GUI application

The `gqlx-gui.jar` is a runnable JAR file – for a standard Java installation this means that simply double-clicking on the file will start the application up in normal mode. If:

- you have a non-standard JRE installation
- have modified your JAR file associations
- want to specify additional command line options

you will either need to create a desktop shortcut or batch file/shell script with the following command:

```
javaw -jar gqlx-gui.jar
```

in order to run the GUI.

In addition, the command line allows you to optionally specify:

- the default Google Docs user ID and password
`-uid <user ID> -pwd <password>`
- an initial script file
`-f <file>`
- to disable persisting the current script across sessions
`-clean`

The full command line specification is:

```
javaw -jar gqlx-gui.jar
      -uid <user ID>
      -pwd <password>
      -f <file>
      -clean
```

Google Doc's authorisation

Unlike the Javascript console, you do not need to login to your Google Doc's account – this is handled by the GQLx OPEN statement, e.g.:

```
OPEN UID='tom.brown' PWD='schooldays';
```

In addition, the GUI allows you to have multiple accounts open at any one time – although you will have to identify spreadsheets with the same name from different accounts using an alias to make it clear which one you have in mind, e.g.:

```
OPEN MySpreadsheet AS XYZ;
```

Executing GQLx scripts

The GUI is implemented as a standard SQL-like console – it displays a 'script' area where you can either type (or paste scripts from another text editor).

As usual, clicking the RUN button will execute either the entire contents of the script area – or if a set of statements is highlighted, then just the highlighted statements.

And the messages tab will display progress information while the results tab will render the returned from SELECT statements.

Notes

1. The GUI application does not store your Google Doc's user ID and password beyond the current session – but it does create a plaintext file (`.gqlx`) that stores the most recent GQLx script which may contain your user ID and password if it was used in an OPEN or SET DEFAULT LOGIN statement. Use the `-clean` command line option to disable this behaviour.

3.5 Java Console

The Java Console is shipped as a standalone JAR file (`gqlx-console.jar`) and can be downloaded from the GQLx website downloads section.

Installation

Installing the application is as simple as downloading the JAR file and moving it to a directory of your choice. It does not require or create additional files so it is quite safe to leave lying around to hand on your desktop.

Invoking the console application

The gqlx-console.jar is a runnable JAR file – for a standard Java installation this means that simply double-clicking on the file will start the application up in normal mode. If:

- you have a non-standard JRE installation
- have modified your JAR file associations
- want to specify additional command line options

you will either need to create a desktop shortcut or batch file/shell script with the following command:

```
javaw -jar gqlx-console.jar
```

in order to run the application.

In addition, you can optionally specify the default Google Docs user ID and password and also an initial script file to load on the command line:

```
java -jar gqlx-console.jar -uid <user ID> -pwd <password> -f <file>
```

Google Doc's authorisation

Like the GUI (and unlike the Javascript console), you do not need to login to your Google Doc's account – this is handled by the GQLx OPEN statement, e.g.:

```
OPEN UID='tom.brown' PWD='schooldays';
```

In addition, the application allows you to have multiple accounts open at any one time – although you will have to identify spreadsheets with the same name from different accounts using an alias to make it clear which one you have in mind, e.g.:

```
OPEN MySpreadsheet AS XYZ;
```

Executing GQLx scripts

The user interface is a command line – at the GQLx> prompt type (or copy and paste from a text editor) the script to run and press ENTER.

'Trace' messages will display the progress of the script execution (see SET TRACE to disable these) and the results of SELECT statements will be displayed inline.

The SET TRACE, SET LINE WIDTH and SET COLUMN WRAP commands allow you to modify how information is rendered.

Notes

1. The console application does not store your Google Doc's user ID and password beyond the current session.

3.6 A Note on Spelling

By default GQLx takes a 'relaxed' approach to the spelling of spreadsheet names, worksheet names and worksheet columns e.g. it will (most probably) recognise MySpredshet as the spreadsheet .

This may occasionally be inconvenient or even incorrect – use SET MATCH EXACT or SET MATCH NORMAL for stricter spelling checking.

4. GQLx LANGUAGE REFERENCE

Using GQLx should be reasonably familiar to anybody with experience of SQL and databases. The basic conventions to follow are:

- every statement must be terminated by a semi-colon.
- unquoted identifiers (i.e. spreadsheet names, worksheet names and column names) can only contains letter and numbers. Identifiers that contains other characters or spaces must be enclosed in single quotes e.g. 'My Spreadsheet'.
- all values are string values and must be enclosed in single quotes.

