SQL Anywhere サーバ
文書バージョン: 17 – 2016-05-11

SQL Anywhere XS JavaScript API リファレンス

目次

1 XS JavaScript アプリケーションプログラミング

SQL Anywhere XS JavaScript ドライバは、SQL Anywhere データベースへの接続、SQL クエリの発行、結果セットの取得に使用できます。

SQL Anywhere XS JavaScript ドライバを使用すると、JavaScript 環境からデータベースと対話できます。SQL Anywhere XS API は、SAPHANA XS エンジンで提供されている SAP HANA XS JavaScript Database API に基づいています。様々なバージョンの Node.js. のドライバが使用可能です。

XS JavaScript ドライバは、SQL Anywhere 外部環境サポートを使用して JavaScript で記述したサーバ側アプリケーションもサポートしています。

> **注記**
>
> XS JavaScript ドライバに加え、最低限の機能を備えた軽量の Node.js ドライバを使用すると、小さな結果セットを処理できます。このドライバを使用する Node.js アプリケーションプログラミングについては、マニュアルの別の場所に記載されています。lightweight ドライバは、小さな結果セットを取得するためにデータベースサーバに簡単にすばやく接続する必要がある単純な Web アプリケーションに最適です。ただし、JavaScript アプリケーションが大きな結果を処理する必要がある場合、制御権限を拡大する必要がある場合、またはより完全なアプリケーションプログラミングインタフェースにアクセスする必要がある場合、XS JavaScript ドライバを使用する必要があります。

Node.js をコンピュータにインストールする必要があります。Node.js 実行ファイルを格納しているフォルダは、PATH に含まれている必要があります。Node.js ソフトウェアは nodejs.org で入手できます。

Node.js がドライバを探す場合、NODE_PATH 環境変数が XS JavaScript ドライバの場所を含んでいることを確認します。次に、Microsoft Windows の例を示します。

```
SET NODE_PATH=%SQLANY17%¥Node
```

> **注記**
>
> Mac OS X 10.11 上で、SQLANY_API_DLL 環境変数を `libdbcapi_r.dylib` へのフルパスに設定します。

次に、XS JavaScript ドライバを使用する単純な Node.js アプリケーションを示します。

```
var sqla = require( 'sqlanywhere-xs' );
var cstr = { Server     : 'demo',
             UserID     : 'DBA',
             Password   : 'sql'
           };
var conn = sqla.createConnection( cstr );
conn.connect();
console.log( 'Connected' );
stmt = conn.prepareStatement( "SELECT * FROM Customers" );
stmt.execute();
var result = stmt.getResultSet();
while ( result.next() )
{
    console.log( result.getString(1) +
            " " + result.getString(3) +
            " " + result.getString(2) );
}
```

```
conn.disconnect();
console.log( 'Disconnected' );
```

このプログラムはサンプルデータベースに接続し、SQL SELECT 文を実行し、結果セットの最初の 3 つのカラム (各顧客の ID、GivenName、Surname) のみを表示します。その後、データベースからの接続を切断します。

この JavaScript コードがファイル `xs-sample.js` に格納されていたとします。このプログラムを実行するには、コマンドプロンプトを開き、次の文を実行します。NODE_PATH 環境変数が適切に設定されていることを確認します。

```
node xs-sample.js
```

次の JavaScript では、準備文とバッチの使用例を示します。この例では、データベーステーブルを作成し、そこに 100 個の正の整数を格納します。

```
var sqla = require( 'sqlanywhere-xs' );
var cstr = { Server     : 'demo',
             UserID     : 'DBA',
             Password   : 'sql'
            };
var conn = sqla.createConnection( cstr );
conn.connect();
conn.prepareStatement( "CREATE OR REPLACE TABLE mytable( col1 INT)" ).execute();
var stmt = conn.prepareStatement( "INSERT INTO mytable VALUES(?)" );
var BATCHSIZE = 100;
stmt.setBatchSize(BATCHSIZE);
for ( var i = 1; i <= BATCHSIZE; i++ )
{
    stmt.setInteger( 1, i );
    stmt.addBatch();
}
stmt.executeBatch();
stmt.close();
conn.commit();
conn.disconnect();
```

次の JavaScript では、準備文と例外処理の使用例を示します。`getcustomer` 関数は、指定した顧客のテーブルローに対応するハッシュまたはエラーメッセージを返します。

```
function getcustomer( conn, customer_id )
{
    try
    {
        var query = 'SELECT * FROM Customers WHERE ID=?';

        var pstmt = conn.prepareStatement(query);
        pstmt.setInteger( 1, customer_id );
        var rs = pstmt.executeQuery();
        if ( rs.next() )
        {
            return(
            {
                id          : rs.getInteger(1),
                surname     : rs.getString(2),
                givenname   : rs.getString(3),
                street      : rs.getString(4),
                city        : rs.getString(5),
                state       : rs.getString(6),
                country     : rs.getString(7),
                postalcode  : rs.getString(8),
                phone       : rs.getString(9),
                companyname : rs.getString(10)
            } );
        }
```

```
        else
        {
            return 'No customer with ID=' + customer_id;
        }
    }
    catch (ex)
    {
        return ex.message;
    }
    finally
    {
        if( pstmt )
        {
            pstmt.close();
        }
    }
}
var sqla = require( 'sqlanywhere-xs' );
var cstr = { Server     : 'demo',
             UserID     : 'DBA',
             Password   : 'sql'
            };
var conn = sqla.createConnection( cstr );
conn.connect();
for ( var i = 200; i < 225; i++ )
{
    console.log( getcustomer( conn, i ) );
}
conn.close();
conn.disconnect();
```

注記

主な SQL Anywhere マニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows のスタートメニューを使用してアクセスするか (Microsoft Windows)、`C:¥Program Files¥SQL Anywhere 17¥Documentation` にナビゲートします。

また、DocCommentXchange の Web で、主な SQL Anywhere API リファレンスマニュアルにアクセスすることもできます。http://dcx.sap.com

このセクションの内容:

CallableStatement クラス [9 ページ]
: 呼び出し可能文オブジェクト。

Connection クラス [40 ページ]
: データベースへの接続を表します。

ParameterMetaData クラス [49 ページ]
: ParameterMetaData は、準備文のメタデータを表します。

PreparedStatement クラス [57 ページ]
: PreparedStatement は、準備された SQL 文を表します。

ResultSet クラス [79 ページ]
: 呼び出し可能文オブジェクト。

ResultSetMetaData クラス [93 ページ]
: ResultSetMetaData は結果セットのメタデータを表します。

1.1 CallableStatement クラス

呼び出し可能文オブジェクト。

構文

```
class CallableStatement
```

メンバー

CallableStatement のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
	close() [13 ページ]	呼び出し可能な準備文を閉じます。
	execute() [14 ページ]	呼び出し可能な準備文を実行します。
Number	getBigInt(Integer) [14 ページ]	指定したパラメータの Number 値を取得します。
Buffer	getBlob(Integer) [15 ページ]	指定したパラメータの Buffer 値を取得します。
String	getClob(Integer) [15 ページ]	指定したパラメータの String 値を取得します。
Date	getDate(Integer) [16 ページ]	指定したパラメータの Date 値を取得します。
Number	getDecimal(Integer) [17 ページ]	指定したパラメータの Number 値を取得します。
Number	getDouble(Integer) [18 ページ]	指定したパラメータの Number 値を取得します。
Integer	getInteger(Integer) [18 ページ]	指定したパラメータの Number 値を取得します。
ResultSetMetaData	getMetaData() [19 ページ]	ResultSetMetaData オブジェクトを作成して返します。
Boolean	getMoreResults() [19 ページ]	次の結果セットを取得します。
ParameterMetaData	getParameterMetaData() [20 ページ]	ParameterMetaData オブジェクトを返します。
Number	getReal(Integer) [20 ページ]	指定したパラメータの Number 値を取得します。
ResultSet	getResultSet() [21 ページ]	ResultSet オブジェクトを作成して返します。

