

GridDB® Advanced Edition
JDBC ドライバ説明書

東芝ソリューション株式会社

© Toshiba Solutions Corporation 2016 All Rights Reserved.

はじめに

本書では GridDB Advanced Edition JDBC ドライバの取り扱い方法および、注意事項について記載しています。
GridDB Advanced Edition をご使用になる前に、必ずお読みください。
なお、本書で説明する機能は、GridDB Advanced Edition ライセンスを保有するユーザのみがご利用いただけます。

商標

- GridDB は、東芝ソリューション株式会社の登録商標です。
- Oracle と Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標または商標です。
- Red Hat は米国およびその他の国における Red Hat, Inc.の登録商標もしくは商標です。
- その他製品名は、それぞれの所有者の商標または登録商標です。

目次

1.	GridDB Advanced Edition とは.....	1
2.	GridDB AE JDBC ドライバの概要.....	1
2.1	接続方法.....	1
2.1.1	ドライバの指定.....	1
2.1.2	接続時の URL 形式.....	1
2.1.3	接続タイムアウトの設定.....	2
2.2	注意点.....	3
3.	GridDB AE JDBC ドライバの仕様.....	5
3.1	共通事項.....	5
3.1.1	サポートされる JDBC バージョン.....	5
3.1.2	エラー処理.....	5
3.2	API 仕様詳細.....	6
3.2.1	Connection インターフェース.....	6
3.2.2	DatabaseMetaData インターフェース.....	8
3.2.3	Statement インターフェース.....	12
3.2.4	PreparedStatement インターフェース.....	13
3.2.5	ParameterMetaData インターフェース.....	14
3.2.6	ResultSet インターフェース.....	15
3.2.7	ResultSetMetaData インターフェース.....	18
4.	サンプル.....	20

1. GridDB Advanced Edition とは

GridDB Advanced Edition では、GridDB のデータに SQL でアクセスできるインターフェースを提供します。本書では、GridDB Advanced Edition (以降 GridDB AE と記載します) のサポートするデータベースにアクセスする Java の API (JDBC) の概要および仕様を説明します。

2. GridDB AE JDBC ドライバの概要

JDBC パラメータのプログラムでの指定形式や使用できるデータ型、使用上の注意点を説明します。なお、4. サンプル に JDBC プログラムのサンプルがありますので、同時に参照してください。

2.1 接続方法

2.1.1 ドライバの指定

JDBC ドライバファイル `gridstore-jdbc.jar` をクラスパスに追加します。これによりドライバが自動的に登録されます。JDBC ドライバファイルは、デフォルトでは `/usr/share/java` 以下にインストールされています。

さらに必要に応じて、以下のようにしてドライバクラスを読み込みます。通常は不要です。

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

2.1.2 接続時の URL 形式

URL は以下の形式となります。クラスタ構成方式がマルチキャスト方式の場合、通常は (A) の形式で接続してください。GridDB クラスタ側で自動的に負荷分散が行われ適切なノードに接続されます。GridDB クラスタとの間でマルチキャストでの通信ができない場合のみ、(B) の形式で接続してください。

