4D ODBC Pro

Introduction

Preface

4D ODBC PRO is a set of 4D external routines that allows a 4D database on Macintosh or Windows to communicate with an ODBC database. Using 4D ODBC PRO, your 4D database can display, manipulate, and modify data stored in an ODBC database.

About this Manual

This manual describes how to implement, use, and modify data sources that can be accessed by ODBC with a 4D database.

The manual is written for users already familiar with the 4D language and with ODBC's SQL language. We recommend that new users familiarize themselves with both products before continuing with the manual.

Cross-Platform

This manual explains the use of 4D ODBC PRO both on the Macintosh and Windows. Although the concepts and functionality of both versions of 4D ODBC PRO are nearly identical, the manual addresses any differences where necessary. Such differences include the graphical user interface and keyboard commands.

4D, 4D Server and 4D ODBC PRO

4D ODBC PRO can be used with either 4D or 4D Server. When used with 4D ODBC PRO, 4D enables you to create a database that can become a client of the ODBC data source. Each user with a copy of the database can connect to and use the ODBC database simultaneously.

4D Server allows you to create a multi-developer database application. When used with 4D ODBC PRO, 4D Server allows multiple developers to connect to an ODBC database.

In this manual, 4D and 4D Server are both referred to as 4D except when there is a difference between the behaviour of the two products.

Conventions

This manual uses certain conventions to help you understand the material.

- The following explanatory notes are used:

Note: Text emphasized like this provides annotations and shortcuts that will help you use 4D more productively.

Warning: Warnings like this alert you to situations where data might be lost.

- Functions: All 4D ODBC PRO functions are preceded by "ODBC_", for exemple: *ODBC_SQLAllocConnect*.
- Table names: In addition, all table names are shown in brackets in the text to help distinguish them from the names of fields, forms, and other items. For instance, the Companies table is written as the [Companies] table.

Introduction

This manual serves as a reference guide for designers, administrators, and users of integrated 4D ODBC PRO. This manual assumes that you are familiar with the overall architecture and capabilities of your ODBC data source and know 4D's procedural language and the functions available in your ODBC driver.

4D is a powerful data management tool for the Macintosh and Windows. Applications developed with 4D ODBC PRO combine the ease-of-use of a graphical interface with the power of a relational database on a microcomputer.

4D ODBC PRO makes it possible to develop applications that take advantage of the strengths of both 4D and the ODBC data source. Using 4D ODBC PRO, data stored in an SQL database can be accessed from 4D.

ODBC Architecture

Open Database Connectivity (ODBC) defines a library of functions that allows an application, such as 4D, to access a Database Management System (DBMS) using Structured Query Language (SQL). The ODBC interface offers vendor-neutral access to different database management system.

The ODBC architecture has four components:
1. the application
2. a driver manager
3. the driver
4. the data source

The main functionalities provided by any ODBC driver include the following:

- Connecting to and detaching from a DBMS
- Performing queries and providing storage areas and data formats for query results
- Allowing for online transaction processing
- Features external to the ODBC interface (DBMS specific features)
- The driver manager is a dynamically linked library (DLL) that loads drivers, providing a single entry point to ODBC functions for different drivers.

Note for Macintosh users: Since the Mac OS built-in ODBC library is not fully operational, it is necessary to install a third-party ODBC framework on this platform. Such a framework can be acquired, for example, from Openlink (http://www.openlinksw.com).

This manual reviews the important aspects of accessing a data source using the low level and control commands in 4D ODBC PRO. These closely resemble native Microsoft ODBC API calls in name, syntax and function. More information regarding the MS ODBC API you can find at the following address:

http://msdn.microsoft.com/en-us/library/ms714562(VS.85).aspx

This manual is not intended to provide a detailed analysis of ODBC operations and functionality.

Connection Choices

The first step when designing any 4D ODBC PRO application is deciding on what database to connect with. ODBC provides many functions that tell what databases are available, describe what type of databases they are, and establish a connection with them.

An application can be designed with a specific target database in mind. For example, an accounting department might have records stored in an ORACLE database. When designing a purchase order system, they know ORACLE drivers are needed. They also know what types of database this is and what attributes are needed to connect to this database. When an application is designed for a target data source, it is possible to take advantage of specific features offered by the DBMS and driver.

Alternatively, an application might need to be designed to work with any database. It will not know beforehand what driver will be used nor which database to connect with. In this case, developers must use caution to only use

those features common to all ODBC data sources.

4D ODBC PRO allows developers to develop applications for either of these two scenarios.

High-level and Low-level ODBC Commands

The high-level ODBC commands integrated into the “External Data Source” theme in 4D allow you to implement simple solutions to make your 4D applications communicate with ODBC data sources. For more information regarding the high-level ODBC commands category, please refer to the 4D Language Reference manual.

If your applications require more advanced ODBC features, you should use the “low-level” and control commands located in the 4D ODBC PRO plug-in and described in this documentation.

The various ODBC functions can be broken down into eleven distinct groups. These following groups of routines allow you to interact with a data source at different stages of communication:

- Connecting to a data source
- Obtaining information about a driver and data source
- Setting and retrieving driver attributes
- Preparing SQL requests
- Submitting requests
- Retrieving results and information about results
- Catalogue functions
- Terminating a statement
- Terminating a connection
- Macro
- Error handling

ODBC_Catalog functions

Catalog Functions, Introduction

The catalog commands enable you to retrieve information such as the list of tables stored in a data source's catalog, the list of column names in specified tables and the indexes associated with a table.

Using the catalog commands, you can:

- Get a list of columns and associated privileges for the specified table (*ODBC_SQLColumnPrivileges*)
- Obtain a list of column names in specified tables (*ODBC_SQLColumns*)
- Retrieve a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table (*ODBC_SQLForeignKeys*)
- Find out the information about data types supported by the data source (*ODBC_SQLGetTypeInfo*)
- Retrieve the column names that make up the primary key for a table (*ODBC_SQLPrimaryKeys*)
- Get the list of input and output parameters, as well as the columns that make up the result set for the specified procedures (*ODBC_SQLProcedureColumns*)
- Obtain the list of procedure names stored in a specific data source (*ODBC_SQLProcedures*)
- Find out information about columns within a specified table. Either the optimal set of columns that uniquely identifies a row in the table or the columns that are automatically updated when any value in the row is updated by a transaction. (*ODBC_SQLSpecialColumns*)
- Get a list of statistics about a single table and the indexes associated with the table (*ODBC_SQLStatistics*)
- Obtain a list of tables and the privileges associated with each table (*ODBC_SQLTablePrivileges*)
- Return a list of table, catalog, or schema names, and table types, stored in a specific data source (*ODBC_SQLTables*)

ODBC_SQLColumnPrivileges

ODBC_SQLColumnPrivileges (stmtID ; catalogName ; schemaName ; tableName ; columnName) -> Function result

Parameter	Type		Description
stmtID	Longint	⇨	Statement ID
catalogName	Text	⇦	Catalog name
schemaName	Text	⇦	Schema name
tableName	Text	⇦	Table name
columnName	Text	⇦	Column name
Function result	Longint	⇩	Returns the result of the MS ODBC API function SQLColumnPrivileges

Description

The *ODBC_SQLColumnPrivileges* command returns a list of columns and associated privileges for the specified table. The driver returns the information as a result set on the specified *stmtID*.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the catalog name.

schemaName is the schema name.

tableName is the table name.

columnName is the string search pattern for column names.