タイプ	メソッド	説明
Integer	getSmallInt(Integer) [21 ページ]	指定したパラメータの Number 値を取得します。
String	getString(Integer) [22 ページ]	指定したパラメータの String 値を取得します。
String	getText(Integer) [23 ページ]	指定したパラメータの String 値を取得します。
Date	getTime(Integer) [23 ページ]	指定したパラメータの Date 値を取得します。
Date	getTimestamp(Integer) [24 ページ]	指定したパラメータの Date 値を取得します。
Integer	getTinyInt(Integer) [25 ページ]	指定したパラメータの Number 値を取得します。
Integer	getUnsigned(Integer) [25 ページ]	指定したパラメータの Number 値を取得します。
Boolean	isClosed() [26 ページ]	文が閉じられているかどうかを確認します。
	setBigInt(Integer, Number) [27 ページ]	指定した Number 値にパラメータを設定します。
	setBlob(Integer, Buffer) [27 ページ]	指定した JavaScript Buffer 値にバイナリパラメータを設定します。
	setBString(Integer, String) [28 ページ]	指定した String 値にパラメータを設定します。
	setClob(Integer, String) [28 ページ]	指定した String 値にパラメータを設定します。
	setDate [29 ページ]	指定した JavaScript 日付値に DATE パラメータを設定します。
	setDecimal(Integer, Number) [31 ページ]	指定した Number 値にパラメータを設定します。
	setDouble(Integer, Number) [31 ページ]	指定した Number 値にパラメータを設定します。
	setInteger(Integer, Integer) [32 ページ]	指定した Number 値にパラメータを設定します。
	setNull(Integer) [33 ページ]	指定したバインドパラメータを NULL に設定します。
	setReal(Integer, Number) [33 ページ]	指定した Number 値にパラメータを設定します。
	setSmallInt(Integer, Integer) [34 ページ]	指定した Number 値にパラメータを設定します。
	setString(Integer, String) [34 ページ]	指定した String 値にパラメータを設定します。
	setText(Integer, String) [35 ページ]	指定した String 値にパラメータを設定します。

タイプ	メソッド	説明
	setTime [35 ページ]	指定した JavaScript 日付値の時間部分に TIME パラメータを設定します。
	setTimestamp [37 ページ]	指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。
	setTinyInt(Integer, Integer) [39 ページ]	指定した Number 値に TINYINT パラメータを設定します。
	setUnsigned(Integer, Integer) [40 ページ]	指定した 32 ビット符号なし整数値にパラメータを設定します。

備考

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=dba;PWD=sql" );
var callStmt = conn.prepareCall( "CALL ShowCustomers()" );
callStmt.execute();
var result = callStmt.getResultSet();
while ( result.next() )
{
   console.log( result.getInteger(1) +
          " " + result.getString(2) );
}
callStmt.close();
conn.close();
```

このセクションの内容:

close() メソッド [13 ページ]
　呼び出し可能な準備文を閉じます。

execute() メソッド [14 ページ]
　呼び出し可能な準備文を実行します。

getBigInt(Integer) メソッド [14 ページ]
　指定したパラメータの Number 値を取得します。

getBlob(Integer) メソッド [15 ページ]
　指定したパラメータの Buffer 値を取得します。

getClob(Integer) メソッド [15 ページ]
　指定したパラメータの String 値を取得します。

getDate(Integer) メソッド [16 ページ]
　指定したパラメータの Date 値を取得します。

getDecimal(Integer) メソッド [17 ページ]
　指定したパラメータの Number 値を取得します。

getDouble(Integer) メソッド [18 ページ]
　指定したパラメータの Number 値を取得します。

指定した Number 値にパラメータを設定します。

setDouble(Integer, Number) メソッド [31 ページ]
指定した Number 値にパラメータを設定します。

setInteger(Integer, Integer) メソッド [32 ページ]
指定した Number 値にパラメータを設定します。

setNull(Integer) メソッド [33 ページ]
指定したバインドパラメータを NULL に設定します。

setReal(Integer, Number) メソッド [33 ページ]
指定した Number 値にパラメータを設定します。

setSmallInt(Integer, Integer) メソッド [34 ページ]
指定した Number 値にパラメータを設定します。

setString(Integer, String) メソッド [34 ページ]
指定した String 値にパラメータを設定します。

setText(Integer, String) メソッド [35 ページ]
指定した String 値にパラメータを設定します。

setTime メソッド [35 ページ]
指定した JavaScript 日付値の時間部分に TIME パラメータを設定します。

setTimestamp メソッド [37 ページ]
指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。

setTinyInt(Integer, Integer) メソッド [39 ページ]
指定した Number 値に TINYINT パラメータを設定します。

setUnsigned(Integer, Integer) メソッド [40 ページ]
指定した 32 ビット符号なし整数値にパラメータを設定します。

1.1.1 close() メソッド

呼び出し可能な準備文を閉じます。

構文

```
callableStatement.close ()
```

備考

このメソッドは準備文を閉じ、割り当てられたリソースを解放します。

非同期コールバックをサポートしています。

1.1.2 execute() メソッド

呼び出し可能な準備文を実行します。

構文

```
callableStatement.execute ()
```

備考

このメソッドは SQL 準備文を実行します。実行前にすべてのパラメータをバインドさせる必要があります。

非同期コールバックをサポートしています。

1.1.3 getBigInt(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getBigInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドは、SQL BIGINT パラメータ値を数値として返します。

JavaScript 数値は、IEEE 754 倍精度浮動小数点を使用して実装されます。64 ビット浮動小数点は mantissa で 53 ビットをデプロイしているため、表現可能な正確な最大整数値は +/- 2^53 (+/- 9007199254740992) です。これより大きな BIGINT 値または UNSIGNED BIGINT 値は近似値です。

1.1.4 getBlob(Integer) メソッド

指定したパラメータの Buffer 値を取得します。

構文

```
callableStatement.getBlob (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Node.js Buffer オブジェクトとして値を返します。

備考

このメソッドは、バイナリパラメータ値を Node.js Buffer として返します。

1.1.5 getClob(Integer) メソッド

指定したパラメータの String 値を取得します。

構文

```
callableStatement.getClob (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、実行後にパラメータ値を文字列として返します。

1.1.6 getDate(Integer) メソッド

指定したパラメータの Date 値を取得します。

構文

```
callableStatement.getDate (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、パラメータ値を JavaScript 日付オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP および DATE SQL データ型の取得に使用できます。

1.1.7 getDecimal(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getDecimal (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドは、SQL 数値パラメータ値を数値として返します。

このメソッドを使用すると、NUMERIC や DECIMAL などの SQL データ型を取得できます。

1.1.8 getDouble(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getDouble (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドを使用すると、REAL、FLOAT、DOUBLE、BIGINT などの数値 SQL データ型を取得できます。

1.1.9 getInteger(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getInteger (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

このメソッドは、SQL 整数パラメータ値を数値として返します。

受け入れ可能な SQL データ型には TINYINT、SMALLINT、INTEGER、BIGINT、およびそれに対応する UNSIGNED があります。ただし、返される SQL 値は 32 ビット整数値に収まる必要があります。

1.1.10 getMetaData() メソッド

ResultSetMetaData オブジェクトを作成して返します。

構文

```
callableStatement.getMetaData ()
```

戻り値

ResultSetMetaData オブジェクトを返します。

1.1.11 getMoreResults() メソッド

次の結果セットを取得します。

構文

```
callableStatement.getMoreResults ()
```

戻り値

さらに結果セットがある場合は true、そうでない場合は false を返します (型: ブール値)。

備考

このメソッドは、結果セットがさらにあるかどうかを確認し、次の使用可能な結果セットをフェッチします。

非同期コールバックをサポートしています。

1.1.12 getParameterMetaData() メソッド

ParameterMetaData オブジェクトを返します。

構文

```
callableStatement.getParameterMetaData ()
```

戻り値

ParameterMetaData オブジェクトを返します。

1.1.13 getReal(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getReal (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドは、SQL REAL パラメータ値を数値として返します。

1.1.14 getResultSet() メソッド

ResultSet オブジェクトを作成して返します。

構文

```
callableStatement.getResultSet ()
```

戻り値

ResultSet オブジェクトを返します。

1.1.15 getSmallInt(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getSmallInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

このメソッドは、SQL SMALLINT または UNSIGNED SMALLINT パラメータ値を数値として返します。

1.1.16 getString(Integer) メソッド

指定したパラメータの String 値を取得します。

構文

```
callableStatement.getString (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、パラメータ値を JavaScript 文字列オブジェクトとして返します。

1.1.17 getText(Integer) メソッド

指定したパラメータの String 値を取得します。

構文

```
callableStatement.getText (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、パラメータ値を JavaScript 文字列オブジェクトとして返します。

1.1.18 getTime(Integer) メソッド

指定したパラメータの Date 値を取得します。

構文

```
callableStatement.getTime (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、パラメータ値を JavaScript 日付オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP、DATE、および TIME SQL データ型の取得に使用できます。

1.1.19 getTimestamp(Integer) メソッド

指定したパラメータの Date 値を取得します。

構文

```
callableStatement.getTimestamp (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、パラメータ値を JavaScript 日付オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP、DATE、および TIME SQL データ型の取得に使用できます。

1.1.20 getTinyInt(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getTinyInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

このメソッドは、SQL TINYINT パラメータ値を数値として返します。

1.1.21 getUnsigned(Integer) メソッド

指定したパラメータの Number 値を取得します。

構文

```
callableStatement.getUnsigned (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

返される SQL 値は 32 ビット符号なし整数値に収まる必要があります。受け入れ可能な SQL データ型には TINYINT、UNSIGNED SMALLINT、UNSIGNED INTEGER、UNSIGNED BIGINT があります。

値が 9007199254740992 (2^53) を超える場合、または -9007199254740992 (-2^53) を下回る場合、例外がスローされます。

1.1.22 isClosed() メソッド

文が閉じられているかどうかを確認します。

構文