(A) マルチキャスト方式の GridDB クラスタの適切なノードへ自動的に接続する場合

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

`multicastAddress` : GridDB クラスタとの接続に使うマルチキャストアドレス。

デフォルトは 239.0.0.1

`portNo` : GridDB クラスタとの接続に使うポート No。デフォルトは 41999

`clusterName` : GridDB クラスタのクラスタ名

`databaseName` : データベース名。省略した場合はデフォルトデータベースに接続します

※`multicastAddress`、`portNo` は `gs_cluster.json` ファイルを編集することで変更可能です。

(B) マルチキャスト方式の GridDB クラスタ内のノードに直接接続する場合

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

`nodeAddress` : ノードのアドレス

`portNo` : ノードとの接続に使うポート No。デフォルト値は 20001

`clusterName` : ノードが属する GridDB クラスタのクラスタ名

`databaseName` : データベース名。省略した場合はデフォルトデータベースに接続します

※nodeAddress、portNo のデフォルト値は gs_node.json ファイルを編集することで変更可能です。

クラスタ構成方式が固定リスト方式の場合、(C)の形式で接続してください。

(C) 固定リスト方式の GridDB クラスタに接続する場合

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

clusterName : GridDB クラスタのクラスタ名

databaseName : データベース名。省略した場合はデフォルトデータベースに接続します

notificationMember : ノードのアドレスリスト(要 URL エンコード)。デフォルトポートは 20001

例 : 192.168.0.10:20001, 192.168.0.11:20001, 192.168.0.12:20001

※notificationMember は gs_cluster.json ファイルを編集することで変更可能です。

アドレスリストで使うポートは、gs_node.json ファイルを編集することで変更可能です。

クラスタ構成方式がプロバイダ方式の場合、(D)の形式で接続してください。

(D) プロバイダ方式の GridDB クラスタに接続する場合

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

clusterName : GridDB クラスタのクラスタ名

databaseName : データベース名。省略した場合はデフォルトデータベースに接続します

notificationProvider : アドレスプロバイダの URL(要 URL エンコード)

※notificationProvider は gs_cluster.json ファイルを編集することで変更可能です。

なお、(A)～(D)いずれの場合でも、ユーザ名・パスワードを URL に含める場合は、URL の末尾に次のように追加してください。

?user=(ユーザ名)&password=(パスワード)

2.1.3 接続タイムアウトの設定

(1)、(2) どちらかの方法で接続タイムアウトを設定できます。

(1)と(2)の両方が設定された場合、(2)の設定が優先されます。

(1)と(2)のどちらとも設定されない場合、設定がない場合のデフォルト値 300 秒(5分)が使用されます。

(1) DriverManager#setLoginTimeout(int seconds)で指定する

seconds の値によって、以下のように設定されます。設定後、DriverManager#getConnection または Driver#connect で取得する全ての GridDB AE への Connection に接続タイムアウトが設定されません。

- 値が 1～Integer.MAX_VALUE の場合
 - 指定した秒数で設定されます
- 値が Integer.MIN_VALUE～0 の場合
 - 設定されません

(2) DriverManager#getConnection(String url, Properties info) または Driver#connect(String url, Properties info) で指定する

引数 info にキー”loginTimeout”でプロパティを追加してください。キー”loginTimeout”に対応する値が数値に変換できた場合、取得した Connection にのみ以下のように接続タイムアウトが設定されます。

- 変換した値が 1~Integer.MAX_VALUE の場合
 - 指定した秒数で設定されます
- 変換した値が 0 以下 または Integer.MAX_VALUE より大きい場合
 - 設定されません

2.2 注意点

- GridDB SE (Standard Edition) クライアントで作成したコンテナは、GridDB AE JDBC ドライバでテーブルとして参照できますが、更新はできません。更新には行の更新だけでなくコンテナのスキーマや索引の変更を含みます。逆に、GridDB AE JDBC ドライバで作成したテーブルは、GridDB SE クライアントからは参照も更新もできません。
- GridDB SE クライアントで作成した時系列コンテナを GridDB AE JDBC ドライバから SQL で検索した場合、GridDB SE クライアントから TQL で検索した場合と異なり、主キーに対する ORDER BY 句がなければ結果は時刻順とはなりません。SQL 結果の時刻順整列が必要な場合には主キーに対する ORDER BY を指定してください。
- 検索時の一貫性について、GridDB SE クライアントから検索した場合 GridDB AE JDBC ドライバから SQL で検索した場合と見え方が異なることがあります。GridDB ではトランザクションの隔離レベルとして READ COMMITTED をサポートしているため、検索では検索開始時点でコミット済みのデータを読み取ります。GridDB SE クライアントから検索した場合は、1 検索要求を 1 回の読み取りで処理するため READ COMMITTED の通り検索開始時点でコミット済みデータのみを検索します。一方、GridDB AE JDBC ドライバから SQL で検索した場合、要求された SQL を複数回の読み取りで処理することがあり、このときそれぞれの読み取りが READ COMMITTED となります。そのため、読み取りと読み取りの間で他クライアントのトランザクションがコミットされると、次の読み取りでその更新内容を読み取ってしまい、SQL 全体としては READ COMMITTED とならないことがあります。
- SQL のヒット件数が多いと、「Memory limit exceeded」というエラーが発生することがあります。その場合は、以下のように gs_node.json ファイルにパラメータ transaction:totalMemoryLimit および dataStore:resultSetMemoryLimit の記述を追加し、SQL 処理で使用するメモリの上限値を拡張してください。ただし、これらのパラメータは将来のバージョンで名前の変更や削除が行われる可能性があります。
 - 例) 上限値を指定する場合

```
"transaction": {  
  "totalMemoryLimit": "2048MB"  
}
```

```
"dataStore": {  
  "resultSetMemoryLimit": "20480MB"  
}
```