For more information, please see the SQLColumnPrivileges function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716336(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLColumns

ODBC_SQLColumns (stmtID ; catalogName ; schemaName ; tableName ; columnName) -> Function result

Parameter	Type		Description
stmtID	Longint	→	Statement ID
catalogName	String	←	Catalog name
schemaName	String	←	String search pattern for schema names
tableName	String	←	String search pattern for table names
columnName	String	←	String search pattern for column names
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLColumns

Description

The *ODBC_SQLColumns* command returns the list of column names in specified tables. The driver returns this information as a result set on the specified *stmtID*.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the catalog name.

schemaName is the string search pattern for schema names.

tableName is the string search pattern for table names.

columnName is the string search pattern for column names.

For more information, please see the SQLColumns function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711683(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLForeignKeys

ODBC_SQLForeignKeys (stmtID ; pkCatalogName ; pkSchemaName ; pkTableName ; fkCatalogName ; fkSchemaName ; fkTableName) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
pkCatalogName	String	⇐	Primary key table catalog name
pkSchemaName	String	⇐	Primary key table schema name
pkTableName	String	⇐	Primary key table name
fkCatalogName	String	⇐	Foreign key table catalog name
fkSchemaName	String	⇐	Foreign key table schema name
fkTableName	String	⇐	Foreign key table name
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLForeignKeys

Description

The *ODBC_SQLForeignKeys* command returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

pkCatalogName is the primary key catalog table name.

pkSchemaName is the primary key schema name.

pkTableName is the primary key table name.

fkCatalogName is the foreign key table catalog name.

fkSchemaName is the foreign key table schema name.

fkTableName is the foreign key table name.

For more information, please see the SQLForeignKeys function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709315(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLGetTypeInfo

ODBC_SQLGetTypeInfo (stmtID ; dataType) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
dataType	Longint	⇐	SQL data type
Function result	Longint	⇒	Returns the result of the MS ODBC API function SQLGetTypeInfo

Description

The *ODBC_SQLGetTypeInfo* command returns information about data types supported by the data source. The driver returns the information in the form of an SQL result set. The data types are intended for use in Data Definition Language (DDL) statements.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

dataType is the SQL data type, such as the constant SQL_ALL_TYPES, which is equal to 0.

For more information, please see the SQLGetTypeInfo function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714632(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLPrimaryKeys

ODBC_SQLPrimaryKeys (stmtID ; catalogName ; schemaName ; tableName) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
catalogName	String	⬅	Catalog name
schemaName	String	⬅	String search pattern for schema names
tableName	String	⬅	String search pattern for table names
Function result	Longint	➡	Returns the result of the MS ODBC API function SQLPrimaryKeys

Description

The *ODBC_SQLPrimaryKeys* command returns column names that make up the primary key for a table. The driver returns the information as a result set. This command does not support returning primary keys from multiple tables in a single call.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the catalog name.

schemaName is the string search pattern for schema names.

tableName is the string search pattern for table names.

For more information, please see the SQLPrimaryKeys function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711005(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLProcedureColumns

ODBC_SQLProcedureColumns (stmtID ; catalogName ; schemaName ; procName ; columnName) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
catalogName	String	⇒	Procedure catalog name
schemaName	String	⇒	String search pattern for procedure schema names
procName	String	⇒	String search pattern for procedure names
columnName	String	⇒	String search pattern for column names
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLProcedureColumns

Description

The *ODBC_SQLProcedureColumns* command returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. The driver returns the information as a result set on the specified statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the procedure catalog name.

schemaName is the string search pattern for procedure schema names.

procName is the string search pattern for procedure names.

columnName is the string search pattern for column names.

For more information, please see the SQLProcedureColumns function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711701(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLProcedures

ODBC_SQLProcedures (stmtID ; catalogName ; schemaName ; procName) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
catalogName	String	➡	Procedure catalog name
schemaName	String	➡	String search pattern for procedure schema names
procName	String	➡	String search pattern for procedure names
Function result	Longint	⮌	Returns the result of the MS ODBC API function SQLProcedures

Description

The *ODBC_SQLProcedures* command returns the list of procedure names stored in a specific data source. Procedure is a generic term used to describe an executable object, or a named entity that can be invoked using input and output parameters.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the procedure catalog name.

schemaName is the string search pattern for procedure schema names.

procName is the string search pattern for procedure names.

For more information, please see the SQLProcedures function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms715368(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLSpecialColumns

ODBC_SQLSpecialColumns (stmtID ; identifierType ; catalogName ; schemaName ; tableName ; scope ; nulllable) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
identifierType	Longint	⇒	Type of column to return
catalogName	String	⇒	Catalog name for the table
schemaName	String	⇒	Schema name for the table
tableName	String	⇒	Table name
scope	Longint	⇒	Minimum required space of the rowid
nulllable	Longint	⇒	Determines whether to return special columns that can have a NULL value
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSpecialColumns

Description

The *ODBC_SQLSpecialColumns* command retrieves the following information about columns within a specified table. Either the optimal set of columns that uniquely identifies a row in the table or the columns that are automatically updated when any value in the row is updated by a transaction.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

identifierType is the type of column to return.

catalogName is the catalog name for the table.

schemaName is the schema name for the table.

tableName is the table name.

scope is the minimum required space of the row and can have one of the following values:

Constant	Description
SQL_SCOPE_CURROW	The rowid is guaranteed to be valid only while positioned on that row. A later reselect using rowid may not return a row if the row was updated or deleted by another transaction.
SQL_SCOPE_TRANSACTION	The rowid is guaranteed to be valid for the duration of the current transaction.
SQL_SCOPE_SESSION	The rowid is guaranteed to be valid for the duration of the session (across transaction boundaries).

nullable determines whether to return special columns that can have a NULL value.

For more information, please see the SQLSpecialColumns function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714602(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLStatistics

ODBC_SQLStatistics (stmtID ; catalogName ; schemaName ; tableName ; unique ; reserved) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
catalogName	String	⇒	Catalog name
schemaName	String	⇒	Schema name
tableName	String	⇒	Table name
unique	Longint	⇒	Type of index
reserved	Longint	⇒	Indicates the importance of the CARDINALITY and PAGES columns
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLStatistics

Description

The *ODBC_SQLStatistics* command retrieves a list of statistics about a single table and the indexes associated with the table.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the catalog name.

schemaName is the schema name.

tableName is the table name.

unique indicates the type of index and can have one of the following values: SQL_INDEX_UNIQUE or SQL_INDEX_ALL.

reserved indicates the importance of the CARDINALITY and PAGES columns.