```
callableStatement.isClosed ()
```

戻り値

閉じられている場合は true、閉じられていない場合は false (型: ブール値)。

1.1.23 setBigInt(Integer, Number) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setBigInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、BIGINT や UNSIGNED BIGINT などの大きな整数型カラムを設定する場合に使用されます。

JavaScript 数値は、IEEE 754 倍精度浮動小数点を使用して実装されます。64 ビット浮動小数点は mantissa で 53 ビットをデプロイしているため、表現可能な正確な最大整数値は +/- 2^53 (+/- 9007199254740992) です。これより大きな BIGINT 値または UNSIGNED BIGINT 値は近似値です。

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、どの JavaScript Number 値でも構いません。また、この値は、SQL DOUBLE としてサーバに送信されます。データベースサーバは、この結果を適切な SQL データ型に変換します。

1.1.24 setBlob(Integer, Buffer) メソッド

指定した JavaScript Buffer 値にバイナリパラメータを設定します。

構文

```
callableStatement.setBlob (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Buffer	value	設定するバイナリ値 (型: バッファ)。

備考

このメソッドは、指定したカラムに BLOB バインドパラメータを設定します。入力したパラメータは Node.js Buffer オブジェクトでなければなりません。

1.1.25 setBString(Integer, String) メソッド

指定した String 値にパラメータを設定します。

構文

```
callableStatement.setBString (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス
String	value	設定する文字列値。

1.1.26 setClob(Integer, String) メソッド

指定した String 値にパラメータを設定します。

構文

```
callableStatement.setClob (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス
String	value	設定する文字列値。

1.1.27 setDate メソッド

指定した JavaScript 日付値に DATE パラメータを設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
	setDate(Integer, Date) [29 ページ]	指定した JavaScript 日付値に DATE パラメータを設定します。
	setDate(Integer, String) [30 ページ]	指定した JavaScript 文字列値に DATE パラメータを設定します。

このセクションの内容:

setDate(Integer, Date) メソッド [29 ページ]
指定した JavaScript 日付値に DATE パラメータを設定します。

setDate(Integer, String) メソッド [30 ページ]
指定した JavaScript 文字列値に DATE パラメータを設定します。

1.1.27.1 setDate(Integer, Date) メソッド

指定した JavaScript 日付値に DATE パラメータを設定します。

構文

```
callableStatement.setDate (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、JavaScript の日付オブジェクトでなければならず、SQL DATE に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL DATE 値を設定するために使用できます。

1.1.27.2 setDate(Integer, String) メソッド

指定した JavaScript 文字列値に DATE パラメータを設定します。

構文

```
callableStatement.setDate (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される文字列値は、データベースサーバによる DATE 値への変換に適した形式でなければなりません。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL DATE 値を設定するために使用できます。

1.1.28 setDecimal(Integer, Number) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setDecimal (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、DECIMAL や NUMERIC などの SQL 数値データ型を設定する場合に使用できます。

JavaScript 数値は、IEEE 754 倍精度浮動小数点を使用して実装されます。64 ビット浮動小数点は mantissa で 53 ビットをデプロイしているため、表現可能な正確な最大整数値は +/- 2^53 (+/- 9007199254740992) です。これより大きな BIGINT 値または UNSIGNED BIGINT 値は近似値です。

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、どの JavaScript Number 値でも構いません。また、この値は、SQL DOUBLE としてサーバに送信されます。データベースサーバは、この結果を適切な SQL データ型に変換します。

1.1.29 setDouble(Integer, Number) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setDouble (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、FLOAT や DOUBLE などの SQL 数値データ型を設定する場合に使用できます。

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、どの JavaScript Number 値でも構いません。また、この値は、SQL DOUBLE としてサーバに送信されます。データベースサーバは、この結果を適切な SQL データ型に変換します。

1.1.30 setInteger(Integer, Integer) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setInteger (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、INTEGER などの SQL 整数データ型を設定する場合に使用できます。

このメソッドは、指定した値にバインドパラメータを設定します。値は 32 ビット符号付き整数値に収まる必要があります。

1.1.31 setNull(Integer) メソッド

指定したバインドパラメータを NULL に設定します。

構文

```
callableStatement.setNull (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス

1.1.32 setReal(Integer, Number) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setReal (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、REAL や FLOAT などの SQL 数値データ型を設定する場合に使用できます。

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、どの JavaScript Number 値でも構いません。また、この値は、SQL DOUBLE としてサーバに送信されます。データベースサーバは、この結果を適切な SQL データ型に変換します。

1.1.33 setSmallInt(Integer, Integer) メソッド

指定した Number 値にパラメータを設定します。

構文

```
callableStatement.setSmallInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、SMALLINT や UNSIGNED SMALLINT などの SQL 整数データ型を設定する場合に使用できます。

このメソッドは、指定した値にバインドパラメータを設定します。値は 16 ビット整数値に収まる必要があります。

1.1.34 setString(Integer, String) メソッド

指定した String 値にパラメータを設定します。

構文

```
callableStatement.setString (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

タイプ	名前	説明
String	value	設定する文字列値 (型: 文字列)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。このメソッドは、データベースの CHAR 文字セットを使用して値を設定します。

1.1.35 setText(Integer, String) メソッド

指定した String 値にパラメータを設定します。

構文

```
callableStatement.setText (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value	設定する文字列値 (型: 文字列)。

1.1.36 setTime メソッド

指定した JavaScript 日付値の時間部分に TIME パラメータを設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
	setTime(Integer, Date) [36 ページ]	指定した JavaScript 日付値の時間部分に TIME パラメータを設定します。

タイプ	オーバーロード名	説明
	setTime(Integer, String) [37 ページ]	指定した JavaScript 文字列値の時間部分に TIME パラメータを設定します。

このセクションの内容:

1.1.36.1 setTime(Integer, Date) メソッド

指定した JavaScript 日付値の時間部分に TIME パラメータを設定します。

構文

```
callableStatement.setTime (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、JavaScript の日付オブジェクトでなければならず、SQL TIME に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIME 値を設定するために使用できます。

1.1.36.2 setTime(Integer, String) メソッド

指定した JavaScript 文字列値の時間部分に TIME パラメータを設定します。

構文

```
callableStatement.setTime (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される文字列値は、データベースサーバによる TIME 値への変換に適した形式でなければなりません。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIME 値を設定するために使用できます。

1.1.37 setTimestamp メソッド

指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
	setTimestamp(Integer, Date) [38 ページ]	指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。
	setTimestamp(Integer, String) [38 ページ]	指定した JavaScript 文字列値に TIMESTAMP パラメータを設定します。

このセクションの内容:

setTimestamp(Integer, Date) メソッド [38 ページ]
指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。

setTimestamp(Integer, String) メソッド [38 ページ]
指定した JavaScript 文字列値に TIMESTAMP パラメータを設定します。

1.1.37.1 setTimestamp(Integer, Date) メソッド

指定した JavaScript 日付値に TIMESTAMP パラメータを設定します。

構文

```
callableStatement.setTimestamp (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される値は、JavaScript の日付オブジェクトでなければならず、SQL TIMESTAMP に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIMESTAMP 値を設定するために使用できます。

1.1.37.2 setTimestamp(Integer, String) メソッド

指定した JavaScript 文字列値に TIMESTAMP パラメータを設定します。

構文

```
callableStatement.setTimestamp (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。送信される文字列値は、データベースサーバによる TIMESTAMP 値への変換に適した形式でなければなりません。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIMESTAMP 値を設定するために使用できます。

1.1.38 setTinyInt(Integer, Integer) メソッド

指定した Number 値に TINYINT パラメータを設定します。

構文

```
callableStatement.setTinyInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、指定した値にバインドパラメータを設定します。値は TINYINT 値 (0～255) に収まる必要があります。

1.1.39 setUnsigned(Integer, Integer) メソッド

指定した 32 ビット符号なし整数値にパラメータを設定します。

構文

```
callableStatement.setUnsigned (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、UNSIGNED INTEGER や UNSIGNED SMALLINT などの SQL 整数データ型を設定する場合に使用できます。

このメソッドは、指定した値にバインドパラメータを設定します。値は 32 ビット符号なし整数値に収まる必要があります。データベースサーバは、この結果を適切な SQL データ型に変換します。