※記述がない場合のデフォルト値はそれぞれ 1024MB、10240MB です。

3. GridDB AE JDBC ドライバの仕様

本章では、GridDB AE JDBC ドライバの仕様について示します。主に、ドライバのサポート範囲ならびに JDBC 標準との相違点について説明します。特記事項がなく JDBC 標準に準拠している API の仕様については、JDK の API リファレンスを参照してください。

将来のバージョンでは、特に次の点に変更される可能性があります。

- JDBC 標準に準拠していない挙動
- 未サポートの機能のサポート状況
- エラーメッセージ

3.1 共通事項

3.1.1 サポートされる JDBC バージョン

JDBC4.1 の一部機能に対応し、次の機能はサポートされません。

- トランザクション制御
- ストアドプロシージャ

3.1.2 エラー処理

3.1.2.1 未サポート機能の使用

- 標準機能
JDBC 仕様に準拠したドライバがサポートすべき標準機能のうち、本ドライバにおいて現状サポートされていない機能を使用した場合、`SQLFeatureNotSupportedException` が発生します。この挙動は、本来の `SQLFeatureNotSupportedException` の仕様とは異なります。エラー名（後述）は `JDBC_NOT_SUPPORTED` となります。
- オプション機能
JDBC 仕様においてオプション機能に位置付けられており、`SQLFeatureNotSupportedException` が発生する可能性のある機能のうち、本ドライバにおいてサポートされていない機能を使用した場合、JDBC 仕様通り `SQLFeatureNotSupportedException` が発生します。エラー名は `JDBC_OPTIONAL_FEATURE_NOT_SUPPORTED` となります。

3.1.2.2 クローズ済みのオブジェクトに対するメソッド呼び出し

JDBC 仕様の通り、`Connection` オブジェクトなど `close()` メソッドを持つオブジェクトに対し、`isClosed()` 以外のメソッドを呼び出すと、`SQLException` が発生します。エラー名は `JDBC_ALREADY_CLOSED` となります。

3.1.2.3 不正な null 引数

API のメソッド引数として、`null` が許容されないにも関わらず指定された場合、`JDBC_EMPTY_PARAMETER` エラーからなる `SQLException` が発生します。

JDBC 仕様で明示的に `null` の受け入れを明記している引数以外は、`null` を許容しません。

3.1.2.4 複数のエラー原因がある場合

複数のエラー原因がある場合は、いずれかのエラーを検知した時点でアプリケーションに制御が戻ります。特に、未サポート機能を使用しようとした場合のエラーは、他のエラーよりも先に検知します。たとえば、クローズ済みの `Connection` オブジェクトに対してストアドプロシージャを作成しようとした場合は、クローズされていることではなく、未サポートであることを示すエラーが返ります。

3.1.2.5 例外の内容

ドライバよりスローされるチェック例外は、SQLException もしくは SQLException のサブクラスのインスタンスからなります。

例外の詳細を取得するには、次のメソッドを使用します。

- `getErrorCode()`
サーバ・クライアントのいずれかで GridDB が検知したエラーについて、エラー番号を返却します。具体的な番号やその原因については、エラーコード一覧を参照してください。
- `getSQLState()`
SQL 処理や JDBC ドライバ内で検出したエラーの場合は対応する SQLSTATE を返却します。それ以外の場合は null を返却します。
- `getMessage()`
エラー番号とエラーの説明を組にした、エラーメッセージを返却します。書式は次のようになります。
[(エラー番号):(エラーコード名)] (補足メッセージ)