4.1 OPEN

Description

The OPEN statement retrieves the information about a spreadsheet and its contained worksheets and makes it available for use by the engine for SELECT, INSERT, UPDATE and DELETE operations.

Syntax

```
OPEN spreadsheet [UID=uid [PWD=pwd]] [AS alias];
```

Opens a 'private' spreadsheet, where:

- *spreadsheet* is the title of a spreadsheet document accessible to the default user ID or to the *uid* user ID.
- *uid* is an optional Google Doc's user ID. If not supplied the default user ID is used.
- *pwd* is the password associated with the Google Doc's user ID *uid*
- *alias* is an optional name by which the spreadsheet will be known. It is mostly useful when:
 - a spreadsheets from different user ID's have the same name
 - a spreadsheet needs to be either opened twice for a complicated JOIN
 - the spreadsheet name is very long.
 - opening a public spreadsheet

```
OPEN * [UID=<uid> [PWD=<pwd>]];
```

Opens all 'private' spreadsheets associated with a Google Doc's user ID:

- *uid* is an optional Google Doc's user ID. If not supplied the default user ID is used.
- *pwd* is the password associated with the Google Doc's user ID *uid*

The spreadsheets are opened under their document titles.

```
OPEN <url> [AS alias];
```

Opens a 'public' spreadsheet using the published URL created when sharing the document:

- *url* is the URL under which the spreadsheet is published.
- *alias* is an optional name by which the spreadsheet will be known. It is particularly useful for opening spreadsheets from different sources that have the same name.

Examples

```
OPEN ;
OPEN 'My Spreadsheet';
OPEN MySpreadsheet AS XYZ;
OPEN UID='tom.brown' PWD='schooldays';
OPEN http://spreadsheets.google.com/pub?key=pVUHqxow-OA_q2paiysqgQg;
```

See Also:

SHOW SPREADSHEETS
SET DEFAULT LOGIN

Notes

1. The online Javascript console only allows a single Google Doc's user ID to be in use at any one time.
2. 'Public' spreadsheets cannot be modified (i.e. the INSERT, UPDATE and DELETE statements will all report an error).

4.2 CLOSE

Description

The CLOSE statement removes a spreadsheet from the current session and makes it unavailable.

It is mostly useful when the same worksheet name occurs in two or more spreadsheets – resolving the ambiguity means the worksheet names have to be qualified by the containing spreadsheet name.

Syntax

```
CLOSE spreadsheet;
```

Closes an open spreadsheet, where:

- *spreadsheet* is the title or alias of an open spreadsheet document.

```
CLOSE *;
```

Closes all open spreadsheets.

Examples

```
CLOSE ;  
CLOSE 'My Spreadsheet';  
CLOSE SpreadsheetKnownAsFred;
```

See Also:

SHOW SPREADSHEETS
SHOW WORKSHEETS

Notes

4.3 SELECT

Description

The SELECT statement retrieves the contents of a worksheet.

Syntax

```
SELECT * FROM [spreadsheet.] worksheet [WHERE clause];
```

Retrieves all data from a worksheet that matches the optional WHERE clause.:

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.

- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

```
SELECT columns FROM [spreadsheet.]worksheet [WHERE clause];
```

Retrieves only the data in the listed columns from a worksheet:

- *columns* is a list of the columns in the worksheet that must be returned.
- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

Examples

```
SELECT * FROM WorksheetX;
SELECT * FROM .WorksheetX;
SELECT * FROM 'Worksheet X';
SELECT * FROM 'My Spreadsheet'. 'Worksheet X';
SELECT ColumnA, ColumnB, ColumnC FROM WorksheetX;
SELECT * FROM WorksheetX WHERE ColumnA='Yes';
SELECT * FROM WorksheetX WHERE ColumnA='Yes' AND ColumnB='Maybe';
SELECT * FROM WorksheetX
      WHERE ColumnA='Yes'
      AND (ColumnB='Maybe' OR ColumnC='No');
```

See Also:

```
SELECT.. JOIN
```

Notes

1. The current implementation of GQLx arbitrarily chooses the first column of the columns in the worksheet with the same name.
2. SELECT returns the values of cells with formula's – not the original formula.

4.4 SELECT..JOIN

Description

The SELECT.. JOIN statement retrieves the contents of a worksheet of several worksheets by matching rows on a JOIN condition.