1.2 Connection クラス

データベースへの接続を表します。

構文

```
class Connection
```

メンバー

Connection のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
	close() [42 ページ]	現在の接続を閉じます。
	commit() [42 ページ]	接続を使用しているデータベースに対するコミットを実行します。
	connect [43 ページ]	データベースへの接続を作成します。
	disconnect() [46 ページ]	現在の接続を閉じます。
Boolean	isClosed() [46 ページ]	接続が閉じられているかどうかを確認します。
CallableStatement	prepareCall(String) [46 ページ]	呼び出し可能文の準備を行います。
PreparedStatement	prepareStatement(String) [47 ページ]	SQL 文の準備を行います。
	rollback() [48 ページ]	接続を使用しているデータベースに対するロールバックを実行します。
	setAutoCommit(Boolean) [48 ページ]	接続のオートコミット設定を変更します。

備考

次の例では、同期呼び出しを使用してデータベースサーバへの新規接続を作成し、サーバに対して SQL クエリを発行し、結果セットを表示し、サーバとの接続を切断します。

```
var sqla = require( 'sqlanywhere-xs' );
var cstr = { Server   : 'demo',
             UserID   : 'DBA',
             Password : 'sql'
};
var conn = sqla.createConnection(cstr);
conn.connect();
console.log('Connected');
stmt = conn.prepareStatement("SELECT * FROM Customers");
stmt.execute();
var result = stmt.getResultSet();
while ( result.next() )
{
  console.log( result.getString(1) +
         " " + result.getString(3) +
         " " + result.getString(2) );
}
conn.disconnect();
console.log('Disconnected');
```

このセクションの内容:

close() メソッド [42 ページ]
現在の接続を閉じます。

commit() メソッド [42 ページ]
接続を使用しているデータベースに対するコミットを実行します。

connect メソッド [43 ページ]
データベースへの接続を作成します。

disconnect() メソッド [46 ページ]
現在の接続を閉じます。

isClosed() メソッド [46 ページ]
接続が閉じられているかどうかを確認します。

prepareCall(String) メソッド [46 ページ]
呼び出し可能文の準備を行います。

prepareStatement(String) メソッド [47 ページ]
SQL 文の準備を行います。

rollback() メソッド [48 ページ]
接続を使用しているデータベースに対するロールバックを実行します。

setAutoCommit(Boolean) メソッド [48 ページ]
接続のオートコミット設定を変更します。

1.2.1 close() メソッド

現在の接続を閉じます。

構文

```
connection.close ()
```

備考

このメソッドは現在の接続を閉じます。リソースを解放するためにプログラムを終了する前に呼び出す必要があります。このメソッドは Connection::disconnect() と同じです。

非同期コールバックをサポートしています。

1.2.2 commit() メソッド

接続を使用しているデータベースに対するコミットを実行します。

構文

```
connection.commit ()
```

備考

非同期コールバックをサポートしています。

1.2.3 connect メソッド

データベースへの接続を作成します。

オーバロードリスト

タイプ	オーバーロード名	説明
	connect(Hash) [43 ページ]	データベースへの接続を作成します。
	connect(Number) [44 ページ]	既存の接続を使用してデータベースへの接続を作成します。
	connect(String) [45 ページ]	データベースへの接続を作成します。

このセクションの内容:

connect(Hash) メソッド [43 ページ]
　データベースへの接続を作成します。

connect(Number) メソッド [44 ページ]
　既存の接続を使用してデータベースへの接続を作成します。

connect(String) メソッド [45 ページ]
　データベースへの接続を作成します。

1.2.3.1 connect(Hash) メソッド

データベースへの接続を作成します。

構文

```
connection.connect (conn_hash)
```

パラメータ

タイプ	名前	説明
Hash	conn_hash	有効な接続パラメータのハッシュ (型: オブジェクト)。

備考

このメソッドは、接続パラメータの指定したハッシュを使用して新しい接続を作成します。プログラムの終了前に、disconnect() または close() メソッドを使用してリソースを解放し、接続を切断する必要があります。";CHARSET='UTF-8'" という文字列は、すべての文字列がそのエンコーディングを使用してデータベースサーバに送信されるため、出力された接続文字列の最後に付加されます。

createConnection 呼び出しで指定した接続パラメータは上書きされます。

非同期コールバックをサポートしています。

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( { uid: 'dba',
                pwd: 'sql',
                server: 'myserver' } );
```

1.2.3.2 connect(Number) メソッド

既存の接続を使用してデータベースへの接続を作成します。

構文

```
connection.connect (dbcapi_handle)
```

パラメータ

タイプ	名前	説明
Number	dbcapi_handle	接続ハンドル (型: 数値)。

備考

このメソッドは、既存の接続を使用してデータベースに接続します。JavaScript 外部環境から取得した接続ハンドルはパラメータとして渡されます。close() または disconnect() メソッドは、リソースを解放するためにプログラム終了前に呼び出す必要があります。

非同期コールバックをサポートしています。

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( sa_dbcapi_handle );
```

1.2.3.3 connect(String) メソッド

データベースへの接続を作成します。

構文

```
connection.connect (conn_string)
```

パラメータ

タイプ	名前	説明
String	conn_string	有効な接続文字列 (型: 文字列)。

備考

このメソッドは、指定した接続文字列を使用して新しい接続を作成します。プログラムの終了前に、disconnect() または close() メソッドを使用してリソースを解放し、接続を切断する必要があります。";CHARSET='UTF-8'" という文字列は、すべての文字列がそのエンコーディングを使用してデータベースサーバに送信されるため、接続文字列の最後に付加されます。

createConnection 呼び出しで指定した接続パラメータは上書きされます。

非同期コールバックをサポートしています。

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=dba;PWD=sql;SERVER=myserver" );
```

1.2.4 disconnect() メソッド

現在の接続を閉じます。

構文

```
connection.disconnect ()
```

備考

このメソッドは現在の接続を閉じます。リソースを解放するためにプログラムを終了する前に呼び出す必要があります。このメソッドは Connection::close() と同じです。

非同期コールバックをサポートしています。

1.2.5 isClosed() メソッド

接続が閉じられているかどうかを確認します。

構文

```
connection.isClosed ()
```

戻り値

閉じられている場合は true、閉じられていない場合は false を返します (型: ブール値)。

1.2.6 prepareCall(String) メソッド

呼び出し可能文の準備を行います。

構文

```
connection.prepareCall (sql_stmt)
```

パラメータ

タイプ	名前	説明
String	sql_stmt	準備する SQL 文 (型: 文字列)。

戻り値

CallableStatement オブジェクトを返します。

備考

このメソッドは、SQL 文を準備し、呼び出し可能文オブジェクトを返します。

非同期コールバックをサポートしています。

1.2.7 prepareStatement(String) メソッド

SQL 文の準備を行います。

構文

```
connection.prepareStatement (sql_stmt)
```

パラメータ

タイプ	名前	説明
String	sql_stmt	準備する SQL 文 (型: 文字列)。

戻り値

PreparedStatement オブジェクトを返します。

備考

このメソッドは、SQL 文を準備し、準備文オブジェクトを返します。

非同期コールバックをサポートしています。

1.2.8 rollback() メソッド

接続を使用しているデータベースに対するロールバックを実行します。

構文

```
connection.rollback ()
```

備考

非同期コールバックをサポートしています。

1.2.9 setAutoCommit(Boolean) メソッド

接続のオートコミット設定を変更します。

構文

```
connection.setAutoCommit (flag)
```

パラメータ

タイプ	名前	説明
Boolean	flag	オートコミットを有効 (1) または無効 (0) にするための値 (型: ブール値)。

備考

オートコミットが有効になると、execute() 呼び出しに続いてコミットが実行されます。

1.3 ParameterMetaData クラス

ParameterMetaData は、準備文のメタデータを表します。

構文

```
class ParameterMetaData
```

メンバー

ParameterMetaData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
Integer	getParameterCount() [51 ページ]	準備文内のパラメータ数を返します。
Integer	getParameterMode(Integer) [51 ページ]	指定したパラメータのモードを返します。
String	getParameterName(Integer) [52 ページ]	指定したパラメータの名前を返します。
Integer	getParameterType(Integer) [52 ページ]	指定したパラメータのデータ型を表す数値を返します。
String	getParameterTypeName(Integer) [53 ページ]	指定したパラメータのデータ型の名称を含む文字列を返します。
Boolean	getPrecision(Integer) [53 ページ]	指定したパラメータの精度を返します。
Boolean	getScale(Integer) [54 ページ]	指定したパラメータの位取りを返します。
Boolean	hasDefault(Integer) [55 ページ]	指定したパラメータにデフォルト値があるかどうかをチェックします。
Boolean	isNullable(Integer) [55 ページ]	指定されたパラメータが NULL 入力可であるかどうかをチェックします。
Boolean	isSigned(Integer) [56 ページ]	指定されたパラメータが符号付きかどうかをチェックします。

備考

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=DBA;PWD=sql" );
var stmt = conn.prepareStatement( "INSERT INTO Departments
VALUES( :id,:name,:head )" );
stmt.setInteger( 1, 201 );
```

```
stmt.setString( 2, "Returns" );
stmt.setInteger( 3, 1250 );
try
{
    stmt.execute();
}
catch (ex)
{
}
var metaData = stmt.getParameterMetaData()
var parms = metaData.getParameterCount();
for ( var i = 1; i <= parms; i++ )
{
    console.log( metaData.getParameterName(i) );
}
stmt.close();
conn.commit();
conn.close();
```