For more information, please see the SQLStatistics function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711022(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLTablePrivileges

ODBC_SQLTablePrivileges (stmtID ; catalogName ; schemaName ; tableName) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
catalogName	String	➡	Table catalog name
schemaName	String	➡	String search pattern for schema names
tableName	String	➡	String search pattern for table names
Function result	Longint	⮌	Returns the result of the MS ODBC API function SQLTablePrivileges

Description

The *ODBC_SQLTablePrivileges* command returns a list of tables and the privileges associated with each table.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the table catalog name.

schemaName is the string search pattern for schema names.

tableName is the string search pattern for table names.

For more information, please see the SQLTablePrivileges function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713565(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLTables

ODBC_SQLTables (stmtID ; catalogName ; schemaName ; tableName ; tableType) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
catalogName	String	⇒	Catalog name
schemaName	String	⇒	String search pattern for schema names
tableName	String	⇒	String search pattern for table names
tableType	String	⇒	List of table types to match
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLTables

Description

The *ODBC_SQLTables* command returns the list of table, catalog, or schema names, and table types, stored in a specific data source.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

catalogName is the catalog name.

schemaName is the string search pattern for schema names.

tableName is the string search pattern for table names.

tableType is the list of table types to match.

For more information, please see the SQLTables function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711831(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_Connection

- Connecting to a Data Source
- ODBC_SQLAllocConnect
- ODBC_SQLAllocStmt
- ODBC_SQLBrowseConnect
- ODBC_SQLConnect

Connecting to a Data Source

The commands in this chapter enable you to connect to an ODBC data source, by allowing you to do the following:

- Allocate a connection handle (*ODBC_SQLAllocConnect*)
- Allocate a statement handle to a valid connection (*ODBC_SQLAllocStmt*)
- Establish a connection to a specific driver (*ODBC_SQLConnect*) or by passing a browse request connection string (*ODBC_SQLBrowseConnect*)

ODBC_SQLAllocConnect

ODBC_SQLAllocConnect (connectionID) -> Function result

Parameter	Type		Description
connectionID	Longint	⇦	Connection ID
Function result	Longint	⇨	Returns the result of the MS ODBC API function SQLAllocHandle

Description

The *ODBC_SQLAllocConnect* command allocates a connection handle to the *connectionID* parameter, which is a Longint variable that you pass to it. After calling this command, you can establish a connection to a specific data source by calling the *ODBC_SQLConnect* command.

For more information, please see the SQLAllocHandle function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714852(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE, or SQL_ERROR.

Example

See the example for the *ODBC_SQLConnect* command.

ODBC_SQLAllocStmt

ODBC_SQLAllocStmt (connectionID ; stmtID) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
stmtID	Longint	⇐	Statement ID
Function result	Longint	⇒	Returns the result of the MS ODBC API function SQLAllocHandle

Description

The *ODBC_SQLAllocStmt* function allocates a statement handle to *connectionID*.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

stmtID is the statement ID returned if the connection is valid. It can then be passed to all other commands that require a *stmtID*, like *ODBC_SQLSetStmtAttr* and *ODBC_SQLExecute*.

For more information, please see the ODBC_SQLAllocStmt function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712649(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE, or SQL_ERROR.

Example

The following method connects you to a data source and then creates a statement handle ID:

```
$result:=ODBC_SQLAllocConnect($connectionID)
$result:=ODBC_SQLConnect($connectionID;"access";"Administrator";"admin1")
$result:=ODBC_SQLAllocStmt($connectionID;$statementID)
```

ODBC_SQLBrowseConnect

ODBC_SQLBrowseConnect (connectionID ; inConnectionStr ; outConnectionStr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
inConnectionStr	Text	⇒	Browse request connection string
outConnectionStr	Text	⇐	Browse result connection string
Function result	Longint	⇒	Returns the result of the MS ODBC API function SQLBrowseConnect

Description

The *ODBC_SQLBrowseConnect* command supports an iterative method of discovering and enumerating the attribute values required to connect to a data source. Each call to this command returns successive levels of attributes and attribute values. When all levels have been enumerated, a connection to the data source is completed and a complete connection string is returned.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect*.

inConnectionStr is the browse request connection string.

outConnectionStr is the browse result connection string.

For more information, please see the SQLBrowseConnect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714565(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLConnect

ODBC_SQLConnect (connectionID ; serverName ; userName ; password) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
serverName	String	⇒	Data source name
userName	String	⇒	User identifier
password	String	⇒	User password
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLConnect

Description

The *ODBC_SQLConnect* function establishes a connection to a specific driver by passing it the *serverName*, *userName*, and *password*. It internally uses the MS ODBC API function SQLConnect.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect*.

serverName is the name of the data source name.

userName is the user name or login name defined when setting up ODBC authorization.

password is the user password.

For more information, please see the SQLConnect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711810(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method connects you to a data source whose name, username, and password are passed to the *ODBC_SQLConnect* command:

```
$result:=ODBC_SQLAllocConnect($connectionID)
$result:=ODBC_SQLConnect($connectionID;"access";"Administrator";"admin1")
If($result=SQL_SUCCESS) `Connection was successful
  `... continue by calling other commands that require a valid $connectionID
End if
```

ODBC_Driver attributes

- Setting and Retrieving Driver Attributes
- ODBC_SQLGetConnectAttr
- ODBC_SQLGetEnvAttr
- ODBC_SQLGetStmtAttr
- ODBC_SQLSetConnectAttr
- ODBC_SQLSetEnvAttr
- ODBC_SQLSetStmtAttr

Setting and Retrieving Driver Attributes

The commands in this chapter enable you to set and retrieve connection, environment, and driver attributes. With them, you can do the following:

- Retrieve and set the current setting of a connection attribute (*ODBC_SQLGetConnectAttr* and *ODBC_SQLSetConnectAttr*)
- Retrieve and set the current setting of an environment attribute (*ODBC_SQLGetEnvAttr* and *ODBC_SQLSetEnvAttr*)
- Retrieve and set the current setting of a statement attribute (*ODBC_SQLGetStmtAttr* and *ODBC_SQLSetStmtAttr*).

ODBC_SQLGetConnectAttr

ODBC_SQLGetConnectAttr (connectionID ; attribute ; valuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	➡	Connection ID
attribute	Longint	➡	Attribute to retrieve
valuePtr	Pointer	➡	Pointer to the current value of the attribute
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLGetConnectAttr

Description

The *ODBC_SQLGetConnectAttr* command returns the current setting of a connection attribute passed in *attribute*. This command can be used in conjunction with **ODBC_SetConnectAttr.**

connectionID is a valid connection ID returned by either *ODBC_SQLAllocConnect* before or *ODBC_SQLConnect* after having established a connection to a data source depending on *attribute*.

attribute is the connection attribute to retrieve and can be one of the following values:

Constant	Description
SQL_ATTR_ACCESS_MODE	Read-only or read-write
SQL_ATTR_ASYNC_ENABLE	Defines if a function can be executed asynchronously
SQL_ATTR_AUTOCOMMIT	Autocommit or manual-commit mode
SQL_ATTR_CONNECTION_TIMEOUT	Number of seconds to wait for a request to complete
SQL_ATTR_CURRENT_CATALOG	Name of the catalog to be used by the data source
SQL_ATTR_LOGIN_TIMEOUT	Number of seconds to wait for a login request
SQL_ATTR_METADATA_ID	Determines how the string arguments of catalog functions are treated
SQL_ATTR_ODBC_CURSORS	Specifies how the Driver Manager uses the ODBC cursor
SQL_ATTR_PACKET_SIZE	Network packet size in bytes
SQL_ATTR_QUIET_MODE	Does not display any dialog boxes, except **ODBC_SQLDriverConnect**
SQL_ATTR_TRACE	Tracing on or off
SQL_ATTR_TRACEFILE	Name of the trace file
SQL_ATTR_TRANSLATE_LIB	Name of a library containing to perform tasks, such as character set translation
SQL_ATTR_TRANSLATE_OPTION	A value passed to the trnaslation DLL
SQL_ATTR_TXN_ISOLATION	Set the transaction isolation level for connection

valuePtr is a pointer to the current value of the attribute defined in the *attribute* parameter. The variable *valuePtr* points to must be a String variable.

For more information, please see the SQLGetConnectAttr function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710297(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method sets a connection attribute and then retrieves it:

```
C_STRING(255;vCatalogName;vNewCatalogName)
vCatalogName:="MyCatalogName"
$result:=ODBC_SQLSetConnectAttr($connectionID;SQL_ATTR_CURRENT_CATALOG;->vCatalogName)
$result:=ODBC_SQLGetConnectAttr($connectionID;SQL_ATTR_CURRENT_CATALOG;->vNewCatalogName)
```

ODBC_SQLGetEnvAttr

ODBC_SQLGetEnvAttr (attribute ; value) -> Function result

Parameter	Type		Description
attribute	Longint	⇒	Attribute to retrieve
value	Longint	⇐	Current value of the attribute
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLGetEnvAttr

Description

The *ODBC_SQLGetEnvAttr* command returns the current setting of an environment attribute.

attribute is the environment attribute to retrieve and can be one of the following values:

Constant	Description
SQL_ATTR_CONNECTION_POOLING	Enables or disables connection pooling at the environment level
SQL_ATTR_CP_MATCH	Determines how a connection is chosen from a connection pool
SQL_ATTR_ODBC_VERSION	Determines whether certain functionality exhibits ODBC 2.x behavior or ODBC 3.x behavior
SQL_ATTR_OUTPUT_NTS	Determines how the driver returns string data

value is the current value of *attribute*.

For more information, please see the SQLGetEnvAttr function in the MS ODBC API Reference to http://msdn.microsoft.com/en-us/library/ms709276(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method sets a environment attribute and then retrieves it:

```
$result:=ODBC_SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION;3)
$result:=ODBC_SQLGetEnvAttr(SQL_ATTR_ODBC_VERSION;vEnvAttribute)
```

ODBC_SQLGetStmtAttr

ODBC_SQLGetStmtAttr (stmtID ; attribute ; valuePtr) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
attribute	Longint	⇒	Attribute to retrieve
valuePtr	Pointer	⇒	Pointer to the current value of the attribute
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLGetStmtAttr

Description

The *ODBC_SQLGetStmtAttr* command returns the current setting of a statement attribute.
stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

attribute is the statement attribute to retrieve and can be one of the following values:

Constant	Description
SQL_ATTR_APP_PARAM_DESC	
SQL_ATTR_APP_ROW_DESC	
SQL_ATTR_ASYNC_ENABLE	Specifies whether a function called with the specified statement is executed asynchronously
SQL_ATTR_CONCURRENCY	Specifies the cursor concurrency
SQL_ATTR_CURSOR_SCROLLABLE	Scrollable cursors are either required or not required for the specified statement
SQL_ATTR_CURSOR_SENSITIVITY	Specifies whether cursors for the specified statement handle make visible the changes made to a result set by another cursor
SQL_ATTR_CURSOR_TYPE	Specifies cursor type, like scrolls forward, static, saves and uses the keys for the number of specified rows specified or only those in the rowset
SQL_ATTR_ENABLE_AUTO_IPD	SQL_TRUE = Turns on automatic population of the IPD after a call to *ODBC_SQLPrepare*. SQL_FALSE = Turns off automatic population of the IPD after a call to *ODBC_SQLPrepare*.
SQL_ATTR_FETCH_BOOKMARK_PTR	Bookmark value
SQL_ATTR_IMP_PARAM_DESC	The value of this attribute is the descriptor allocated when the statement was initially allocated.
SQL_ATTR_IMP_ROW_DESC	The value of this attribute is the descriptor allocated when the statement was initially allocated.
SQL_ATTR_KEYSET_SIZE	Number of rows in the keyset for a keyset-driven cursor
SQL_ATTR_MAX_LENGTH	Maximum amount of data that the driver returns from a character or binary column.
SQL_ATTR_MAX_ROWS	Maximum number of rows to return to the application for a SELECT statement.
SQL_ATTR_METADATA_ID	Determines how the string arguments of catalog functions are treated.
SQL_ATTR_NOSCAN	Indicates whether the driver should scan SQL strings for escape sequences.
SQL_ATTR_PARAM_BIND_OFFSET_PTR	Bind offset.
SQL_ATTR_PARAM_BIND_TYPE	Indicates the binding orientation to be used for dynamic parameters.
SQL_ATTR_PARAM_OPERATION_PTR	Indicates if a parameter is to be ignored during execution of an SQL statement.
SQL_ATTR_PARAM_STATUS_PTR	Status information for each row of parameter values.
SQL_ATTR_PARAMS_PROCESSED_PTR	Number of sets of parameters that have been processed, including error sets.
SQL_ATTR_PARAMSET_SIZE	Specifies the number of values for each parameter.
SQL_ATTR_QUERY_TIMEOUT	Number of seconds to wait for an SQL statement to execute.
SQL_ATTR_RETRIEVE_DATA	Either retrieve or do not retrieve data after it positions the cursor to the specified location.
SQL_ATTR_ROW_ARRAY_SIZE	Number of rows returned by each call to **ODBC_SQLFetch** or *ODBC_SQLFetchScroll*
SQL_ATTR_ROW_BIND_OFFSET_PTR	An offset to change binding of column data.
SQL_ATTR_ROW_BIND_TYPE	The binding orientation to be used when **ODBC_SQLFetch** or *ODBC_SQLFetchScroll* is called on the specified statement.
SQL_ATTR_ROW_NUMBER	Number of the current row in the entire result set.
SQL_ATTR_ROW_OPERATION_PTR	Values used to ignore a row during a bulk operation using *ODBC_SQLSetPos*.
SQL_ATTR_ROW_STATUS_PTR	Row status values after a call to *ODBC_SQLFetch* or *ODBC_SQLFetchScroll*.
SQL_ATTR_ROWS_FETCHED_PTR	Number of rows fetched after a call to **ODBC_SQLFetch** or *ODBC_SQLFetchScroll*.