Syntax

```
SELECT * FROM [spreadsheet.]worksheet
      [INNER] JOIN [spreadsheet.]worksheet
      ON [spreadsheet.]worksheet.column =
      [spreadsheet.]worksheet.column
      [WHERE clause];
```

Retrieves all data from the worksheets that matches the optional WHERE clause and where the rows in the worksheet match on the join condition:

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.

- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

```
SELECT * FROM [spreadsheet.]worksheet
        INNER JOIN [spreadsheet.]worksheet
                ON [spreadsheet.]worksheet.column =
                [spreadsheet.]worksheet.column
        [WHERE clause];
```

Retrieves only the data in the listed columns from the joined worksheets:

- *columns* is a list of the columns in the worksheet that must be returned.
- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

Examples

```
SELECT * FROM WorksheetX
        INNER JOIN WorksheetY
                ON WorksheetX.ColumnA = WorksheetY.ColumnA;

SELECT WorksheetX.ColumnA AS ColXA,
        WorksheetY.ColumnA AS ColYA
        FROM WorksheetX
                INNER JOIN WorksheetY
                        ON WorksheetX.ColumnA = WorksheetY.ColumnA;

SELECT * FROM WorksheetX
        INNER JOIN WorksheetY
                ON WorksheetX.ColumnA = WorksheetY.ColumnA
        WHERE WorksheetX.ColumnA = 'Maybe';
```

See Also:

```
SELECT .. JOIN
```

Notes

1. The current implementation of GQLx only supports INNER JOINS and the JOIN condition can only be an EQUALS operator.
2. SELECT .. JOIN returns the values of cells with formula's – not the original formula.

4.5 INSERT

Description

The INSERT statement appends a row to a worksheet.

Syntax

```
INSERT INTO [spreadsheet.]worksheet [(columns)] VALUES (values);
```

Appends a new row the worksheet that contains the values in the VALUES list.:

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *columns* is an optional list of column names that matches the values. If the column names are not supplied the values are assigned to columns from left to right in order.

- *values* is an list of string values to be inserted into the new row..

Examples

```
INSERT INTO WorksheetX VALUES ('Yes', 'No', 'Maybe');
INSERT INTO WorksheetX (ColumnA, ColumnB, ColumnX)
VALUES ('Yes', 'No', 'Maybe');
```

See Also:

```
INSERT .. SELECT
```

Notes

1. It is not currently possible to use INSERT to create a formula into the worksheet – it is intended to add this into a future release.
The 'workaround' for now is to use a data worksheet to store data values and a separate formula worksheets that references the data stored in the data worksheet.

4.6 INSERT..SELECT

Description

The INSERT .. SELECT statement appends the rows retrieved from one worksheet to another worksheet.

Syntax

```
INSERT INTO [spreadsheet.]worksheet
SELECT * FROM [spreadsheet.]worksheet
[WHERE (clause)];
```

Appends all the rows returned by the SELECT statement to the destination worksheet.

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *values* is an list of string values to be inserted into the new row, using column names in the source and destination worksheets to match columns.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

```
INSERT INTO [spreadsheet.]worksheet
SELECT columns FROM [spreadsheet.]worksheet
[WHERE (clause)];
```

Appends all the rows returned by the SELECT statement to the destination worksheet.

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *values* is an list of string values to be inserted into the new row, using column names in the source and destination worksheets to match columns.
- *columns* is an list of column names from the source worksheet. By using column aliases (see examples) values can be copied to columns of a different name.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

Examples

```

INSERT INTO WorksheetX
  SELECT * FROM WorksheetY;

INSERT INTO WorksheetX
  SELECT * FROM WorksheetY
    WHERE ColumnA = 'Yes';

INSERT INTO WorksheetX
  SELECT ColumnA,ColumnC
    FROM WorksheetY;

INSERT INTO WorksheetX
  SELECT ColumnA AS ColXA,
    ColumnC AS ColXC
    FROM WorksheetY;

```

See Also:

```

SELECT
INSERT

```

Notes

1. It is not currently possible to use INSERT . . SELECT to copy formula's between worksheets.

4.7 UPDATE*Description*

The UPDATE statement updates the contents of selected the rows in a worksheet.

Syntax

```

UPDATE [spreadsheet.]worksheet
  SET column=value[,column=value...]
  [WHERE (clause)];

```

Appends all the rows returned by the SELECT statement to the destination worksheet.