このセクションの内容:

1.3.1 getParameterCount() メソッド

準備文内のパラメータ数を返します。

構文

```
parameterMetaData.getParameterCount ()
```

戻り値

バインドパラメータの数を返します (型: 整数)。

1.3.2 getParameterMode(Integer) メソッド

指定したパラメータのモードを返します。

構文

```
parameterMetaData.getParameterMode (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

バインドパラメータのモードを返します (型: 整数)。

備考

0 は INVALID、1 は IN、2 は OUT、3 は INOUT です。

1.3.3 getParameterName(Integer) メソッド

指定したパラメータの名前を返します。

構文

```
parameterMetaData.getParameterName (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

バインドパラメータの名前を返します (型: 文字列)。

1.3.4 getParameterType(Integer) メソッド

指定したパラメータのデータ型を表す数値を返します。

構文

```
parameterMetaData.getParameterType (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

バインドパラメータの種類を返します (型: 整数)。

備考

対応するデータ型については、a_sqlany_data_type 列挙体を参照してください。

1.3.5 getParameterTypeName(Integer) メソッド

指定したパラメータのデータ型の名称を含む文字列を返します。

構文

```
parameterMetaData.getParameterTypeName (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

バインドパラメータの種類を返します (型: 文字列)。

1.3.6 getPrecision(Integer) メソッド

指定したパラメータの精度を返します。

構文

```
parameterMetaData.getPrecision (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

符号付きの場合は true、符号なしの場合は false を返します (型: ブール値)。

備考

注意: このメソッドはサポートされていません。

1.3.7 getScale(Integer) メソッド

指定したパラメータの位取りを返します。

構文

```
parameterMetaData.getScale (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

符号付きの場合は true、符号なしの場合は false を返します (型: ブール値)。

備考

注意: このメソッドはサポートされていません。

1.3.8 hasDefault(Integer) メソッド

指定したパラメータにデフォルト値があるかどうかをチェックします。

構文

```
parameterMetaData.hasDefault (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

符号付きの場合は true、符号なしの場合は false を返します (型: ブール値)。

備考

注意: このメソッドはサポートされていません。

1.3.9 isNullable(Integer) メソッド

指定されたパラメータが NULL 入力可であるかどうかをチェックします。

構文

```
parameterMetaData.isNullable (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

符号付きの場合は true、符号なしの場合は false を返します (型: ブール値)。

備考

注意: このメソッドはサポートされていません。

1.3.10 isSigned(Integer) メソッド

指定されたパラメータが符号付きかどうかをチェックします。

構文

```
parameterMetaData.isSigned (index)
```

パラメータ

タイプ	名前	説明
Integer	index	1 から始まるパラメータのインデックス (型: 整数)。

戻り値

符号付きの場合は true、符号なしの場合は false を返します (型: ブール値)。

1.4 PreparedStatement クラス

PreparedStatement は、準備された SQL 文を表します。

構文

```
class PreparedStatement
```

メンバー

PreparedStatement のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
PreparedStatement	addBatch() [60 ページ]	パラメータのセットを追加します。
PreparedStatement	close() [61 ページ]	準備文を閉じます。
Boolean PreparedStatement	execute() [61 ページ]	準備された SQL 文を実行します。
PreparedStatement	executeBatch() [62 ページ]	バッチを実行します。
ResultSet PreparedStatement	executeQuery() [62 ページ]	準備された SQL 文を実行します。
Integer PreparedStatement	executeUpdate() [63 ページ]	準備された SQL UPDATE 文を実行します。
ResultSetMetaData PreparedStatement	getMetaData() [63 ページ]	ResultSetMetaData オブジェクトを作成して返します。
Boolean PreparedStatement	getMoreResults() [64 ページ]	次の結果セットを取得します。
ParameterMetaData PreparedStatement	getParameterMetaData() [64 ページ]	ParameterMetaData オブジェクトを作成して返します。
ResultSet PreparedStatement	getResultSet() [64 ページ]	ResultSet オブジェクトを作成して返します。
Boolean PreparedStatement	isClosed() [65 ページ]	準備文が閉じられているかどうかを確認します。
PreparedStatement	setBatchSize(Integer) [65 ページ]	指定数のパラメータセット用のスペースを確保します。
PreparedStatement	setBigInt(Integer, Number) [66 ページ]	指定したパラメータの数値を設定します。
PreparedStatement	setBlob(Integer, Buffer) [66 ページ]	指定したパラメータのバイナリ値を設定します。
PreparedStatement	setBString(Integer, String) [67 ページ]	BINARY、VARBINARY カラム型で使用される文字列パラメータを設定します。 UNICODE 文字を含む文字列で使用する必要があります。

タイプ	メソッド	説明
PreparedStatement	setClob(Integer, String) [67 ページ]	setClob は、CLOB (LONG VARCHAR) カラム型の値を指定する場合に使用します。
PreparedStatement	setDate [68 ページ]	指定したパラメータの日付値を設定します。
PreparedStatement	setDecimal(Integer, Number) [70 ページ]	指定したパラメータの数値を設定します。
PreparedStatement	setDouble(Integer, Number) [70 ページ]	指定したパラメータの数値を設定します。
PreparedStatement	setInteger(Integer, Integer) [71 ページ]	TINYINT、SMALLINT、INT などの整数カラム型で使用される整数パラメータを設定します。
PreparedStatement	setNull(Integer) [71 ページ]	指定したパラメータの NULL 値を設定します。
PreparedStatement	setReal(Integer, Number) [72 ページ]	指定したパラメータの浮動小数点値を設定します。
PreparedStatement	setSmallInt(Integer, Integer) [73 ページ]	SMALLINT カラム型で使用される整数パラメータを設定します。
PreparedStatement	setString(Integer, String) [73 ページ]	CHAR、VARCHAR、LONG VARCHAR、および他の文字カラムタイプで使用する文字列パラメータを設定します。
PreparedStatement	setText(Integer, String) [74 ページ]	setText は、TEXT (LONG VARCHAR) カラム型の値を指定する場合に使用します。
PreparedStatement	setTime [74 ページ]	指定したパラメータの時間値を設定します。
PreparedStatement	setTimestamp [76 ページ]	指定したパラメータのタイムスタンプ値を設定します。
PreparedStatement	setTinyInt(Integer, Integer) [78 ページ]	指定したパラメータの TINYINT 値を設定します。
PreparedStatement	setUnsigned(Integer, Integer) [79 ページ]	指定したパラメータの符号なし整数値を設定します。

備考

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=DBA;PWD=sql" );
var stmt = conn.prepareStatement( "DROP TABLE mytable" );
stmt.execute();
stmt.close();
conn.close();
```

このセクションの内容:

addBatch() メソッド [60 ページ]
パラメータのセットを追加します。

close() メソッド [61 ページ]
準備文を閉じます。

execute() メソッド [61 ページ]
準備された SQL 文を実行します。

executeBatch() メソッド [62 ページ]
バッチを実行します。

executeQuery() メソッド [62 ページ]
準備された SQL 文を実行します。

executeUpdate() メソッド [63 ページ]
準備された SQL UPDATE 文を実行します。

getMetaData() メソッド [63 ページ]
ResultSetMetaData オブジェクトを作成して返します。

getMoreResults() メソッド [64 ページ]
次の結果セットを取得します。

getParameterMetaData() メソッド [64 ページ]
ParameterMetaData オブジェクトを作成して返します。

getResultSet() メソッド [64 ページ]
ResultSet オブジェクトを作成して返します。

isClosed() メソッド [65 ページ]
準備文が閉じられているかどうかを確認します。

setBatchSize(Integer) メソッド [65 ページ]
指定数のパラメータセット用のスペースを確保します。

setBigInt(Integer, Number) メソッド [66 ページ]
指定したパラメータの数値を設定します。

setBlob(Integer, Buffer) メソッド [66 ページ]
指定したパラメータのバイナリ値を設定します。

setBString(Integer, String) メソッド [67 ページ]
BINARY、VARBINARY カラム型で使用される文字列パラメータを設定します。UNICODE 文字を含む文字列で使用する必要があります。

setClob(Integer, String) メソッド [67 ページ]
setClob は、CLOB (LONG VARCHAR) カラム型の値を指定する場合に使用します。

setDate メソッド [68 ページ]
指定したパラメータの日付値を設定します。

setDecimal(Integer, Number) メソッド [70 ページ]
指定したパラメータの数値を設定します。

setDouble(Integer, Number) メソッド [70 ページ]
指定したパラメータの数値を設定します。

setInteger(Integer, Integer) メソッド [71 ページ]
TINYINT、SMALLINT、INT などの整数カラム型で使用される整数パラメータを設定します。

setNull(Integer) メソッド [71 ページ]

指定したパラメータの NULL 値を設定します。

setReal(Integer, Number) メソッド [72 ページ]
指定したパラメータの浮動小数点値を設定します。

setSmallInt(Integer, Integer) メソッド [73 ページ]
SMALLINT カラム型で使用される整数パラメータを設定します。

setString(Integer, String) メソッド [73 ページ]
CHAR、VARCHAR、LONG VARCHAR、および他の文字カラムタイプで使用する文字列パラメータを設定します。

setText(Integer, String) メソッド [74 ページ]
setText は、TEXT (LONG VARCHAR) カラム型の値を指定する場合に使用します。

setTime メソッド [74 ページ]
指定したパラメータの時間値を設定します。

setTimestamp メソッド [76 ページ]
指定したパラメータのタイムスタンプ値を設定します。

setTinyInt(Integer, Integer) メソッド [78 ページ]
指定したパラメータの TINYINT 値を設定します。

setUnsigned(Integer, Integer) メソッド [79 ページ]
指定したパラメータの符号なし整数値を設定します。

1.4.1 addBatch() メソッド

パラメータのセットを追加します。

構文