SQL_ATTR_SIMULATE_CURSOR	Specifies whether drivers that simulate positioned update and delete statements guarantee that such statements affect only one single row.
SQL_ATTR_USE_BOOKMARKS	Specifies whether an application will use bookmarks with a cursor.

valuePtr is a pointer to a variable that will contain the current value of *attribute*.

For more information, please see the SQLGetStmtAttr function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms715438(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method sets a statement attribute and then retrieves it:

```
vAttributeVal:=SQL_CURSOR_KEYSET_DRIVEN
$result:=ODBC_SQLSetStmtAttr($newStmt;SQL_ATTR_CURSOR_TYPE;->vAttributeVal)
$result:=ODBC_SQLGetStmtAttr($newStmt;SQL_ATTR_CURSOR_TYPE;->vNewAttributeValue)
```

ODBC_SQLSetConnectAttr

ODBC_SQLSetConnectAttr (connectionID ; attribute ; valuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
attribute	Longint	⇒	Attribute to set
valuePtr	Pointer	⇒	Pointer to the value to set the attribute
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetConnectAttr

Description

The *ODBC_SQLSetConnectAttr* command sets the attributes that govern aspects of connections.

connectionID is a valid connection ID returned by either *ODBC_SQLAllocConnect* before or *ODBC_SQLConnect* after having established a connection to a data source depending on *attribute*.

attribute is a connection attribute to set and can be one of the following values:

Constant	Description
SQL_ATTR_ACCESS_MODE*	Read-only or read-write
SQL_ATTR_ASYNC_ENABLE	Defines if a function can be executed asynchronously
SQL_ATTR_AUTOCOMMIT*	Autocommit or manual-commit mode
SQL_ATTR_CONNECTION_TIMEOUT	Number of seconds to wait for a request to complete
SQL_ATTR_CURRENT_CATALOG	Name of the catalog to be used by the data source
SQL_ATTR_LOGIN_TIMEOUT*	Number of seconds to wait for a login request
SQL_ATTR_METADATA_ID	Determines how the string arguments of catalog functions are treated
SQL_ATTR_ODBC_CURSORS*	Specifies how the Driver Manager uses the ODBC cursor
SQL_ATTR_PACKET_SIZE	Network packet size in bytes
SQL_ATTR_QUIET_MODE	Does not display any dialog boxes, except ODBC_SQLDriverConnect
SQL_ATTR_TRACE*	Tracing on or off
SQL_ATTR_TRACEFILE*	Name of the trace file
SQL_ATTR_TRANSLATE_LIB	Name of a library containing to perform tasks, such as character set translation
SQL_ATTR_TRANSLATE_OPTION**	A value passed to the trnaslation DLL
SQL_ATTR_TXN_ISOLATION	Set the transaction isolation level for connection

Notes:
* These attributes must be set before a connection is established.
** This attribute must be set after connecting.

valuePtr is a pointer to a variable containing the value at which to set the *attribute* parameter.

For more information, please see the SQLSetConnectAttr function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713605(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLGetConnectAttr* command.

ODBC_SQLSetEnvAttr

ODBC_SQLSetEnvAttr (attribute ; value) -> Function result

Parameter	Type		Description
attribute	Longint	⇒	Attribute to set
value	Longint	⇒	Value to set the attribute
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetEnvAttr

Description

The *ODBC_SQLSetEnvAttr* command sets an attribute that governs the different aspects of environments.

attribute is the environment attribute to set and can be one of the following values:

Constant	Description
SQL_ATTR_CONNECTION_POOLING	Enables or disables connection pooling at the environment level
SQL_ATTR_CP_MATCH	Determines how a connection is chosen from a connection pool
SQL_ATTR_ODBC_VERSION	Determines whether certain functionality exhibits ODBC 2.x behavior or ODBC 3.x behavior
SQL_ATTR_OUTPUT_NTS	Determines how the driver returns string data

value is a Longint value at which to set *attribute*.

For more information, please see the SQLSetEnvAttr function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709285(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLGetEnvAttr* command.

ODBC_SQLSetStmtAttr

ODBC_SQLSetStmtAttr (stmtID ; attribute ; valuePtr) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
attribute	Longint	⇒	Attribute to set
valuePtr	Pointer	⇒	Pointer to the value to set the attribute
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetStmtAttr

Description

The *ODBC_SQLSetStmtAttr* command sets *attribute* related to a statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

attribute is a statement attribute to set and can be one of the following values:

Constant	Description
SQL_ATTR_APP_PARAM_DESC	
SQL_ATTR_APP_ROW_DESC	
SQL_ATTR_ASYNC_ENABLE	Specifies whether a function called with the specified statement is executed asynchronously
SQL_ATTR_CONCURRENCY	Specifies the cursor concurrency
SQL_ATTR_CURSOR_SCROLLABLE	Scrollable cursors are either required or not required for the specified statement
SQL_ATTR_CURSOR_SENSITIVITY	Specifies whether cursors for the specified statement handle make visible the changes made to a result set by another cursor
SQL_ATTR_CURSOR_TYPE	Specifies cursor type, like scrolls forward, static, saves and uses the keys for the number of specified rows specified or only those in the row set
SQL_ATTR_ENABLE_AUTO_IPD	SQL_TRUE = Turns on automatic population of the IPD after a call to *ODBC_SQLPrepare*. SQL_FALSE = Turns off automatic population of the IPD after a call to *ODBC_SQLPrepare*.
SQL_ATTR_FETCH_BOOKMARK_PTR	Bookmark value
SQL_ATTR_IMP_PARAM_DESC	The value of this attribute is the descriptor allocated when the statement was initially allocated.
SQL_ATTR_IMP_ROW_DESC	The value of this attribute is the descriptor allocated when the statement was initially allocated.
SQL_ATTR_KEYSET_SIZE	Number of rows in the keyset for a keyset-driven cursor
SQL_ATTR_MAX_LENGTH	Maximum amount of data that the driver returns from a character or binary column.
SQL_ATTR_MAX_ROWS	Maximum number of rows to return to the application for a SELECT statement.
SQL_ATTR_METADATA_ID	Determines how the string arguments of catalog functions are treated.
SQL_ATTR_NOSCAN	Indicates whether the driver should scan SQL strings for escape sequences.
SQL_ATTR_PARAM_BIND_OFFSET_PTR	Bind offset.
SQL_ATTR_PARAM_BIND_TYPE	Indicates the binding orientation to be used for dynamic parameters.
SQL_ATTR_PARAM_OPERATION_PTR	Indicates if a parameter is to be ignored during execution of an SQL statement.
SQL_ATTR_PARAM_STATUS_PTR	Status information for each row of parameter values.
SQL_ATTR_PARAMS_PROCESSED_PTR	Number of sets of parameters that have been processed, including error sets.
SQL_ATTR_PARAMSET_SIZE	Specifies the number of values for each parameter.
SQL_ATTR_QUERY_TIMEOUT	Number of seconds to wait for an SQL statement to execute.
SQL_ATTR_RETRIEVE_DATA	Either retrieve or do not retrieve data after it positions the cursor to the specified location.
SQL_ATTR_ROW_ARRAY_SIZE	Number of rows returned by each call to **ODBC_SQLFetch** or *ODBC_SQLFetchScroll*
SQL_ATTR_ROW_BIND_OFFSET_PTR	An offset to change binding of column data.
SQL_ATTR_ROW_BIND_TYPE	The binding orientation to be used when **ODBC_SQLFetch** or *ODBC_SQLFetchScroll* is called on the specified statement.
SQL_ATTR_ROW_NUMBER	Number of the current row in the entire result set.
SQL_ATTR_ROW_OPERATION_PTR	Values used to ignore a row during a bulk operation using *ODBC_SQLSetPos*.
SQL_ATTR_ROW_STATUS_PTR	Row status values after a call to *ODBC_SQLFetch*or *ODBC_SQLFetchScroll*.
SQL_ATTR_ROWS_FETCHED_PTR	Number of rows fetched after a call to **ODBC_SQLFetch** or *ODBC_SQLFetchScroll*.