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *column* is the name of column in the worksheet
- *value* is the new string value of the column.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

Examples

```

UPDATE WorksheetX
  SET ColumnA='One',ColumnB='Two';

UPDATE WorksheetX
  SET 'Column A'='One',ColumnB='Two'
  WHERE ColumnC = 'Three';

UPDATE MySpreadsheet.WorksheetX
  SET ColumnA='One',ColumnB='Two';

UPDATE 'My Spreadsheet'. 'Worksheet X'
  SET ColumnA='One',ColumnB='Two';

```

See Also:

Notes

1. It is not currently possible to use UPDATE to create or modify formula's in a worksheet.
2. Rows are updated in 'blocks' and are thus not guaranteed to be updated in the same order in which they occur in the worksheet. If sequential row updates is required however, setting the initial block size to a size larger than the number of rows in the worksheet will ensure that rows are updated sequentially.

4.8 DELETE

Description

The DELETE statement removes selected the rows in a worksheet.

Syntax

```
DELETE FROM [spreadsheet.] worksheet;
```

Deletes all rows in the worksheet.

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.

```
DELETE FROM [spreadsheet.] worksheet  
WHERE (clause);
```

Deletes rows in the worksheet that match the WHERE clause:

- *spreadsheet* is the title or alias of an already open spreadsheet. It is only required if the *worksheet* name exists in another open spreadsheet.
- *worksheet* is the title of a worksheet in an already open spreadsheet.
- *clause* is a set of conditions that a row in the worksheet must meet for inclusion in the returned data

Examples

```
DELETE FROM WorksheetX;  
DELETE FROM .WorksheetX;  
DELETE FROM WorksheetX  
WHERE ColumnC = 'Three';  
DELETE FROM 'Worksheet X';  
DELETE FROM 'My Spreadsheet'.'Worksheet X';
```

See Also:

Notes

1. Rows are deleted in 'blocks' and are thus not guaranteed to be removed in the same order in which they occur in the worksheet. If sequential row deletes is required however, setting the initial block size to a size larger than the number of rows in the worksheet will ensure that rows are deleted sequentially.

4.9 SET

Description

The SET statement sets an operational parameter.

Syntax

```
SET DEFAULT LOGIN UID=uid PWD=pwd;
```

Sets the Google Doc's user ID and password to use if no user ID or password is specified in the OPEN statement.

```
SET MATCH EXACT;
```

Sets the spelling matching algorithm to EXACT i.e. case- and whitespace sensitive (with the exception of column names).

```
SET MATCH NORMAL;
```

Sets the spelling matching algorithm to NORMAL i.e. case- and whitespace insensitive but the spelling must otherwise match exactly.

```
SET MATCH SIMILAR;
```

Sets the spelling matching algorithm to SIMILAR i.e. case- and whitespace and spelling insensitive.

```
SET TRACE ON;
```

Displays the query progress messages – which can be reassuring if a query is taking a long time or for watching complex scripts execute.

```
SET TRACE OFF;
```

Disables the display of query progress messages.

```
SET LINE WIDTH width;
```

```
SET LINE WIDTH DEFAULT;
```

Sets the line width for the command line console and online Javascript console.

For the console the default value is 132 characters, but long records can easily exceed this length, leading to text with ellipsis (...).

For the online Javascript console, the default value is 800px.

```
SET LINE WRAP ON;
```

Enables the wrapping of lines that exceed the line width and is only applicable to the console application. The default is OFF.

```
SET LINE WRAP OFF;
```

Disables the wrapping of lines that exceed the line width and is only applicable to the console application.

```
SET COLUMN WRAP ON;
```

Enables the wrapping of column text that exceeds the width of a column. Only applicable to the console and GUI applications. The default is OFF.

```
SET COLUMN WRAP OFF;
```

Disables the wrapping of column text that exceeds the width of a column. Only applicable to the console and GUI applications.

```
SET BLOCK SIZE size;  
SET BLOCK SIZE initial,size;  
SET BLOCK SIZE DEFAULT;
```

Sets the size of the 'chunks' in which the GQLx engine retrieves data from a worksheet.

Retrieving data from a large worksheet can be very slow and splitting up the data into chunks makes the applications and the online console more useable and responsive (Firefox in particular seems to lock up completely when fetching rows from a large'ish worksheet).

The 'initial chunk' is the number of rows retrieved on the first 'fetch' and the block size is the number of rows retrieved on subsequent 'fetches'.

The first form sets both the initial and subsequent block sizes to the same number of rows. The second form allows the first 'fetch' to be a larger (or at least different) number to subsequent 'fetches', which can be useful when mostly retrieving data from small worksheets interspersed with the occasional large sheet.