```
preparedStatement.addBatch ()
```

備考

このメソッドは、パラメータ値の最後のセットを追加し、次のバッチスロットまで反復します。setBatchSize() メソッドを使用し、十分なスロットを確保しておく必要があります。

```
var stmt = conn.prepareStatement( "INSERT INTO mytable VALUES(?)" );
var BATCHSIZE = 100;
stmt.setBatchSize(BATCHSIZE);
for ( var i = 1; i <= BATCHSIZE; i++ )
{
   stmt.setInteger( 1, i );
   stmt.addBatch();
}
stmt.executeBatch();
```

1.4.2 close() メソッド

準備文を閉じます。

構文

```
preparedStatement.close ()
```

備考

このメソッドは準備文を閉じ、割り当てられたリソースを解放します。

1.4.3 execute() メソッド

準備された SQL 文を実行します。

構文

```
preparedStatement.execute ()
```

戻り値

結果セットがある場合は true、そうでない場合は false を返します

備考

このメソッドは、準備された SQL 文を実行します。結果がある場合は true、そうでない場合は false を返します。

非同期コールバックをサポートしています。

1.4.4 executeBatch() メソッド

バッチを実行します。

構文

```
preparedStatement.executeBatch ()
```

備考

このメソッドはバッチを実行します。setBatchSize() と addBatch() を使用してバッチを準備します。

非同期コールバックをサポートしています。

```
var stmt = conn.prepareStatement( "INSERT INTO mytable VALUES(?)" );
var BATCHSIZE = 100;
stmt.setBatchSize(BATCHSIZE);
for ( var i = 1; i <= BATCHSIZE; i++ )
{
   stmt.setInteger( 1, i );
   stmt.addBatch();
}
stmt.executeBatch();
```

1.4.5 executeQuery() メソッド

準備された SQL 文を実行します。

構文

```
preparedStatement.executeQuery ()
```

戻り値

ResultSet オブジェクトを返します。

備考

このメソッドは、準備された SQL 文を実行し、ResultSet オブジェクトを返します。

非同期コールバックをサポートしています。

1.4.6 executeUpdate() メソッド

準備された SQL UPDATE 文を実行します。

構文

```
preparedStatement.executeUpdate ()
```

戻り値

文の影響を受けるローの数を返します。

備考

このメソッドは、準備された SQL UPDATE 文を実行し、影響を受けるローの数を返します。

非同期コールバックをサポートしています。

1.4.7 getMetaData() メソッド

ResultSetMetaData オブジェクトを作成して返します。

構文

```
preparedStatement.getMetaData ()
```

戻り値

ResultSetMetaData オブジェクトを返します。

1.4.8 getMoreResults() メソッド

次の結果セットを取得します。

構文

```
preparedStatement.getMoreResults ()
```

戻り値

さらに結果セットがある場合は true、そうでない場合は false を返します (型: ブール値)。

備考

このメソッドは、結果セットがさらにあるかどうかを確認し、ある場合は次の結果セットまで反復します。

1.4.9 getParameterMetaData() メソッド

ParameterMetaData オブジェクトを作成して返します。

構文

```
preparedStatement.getParameterMetaData ()
```

戻り値

ParameterMetaData オブジェクトを返します。

1.4.10 getResultSet() メソッド

ResultSet オブジェクトを作成して返します。

構文

```
preparedStatement.getResultSet ()
```

戻り値

ResultSet オブジェクトを返します。

1.4.11 isClosed() メソッド

準備文が閉じられているかどうかを確認します。

構文

```
preparedStatement.isClosed ()
```

戻り値

閉じられている場合は true、閉じられていない場合は false

1.4.12 setBatchSize(Integer) メソッド

指定数のパラメータセット用のスペースを確保します。

構文

```
preparedStatement.setBatchSize (size)
```

パラメータ

タイプ	名前	説明
Integer	size	割り当てるスロットの数 (型: 整数)。

備考

```
var stmt = conn.prepareStatement( "INSERT INTO mytable VALUES(?)" );
var BATCHSIZE = 100;
stmt.setBatchSize(BATCHSIZE);
for ( var i = 1; i <= BATCHSIZE; i++ )
```

```
{
   stmt.setInteger( 1, i );
   stmt.addBatch();
}
stmt.executeBatch();
```

1.4.13 setBigInt(Integer, Number) メソッド

指定したパラメータの数値を設定します。

構文

```
preparedStatement.setBigInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、BIGINT や UNSIGNED BIGINT などの大きな整数型カラムを設定する場合に使用されます。

JavaScript 数値は、IEEE 754 倍精度浮動小数点を使用して実装されます。64 ビット浮動小数点は mantissa で 53 ビットをデプロイしているため、表現可能な正確な最大整数値は +/- 2^53 (+/- 9007199254740992) です。これより大きな BIGINT 値または UNSIGNED BIGINT 値は近似値です。

1.4.14 setBlob(Integer, Buffer) メソッド

指定したパラメータのバイナリ値を設定します。

構文

```
preparedStatement.setBlob (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Buffer	value	設定するバイナリ値 (型: バッファ)。

備考

このメソッドは、指定したカラムに BLOB (LONG BINARY) パラメータを設定します。入力したパラメータは Node.js Buffer オブジェクトでなければなりません。

1.4.15 setBString(Integer, String) メソッド

BINARY、VARBINARY カラム型で使用される文字列パラメータを設定します。UNICODE 文字を含む文字列で使用する必要があります。

構文

```
preparedStatement.setBString (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value	設定する文字列値 (型: 文字列)。

1.4.16 setClob(Integer, String) メソッド

setClob は、CLOB (LONG VARCHAR) カラム型の値を指定する場合に使用します。

構文

```
preparedStatement.setClob (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value	設定する文字列値 (型: 文字列)。

1.4.17 setDate メソッド

指定したパラメータの日付値を設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
PreparedStatement	setDate(Integer, Date) [68 ページ]	指定したパラメータの日付値を設定します。
PreparedStatement	setDate(Integer, String) [69 ページ]	指定したパラメータの日付値を設定します。

このセクションの内容:

1.4.17.1 setDate(Integer, Date) メソッド

指定したパラメータの日付値を設定します。

構文

```
preparedStatement.setDate (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、JavaScript の日付オブジェクトでなければならず、SQL DATE に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL DATE 値を設定するために使用されます。

1.4.17.2 setDate(Integer, String) メソッド

指定したパラメータの日付値を設定します。

構文

```
preparedStatement.setDate (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、データベースサーバで認識できるタイムスタンプ文字列でなければならず、サーバに送信されるときに適切な DATE 値に変換されます。

このメソッドは、SQL DATE 値を設定するために使用されます。

1.4.18 setDecimal(Integer, Number) メソッド

指定したパラメータの数値を設定します。

構文

```
preparedStatement.setDecimal (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、NUMERIC および DECIMAL などの数値カラムを設定するために使用されます。

1.4.19 setDouble(Integer, Number) メソッド

指定したパラメータの数値を設定します。

構文

```
preparedStatement.setDouble (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、指定した値にパラメータを設定します。値は JavaScript 数値で、SQL DOUBLE に変換されます。データベースサーバは、適切な SQL 値に結果をキャストします。

このメソッドは、REAL、FLOAT、DOUBLE、DECIMAL、BIGINT などの数値カラムを設定するために使用されます。

1.4.20 setInteger(Integer, Integer) メソッド

TINYINT、SMALLINT、INT などの整数カラム型で使用される整数パラメータを設定します。

構文

```
preparedStatement.setInteger (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、指定した値にパラメータを設定します。値は 32 ビット整数値に収まる必要があります。