3.1.2.6 エラー一覧

ドライバ内部で検出される主なエラーの一覧は次の通りです。

表 1: GridDB AE JDBC ドライバ エラー一覧

エラー番号	エラーコード名	エラー説明の書式
(別記)	JDBC_NOT_SUPPORTED	Currently not supported
(別記)	JDBC_OPTIONAL_FEATURE_NOT_SUPPORTED	Optional feature not supported
(別記)	JDBC_EMPTY_PARAMETER	The parameter (引数名) must not be null
(別記)	JDBC_ALREADY_CLOSED	Already closed
(別記)	JDBC_COLUMN_INDEX_OUT_OF_RANGE	Column index out of range
(別記)	JDBC_VALUE_TYPE_CONVERSION_FAILED	Failed to convert value type
(別記)	JDBC_UNWRAPPING_NOT_SUPPORTED	Unwrapping interface not supported
(別記)	JDBC_ILLEGAL_PARAMETER	Illegal value: (引数名)
(別記)	JDBC_UNSUPPORTED_PARAMETER_VALUE	Unsupported (パラメータ名)
(別記)	JDBC_ILLEGAL_STATE	Protocol error occurred
(別記)	JDBC_INVALID_CURSOR_POSITION	Invalid cursor position
(別記)	JDBC_STATEMENT_CATEGORY_UNMATCHED	Writable query specified for read only request Read only query specified for writable request
(別記)	JDBC_MESSAGE_CORRUPTED	Protocol error

エラーの発生源となる元のエラーがある場合などは、上記のエラー説明の末尾に追加の詳細メッセージが追加されることがあります。

3.2 API 仕様詳細

3.2.1 Connection インターフェース

Connection インターフェースの各メソッドについて説明します。特に説明のない限り、Connection がクローズされていない場合の説明のみを記載します。

3.2.1.1 トランザクション制御

トランザクション制御はサポートしません。ただし、トランザクションを使用するアプリケーションにおいても疑似的に動作するよう、機能制限を原因とするエラーを引き起こしません。

- `getAutoCommit()`

常に true を返します。JDBC 仕様とは異なります。

- `setAutoCommit(autoCommit)`
パラメータを無視します。JDBC 仕様とは異なります。
- `commit()/rollback()`
要求を無視します。JDBC 仕様とは異なります。
- `setTransactionIsolation(level)`
`TRANSACTION_READ_COMMITTED` のみサポートしているかのように振る舞います。
`DatabaseMetaData#supportsTransactionIsolationLevel(level)` と対応します。

3.2.1.2 各種属性の設定・変更

- `isReadOnly()`
常に false を返します。JDBC 仕様とは異なります。
- `setReadOnly(readOnly)`
パラメータを無視します。JDBC 仕様とは異なります。
- `setHoldability(holdability)`
`CLOSE_CURSORS_AT_COMMIT` のみサポートします。
- `createStatement(resultSetType, resultSetConcurrency)`
`resultSetType` は `TYPE_FORWARD_ONLY` のみ、`resultSetConcurrency` は `CONCUR_READ_ONLY` のみサポートします。
- `createStatement(resultSetType, resultSetConcurrency, resultSetHoldability)`
`resultSetType` は `TYPE_FORWARD_ONLY` のみ、`resultSetConcurrency` は `CONCUR_READ_ONLY` のみ、`resultSetHoldability` は `CLOSE_CURSORS_AT_COMMIT` のみサポートします。
- `prepareStatement(sql, resultSetType, resultSetConcurrency)`
`resultSetType` は `TYPE_FORWARD_ONLY` のみ、`resultSetConcurrency` は `CONCUR_READ_ONLY` のみサポートします。
- `prepareStatement(sql, resultSetType, resultSetConcurrency, resultSetHoldability)`
`resultSetType` は `TYPE_FORWARD_ONLY` のみ、`resultSetConcurrency` は `CONCUR_READ_ONLY` のみ、`resultSetHoldability` は `CLOSE_CURSORS_AT_COMMIT` のみサポートします。

3.2.1.3 未サポートの機能

- 標準機能
 - `abort(executor)`
 - `prepareCall(sql)`
 - `getSchema()`
 - `setSchema(schema)`
- オプション機能
 - `createArrayOf(typeName, elements)`
 - `createBlob()`
 - `createClob()`
 - `createNClob()`
 - `createSQLXML()`
 - `createStruct(typeName, attributes)`
 - `getNetworkTimeout()`
 - `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()`
- `setSavepoint(name)`
- `setTypeMap(map)`

