GMA014G0

GridDB Advanced Edition
JDBC Driver Instructions

Introduction

This manual describes how to operate GridDB Advanced Edition JDBC driver.

Please read this manual prior to using the GridDB Advanced Edition / Vector Edition.

The functions described in this manual can be used exclusively by users with GridDB Advanced Edition / Vector Edition license.

Trademarks

- GridDB is a trademark of Toshiba Corporation.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates.
- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.
- Other product names are trademarks or registered trademarks of the respective owners.

Table of Contents

1. About the GridDB Advanced Edition

An interface that can access GridDB data using SQL is provided in the GridDB Advanced Edition.

This manual provides an overview and specifications of the Java API (JDBC) used to access a database supported by the GridDB Advanced Edition (hereinafter referred to as GridDB AE) and the GridDB Vector Edition (hereinafter referred to as GridDB VE).

In this document, when the edition type is not stated explicitly, the specification refers to common function of Advanced / Vector Edition.

2. Overview of GridDB AE JDBC driver

This chapter consists of a description of the specified format and data types that can be used in a program using JDBC parameters, and the points to note during use. Please refer to the JDBC program samples provided in section 4 while going through this section.

2.1 Connection method

2.1.1 Driver specification

Add the JDBC driver file gridstore-jdbc.jar to the class path. When added, the driver will be registered automatically. The JDBC driver file has been installed under /usr/share/java by default.

In addition, import the driver class as follows if necessary (Normally not required).

```
Class.forName("com.toshiba.mwcloud.gs.sql.Driver");
```

2.1.2 URL format when connected

URL format is as follows. If the multicast method is used to compose a cluster, normally it is connected using method (A). The load will be automatically distributed on the GridDB cluster side and the appropriate nodes will be connected. Connect using method (B) only if multicast communication with the GridDB cluster is not possible.

(A) If connecting automatically to a suitable node in a GridDB cluster using the multicast method

jdbc:gs:///(multicastAddress):(portNo)/(clusterName)/(databaseName)

multicastAddress: Multicast address used in connecting with a GridDB cluster.
Default is 239.0.0.1

portNo: Port number used in connecting with a GridDB cluster. Default is 41999

clusterName: Cluster name of GridDB cluster

databaseName: Database name. Connect to the default database if omitted.

*multicastAddress, portNo can be amended by editing the gs_cluster.json file.

(B) If connecting directly to a node in a GridDB cluster using the multicast method

jdbc:gs://(nodeAddress):(portNo)/(clusterName)/(databaseName)

nodeAddress: Address of node

portNo: Port number used in connecting with a node Default value is 20001

clusterName: Cluster name of GridDB cluster that node belongs to

databaseName: Database name. Connect to the default database if omitted.

*Default values of nodeAddress, portNo can be amended by editing the gs_node.json file.

If the fixed list method is used to compose a cluster, use method (C) to connect.

(C) If connecting to a GridDB cluster using the fixed list method

jdbc:gs:///(clusterName)/(databaseName)?notificationMember=(notificationMember)

clusterName: Cluster name of GridDB cluster

databaseName: Database name. Connect to the default database if omitted.

notificationMember: Address list of node (URL encoding required). Default port is 20001

Example: 192.168.0.10:20001,192.168.0.11:20001,192.168.0.12:20001

*notificationMember can be amended by editing the gs_cluster.json file.

Port used in the address list can be amended by editing the gs_node.json file.

If the provider method is used to compose a cluster, use method (D) to connect.

(D) If connecting to a GridDB cluster using the provider method

jdbc:gs:///(clusterName)/(databaseName)?notificationProvider=(notificationProvider)

clusterName: Cluster name of GridDB cluster

databaseName: Database name. Connect to the default database if omitted.

notificationProvider: URL of address provider (URL encoding required)

*notificationProvider can be amended by editing the gs_cluster.json file.

If the user name and password are going to be included in the URL in either one of the cases (A) to (D), add them at the end of the URL as shown below.

?user=(user name)&password=(password)

2.1.3 Connection timeout settings

The connection timeout can be set in either of the methods (1) and (2).

Setting (2) is prioritized if both (1) and (2) are set.

Default value of 300 seconds (5 minutes) is used if neither (1) or (2) has been set, or if there are no settings.

(1) Specify with the DriverManager#setLoginTimeout (int seconds)

The value in seconds is set as follows. After setting, connection timeout will be set in the connections to all the GridDB AE acquired by the DriverManager#getConnection or Driver#connect.

- If the value is 1 to Integer.MAX_VALUE

- Set by the specified number of seconds
- If the value is Integer.MIN_VALUE to 0
  - Not set

(2) Specify with DriverManager#getConnection(String url, Properties info) or Driver#connect(String url, Properties info)

Add a property to argument info in the key “loginTimeout”. If the value corresponding to the key ”loginTimeout” could be converted to a numerical value, the connection timeout will be set in the connection obtained as follows.