SQL_ATTR_SIMULATE_CURSOR	Specifies whether drivers that simulate positioned update and delete statements guarantee that such statements affect only one single row.
SQL_ATTR_USE_BOOKMARKS	Specifies whether an application will use bookmarks with a cursor.

valuePtr is a pointer to a variable containing the value at which to set the *attribute* parameter.

For more information, please see the SQLSetStmtAttr function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712631(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLBulkOperations* command.

ODBC_End connection

- Terminating a Connection, Introduction
- ODBC_SQLDisconnect
- ODBC_SQLEndTran
- ODBC_SQLFreeConnect

Terminating a Connection, Introduction

The commands in this chapter enable you to terminate a connection, so you can:

- Close the connection (*ODBC_SQLDisconnect*)
- Request a commit or rollback operation for all active operations on all statements associated with a connection (*ODBC_SQLEndTran*)
- Free up resources associated with a connection handle (*ODBC_SQLFreeConnect*)

ODBC_SQLDisconnect

ODBC_SQLDisconnect (connectionID) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLDisconnect

Description

The *ODBC_SQLDisconnect* command closes the connection specified by *connectID*.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

For more information, please see the SQLDisconnect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713946(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLEndTran

ODBC_SQLEndTran (handleType ; handleID ; completionType) -> Function result

Parameter	Type		Description
handleType	Longint	⇒	Type of ID to pass to handleID
handleID	Longint	⇒	Either the statement ID or the connection ID
completionType	Longint	⇒	Either SQL_COMMIT or SQL_ROLLBACK
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLEndTran

Description

The *ODBC_SQLEndTran* command requests a commit or rollback operation for all active operations on all statements associated with a connection. *ODBC_SQLEndTran* can also request that a commit or rollback operation be performed for all connections associated with an environment.

If no transactions are active, *ODBC_SQLEndTran* returns SQL_SUCCESS with no effect on any data sources.

handleType defines which type of ID to pass to *handleID* and can have one of the following two values:

Constant	Description
SQL_HANDLE_STMT	Statement ID
SQL_HANDLE_DBC	Connection ID

handleID is the *connectionID* if *handleType* is equal to SQL_HANDLE_DBC. *connectionID* is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

Otherwise, *handleID* is the *stmtID*, which is a valid statement ID returned by *ODBC_SQLAllocStmt*.

completionType can have one of the following two values:

Constant	Description
SQL_COMMIT	Commit changes
SQL_ROLLBACK	Rollback changes

For more information, please see the SQLEndTran function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716544(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLFreeConnect

ODBC_SQLFreeConnect (connectionID) -> Function result

Parameter	Type		Description
connectionID	Longint	→	Connection ID
Function result	Longint	←	Returns the result of the MS ODBC API function SQLFreeConnect

Description

The *ODBC_SQLFreeConnect* command frees resources associated with a connection handle.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect*.

For more information, please see the SQLFreeConnect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709370(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_End statement

- Terminating a Statement, Introduction
- ODBC_SQLCancel
- ODBC_SQLCloseCursor
- ODBC_SQLFreeStmt

Terminating a Statement, Introduction

The commands in this chapter enable you to terminate a statement, by allowing you to do the following:

- Get a list of columns and associated privileges for the specified table (*ODBC_SQLCancel*)
- Close a cursor that has been opened on a statement and discards pending results (*ODBC_SQLCloseCursor*)
- Stop the processing associated with a specific statement, closes any open cursors associated with the statement, discards pending results, or, optionally, frees all resources associated with the statement handle (*ODBC_SQLFreeStmt*)

ODBC_SQLCancel

ODBC_SQLCancel (stmtID) -> Function result

Parameter	Type		Description
stmtID	Longint	→	Statement ID
Function result	Longint	←	Returns the result of the MS ODBC API function SQLCancel

Description

The *ODBC_SQLCancel* command cancels the processing on a statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

For more information, please see the SQLCancel function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714112(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLCloseCursor

ODBC_SQLCloseCursor (stmtID) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLCloseCursor

Description

The *ODBC_SQLCloseCursor* command closes a cursor that has been opened on a statement and discards pending results.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

For more information, please see the SQLCloseCursor function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709301(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLFreeStmt

ODBC_SQLFreeStmt (stmtID ; option) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
option	Longint	⇒	Option to execute
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLFreeStmt

Description

The *ODBC_SQLFreeStmt* command stops the processing associated with a specific statement, closes any open cursors associated with the statement, discards pending results, or, optionally, frees all resources associated with the statement handle.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

The *option* parameter can take one of the following values:

Constants	Description
SQL_CLOSE	Closes the cursor associated with StatementHandle (if one was defined) and discards all pending results
SQL_UNBIND	Sets the SQL_DESC_COUNT field of the ARD to 0, releasing all column buffers bound by SQLBindCol for the given stmtID
SQL_RESET_PARAMS	Sets the SQL_DESC_COUNT field of the APD to 0, releasing all parameter buffers set by SQLBindParameter for the given stmtID
SQL_DROP	Frees the statement handle

For more information, please see the SQLFreeStmt function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709284(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLFetch* command.

ODBC_Error handling

Error Handling, Introduction
ODBC_SetErrorHandler

Error Handling, Introduction

The command in this chapter enables you to install an error method that will handle the errors (*ODBC_SetErrorHandler*).

ODBC_SetErrorHandler

ODBC_SetErrorHandler (errorMethod)

Parameter	Type		Description
errorMethod	String	➡	Error method to be invoked, or Empty string to stop trapping errors

Description

The *ODBC_SetErrorHandler* command installs an error-handling method that will handle the errors for all processes and 4D ODBC PRO will no longer display the alert informing you of invalid connection and/or statement IDs.

The called method will receive 2 parameters in $1 and $2:

- $1 gets the error number,
- $2 gets the error text.

Note: Due to database compilation, $1 and $2 must be declared respectively as longint (**C_LONGINT**) and text (**C_TEXT**) in the error method.

The method installed by this command can use the *ODBC_SQLGetDiagRec* routine to get more information on the error.

To uninstall the error handling method, pass an empty string to *ODBC_SetErrorHandler*.