3.2.2 DatabaseMetaData インターフェース

- ResultSet を返す属性

- `getColumns(catalog, schemaPattern, tableNamePattern, columnNamePattern)`
 指定の `tableNamePattern` に対応する ResultSet を返却します。`tableNamePattern` にはワイルドカードは指定できず、完全一致するテーブルがなければ空の結果を返却します。指定された他の絞り込み条件は無視されます。

表 2 : `getColumns()` 結果列一覧

結果カラム名	値	JDBC仕様との一致
TABLE_CAT	null	○
TABLE_SCHEM	null	○
TABLE_NAME	(テーブル名)	○
COLUMN_NAME	(カラム名)	○
DATA_TYPE	(DatabaseMetaData#getTypeInfo()参照)	○
TYPE_NAME	(ResultSetMetaData#getColumnTypeName()の結果を大文字にした値と対	○
COLUMN_SIZE	2000000000	○
BUFFER_LENGTH	2000000000	○
DECIMAL_DIGITS	10	○
NUM_PREC_RADIX	10	○
NULLABLE	1 (DatabaseMetaData#columnNullable)	○
REMARKS	null	○
COLUMN_DEF	null	○
SQL_DATA_TYPE	0	○
SQL_DATETIME_SUB	0	○
CHAR_OCTET_LENGTH	2000000000	○
ORDINAL_POSITION	(1から開始)	○
IS_NULLABLE	'YES'	○
SCOPE_CATALOG	null	○
SCOPE_SCHEMA	null	○
SCOPE_TABLE	null	○
SOURCE_DATA_TYPE	null	○
IS_AUTOINCREMENT	'NO'	○
IS_GENERATEDCOLUMN	'NO'	○

- `getPrimaryKeys(catalog, schema, table)`
 主キーの有無に応じて、TABLE_NAME、COLUMN_NAME、KEY_SEQ が設定された ResultSet を返却します。返却される ResultSet の他のカラムには全て null が設定されます。指定された他の絞り込み条件は無視されます。
- `getTables(catalog, schemaPattern, tableNamePattern, types)`

TABLE_NAME と TABLE_TYPE のみ設定された ResultSet を返却します。TABLE_TYPE には常に 'TABLE' が設定されます。types が指定された場合は、types に 'TABLE' と一致する要素が存在しなければ常に空の結果を返します。types 内の文字列要素の大文字小文字表記の違いは無視されます。tableNamePattern による絞り込みは '%' のみ有効です。

- getTableTypes()
 - TABLE_TYPE として 'TABLE' が設定された、1つの行からなる ResultSet を返却します。
- getTypeInfo()
 - JDBC 仕様通りに振る舞います。
 - 型別の情報は以下の通りです。

表 3 : getTypeInfo() 結果 TYPE_NAME, DATA_TYPE 列一覧

TYPE_NAME	DATA_TYPE
'BLOB'	2004 (Types#BLOB)
'NULL'	0 (Types#NULL)
'REAL'	7 (Types#REAL)
'TEXT'	12 (Types#VARCHAR)
'INTEGER'	4 (Types#BIGINT)

すべての型で共通の情報は以下の通りです。

表 4 : getTypeInfo() 結果列一覧

結果カラム名	値
PRECISION	0
LITERAL_PREFIX	null
LITERAL_SUFFIX	null
CREATE_PARAMS	null
NULLABLE	1 (DatabaseMetaData#typeNullable)
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 単純値を返す属性

表 5 : DatabaseMetaData 単純値を返すメソッドと結果一覧(1)

メソッド	結果
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()	(ユーザ名)
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

表 6 : DatabaseMetaData 単純値を返すメソッドと結果一覧(2)

メソッド	結果
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)	typeがTYPE_FORWARD_ONLY、 concurrencyがCONCUR_READ_ONLYの場
supportsResultSetHoldability(holdability)	CLOSE_CURSORS_AT_COMMITの場合のみ
supportsResultSetType()	TYPE_FORWARD_ONLYの場合のみ
supportsSavepoints()	FALSE
supportsSchemasInDataManipulation()	FALSE
supportsSchemasInIndexDefinitions()	FALSE