- If the converted value is 1 to Integer.MAX_VALUE
  - Set by the specified number of seconds
- If the converted value is 0 or less or larger than Integer.MAX_VALUE
  - Not set

2.2 Observations

- A container created by a GridDB SE (Standard Edition) client can be referenced as a table by the GridDB AE JDBC driver but it cannot be updated. Besides updating the rows, changes in the schema and index of a container are also included in an update. Conversely, a table created with the GridDB AE JDBC driver can neither be referenced nor updated from a GridDB SE client.
- If a time series container created by a GridDB client is searched with a SQL command from the GridDB AE JDBC driver, this is different from a search conducted with a TQL command from a GridDB SE client, and the results will not be in chronological order if no ORDER BY phrase is specified for the main key. Specify an ORDER BY against the main key if a chronological series of the SQL results is required.
- Regarding the consistency during a search, the way the results are viewed may differ between a search conducted from a GridDB SE client and a search conducted from a GridDB AE JDBC driver. As GridDB supports READ COMMITTED as an isolation level of the transaction, data committed at the point the search is started will be read in a search. If a search is conducted from a GridDB SE client, since each search request is read once in the process, only data committed at the point the search is started will be searched as per READ COMMITTED. On the other hand, if a search is conducted with a SQL command from a GridDB AE JDBC driver, the requested SQL may be read and processed several times. In this case, the respective readings will become READ COMMITTED. As a result, when a transaction of another client is committed in-between the readings, the update details are read by the next reading and the entire SQL may not become READ COMMITTED.
- When the number of SQL hits is large, the error “Memory limit exceeded” may occur. In this case, add the descriptions of parameter transaction:totalMemoryLimit and dataStore:resultSetMemoryLimit in the gs_node.json file as shown below, and expand the upper limit value of the memory used in the SQL process. However, the names of these parameters may be changed or deleted in future versions.

- Example: When specifying the upper limit value

```
"transaction":{
   "totalMemoryLimit":"2048MB"
}
"dataStore":{
   "resultSetMemoryLimit":"20480MB"
}
```

*Default value is 1024 MB and 10240 MB respectively if there is no description.

3. Specifications of GridDB AE JDBC driver

The specifications of the GridDB AE JDBC driver are shown in this chapter. The chapter explains mainly the support range of the driver as well as the differences with the JDBC standard. See the JDK API reference for the API specifications that conform to the JDBC standard unless otherwise stated.
Please note that the following could be revised in the future versions.

- Actions not conforming to the JDBC standard
- Support status of unsupported functions
- Error message

3.1 Common items

3.1.1 Supported JDBC version

The following functions corresponding to some of the functions of JDBC4.1 are not supported.

- Transaction control
- Stored procedure
- Batch instructions

3.1.2 Error processing

3.1.2.1 Use of unsupported functions

- Standard functions
  A SQLFeatureNotSupportedException occurs if a function that ought to be but is currently not supported by a driver conforming to the JDBC specifications is used. This action differs from the original SQLFeatureNotSupportedException specifications. Error name (to be described later) is JDBC_NOT_SUPPORTED.
- Optional functions
  If a function not supported by this driver that is positioned as an optional function in the JDBC specifications and for which a SQLFeatureNotSupportedException may occur is used, a SQLFeatureNotSupportedException will occur as per the JDBC specifications. Error name is JDBC_OPTIONAL_FEATURE_NOT_SUPPORTED.

3.1.2.2 Invoke a method against a closed object

As per the JDBC specifications, when a method other than isClosed() is invoked for an object that has a close() method, e.g. a connection object, etc., a SQLException will occur. Error name is JDBC_ALREADY_CLOSED.

3.1.2.3 Invalid null argument

If null is specified as the API method argument despite not being permitted, a SQLException due to a JDBC_EMPTY_PARAMETER error will occur.
Null is not permitted except for arguments which explicitly accepts null in the JDBC specifications or this guide.

3.1.2.4 If there are multiple error causes

If there are multiple error causes, control will be returned to the application at the point either one of the errors is detected.
In particular, if use of an unsupported function is attempted, it will be detected earlier than other errors. For example, if there is an attempt to create a stored procedure for a closed connection object, an error indicating that the operation is “not supported” instead of “closed” will be returned.

3.1.2.5 Description of exception

A check exception thrown from the driver is made up of a SQLException or a subclass instance of the SQLException.

Use the following method to get the exception details.

- getErrorCode()
  For errors detected by GridDB in either the server or client, an error number will be returned. See the list of error codes for the specific number and cause.
- getSQLState()
  Null is always returned.
- getMessage()
  Return an error message containing the error number and error description as a set. The format is as follows.

  [Code: (Error number)] (Error description)

  When the error number is not on the list, the following error message format will be used instead:
  .

  (Error Details)

3.1.2.6 Error list

The list of main errors detected inside the driver is as follows.

