Oracledb

Project sample

var oracledb = require('oracledb');

oracledb.maxRows = dbConfig.maxRows;

oracledb.poolMax = dbConfig.poolMax;

oracledb.outFormat = oracledb.OBJECT;

oracledb.autoCommit = true;

var pool;

oracledb.createPool({

user: dbConfig.user,

password: decodePass,

connectString: dbConfig.connectstring

}, function(err, pool) {

if (err) {

logger.error('Error while executing getPool method, ', sanitize.sanitize(err), metadata);

callback(err, null);

} else {

callback(null, pool);

}

});

pool.getConnection(function(err, connection) {

connection.execute(whereQuery, callbackExecute);

}

Ideal

// myscript.js

var oracledb = require('oracledb');

oracledb.getConnection(

{

user : "hr",

password : "welcome",

connectString : "localhost/XE"

},

function(err, connection)

{

if (err) {

console.error(err.message);

return;

}

connection.execute(

`SELECT department\_id, department\_name

FROM departments

WHERE manager\_id < :id`,

[110], // bind value for :id

function(err, result)

{

if (err) {

console.error(err.message);

doRelease(connection);

return;

}

console.log(result.rows);

doRelease(connection);

});

});

function doRelease(connection)

{

connection.close(

function(err) {

if (err)

console.error(err.message);

});

}

**oracledb.autoCommit**

Boolean autoCommit

If this property is *true*, then the transaction in the current connection is automatically committed at the end of statement execution.

The default value is *false*.

#### oracledb.maxRows

Number maxRows

The maximum number of rows that are fetched by a query with [connection.execute()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "execute) when not using a [ResultSet](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "resultsetclass). Rows beyond this limit are not fetched from the database. A value of 0 means there is no limit.

#### oracledb.poolMax

Number poolMax

The maximum number of connections to which a connection pool can grow.

The default value is 4.

#### oracledb.outFormat

Number outFormat

The format of query rows fetched when using [connection.execute()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "execute) or [connection.queryStream()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "querystream). It affects both [ResultSet](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "propexecresultset)and non-ResultSet queries. It can be used for top level queries and REF CURSOR output.

This can be either of the [Oracledb constants](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "oracledbconstantsoutformat) oracledb.ARRAY or oracledb.OBJECT. The default value is oracledb.ARRAY which is more efficient.

If specified as oracledb.ARRAY, each row is fetched as an array of column values.

If specified as oracledb.OBJECT, each row is fetched as a JavaScript object. The object has a property for each column name, with the property value set to the respective column value. The property name follows Oracle's standard name-casing rules. It will commonly be uppercase, since most applications create tables using unquoted, case-insensitive names.

In TD project oracledb.OBJECT has been set

Bindvars

#### Execute Bind Direction Constants

Constants for execute() [bind parameter](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#executebindParams) dir properties.

These specify whether data values bound to SQL or PL/SQL bind parameters are passed into, or out from, the database:

oracledb.BIND\_IN // (3001) Direction for IN binds

oracledb.BIND\_INOUT // (3002) Direction for IN OUT binds

oracledb.BIND\_OUT // (3003) Direction for OUT binds

#### oracledb.createPool()

##### Prototype

Callback:

createPool(Object poolAttrs, function(Error error, Pool pool){});

Promise:

promise = createPool(Object poolAttrs);

##### Description

This method creates a pool of connections with the specified username, password and connection string.

Internally, createPool() creates an [Oracle Call Interface Session Pool](https://docs.oracle.com/database/122/LNOCI/oci-programming-advanced-topics.htm#LNOCI16617) for each Pool object.

The default properties may be overridden by specifying new properties in the poolAttrs parameter.

It is possible to add pools to the pool cache when calling createPool(). This allows pools to later be accessed by name, removing the need to pass the pool object through code. See [Connection Pool Cache](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#connpoolcache) for more details.

A pool should be terminated with the [pool.close()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "poolclose) call, but only after all connections have been released.

##### Parameters

Object poolAttrs

The poolAttrs parameter provides connection credentials and pool-specific configuration properties, such as the maximum or minimum number of connections for the pool, or the statement cache size for the connections.

The properties provided in the poolAttrs parameter override the default pooling properties of the Oracledb object. If an attribute is not set, or is null, the value of the related Oracledb property will be used.

Note that the poolAttrs parameter may have configuration properties that are not used by the createPool() method. These are ignored.

The properties of poolAttrs are described below.

String user

The database user name. Can be a simple user name or a proxy of the form alison[fred]. See the [Client Access Through a Proxy](https://docs.oracle.com/database/122/LNOCI/oci-programming-basics.htm#GUID-D77D0D4A-7483-423A-9767-CBB5854A15CC) section in the Oracle Call Interface manual for more details about proxy authentication.