The default value is 25 rows for both the initial and subsequent blocks.

Examples

```
SET BLOCK SIZE 50;  
SET BLOCK SIZE 50,25;  
SET BLOCK SIZE DEFAULT;
```

See Also:

SHOW

Notes

4.10 SHOW

Description

The SHOW statement displays the current value of an operational parameter.

Syntax

```
SHOW SPREADSHEETS;
```

Displays a list of the opened spreadsheets.

```
SHOW SPREADSHEETS FOR UID=uid PWD=pwd;
```

Displays a list of the spreadsheets (both open and not yet opened) associated with the user ID.

```
SHOW WORKSHEETS;
```

Displays a list of all the worksheets in the current open spreadsheets.

```
SHOW WORKSHEETS IN spreadsheet;
```

Displays a list of all the worksheets in the selected spreadsheet.

```
SHOW MATCH;
```

Displays the current spelling match algorithm (EXACT, NORMAL or SIMILAR).

```
SHOW TRACE;
```

Displays the current progress message 'trace' setting (ON or OFF).

`SHOW LINE WIDTH;`

Displays the current console line width (only applicable to the command line and online Javascript console applications).

`SHOW LINE WRAP;`

Displays the current console line wrap setting (ON or OFF) and is only applicable to the command line and online Javascript console applications.

`SHOW COLUMN WRAP;`

Displays the current column wrap setting (ON or OFF) and is only applicable to the console and GUI applications.

`SHOW BLOCK SIZE;`

Displays the number of rows fetched during the initial and subsequent 'fetch' operations when retrieving data from a worksheet.

Examples

```
SHOW SPREADSHEETS;  
SHOW SPREADSHEETS FOR UID='wild.thing' PWD='lionking';  
SHOW WORKSHEETS;  
SHOW WORKSHEETS IN ;  
SHOW WORKSHEETS IN 'My Spreadsheet';
```

See Also:

SET

Notes

5. RESERVED WORDS

The following key words are reserved – if you need to use them as a spreadsheet, worksheet, column name or value then enclose them in single quotes:

- OPEN
- CLOSE
- SELECT
- INSERT
- UPDATE
- DELETE
- SET
- SHOW

6. LIMITATIONS

The current definition of GLQx is a fairly minimal set of operations considered sufficient for most day-to-day use.

There are however a couple of glaring limitations:

- there is currently no way to discriminate between columns with the same name.
- there is no support for cell formula's

- only INNER JOIN's are supported
- JOIN's are memory-based
- worksheet information is not cached at present and retrieving a large worksheet is slow

6.1 Duplicate column names

Unlike a database, Google Doc's spreadsheets do not enforce unique column names so it is quite possible (and not unusual) for column names within a spreadsheet to be ambiguous.

Where duplicate column names exist in a worksheet, the current implementations GQLx arbitrarily use the first occurrence as the column of choice.

Future implementations may offer an array like subscript if this proves to be workable in practice – for the moment either ensure the column names are unique or resign yourself to manually updating the orphaned columns.

6.2 Cell Formula's

There is currently no support for retrieving, inserting or updating a cell formula – GQLx only works with cell values.

This is primarily a notation problem i.e. how to reference neighbouring cells in a formula when the row/column value of a value is not visible.

Suggestions are welcome – but for the moment 'best practice' is to use a 'calculation sheet' with the required calculations and formula's that references values in an underlying 'data sheet' that can be accessed using GQLx.

6.3 JOIN's

The JOIN support in GQLx is experimental and still very basic and not particularly fast:

- only INNER JOIN's are supported
- the only JOIN operator supported is "="

This is primarily because implementations of arbitrary JOIN's can become frighteningly complex and GQLx is still an infant and not ready for the experience of outright terror. Support for other operators and join types is planned.

In addition, the JOIN operations are implemented entirely in memory – partly because spreadsheets seldom get to the size that databases routinely grow too and partly for simplified implementation.

However, without care a JOIN could result in a dataset of several million rows, which will take a very long time to process and is almost bound to crash the engine.

In other words, "feel free to use JOIN's but be gentle with them" – at least until the implementation is more mature.

7. COMING SOON

The following additional functionality is planned for the next major release:

CREATE WORKSHEET

Creates a new worksheet in an open spreadsheet.

DROP WORKSHEET

Removes a worksheet from an open spreadsheet.

CREATE PROCEDURE

Creates a Javascript procedure for complex sequential processing of spreadsheet data.

[DROP PROCEDURE](#)

Removes an existing Javascript procedure.