Table 1: GridDB AE JDBC driver error list

Error no.	Error code name	Error explanation format
(Appended)	JDBC_NOT_SUPPORTED	Currently not supported
(Appended)	JDBC_OPTIONAL_FEATURE_NOT_SUPPORTED	Optional feature not supported
(Appended)	JDBC_EMPTY_PARAMETER	The parameter (parameter name) must not be null
(Appended)	JDBC_ALREADY_CLOSED	Already closed
(Appended)	JDBC_COLUMN_INDEX_OUT_OF_RANGE	Column index out of range
(Appended)	JDBC_VALUE_TYPE_CONVERSION_FAILED	Failed to convert value type
(Appended)	JDBC_UNWRAPPING_NOT_SUPPORTED	Unwrapping interface not supported
(Appended)	JDBC_ILLEGAL_PARAMETER	Illegal value: (parameter name)
(Appended)	JDBC_UNSUPPORTED_PARAMETER_VALUE	Unsupported (parameter name)
(Appended)	JDBC_ILLEGAL_STATE	Protocol error occurred
(Appended)	JDBC_INVALID_CURSOR_POSITION	Invalid cursor position
(Appended)	JDBC_STATEMENT_CATEGORY_UNMATCHED	Writable query specified for read only request Read only query specified for writable request

When there is an error in the source generating the error and so on, additional details may be added to the end of the error description mentioned above. See GridDB error codes for the other error occurred.

3.2 API detailed specifications

3.2.1 Connection interface

Describes each method of the connection interface. Unless otherwise stated, only the description for the case when connection has not been closed is included.

3.2.1.1 Transaction control

Transaction control is not supported. However, errors that are caused by functional limits will not arise so that pseudo operations are carried out in an application using transactions as well.

- getAutoCommit()
  Always return “true”. Different from JDBC specifications.
- setAutoCommit(autoCommit)
  Ignore parameters. Different from JDBC specifications.
- commit()/rollback()
  Ignore request. Different from JDBC specifications.
- setTransactionIsolation(level)
  Behaves as if only TRANSACTION_READ_COMMITTED is supported. Compatible with DatabaseMetaData#supportsTransactionIsolationLevel (level).

3.2.1.2 Set or change various attributes

- isReadOnly()
  Always return “false”. Different from JDBC specifications.
- setReadOnly(readOnly)
  Ignore parameters. Different from JDBC specifications.
- setHoldability(holdability)
  Support only CLOSE_CURSORS_AT_COMMIT.
- createStatement(resultSetType, resultSetConcurrency)
  resultSetType supports only TYPE_FORWARD_ONLY, while resultSetConcurrency supports only CONCUR_READ_ONLY.
- createStatement(resultSetType, resultSetConcurrency, resultSetHoldability)
  resultSetType supports only TYPE_FORWARD_ONLY, resultSetConcurrency supports only CONCUR_READ_ONLY, and resultSetHoldability supports only CLOSE_CURSORS_AT_COMMIT.
- prepareStatement(sql, resultSetType, resultSetConcurrency)
  resultSetType supports only TYPE_FORWARD_ONLY, while resultSetConcurrency supports only CONCUR_READ_ONLY.
- prepareStatement(sql, resultSetType, resultSetConcurrency, resultSetHoldability)
  resultSetType supports only TYPE_FORWARD_ONLY, resultSetConcurrency supports only CONCUR_READ_ONLY, and resultSetHoldability supports only CLOSE_CURSORS_AT_COMMIT.

3.2.1.3 Unsupported function

- Standard functions
  - prepareCall(sql)
- Optional functions
  - abort(executor)
  - createArrayOf(typeName, elements)
  - createBlob()
  - createClob()
  - createNClob()
  - createSQLXML()
  - createStruct(typeName, attributes)
  - getNetworkTimeout()
  - getSchema()
  - getTypeMap()
  - prepareCall(sql, resultSetType, resultSetConcurrency)
  - prepareCall(sql, resultSetType, resultSetConcurrency, resultSetHoldability)
  - prepareStatement(sql, autoGeneratedKeys)
  - prepareStatement(sql, columnIndexes)
  - prepareStatement(sql, columnNames)

- releaseSavepoint(savepoint)
- rollback(savepoint)
- setNetworkTimeout(executor, milliseconds)
- setSavepoint()

3.2.2 DatabaseMetaData interface

- Attribute to return ResultSet
  - getColumns(catalog, schemaPattern, tableNamePattern, columnNamePattern)
    Return ResultSet corresponding to specified tableNamePattern. Wild cards cannot be specified in the tableNamePattern, and a blank result will be returned if there are no fully matching table. Other filter conditions specified are ignored.