3.2.2.2 未サポートの属性

表 7: DatabaseMetaData 未サポートのメソッド一覧

メソッド名	結果
generatedKeyAlwaysReturned()	SQLFeatureNotSupportedException
getAttributes	空のResultSet
getBestRowIdentifier	空のResultSet
getCatalogs	空のResultSet
getClientInfoProperties	空のResultSet
getColumnPrivileges	空のResultSet
getCrossReference	空のResultSet
getExportedKeys	空のResultSet
getFunctionColumns	空のResultSet
getFunctions	空のResultSet
getImportedKeys	空のResultSet
getIndexInfo	空のResultSet
getProcedureColumns	空のResultSet
getProcedures	空のResultSet
getPseudoColumns	空のResultSet
getSchemas()	空のResultSet
getSchemas(catalog, schemaPattern)	空のResultSet
getSuperTables	空のResultSet
getSuperTypes	空のResultSet
getTablePrivileges	空のResultSet
getUDTs	空のResultSet
getVersionColumns	空のResultSet

3.2.3 Statement インターフェース

3.2.3.1 フェッチサイズの設定・取得

値のチェックでは、この Statement の getMaxRows() で得られるロウ数より超えないこともチェックします。この値に関する制限は、JDBC4.0 より前の JDBC 仕様でのみ明記されていました。ただし、以前の JDBC 仕様とは異なり、getMaxRows() の結果がデフォルト値 0 に設定されている場合を除きます。

3.2.3.2 フェッチ方向の設定・取得

フェッチ方向は FETCH_FORWARD のみをサポートします。FETCH_FORWARD 以外が指定された場合、SQLException が発生します。この挙動は JDBC 仕様とは異なります。

3.2.3.3 未サポートの機能

- 標準機能
 - 以下のメソッドの呼び出しは無視されます。
 - ◇ setCursorName(name)
 - ◇ setEscapeProcessing(enable)
 - ◇ setPoolable(poolable)
- オプション機能
 - 以下のメソッドを呼び出すと SQLFeatureNotSupportedException が発生します。

- ◇ `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 インターフェース

3.2.4.1 パラメータの設定・取得

以下のメソッドをサポートします。ただし、一部のメソッドについては JDBC 仕様とは異なる振る舞いをします。

- `clearParameters()`
- `getMetaData()`
- `getParameterMetaData()`
- `setBlob(int parameterIndex, Blob x)`
- `setBoolean(int parameterIndex, boolean x)`
 - JDBC 仕様とは異なり、パラメータに `x` が `true` のとき 1、`false` のとき 0 を設定します。
- `setByte(int parameterIndex, byte x)`
- `setDate(int parameterIndex, Date x)`
 - JDBC 仕様とは異なり、パラメータに `x.getTime()` で得られる値を設定します。
- `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)`
 - JDBC 仕様とは異なり、`x` が `Blob`、`Boolean`、`Byte`、`Date`、`Double`、`Float`、`Integer`、`Long`、`Short`、`String`、`Time`、`TimeStamp` のオブジェクトの場合、対応する `set` メソッドと同じ結果になり、その他のクラスのオブジェクトの場合、エラーとなります。
- `setShort(int parameterIndex, short x)`
- `setString(int parameterIndex, String x)`
- `setTime(int parameterIndex, Time x)`
 - JDBC 仕様とは異なり、パラメータに `x.getTime()` で得られる値を設定します。
- `setTimestamp(int parameterIndex, Timestamp x)`
 - JDBC 仕様とは異なり、パラメータに `x.getTime()` で得られる値を設定します。

3.2.4.2 SQL の実行

以下のメソッドをサポートします。

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

3.2.4.3 未サポートの機能

- 標準機能
 - 以下のメソッドを呼び出すと `SQLFeatureNotSupportedException` が発生します。この挙動は JDBC 仕様とは異なります。
 - ◇ `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)`
- オプション機能
 - 以下のメソッドを呼び出すと `SQLFeatureNotSupportedException` が発生します。引数を省略しているものは、全てのオーバーロードが未サポートです。
 - ◇ `setArray`
 - ◇ `setAsciiStream`
 - ◇ `setBinaryStream`
 - ◇ `setBlob(int parameterIndex, InputStream inputStream)`
 - ◇ `setBlob(int parameterIndex, InputStream inputStream, long length)`
 - ◇ `setBytes`
 - ◇ `setCharactorStream`
 - ◇ `setClob`
 - ◇ `setNCharactorStream`
 - ◇ `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 インターフェース