このメソッドは、整数値を設定するために使用されます。

1.4.21 setNull(Integer) メソッド

指定したパラメータの NULL 値を設定します。

構文

```
preparedStatement.setNull (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。

備考

このメソッドは、指定したパラメータを NULL に設定します。

1.4.22 setReal(Integer, Number) メソッド

指定したパラメータの浮動小数点値を設定します。

構文

```
preparedStatement.setReal (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Number	value	設定する数値 (型: 数値)。

備考

このメソッドは、REAL、DOUBLE、FLOAT などの浮動小数点カラムを設定するために使用されます。

1.4.23 setSmallInt(Integer, Integer) メソッド

SMALLINT カラム型で使用される整数パラメータを設定します。

構文

```
preparedStatement.setSmallInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、指定した値にパラメータを設定します。値は 16 ビット整数値に収まる必要があります。

このメソッドは、小さな整数値を設定するために使用されます。

1.4.24 setString(Integer, String) メソッド

CHAR、VARCHAR、LONG VARCHAR、および他の文字カラムタイプで使用する文字列パラメータを設定します。

構文

```
preparedStatement.setString (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value	設定する文字列値 (型: 文字列)。

備考

このメソッドは、指定した文字カラムの文字列値を設定します。setString() メソッドは、データベースの CHAR 文字セットの値を設定します。

1.4.25 setText(Integer, String) メソッド

setText は、TEXT (LONG VARCHAR) カラム型の値を指定する場合に使用します。

構文

```
preparedStatement.setText (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value	設定する文字列値 (型: 文字列)。

1.4.26 setTime メソッド

指定したパラメータの時間値を設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
PreparedStatement	setTime(Integer, Date) [75 ページ]	指定したパラメータの時間値を設定します。
PreparedStatement	setTime(Integer, String) [75 ページ]	指定したパラメータの時間値を設定します。

このセクションの内容:

指定したパラメータの時間値を設定します。

1.4.26.1 setTime(Integer, Date) メソッド

指定したパラメータの時間値を設定します。

構文

```
preparedStatement.setTime (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、JavaScript の日付オブジェクトでなければならず、SQL TIME に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIME 値を設定するために使用されます。

1.4.26.2 setTime(Integer, String) メソッド

指定したパラメータの時間値を設定します。

構文

```
preparedStatement.setTime (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、データベースサーバで認識できるタイムスタンプ文字列でなければならず、サーバに送信されるときに適切な TIME 値に変換されます。

このメソッドは、SQL TIME 値を設定するために使用されます。

1.4.27 setTimestamp メソッド

指定したパラメータのタイムスタンプ値を設定します。

オーバロードリスト

タイプ	オーバーロード名	説明
PreparedStatement	setTimestamp(Integer, Date) [77 ページ]	指定したパラメータのタイムスタンプ値を設定します。
PreparedStatement	setTimestamp(Integer, String) [77 ページ]	指定したパラメータのタイムスタンプ値を設定します。

このセクションの内容:

1.4.27.1 setTimestamp(Integer, Date) メソッド

指定したパラメータのタイムスタンプ値を設定します。

構文

```
preparedStatement.setTimestamp (colIndex, value_obj)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
日付	value_obj	SQL 値を設定するための値 (型: 日付オブジェクト)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、JavaScript の日付オブジェクトでなければならず、SQL TIMESTAMP データ型に変換されます。いずれの場合でも、ローカルタイムゾーンが想定されています。

このメソッドは、SQL TIMESTAMP 値を設定するために使用されます。

1.4.27.2 setTimestamp(Integer, String) メソッド

指定したパラメータのタイムスタンプ値を設定します。

構文

```
preparedStatement.setTimestamp (colIndex, value_str)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
String	value_str	SQL 値を設定するための値 (型: 文字列)。

備考

このメソッドは、指定した値にパラメータを設定します。値は、データベースサーバで認識できるタイムスタンプ文字列でなければならず、サーバに送信されるときに適切な TIMESTAMP 値に変換されます。

このメソッドは、SQL TIMESTAMP 値を設定するために使用されます。

1.4.28 setTinyInt(Integer, Integer) メソッド

指定したパラメータの TINYINT 値を設定します。

構文

```
preparedStatement.setTinyInt (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、指定した値にパラメータを設定します。値は TINYINT カラム (0〜255) に合わせる必要があります。

このメソッドは、SQL TINYINT カラムを設定するために使用されます。

1.4.29 setUnsigned(Integer, Integer) メソッド

指定したパラメータの符号なし整数値を設定します。

構文

```
preparedStatement.setUnsigned (colIndex, value)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる準備文のパラメータのインデックス (型: 整数)。
Integer	value	設定する整数値 (型: 整数)。

備考

このメソッドは、指定した値にパラメータを設定します。値は 32 ビット符号なし整数値に収まる必要があります。このメソッドは、UNSIGNED INTEGER、INTEGER、SMALLINT、または UNSIGNED SMALLINT などの整数カラムを設定するために使用されます。

1.5 ResultSet クラス

呼び出し可能文オブジェクト。

構文

```
class ResultSet
```

メンバー

ResultSet のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
	close() [82 ページ]	このメソッドは ResultSet オブジェクトを閉じ、リソースを解放します。
Number	getBigInt(Integer) [82 ページ]	結果セットの指定カラムの数値を取得します。
Buffer	getBlob(Integer) [83 ページ]	結果セットの指定カラムのバイナリ値を取得します。
String	getClob(Integer) [83 ページ]	結果セットの指定カラムの文字列値を取得します。
Date	getDate(Integer) [84 ページ]	結果セットの指定カラムの日付値を取得します。
Number	getDecimal(Integer) [85 ページ]	結果セットの指定カラムの数値を取得します。
Number	getDouble(Integer) [85 ページ]	結果セットの指定カラムの数値を取得します。
Integer	getInteger(Integer) [86 ページ]	結果セットの指定カラムの整数値を取得します。
Number	getReal(Integer) [87 ページ]	結果セットの指定カラムの数値を取得します。
Integer	getSmallInt(Integer) [87 ページ]	結果セットの指定カラムの整数値を取得します。
String	getString(Integer) [88 ページ]	結果セットの指定カラムの文字列値を取得します。
String	getText(Integer) [89 ページ]	結果セットの指定カラムの文字列値を取得します。
Date	getTime(Integer) [89 ページ]	結果セットの指定カラムの時間値を取得します。
Date	getTimestamp(Integer) [90 ページ]	結果セットの指定カラムのタイムスタンプ値を取得します。
Integer	getTinyInt(Integer) [91 ページ]	結果セットの指定カラムの整数値を取得します。
Integer	getUnsigned(Integer) [91 ページ]	結果セットの指定カラムの整数値を取得します。
Boolean	isClosed() [92 ページ]	このメソッドは、結果セットが閉じられているかどうかをチェックします。
Boolean	next() [92 ページ]	結果セットの次のローを返します。

備考

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=dba;PWD=sql" );
var stmt = conn.prepareStatement( "SELECT * FROM Customers" );
```

```
stmt.execute();
var result = stmt.getResultSet();
while ( result.next() )
{
   console.log( result.getInteger(1) +
           " " + result.getString(2) );
}
stmt.close();
conn.close();
```

このセクションの内容:

isClosed() メソッド [92 ページ]
　このメソッドは、結果セットが閉じられているかどうかをチェックします。

next() メソッド [92 ページ]
　結果セットの次のローを返します。

1.5.1 close() メソッド

このメソッドは ResultSet オブジェクトを閉じ、リソースを解放します。

構文

```
resultSet.close ()
```

1.5.2 getBigInt(Integer) メソッド

結果セットの指定カラムの数値を取得します。

構文

```
resultSet.getBigInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドを使用すると、BIGINT、DOUBLE、REAL、FLOAT、DOUBLE などの数値 SQL 型を取得できます。

1.5.3 getBlob(Integer) メソッド

結果セットの指定カラムのバイナリ値を取得します。

構文

```
resultSet.getBlob (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Node.js Buffer オブジェクトとして値を返します。

備考

このメソッドは、指定されたカラムの値を Node.js Buffer オブジェクトとして返します。

1.5.4 getClob(Integer) メソッド

結果セットの指定カラムの文字列値を取得します。

構文

```
resultSet.getClob (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、結果セットの値を文字列として返します。

1.5.5 getDate(Integer) メソッド

結果セットの指定カラムの日付値を取得します。

構文

```
resultSet.getDate (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、指定されたカラムの値を JavaScript Date オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP、TIME、DATE SQL 型を取得するために使用されます。

1.5.6 getDecimal(Integer) メソッド

結果セットの指定カラムの数値を取得します。

構文

```
resultSet.getDecimal (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドを使用すると、BIGINT、DOUBLE、REAL、FLOAT、DOUBLE などの数値 SQL 型を取得できます。

1.5.7 getDouble(Integer) メソッド

結果セットの指定カラムの数値を取得します。

構文