Table 2: List of getColumns () result columns

Result column name	Value
TABLE_CAT	Null
TABLE_SCHEM	Null
TABLE_NAME	(Table name)
COLUMN_NAME	(Column name)
DATA_TYPE	(Refer to DatabaseMetaData#getTypeInfo())
TYPE_NAME	(Corresponds to the value that is in uppercase of the result of ResultSetMetaData#getColumnTypeName())
COLUMN_SIZE	2000000000
BUFFER_LENGTH	2000000000
DECIMAL_DIGITS	10
NUM_PREC_RADIX	10
NULLABLE	0 (DatabaseMetaData#columnNoNulls)
REMARKS	Null
COLUMN_DEF	Null
SQL_DATA_TYPE	0
SQL_DATETIME_SUB	0
CHAR_OCTET_LENGTH	2000000000
ORDINAL_POSITION	(Start from 1)
IS_NULLABLE	'NO'
SCOPE_CATLOG	Null
SCOPE_SCHEMA	Null
SCOPE_TABLE	Null
SOURCE_DATA_TYPE	Null

IS_AUTOINCREMENT	'NO'
IS_GENERATEDCOLUMN	'NO'

- getIndexInfo(catalog, schema, table, unique, approximate)
  A ResultSet corresponds to the specified table is returned. A table must match an existing table name. When unique is not false, an empty ResultSet is returned. Other specified conditions and parameters are ignored. Note that indices other than created by the CREATE INDEX statement are not included in the result.

  The value of TYPE column, tableIndexOther(3), is set for indices created with USING VNN specification in the Vector Edition.

Table 3 : List of getIndexInfo() result columns

Result column name	Value
TABLE_CAT	null
TABLE_SCHEM	null
TABLE_NAME	(Table name)
NON_UNIQUE	true
INDEX_QUALIFIER	null
INDEX_NAME	(Index name)
TYPE	tableIndexHashed(2) or tableIndexOther(3)
ORDINAL_POSITION	(Start from 1)
COLUMN_NAME	(Column name)
ASC_OR_DESC	null
CARDINALITY	0
PAGES	0
FILTER_CONDITION	null

- getPrimaryKeys(catalog, schema, table)
  ResultSet set by TABLE_NAME, COLUMN_NAME, KEY_SEQ is returned according to whether a main key is present or not. Null is set in all other columns of the ResultSet to be returned. Other filter conditions specified are ignored.
- getTables(catalog, schemaPattern, tableNamePattern, types)
  A ResultSet with only the TABLE_NAME and TABLE_TYPE set is returned. TABLE_TYPE is always set to 'TABLE'. If types is specified, an empty result will always be returned so long as no element matching the 'TABLE' exists in types. String elements inside types are not case sensitive. Filtering using tableNamePattern is valid for '%' only.
- getTableTypes()
  A ResultSet made up of a single row with 'TABLE' set as the TABLE_TYPE is returned.
- getTypeInfo()
  Behaves as per the JDBC specifications.
  The information by data type is as follows.

Table 4: getTypeInfo() result TYPE_NAME, DATA_TYPE column list

TYPE_NAME	DATA_TYPE
'BOOL'	-7 (Types#BIT)
'BYTE'	-6 (Types#TINYINT)
'SHORT'	5 (Types#SMALLINT)
'INTEGER'	4 (Types#INTEGER)
'LONG'	-5 (Types#BIGINT)
'FLOAT'	6 (Types#FLOAT)
'DOUBLE'	8 (Types#DOUBLE)
'TIMESTAMP'	93 (Types#TIMESTAMP)
'STRING'	12 (Types#VARCHAR)
'BLOB'	2004 (Types#BLOB)
'UNKNOWN'	1111 (Types#OTHER)

The information common to all data types is as follows.

Table 5: List of getTypeInfo() result column

Result column name	Value
PRECISION	0
LITERAL_PREFIX	null
LITERAL_SUFFIX	null
CREATE_PARAMS	null
NULLABLE	1 (DatabaseMetaData#typeNoNulls)
CASE_SENSITIVE	1
SEARCHABLE	3 (DatabaseMetaData#typeSearchable)
UNSIGNED_ATTRIBUTE	0
FIXED_PREC_SCALE	0
AUTO_INCREMENT	0
LOCAL_TYPE_NAME	null
MINIMUM_SCALE	0
MAXIMUM_SCALE	0
SQL_DATA_TYPE	0
SQL_DATETIME_SUB	0
NUM_PREC_RADIX	10

3.2.2.1 Attribute to return simple value

Table 6: Method to return a DatabaseMetaData simple value and result list (1)