全てのメソッドをサポートしますが、以下のメソッドは引数 `param` の値によらず常に固定の値を返します。

表 8 : ParameterMetaData 固定値を返すメソッド一覧

メソッド	結果
<code>getParameterType</code>	<code>Types.NULL</code>
<code>getParameterTypeName</code>	"NULL"
<code>getPrecision</code>	0
<code>getScale</code>	0
<code>isSinged</code>	FALSE

3.2.6 ResultSet インターフェース

3.2.6.1 フェッチサイズの設定・取得

指定された値のチェックのみ行い、設定の変更は実際のフェッチ処理には影響しません。値のチェックでは、対象の ResultSet の生成元の Statement の `getMaxRows()` で得られるロウ数より超えないこともチェックします。この制限は、JDBC4.0 より前の JDBC 仕様でのみ明記されていました。ただし、以前の JDBC 仕様とは異なり、`getMaxRows()` の結果がデフォルト値 0 に設定されている場合を除きます。実際のフェッチ処理には影響しませんが、変更した設定値を取得できます。

3.2.6.2 フェッチ方向の設定・取得

フェッチ方向は `FETCH_FORWARD` のみをサポートします。`FETCH_FORWARD` 以外が指定された場合、`SQLException` が発生します。この挙動は JDBC 仕様とは異なります。

3.2.6.3 カーソル情報の取得

カーソルに関する以下のメソッドは JDBC 仕様通りに振る舞います。`isAfterLast()` は、次の行が存在しない場合は `false` を返します。

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

フェッチ方向は `FETCH_FORWARD` のみをサポートしているため、次のメソッドを呼び出すと `FETCH_FORWARD` タイプの ResultSet に対する呼び出しを原因とする `SQLException` が発生します。

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

3.2.6.4 警告の管理

警告は記録されないため、警告を管理するメソッドの挙動は次のようになります。

表 9 : ResultSet 警告に関するメソッドの挙動

メソッド	挙動
<code>getWarnings()</code>	nullを返却
<code>clearWarnings()</code>	警告はクリアされない

3.2.6.5 固定値を返す属性

ResultSet がオープンされている間、常に固定の値を返すメソッドのサポート状況は次の通りです。

表 10 : ResultSet 固定値を返すメソッド一覧

メソッド	結果
getCursorName()	nullを返却
getType()	TYPE_FORWARD_ONLYを返却
getConcurrency()	CONCUR_READ_ONLYを返却
getMetaData()	JDBC準拠
getStatement()	JDBC準拠

3.2.6.6 型変換

指定の列の値を取得する際、ResultSet が保持する型と求める型とが異なる場合は、次の組み合わせに限り型変換を試みます。

表 11 : 型変換を行う組み合わせ一覧

変換先のJava型	INTEGER	REAL	BLOB	NULL	TEXT
boolean	○ ※1	○ ※1	○ ※2 ※3	○	○ ※3
byte	○	○	○ ※2	○	○ ※4
short	○	○	○ ※2	○	○ ※4
int	○	○	○ ※2	○	○ ※4
long	○	○	○ ※2	○	○ ※4
float	○	○	○ ※2	○	○ ※4
double	○	○	○ ※2	○	○ ※4
byte[]	○ ※2	○ ※2	○	○	○ ※2
java.sql.Date	○ ※5	○ ※5	○ ※2 ※4	○	○ ※4 ※6
Time	○ ※5	○ ※5	○ ※2 ※4	○	○ ※4 ※6
Timestamp	○ ※5	○ ※5	○ ※2 ※4	○	○ ※4 ※6
String	○	○	○ ※2	○	○
Blob	○ ※2	○ ※2	○	○	○ ※2
Object	○ (Long)	○ (Double)	○ (Blob)	○	○ (String)