```
resultSet.getDouble (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドを使用すると、BIGINT、DOUBLE、REAL、FLOAT、DOUBLE などの数値 SQL 型を取得できます。

1.5.8 getInteger(Integer) メソッド

結果セットの指定カラムの整数値を取得します。

構文

```
resultSet.getInteger (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

このメソッドは、結果セットの値を整数として返します。返される SQL 値は 32 ビット整数値に収まる必要があります。受け入れ可能な SQL 型には、SMALLINT、INTEGER、TINYINT などがあります。

1.5.9 getReal(Integer) メソッド

結果セットの指定カラムの数値を取得します。

構文

```
resultSet.getReal (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

数値として値を返します。

備考

このメソッドを使用すると、BIGINT、DOUBLE、REAL、FLOAT、DOUBLE などの数値 SQL 型を取得できます。

1.5.10 getSmallInt(Integer) メソッド

結果セットの指定カラムの整数値を取得します。

構文

```
resultSet.getSmallInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Integer として値を返します。

1.5.11 getString(Integer) メソッド

結果セットの指定カラムの文字列値を取得します。

構文

```
resultSet.getString (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、結果セットの値を文字列として返します。

1.5.12 getText(Integer) メソッド

結果セットの指定カラムの文字列値を取得します。

構文

```
resultSet.getText (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

String として値を返します。

備考

このメソッドは、結果セットの値を文字列として返します。

1.5.13 getTime(Integer) メソッド

結果セットの指定カラムの時間値を取得します。

構文

```
resultSet.getTime (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、指定されたカラムの値を JavaScript Date オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP、TIME、DATE SQL 型を取得するために使用されます。

1.5.14 getTimestamp(Integer) メソッド

結果セットの指定カラムのタイムスタンプ値を取得します。

構文

```
resultSet.getTimestamp (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Date オブジェクトとして値を返します。

備考

このメソッドは、指定されたカラムの値を JavaScript Date オブジェクトとして返します。ローカルタイムゾーンが常に想定されています。

このメソッドは、TIMESTAMP、TIME、DATE SQL 型を取得するために使用されます。

1.5.15 getTinyInt(Integer) メソッド

結果セットの指定カラムの整数値を取得します。

構文

```
resultSet.getTinyInt (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Integer として値を返します。

1.5.16 getUnsigned(Integer) メソッド

結果セットの指定カラムの整数値を取得します。

構文

```
resultSet.getUnsigned (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まる結果セットのカラムのインデックス (型: 整数)。

戻り値

Integer として値を返します。

備考

SQL 値は 32 ビット符号なし整数値に収まる必要があります。受け入れ可能な SQL 型には、UNSIGNED SMALLINT、UNSIGNED INTEGER、TINYINT などがあります。

1.5.17 isClosed() メソッド

このメソッドは、結果セットが閉じられているかどうかをチェックします。

構文

```
resultSet.isClosed ()
```

戻り値

閉じられている場合は true、閉じられていない場合は false を返します。(型: ブール値)。

1.5.18 next() メソッド

結果セットの次のローを返します。

構文

```
resultSet.next ()
```

戻り値

成功した場合は true を返します(型: ブール値)。

備考

このメソッドは、結果セットの次のローをフェッチしようとします。成功した場合は true、失敗した場合は false を返します。

非同期コールバックをサポートしています。

1.6 ResultSetMetaData クラス

ResultSetMetaData は結果セットのメタデータを表します。

構文

```
class ResultSetMetaData
```

メンバー

ResultSetMetaData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

タイプ	メソッド	説明
String	getCatalogName() [95 ページ]	データベースサーバ名を返します。
Integer	getColumnCount() [95 ページ]	結果セット内のカラム数を返します。
String	getColumnLabel(Integer) [95 ページ]	指定されたカラムのエイリアスまたは名前を返します。
String	getColumnName(Integer) [96 ページ]	指定されたカラムの名前を返します。
Integer	getColumnType(Integer) [96 ページ]	指定したカラムのデータ型を表す数値を返します。
String	getColumnTypeName(Integer) [97 ページ]	指定したカラムのデータ型の名称を含む文字列を返します。
Integer	getDisplaySize(Integer) [98 ページ]	指定したカラムの最大表示サイズを返します。
Integer	getPrecision(Integer) [98 ページ]	指定したカラムの精度を返します。

タイプ	メソッド	説明
Integer	getScale(Integer) [99 ページ]	指定したカラムの位取りを返します。
Integer	getTableName(Integer) [99 ページ]	指定したカラムのテーブル名を返します。

備考

```
var sqla = require( 'sqlanywhere-xs' );
var conn = sqla.createConnection();
conn.connect( "UID=DBA;PWD=sql" );
var stmt = conn.prepareStatement( "SELECT * FROM Customers" );
stmt.execute();
var resultMeta = stmt.getMetaData()
var columns = resultMeta.getColumnCount();
for ( var i = 1; i <= columns; i++ )
{
   console.log( resultMeta.getColumnName(i) );
}
stmt.close();
conn.close();
```

このセクションの内容:

getCatalogName() メソッド [95 ページ]
データベースサーバ名を返します。

getColumnCount() メソッド [95 ページ]
結果セット内のカラム数を返します。

getColumnLabel(Integer) メソッド [95 ページ]
指定されたカラムのエイリアスまたは名前を返します。

getColumnName(Integer) メソッド [96 ページ]
指定されたカラムの名前を返します。

getColumnType(Integer) メソッド [96 ページ]
指定したカラムのデータ型を表す数値を返します。

getColumnTypeName(Integer) メソッド [97 ページ]
指定したカラムのデータ型の名称を含む文字列を返します。

getDisplaySize(Integer) メソッド [98 ページ]
指定したカラムの最大表示サイズを返します。

getPrecision(Integer) メソッド [98 ページ]
指定したカラムの精度を返します。

getScale(Integer) メソッド [99 ページ]
指定したカラムの位取りを返します。

getTableName(Integer) メソッド [99 ページ]
指定したカラムのテーブル名を返します。

1.6.1 getCatalogName() メソッド

データベースサーバ名を返します。

構文

```
resultSetMetaData.getCatalogName ()
```

戻り値

データベースサーバ名を返します。(型: 文字列)。

1.6.2 getColumnCount() メソッド

結果セット内のカラム数を返します。

構文

```
resultSetMetaData.getColumnCount ()
```

戻り値

カラム数を返します。(型: 整数)。

1.6.3 getColumnLabel(Integer) メソッド

指定されたカラムのエイリアスまたは名前を返します。

構文

```
resultSetMetaData.getColumnLabel (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス(型: 整数)。

戻り値

カラムのエイリアスまたは名前を返します(型: 文字列)。

1.6.4 getColumnName(Integer) メソッド

指定されたカラムの名前を返します。

構文

```
resultSetMetaData.getColumnName (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス(型: 整数)。

戻り値

カラム名を返します。(型: 文字列)。

1.6.5 getColumnType(Integer) メソッド

指定したカラムのデータ型を表す数値を返します。

構文

```
resultSetMetaData.getColumnType (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス(型: 整数)。

戻り値

カラム型を Integer として返します。(型: 整数)。

備考

対応するデータ型については、a_sqlany_data_type 列挙体を参照してください。

1.6.6 getColumnTypeName(Integer) メソッド

指定したカラムのデータ型の名称を含む文字列を返します。

構文

```
resultSetMetaData.getColumnTypeName (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス(型: 整数)。

戻り値

カラム型の名前を返します(型: 文字列)。

1.6.7 getDisplaySize(Integer) メソッド

指定したカラムの最大表示サイズを返します。

構文

```
resultSetMetaData.getDisplaySize (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス(型: 整数)。

戻り値

最大表示サイズを返します(型: 整数)。

1.6.8 getPrecision(Integer) メソッド

指定したカラムの精度を返します。

構文

```
resultSetMetaData.getPrecision (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス (型: 整数)。

戻り値

精度を返します (型: 整数)。

1.6.9 getScale(Integer) メソッド

指定したカラムの位取りを返します。

構文

```
resultSetMetaData.getScale (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス (型: 整数)。

戻り値

位取りを返します (型: 整数)。

1.6.10 getTableName(Integer) メソッド

指定したカラムのテーブル名を返します。

構文

```
resultSetMetaData.getTableName (colIndex)
```

パラメータ

タイプ	名前	説明
Integer	colIndex	1 から始まるカラムのインデックス (型: 整数)。

戻り値

テーブル名を返します (型: 文字列)。

備考

注意: このメソッドはサポートされていません。

2　このマニュアルの印刷、再生、および再配布

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。

1. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
2. マニュアルに変更を加えないこと。
3. SAP 以外の人間がマニュアルの著者または情報源であるかのように示す一切の行為をしないこと。

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html