Method	Result
allProceduresAreCallable()	FALSE
allTablesAreSelectable()	TRUE
autoCommitFailureClosesAllResultSets()	FALSE
dataDefinitionCausesTransactionCommit()	FALSE
dataDefinitionIgnoredInTransactions()	TRUE
deletesAreDetected(type)	FALSE
doesMaxRowSizeIncludeBlobs()	FALSE
getCatalogSeparator()	""
getCatalogTerm()	"catalog"
getDefaultTransactionIsolation()	TRANSACTION_READ_COMMITTED
getExtraNameCharacters()	""
getIdentifierQuoteString()	""
getMaxBinaryLiteralLength()	0
getMaxCatalogNameLength()	0
getMaxCharLiteralLength()	0
getMaxColumnNameLength()	0
getMaxColumnsInGroupBy()	0
getMaxColumnsInIndex()	0
getMaxColumnsInOrderBy()	0
getMaxColumnsInSelect()	0
getMaxColumnsInTable()	0
getMaxConnections()	0
getMaxCursorNameLength()	0
getMaxIndexLength()	0
getMaxSchemaNameLength()	0
getMaxProcedureNameLength()	0
getMaxRowSize()	0
getMaxStatementLength()	0
getMaxStatements()	0
getMaxTableNameLength()	0
getMaxTablesInSelect()	0
getMaxUserNameLength()	0
getProcedureTerm()	"procedure"
getResultSetHoldability()	CLOSE_CURSORS_AT_COMMIT
getRowIdLifetime()	TRUE
getSchemaTerm()	"schema"
getSearchStringEscape()	"%"
getSQLKeywords()	""
getSQLStateType()	sqlStateSQL99
getStringFunctions()	""
getSystemFunctions()	""
getURL()	null
getUserName()	(User name)
insertsAreDetected(type)	FALSE
isCatalogAtStart()	TRUE
isReadOnly()	FALSE
locatorsUpdateCopy()	FALSE
nullPlusNonNullIsNull()	TRUE
nullsAreSortedAtEnd()	FALSE
nullsAreSortedAtStart()	TRUE
nullsAreSortedHigh()	TRUE
nullsAreSortedLow()	FALSE
othersDeletesAreVisible(type)	FALSE
othersInsertsAreVisible(type)	FALSE
othersUpdatesAreVisible(type)	FALSE
ownDeletesAreVisible(type)	FALSE
ownInsertsAreVisible(type)	FALSE

Table 7: Method to return a DatabaseMetaData simple value and result list (2)

Method	Result
ownUpdatesAreVisible(type)	FALSE
storesLowerCaseIdentifiers()	FALSE
storesLowerCaseQuotedIdentifiers()	FALSE
storesMixedCaseIdentifiers()	TRUE
storesMixedCaseQuotedIdentifiers()	FALSE
storesUpperCaseIdentifiers()	FALSE
storesUpperCaseQuotedIdentifiers()	FALSE
supportsAlterTableWithAddColumn()	FALSE
supportsAlterTableWithDropColumn()	FALSE
supportsANSI92EntryLevelSQL()	FALSE
supportsANSI92FullSQL()	FALSE
supportsANSI92IntermediateSQL()	FALSE
supportsBatchUpdates()	FALSE
supportsCatalogsInDataManipulation()	FALSE
supportsCatalogsInIndexDefinitions()	FALSE
supportsCatalogsInPrivilegeDefinitions()	FALSE
supportsCatalogsInProcedureCalls()	FALSE
supportsCatalogsInTableDefinitions()	FALSE
supportsColumnAliasing()	TRUE
supportsConvert()	FALSE
supportsConvert(fromType, toType)	FALSE
supportsCoreSQLGrammar()	TRUE
supportsCorrelatedSubqueries()	FALSE
supportsDataDefinitionAndDataManipulationTransactions()	FALSE
supportsDataManipulationTransactionsOnly()	FALSE
supportsDifferentTableCorrelationNames()	FALSE
supportsExpressionsInOrderBy()	TRUE
supportsExtendedSQLGrammar()	FALSE
supportsFullOuterJoins()	FALSE
supportsGetGeneratedKeys()	FALSE
supportsGroupBy()	TRUE
supportsGroupByBeyondSelect()	FALSE
supportsGroupByUnrelated()	FALSE
supportsIntegrityEnhancementFacility()	FALSE
supportsLikeEscapeClause()	FALSE
supportsLimitedOuterJoins()	TRUE
supportsMinimumSQLGrammar()	TRUE
supportsMixedCaseIdentifiers()	TRUE
supportsMixedCaseQuotedIdentifiers()	FALSE
supportsMultipleOpenResults()	FALSE
supportsMultipleResultSets()	FALSE
supportsMultipleTransactions()	FALSE
supportsNamedParameters()	FALSE
supportsNonNullableColumns()	TRUE
supportsOpenCursorsAcrossCommit()	FALSE
supportsOpenCursorsAcrossRollback()	FALSE
supportsOpenStatementsAcrossCommit()	FALSE
supportsOpenStatementsAcrossRollback()	FALSE
supportsOrderByUnrelated()	FALSE
supportsOuterJoins()	TRUE
supportsPositionedDelete()	FALSE
supportsPositionedUpdate()	FALSE
supportsResultSetConcurrency(type, concurrency)	Only in case of the type is TYPE_FORWARD_ONLY or concurrency is
supportsResultSetHoldability(holdability)	Only in case of the
supportsResultSetType()	Only in case of the TYPE_FORWARD_ONLY
supportsSavepoints()	FALSE
supportsSchemasInDataManipulation()	FALSE