- (※1). long に変換したうえで変換。変換した値が0または1の場合は JDBC 仕様通り。その他の場合は true
- (※2). UTF-8 文字列とみなして変換
- (※3). "0"も"1"も含まない場合は SQLException。"0"も"1"も含む場合は前者を優先
- (※4). パースに失敗した場合は SQLException。前後に数値以外の文字列が含まれている場合や、long もしくは double としてオーバーフローする値が含まれていた場合にパースに失敗する
- (※5). エポック時刻 (UTC での 1970 年 1 月 1 日の開始時点) からの経過ミリ秒とみなして変換。java.util.Calendar の指定は無視される。JDBC 仕様とは異なる。
- (※6). 以下のルールで文字列表現の時刻を変換する。受付可能な表現を java.text.SimpleDateFormat と同様のパターン文字列であらわしたものは次の通り (ただしタイムゾーンを除く)。タイムゾーンについては、文字列に含まれるものを優先し、指定の java.util.Calendar 指定時はその内容を参照する。タイムゾーン文字

列としては、`java.text.SimpleDateFormat` の「z」または「Z」パターンで解釈できる表現のほか、UTCであることを示す「Z」表現を受け付ける。タイムゾーン情報が存在しない場合は、システム設定によらず UTC とみなされる

- 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

3.2.6.7 カラムの値取得

サポートされている型変換先の型と対応するメソッドより、カラムの値を取得できます。カラムの指定方法としては、カラムラベルとカラムインデックスの両方をサポートします。

その他、次の機能を使用できます。

- `getBinaryStream`
byte[]への型変換結果に相当します
- `wasNull`
JDBC 準拠

3.2.6.8 エラー処理

- 不正なカラムインデックス
不正なカラムインデックスを指定して値を取得しようとした場合、`JDBC_COLUMN_INDEX_OUT_OF_RANGE` エラーからなる `SQLException` が発生します。
- 型変換エラー
型変換に失敗した場合、`JDBC_VALUE_TYPE_CONVERSION_FAILED` エラーからなる `SQLException` が発生します。

3.2.6.9 未サポートの機能

次のオプション機能は未サポートです。引数を省略しているものは、全てのオーバーロードが未サポートです。

- `cancelRowUpdates()`
- `getArray`
- `getAsciiStream`
- `getBigDecimal`
- `getBinaryStream`
- `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()`
- `insert` で始まる全メソッド
- `update` で始まる全メソッド
- `delete` で始まる全メソッド

3.2.7 ResultSetMetaData インターフェース

3.2.7.1 カラムの型

ResultSetMetaData が返却するカラムの型について説明します。ここでは、DatabaseMetaData#getTypeInfo() が返却する型名を基準として説明します。カラムの型は、ResultSetMetaData の生成元の状態によって変動することがあります。このドライバのサポート範囲では、ResultSetMetaData の生成元は ResultSet に限定されます。ResultSet より生成された ResultSetMetaData の場合、ResultSet のカーソル位置によってカラムの型が異なります。

表 12 : カーソル位置による ResultSetMetaData のカラム型

カーソル位置	カラムの型
行を持たない空のResultSet	常にNULL
先頭行の直前	先頭行と同一
最終行の直後	最終行と同一
有効な現在行に位置する場合	現在行の指定カラムの値の型

カラムの型を取得する各種メソッドの仕様は次の通りです。

表 13 : カラムの型を取得するメソッドの戻り値一覧

メソッド名	INTEGER	REAL	BLOB	NULL	TEXT
<code>getColumnClassName</code>	<code>java.lang.Long</code>	<code>java.sql.Double</code>	<code>java.sql.Blob</code>	<code>java.lang.Object</code>	<code>java.lang.String</code>
<code>getColumnType</code>	<code>Types#BIGINT</code>	<code>Types#REAL</code>	<code>Types#BLOB</code>	<code>Types#NULL</code>	<code>Types#VARCHAR</code>
<code>getColumnTypeName</code>	<code>"integer"</code>	<code>"float"</code>	<code>"blob"</code>	<code>"null"</code>	<code>"text"</code>

3.2.7.2 単純値を返す属性

表 14 : ResultSetMetaData 単純値を返すメソッドと戻り値一覧

メソッド名	結果
getCatalogName	""
getColumnDisplaySize	Integer#MAX_VALUE
getPrecision	0
getScale	0
getSchemaName	""
getTableName	""
isAutoIncrement	FALSE
isCaseSensitive	TRUE
isCurrency	FALSE

4. サンプル

JDBC のサンプルプログラムは以下のとおりです。

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

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

        // GridDB クラスタとの接続
        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();
        }
    }
}
```

東芝ソリューション株式会社