String password

The password of the database user. A password is also necessary if a proxy user is specified.

String connectString

The Oracle database instance to connect to. The string can be an Easy Connect string, or a Net Service Name from a tnsnames.ora file, or the name of a local Oracle database instance. See [Connection Strings](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#connectionstrings) for examples.

Boolean externalAuth

Indicate whether connections should be established using [External Authentication](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#extauth).

The default is false.

This optional property overrides the [oracledb.externalAuth](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "propdbisexternalauth) property.

The user and password properties should not be set when externalAuth is true.

Note prior to node-oracledb 0.5 this property was called isExternalAuth.

Number stmtCacheSize

The number of statements to be cached in the [statement cache](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#stmtcache) of each connection.

This optional property overrides the [oracledb.stmtCacheSize](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "propdbstmtcachesize) property.

String poolAlias

The poolAlias is an optional property that is used to explicitly add pools to the connection pool cache. If a pool alias is provided, then the new pool will be added to the connection pool cache and the poolAlias value can then be used with methods that utilize the connection pool cache, such as [oracledb.getPool()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "getpool) and [oracledb.getConnection()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "getconnectiondb).

See [Connection Pool Cache](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#connpoolcache) for details and examples.

Number poolIncrement

The number of connections that are opened whenever a connection request exceeds the number of currently open connections.

The default value is 1.

This optional property overrides the [oracledb.poolIncrement](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "propdbpoolincrement) property.

Number poolMax

The maximum number of connections to which a connection pool can grow.

The default value is 4.

This optional property overrides the [oracledb.poolMax](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "propdbpoolmax) property.

Number poolMin

The minimum number of connections a connection pool maintains, even when there is no activity to the target database.

The default value is 0.

#### oracledb.getConnection()

##### Prototype

Callback:

getConnection([String poolAlias | Object connAttrs], function(Error error, Connection conn){});

Promise:

promise = getConnection([String poolAlias | Object connAttrs]);

##### Description

Obtains a connection from a pool in the [connection pool cache](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#connpoolcache) or creates a new, non-pooled connection.

For situations where connections are used infrequently, creating a new connection may be more efficient than creating and managing a connection pool. However, in most cases, Oracle recommends getting connections from a [connection pool](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#createpool).

#### oracledb.getPool()

##### Prototype

getPool([String poolAlias]);

##### Description

Retrieves a pool from the [connection pool cache](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#connpoolcache). Note that this is a synchronous method.

##### Parameters

String poolAlias

The pool alias of the pool to retrieve from the connection pool cache. The default value is 'default' which will retrieve the default pool.

#### connection.execute()

##### Prototype

Callback:

execute(String sql, [Object bindParams, [Object options,]] function(Error error, [Object result]){});

Promise:

promise = execute(String sql, [Object bindParams, [Object options]]);

##### Description

This call executes a single SQL or PL/SQL statement. See [SQL Execution](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#sqlexecution) for examples. Also see [queryStream()](https://github.com/oracle/node-oracledb/blob/master/doc/api.md" \l "querystream) for an alternative way of executing queries.

The statement to be executed may contain [IN binds](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#inbind), [OUT or IN OUT](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#outbind) bind values or variables, which are bound using either an object or an array.

A callback function returns a [result](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#executecallback) object, containing any fetched rows, the values of any OUT and IN OUT bind variables, and the number of rows affected by the execution of [DML](https://docs.oracle.com/database/122/CNCPT/sql.htm#CNCPT516) statements.

##### execute(): Bind Parameters

Object bindParams

The execute() function bindParams parameter is needed if there are bind variables in the statement, or if [options](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#executeoptions) are used. It can be either an object that associates values or JavaScript variables to the statement's bind variables by name, or an array of values or JavaScript variables that associate to the statement's bind variables by their relative positions. See [Bind Parameters for Prepared Statements](https://github.com/oracle/node-oracledb/blob/master/doc/api.md#bind) for more details on binding.

var sp = storedProcedures.BUSINESSES.GETCERTSFORM;

"GET": "SVC\_BOCP.PKG\_BOCP\_COMMON.PRC\_SEARCH\_BUSINESS(:i\_pagination\_paramater,:i\_counter,:i\_business\_id\_type,:i\_entity\_tin,:i\_rm\_number,:i\_business\_name)

var bindvars = {};

var counter = 0;

if(params.restartToken){

counter = parseInt(params.restartToken);

}

bindvars[constants.I\_PAGINATION\_PARAMATER] = {

val: params.dbRecordCount,

type: oracledb.NUMBER,

dir: oracledb.BIND\_IN

};

bindvars[constants.I\_COUNTER] = {

val: counter,

type: oracledb.NUMBER,

dir: oracledb.BIND\_IN

};

result.sp = sp;

result.bindvars = bindvars;