Table 8: Method to return a DatabaseMetaData simple value and result list (3)

Method	Result
supportsSchemasInIndexDefinitions()	FALSE
supportsSchemasInPrivilegeDefinitions()	FALSE
supportsSchemasInProcedureCalls()	FALSE
supportsSchemasInTableDefinitions()	FALSE
supportsSelectForUpdate()	FALSE
supportsStatementPooling()	FALSE
supportsStoredFunctionsUsingCallSyntax()	FALSE
supportsStoredProcedures()	FALSE
supportsSubqueriesInComparisons()	FALSE
supportsSubqueriesInExists()	TRUE
supportsSubqueriesInIns()	TRUE
supportsSubqueriesInQuantifieds()	FALSE
supportsTableCorrelationNames()	FALSE
supportsTransactionIsolationLevel(level)	Only in case of the TRANSACTION_READ_COMMITTED
supportsTransactions()	TRUE
supportsUnion()	TRUE
supportsUnionAll()	TRUE
updatesAreDetected(type)	FALSE
usesLocalFilePerTable()	FALSE
usesLocalFiles()	FALSE

3.2.2.2 Unsupported attribute

Table 9: List of DatabaseMetaData unsupported methods

Method name	Result
getAttributes	Empty ResultSet
getBestRowIdentifier	Empty ResultSet
getCatalogs	Empty ResultSet
getClientInfoProperties	Empty ResultSet
getColumnPrivileges	Empty ResultSet
getCrossReference	Empty ResultSet
getExportedKeys	Empty ResultSet
getFunctionColumns	Empty ResultSet
getFunctions	Empty ResultSet
getImportedKeys	Empty ResultSet
getProcedureColumns	Empty ResultSet
getProcedures	Empty ResultSet
getPseudoColumns	Empty ResultSet
getSchemas()	Empty ResultSet
getSchemas(catalog, schemaPattern)	Empty ResultSet
getSuperTables	Empty ResultSet
getSuperTypes	Empty ResultSet
getTablePrivileges	Empty ResultSet
getUDTs	Empty ResultSet
getVersionColumns	Empty ResultSet

3.2.3 Statement interface

3.2.3.1 Set/get fetch size

When checking this value, check that the number of rows obtained by getMaxRows() of the statement is not exceeded as well. Limits related to this value are stated only in the JDBC specifications from JDBC4.0 or earlier. However, unlike the previous JDBC specifications, this excludes the case in which the result of getMaxRows() has been set to the default value 0.

3.2.3.2 Set/get fetch direction

Only FETCH_FORWARD is supported for the fetch direction. A SQLException occurs if FETCH_FORWARD is not specified.

3.2.3.3 Unsupported function

- Batch instructions
  - Batch instructions are unsupported. When using the following functions, the same error occurs as when using unsupported standard functions.
    - addBatch(sql)
    - clearBatch()
    - executeBatch()
- Standard functions
  - The following methods are ignored when invoked. This behavior is different from the JDBC specifications.
    - setEscapeProcessing(enable)
- Optional functions
  - A SQLFeatureNotSupportedException occurs when the method below is invoked.
    - closeOnCompletion()
    - execute(sql, autoGeneratedKeys)
    - execute(sql, columnIndexes)
    - execute(sql, columnNames)
    - executeUpdate(sql, autoGeneratedKeys)
    - executeUpdate(sql, columnIndexes)
    - executeUpdate(sql, columnNames)
    - getGeneratedKeys()
    - getMoreResults(current)
    - isCloseOnCompletion()

3.2.4 PreparedStatement interface

3.2.4.1 Set/get parameter

The following methods are supported.

- clearParameters()
- getMetaDeta()
- getParameterMetaData()
- setBlob(int parameterIndex, Blob x)
- setBoolean(int parameterIndex, boolean x)
- setByte(int parameterIndex, byte x)
- setDate(int parameterIndex, Date x)
- setDouble(int parameterIndex, double x)
- setFloat(int parameterIndex, float x)
- setInt(int parameterIndex, int x)
- setLong(int parameterIndex, long x)
- setObject(int parameterIndex, Object x)

- The value set for TIMESTAMP accepts an object of a subclass of java.util.Date.
- setShort(int parameterIndex, short x)
- setString(int parameterIndex, String x)
- setTime(int parameterIndex, Time x)
- setTimestamp(int parameterIndex, Timestamp x)

3.2.4.2 SQL execution

The following methods are supported.

- execute()
- executeQuery()
- executeUpdate()

3.2.4.3 Unsupported function

- Standard functions
  - A SQLFeatureNotSupportedException occurs when the method below is invoked. This behavior is different from the JDBC specifications.
    - addBatch()
    - setBigDecimal(int parameterIndex, BigDecimal x)
    - setDate(int parameterIndex, Date x, Calendar cal)
    - setTime(int parameterIndex, Time x, Calendar cal)
    - setTimestamp(int parameterIndex, Timestamp x, Calendar cal)

- Optional functions
  - A SQLFeatureNotSupportedException occurs when the method below is invoked. All overloads for which the argument has been omitted are not supported.
    - setArray
    - setAsciiStream
    - setBinaryStream
    - setBlob(int parameterIndex, InputStream inputStream)
    - setBlob(int parameterIndex, InputStream inputStream, long length)
    - setBytes
    - setCharacterStream
    - setClob
    - setNCharacterStream
    - setNClob
    - setNString
    - setNull
    - setObject(int parameterIndex, Object x, int targetSqlType)
    - setObject(int parameterIndex, Object x, int targetSqlType, int scaleOrLength)
    - setRef
    - setRowId
    - setSQLXML
    - setUnicodeStream
    - setURL

3.2.5 ParameterMetaData interface

All methods are supported but the methods below will always return a fixed value regardless of the value of argument param.

Table 10: List of methods that return a fixed value for ParameterMetaData

Method	Result
getParameterType	Types.OTHE
getParamterTypeName	"UNKNOWN
getPrecision	0
getScale	0
isSigned	FALSE

3.2.6 ResultSet interface

3.2.6.1 Set/get fetch size

Only the specified value is checked and configuration changes will not affect the actual fetch process.
When checking the value, check that the number of rows obtained by getMaxRows() of the statement in the source generating the target ResultSet is not exceeded as well. This limit is stated only in the JDBC specifications from JDBC4.0 or earlier. However, unlike the previous JDBC specifications, this excludes the case in which the result of getMaxRows() has been set to the default value 0.
Actual fetch process is not affected but the revised setting can be acquired.

3.2.6.2 Set/get fetch direction

Only FETCH_FORWARD is supported for the fetch direction. A SQLException occurs if FETCH_FORWARD is not specified. This behavior is different from the JDBC specifications.

3.2.6.3 Get cursor data

The following cursor-related methods are supported.

- isAfterLast()
- isBeforeFirst()
- isFirst()
- isLast()
- next()

Since the only fetch direction supported is FETCH_FORWARD, when the following method is invoked, a SQLException caused by a command being invoked against a FETCH_FORWARD type ResultSet will occur.

- absolute(row)
- afterLast()
- beforeFirst()
- first()
- last()
- previous()
- relative(rows)

3.2.6.4 Management of warnings

As warnings will not be recorded, the actions to manage warnings are therefore as follows.

Table 11: Actions of the ResultSet warning-related method

Method	Behavior
getWarnings()	Return null
clearWarnings()	Warning is not clear

3.2.6.5 Attribute to return fixed value

The support status of a method to return a fixed value all the time while the ResultSet remains open is as follows.

Table 12: List of methods to return a fixed value for the ResultSet

Method	Result
getCursorName()	Return null
getType()	Return TYPE_FORWARD_ONLY
getConcurrency()	Return CONCUR_READ_ONLY
getMetaData()	JDBC-compliant
getStatement()	JDBC-compliant

3.2.6.6 Data type conversion

When getting the value of a specified column, if the data type maintained by the ResultSet differs from the requested data type, data type conversion will be attempted for the following combinations only.

Table 13: List of combinations to perform data type conversion

The Java type of the destination	BOOL	INTEGRAL ※1	FLOATING ※2	TIMESTAMP	STRING	BLOB
boolean	○	○ ※4			○ ※3	
byte	○	○	○		○	
short	○	○	○		○	
int	○	○	○		○	
long	○	○	○		○	
float		○	○		○	
double		○	○		○	
byte[]	○	○	○		○	
java.sql.Date				○ ※5	○	
Time				○ ※5	○	
Timestamp				○ ※5	○	
String	○	○	○	○	○	○ ※6
Blob					○ ※6	○
Object	○	○	○	○	○	○

(*1). INTEGRAL: Any one of BYTE/SHORT/INTEGER/LONG
(*2). FLOATING: Any one of FLOAT/DOUBLE
(*3). Convert to false if “false” and to true if “true". ASCII letters are case insensitive. Otherwise it can not be converted and an error occurs.
(*4). Convert to false if 0 and to true if not 0.
(*5). Treat as time passed in milliseconds starting from the epoch time (UTC time starting from January 1, 1970) and convert. java.util.Calendar is ignored if specified. Different from JDBC specifications.
(*6). Convert the time of a string expression according to the following rules. Acceptable expressions expressed in a pattern string like java.command.SimpleDateFormat are as follows (timezone is excluded though).
- yyyy-MM-dd'T'HH:mm:ss.SSS
- yyyy-MM-dd'T'HH:mm:ss

- yyyy-MM-dd HH:mm:ss.SSS
- yyyy-MM-dd HH:mm:ss
- yyyy-MM-dd
- HH:mm:ss.SSS
- HH:mm:ss

Regarding the timezone, prioritize those contained in a string, and see the descriptions when the specified java.util.Calendar is specified. Besides expressions that can be interpreted by a java.command.SimpleDateFormat “z” or “Z” pattern, “Z” expressions indicating that the time is UTC will be accepted as a timezone string. If timezone data does not exist, UTC time will be assumed regardless of the system setting.

3.2.6.7 Get column value

The column value can be acquired using a method that corresponds to the data type of the supported data type conversion address. Both column label and column index are supported as methods to specify a column.

Besides this, the following functions can be used.

- getBinaryStream
  Equivalent to the data type conversion result in byte []
- wasNull
  JDBC-compliant

3.2.6.8 Error processing

- Invalid column index
  If an invalid column index is specified in an attempt to get the value, a SQLException due to a JDBC_COLUMN_INDEX_OUT_OF_RANGE error will occur.
- Data type conversion error
  If data type conversion failed, a SQLException due to a JDBC_VALUE_TYPE_CONVERSION_FAILED error will occur.

3.2.6.9 Unsupported function

The following optional functions are not supported. All overloads for which the argument has been omitted are not supported.

- cancelRowUpdates()
- getArray
- getAsciiStream
- getBigDecimal
- getClob
- getNClob
- getNCharacterStream
- getNString
- getObject(columnIndex, map)
- getObject(columnLabel, map)
- getObject(columnIndex, type)
- getObject(columnLabel, type)
- getRef
- getRow()
- getRowId
- getSQLXML
- getUnicodeStream
- getURL

- moveToInsertRow()
- moveToCurrentRow()
- refreshRow()
- rowInserted()
- rowDeleted()
- rowUpdated()
- All methods starting with insert
- All methods starting with update
- All methods starting with delete

3.2.7 ResultSetMetaData interface

3.2.7.1 Data type of column

Describes the data type of the column returned by ResultSetMetaData. The data type name returned by DatabaseMetaData#getTypeInfo() is used as a reference here.

The specifications of the various methods to get the data type of a column are as follows.

Table 14: List of return values of the method to get data type of column

Type name	getColumnClassName	getColumnType	getColumnTypeName
BOOL	"java.lang.Boolean"	Types#BIT	"BOOL"
BYTE	"java.lang.Byte"	Types#TINYINT	"BYTE"
SHORT	"java.lang.Short"	Types#SMALLINT	"SHORT"
INTEGER	"java.lang.Integer"	Types#INTEGER	"INTEGER"
LONG	"java.lang.Long"	Types#BIGINT	"LONG"
FLOAT	"java.lang.Float"	Types#FLOAT	"FLOAT"
DOUBLE	"java.lang.Double"	Types#DOUBLE	"DOUBLE"
TIMESTAMP	"java.util.Date"	Types#TIMESTAMP	"TIMESTAMP"
STRING	"java.lang.String"	Types#VARCHAR	"STRING"
BLOB	"java.sql.Blob"	Types#BLOB	"BLOB"
UNKNOWN ※1	"java.lang.Object"	Types#OTHER	"UNKNOWN"

(※1). UNKNOWN: It is used when the column type cannot be specified like a ResultSet obtained by executing "SELECT NULL"

3.2.7.2 Attribute to return simple value

Table 15: List of return values and methods to return ResultSetMetaData simple value

Method name	Result
getCatalogName	""
getColumnDisplaySize	Integer#MAX_VALUE
getPrecision	0
getScale	0
getSchemaName	""
getTableName	""
isAutoIncrement	FALSE
isCaseSensitive	TRUE
isCurrency	FALSE
isDefinitelyWritable	TRUE
isNullable	※1
isReadOnly	FALSE
isSearchable	TRUE
isSigned	FALSE
isWritable	TRUE

(※1). Depending on the operation result, it takes either columnNullable (= 1) or columnNoNulls (= 0)

4. Sample

A JDBC sample programs is given below.

```
import java.sql.*;

public class SampleJDBC {
   public static void main(String[] args) throws SQLException {
      String url = null, user = "", password = "";

      if (args.length != 3) {
         System.err.println(
               "usage: java SampleJDBC (multicastAddress) (port) (clusterName)");
         System.exit(1);
      }

      // url is "jdbc:gs:// (multicastAddress): (portNo)/(clusterName)" format
      url = "jdbc:gs://" + args[0] + ":" + args[1] + "/" + args[2];
      user = "system";
      password = "manager";

      System.out.println("DB Connection Start");

      // Connection with GridDB cluster
      Connection con = DriverManager.getConnection(url, user, password);
      try {
         System.out.println("Start");
         Statement st = con.createStatement();
         ResultSet rs = st.executeQuery("SELECT * FROM table01");
         ResultSetMetaData md = rs.getMetaData();
         while (rs.next()) {
            for (int i = 0; i < md.getColumnCount(); i++) {
               System.out.print(rs.getString(i + 1) + "|");
            }
            System.out.println("");
         }
         rs.close();
         System.out.println("End");
         st.close();
      }
      finally {
         System.out.println("DB Connection Close");
         con.close();
      }
   }
}
```