ODBC_Information

- Obtaining Information about a Driver and Data Source
- ODBC_SQLDataSources
- ODBC_SQLDriverConnect
- ODBC_SQLDrivers
- ODBC_SQLGetFunctions
- ODBC_SQLGetInfo

Obtaining Information about a Driver and Data Source

The commands in this chapter enable you to obtain information about a driver and data source, such as:

- Retrieve information about the data source defined (*ODBC_SQLDataSources*)
- Obtain a list of driver descriptions and driver attribute keywords (*ODBC_SQLDrivers*)
- Find out if a specific ODBC function is supported by the driver (*ODBC_SQLGetFunctions*)
- Obtain general information about the driver and data source associated with a connection (*ODBC_SQLGetInfo*)

ODBC_SQLDataSources

ODBC_SQLDataSources (direction ; serverName ; description) -> Function result

Parameter	Type		Description
direction	Longint	⇒	Which data source the Driver Manager returns information for
serverName	Text	⇐	Data source name
description	Text	⇐	Description of the driver associated with the data source
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLDataSources

Description

The *ODBC_SQLDataSources* command returns information about the User and System data sources defined in the Driver Manager.

The *direction* parameter defines how to fetch the data sources and can be one of the following values:

Constant	Description
SQL_FETCH_NEXT	Fetch the next data source name in the list
SQL_FETCH_FIRST	Fetch from the beginning of the list
SQL_FETCH_FIRST_USER	Fetch the first User data sources
SQL_FETCH_FIRST_SYSTEM	Fetch the first System data sources

serverName is the name of the data source, such as "MS Access Database."

description is the description of the driver associated with the data source, like "Microsoft Access Driver (*.mdb)."

For more information, please see the SQLDataSources function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711004(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method retrieves all the data sources and their descriptions and puts them into two arrays:

```
ARRAY TEXT(arServer;0)
ARRAY TEXT(arDescription;0)
$result:=ODBC_SQLDataSources(SQL_FETCH_FIRST;vServer;vDescription)
If($result=SQL_SUCCESS) `If it's successful, there might be other data sources
   Repeat
      APPEND TO ARRAY(arServer;vServer) `add server name to end of the array
      APPEND TO ARRAY(arDescription;vDescription) `add description to end of the array
      $result:=ODBC_SQLDataSources(SQL_FETCH_NEXT;vServer;vDescription)
   Until($result=SQL_NO_DATA) `loop until no data is retrieved by the command
End if
```

ODBC_SQLDriverConnect

ODBC_SQLDriverConnect -> Function result

Parameter	Type	Description
Function result	Longint	Returns the result of the MS ODBC API function SQLDriverConnect

Description

The *ODBC_SQLDriverConnect* command is not yet documented.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.
For more information, please see the SQLDriverConnect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms715433(VS.85).aspx.

ODBC_SQLDrivers

ODBC_SQLDrivers (direction ; driverDescription ; driverAttributes) -> Function result

Parameter	Type		Description
direction	Longint	➡	Direction from which to retrieve the drivers
driverDescription	Text	⬅	Description of the driver
driverAttributes	Text	⬅	List of driver attribute value pairs
Function result	Longint	➲	Returns the result of the MS ODBC API function SQLDrivers

Description

The *ODBC_SQLDrivers* command lists driver descriptions and driver attribute keywords.

The *direction* parameter determines which driver in the Driver Manager list to retrieve and can be one of the following values:

Constant	Description
SQL_FETCH_NEXT	Fetch the next driver description in the list
SQL_FETCH_FIRST	Fetch from the beginning of the driver description list

driverDescription is the description of the driver, such as "SQL Server", "Microsoft Access Driver (*.mdb)", and "Microsoft ODBC for Oracle".

driverAttributes returns the list of driver attribute pairs, such as "UsageCount", "SQLLevel", "FileUsage", "DirverODBCVer", "ConnectFunctions", "APILevel", "CPTimeout", and "FileExtns" along with their values, such as "UsageCount=2" and each pair is delimited by a **Char**(0).

For more information, please see the SQLDrivers function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712400(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method puts the driver description in one array and the driver's attribute value pairs into two-dimensional arrays:

```
ARRAY TEXT(arDriverDesc;0)
ARRAY TEXT(arAttrName;0;0) `Two-dimensional arrays to store the attribute pairs
ARRAY TEXT(arAttrValue;0;0)
$result:=ODBC_SQLDrivers(SQL_FETCH_FIRST;vDriverDesc;vDriverAttrPair)
If($result=SQL_SUCCESS)
   Repeat
      APPEND TO ARRAY(arDriverDesc;vDriverDesc)
      $size:=Size of array(arDriverDesc)
      INSERT ELEMENT(arAttrName;$size)
      INSERT ELEMENT(arAttrValue;$size)
      ParseDriverAttributePairs(vDriverAttrPair;->arAttrName{$size};->arAttrValue{$size})
      $result:=ODBC_SQLDrivers(SQL_FETCH_NEXT;vDriverDesc;vDriverAttrPair)
   Until($result=SQL_NO_DATA)
End if
```

Here is the code for the ParseDriverAttributePairs method:

```
` Method: ParseDriverAttributePairs
` $1 : Text : Input text to parse
` $2 : Pointer : A text array to hold the names
` $3 : Pointer : A text array to hold the values
```

```
C_TEXT($1;$input_t;$valuePair_t)
C_POINTER($2;$3;$names_aptr;$values_aptr)
C_LONGINT($position_i)
$input_t:=$1
$names_aptr:=$2
$values_aptr:=$3
ARRAY TEXT($names_aptr->;0)
ARRAY TEXT($values_aptr->;0)
Repeat
   $position_i:=Position(Char(0);$input_t)
   If($position_i>0)
      $valuePair_t:=Substring($input_t;1;$position_i)
      $input_t:=Substring($input_t;$position_i+1)
      $position_i:=Position("=";$valuePair_t)
      If($position_i>0)
         APPEND TO ARRAY($names_aptr->;Substring($valuePair_t;1;$position_i-1))
         APPEND TO ARRAY($values_aptr->;Substring($valuePair_t;$position_i+1))
      End if
   End if
Until($position_i=0)
```

ODBC_SQLGetFunctions

ODBC_SQLGetFunctions (connectionID ; functionIdentifier ; infoValuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
functionIdentifier	Longint	⇒	Function identifier
infoValuePtr	Pointer	⇐	Indicates if a function is supported or not by the driver
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLGetFunctions

Description

The *ODBC_SQLGetFunctions* command allows you to determine which specific ODBC functions a driver supports.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

functionIdentifier is the identifier of the function to test to see if it is supported by the driver and can be one of the values below:

Constant	Value
SQL_API_SQLALLOCCONNECT	1
SQL_API_SQLALLOCENV	2
SQL_API_SQLALLOCHANDLE	1001
SQL_API_SQLALLOCSTMT	3
SQL_API_SQLBINDCOL	4
SQL_API_SQLBINDPARAM	1002
SQL_API_SQLCANCEL	5
SQL_API_SQLCLOSECURSOR	1003
SQL_API_SQLCOLATTRIBUTE	6
SQL_API_SQLCOLUMNS	40
SQL_API_SQLCONNECT	7
SQL_API_SQLCOPYDESC	1004
SQL_API_SQLDATASOURCES	57
SQL_API_SQLDESCRIBECOL	8
SQL_API_SQLDISCONNECT	9
SQL_API_SQLENDTRAN	1005
SQL_API_SQLERROR	10
SQL_API_SQLEXECDIRECT	11
SQL_API_SQLEXECUTE	12
SQL_API_SQLFETCH	13
SQL_API_SQLFETCHSCROLL	1021
SQL_API_SQLFREECONNECT	14
SQL_API_SQLFREEENV	15
SQL_API_SQLFREEHANDLE	1006
SQL_API_SQLFREESTMT	16
SQL_API_SQLGETCONNECTATTR	1007
SQL_API_SQLGETCONNECTOPTION	42
SQL_API_SQLGETCURSORNAME	17
SQL_API_SQLGETDATA	43
SQL_API_SQLGETDESCFIELD	1008
SQL_API_SQLGETDESCREC	1009
SQL_API_SQLGETDIAGFIELD	1010
SQL_API_SQLGETDIAGREC	1011
SQL_API_SQLGETENVATTR	1012
SQL_API_SQLGETFUNCTIONS	44
SQL_API_SQLGETINFO	45
SQL_API_SQLGETSTMTATTR	1014
SQL_API_SQLGETSTMTOPTION	46
SQL_API_SQLGETTYPEINFO	47
SQL_API_SQLNUMRESULTCOLS	18
SQL_API_SQLPARAMDATA	48
SQL_API_SQLPREPARE	19
SQL_API_SQLPUTDATA	49
SQL_API_SQLROWCOUNT	20
SQL_API_SQLSETCONNECTATTR	1016
SQL_API_SQLSETCONNECTOPTION	50
SQL_API_SQLSETCURSORNAME	21
SQL_API_SQLSETDESCFIELD	1017

SQL_API_SQLSETDESCREC	1018
SQL_API_SQLSETENVATTR	1019
SQL_API_SQLSETPARAM	22
SQL_API_SQLSETSTMTATTR	1020
SQL_API_SQLSETSTMTOPTION	51
SQL_API_SQLSPECIALCOLUMNS	52
SQL_API_SQLSTATISTICS	53
SQL_API_SQLTABLES	54
SQL_API_SQLTRANSACT	23

infoValuePtr is a pointer to a Longint variable that will be equal to either SQL_TRUE if the specified function is supported by the driver or SQL_FALSE if it is not supported.

ODBC_SQLGetFunctions returns SQL_SUCCESS if the function executes correctly.

For more information, please see the SQLGetFunctions function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms709291(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLGetInfo

ODBC_SQLGetInfo (connectionID ; infoType ; infoValuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
infoType	Longint	⇒	Type of information
infoValuePtr	Pointer	⇐	Information regarding the driver and data source
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLGetInfo

Description

The *ODBC_SQLGetInfo* command returns general information about the driver and data source associated with a connection.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

If the *infoType* defines the type of information regarding a driver and data source and can be one of the following values:

Constant	Value
SQL_ACTIVE_CONNECTIONS	0
SQL_ACTIVE_ENVIRONMENTS	116
SQL_ACTIVE_STATEMENTS	1
SQL_AGGREGATE_FUNCTIONS	169
SQL_ALTER_DOMAIN	117
SQL_ASYNC_MODE	10021
SQL_BATCH_ROW_COUNT	120
SQL_BATCH_SUPPORT	121
SQL_BOOKMARK_PERSISTENCE	82
SQL_CATALOG_LOCATION	114
SQL_CATALOG_NAME_SEPARATOR	41
SQL_CATALOG_TERM	42
SQL_CATALOG_USAGE	92
SQL_COLUMN_ALIAS	87
SQL_CONCAT_NULL_BEHAVIOR	22
SQL_CONVERT_BIGINT	53
SQL_CONVERT_BINARY	54
SQL_CONVERT_BIT	55
SQL_CONVERT_CHAR	56
SQL_CONVERT_DATE	57
SQL_CONVERT_DECIMAL	58
SQL_CONVERT_DOUBLE	59
SQL_CONVERT_FLOAT	60
SQL_CONVERT_FUNCTIONS	48
SQL_CONVERT_GUID	173
SQL_CONVERT_INTEGER	61
SQL_CONVERT_INTERVAL_DAY_TIME	123
SQL_CONVERT_INTERVAL_YEAR_MONTH	124
SQL_CONVERT_LONGVARBINARY	71
SQL_CONVERT_LONGVARCHAR	62
SQL_CONVERT_NUMERIC	63
SQL_CONVERT_REAL	64
SQL_CONVERT_SMALLINT	65
SQL_CONVERT_TIME	66
SQL_CONVERT_TIMESTAMP	67
SQL_CONVERT_TINYINT	68
SQL_CONVERT_VARBINARY	69
SQL_CONVERT_VARCHAR	70
SQL_CONVERT_WCHAR	122
SQL_CONVERT_WLONGVARCHAR	125
SQL_CONVERT_WVARCHAR	126
SQL_CORRELATION_NAME	74
SQL_CREATE_ASSERTION	127
SQL_CREATE_CHARACTER_SET	128
SQL_CREATE_COLLATION	129
SQL_CREATE_DOMAIN	130
SQL_CREATE_SCHEMA	131
SQL_CREATE_TABLE	132
SQL_CREATE_TRANSLATION	133

SQL_CREATE_VIEW	134
SQL_CURSOR_ROLLBACK_BEHAVIOR	24
SQL_DATETIME_LITERALS	119
SQL_DDL_INDEX	170
SQL_DM_VER	171
SQL_DRIVER_HDBC	3
SQL_DRIVER_HDESC	135
SQL_DRIVER_HENV	4
SQL_DRIVER_HLIB	76
SQL_DRIVER_HSTMT	5
SQL_DRIVER_NAME	6
SQL_DRIVER_ODBC_VER	77
SQL_DRIVER_VER	7
SQL_DROP_ASSERTION	136
SQL_DROP_CHARACTER_SET	137
SQL_DROP_COLLATION	138
SQL_DROP_DOMAIN	139
SQL_DROP_SCHEMA	140
SQL_DROP_TABLE	141
SQL_DROP_TRANSLATION	142
SQL_DROP_VIEW	143
SQL_DYNAMIC_CURSOR_ATTRIBUTES1	144
SQL_DYNAMIC_CURSOR_ATTRIBUTES2	145
SQL_EXPRESSIONS_IN_ORDERBY	27
SQL_FILE_USAGE	84
SQL_FORWARD_ONLY_CURSOR_ATTRS1	146
SQL_FORWARD_ONLY_CURSOR_ATTRS2	147
SQL_GROUP_BY	88
SQL_INDEX_KEYWORDS	148
SQL_INFO_SCHEMA_VIEWS	149
SQL_INSERT_STATEMENT	172
SQL_KEYSET_CURSOR_ATTRIBUTES1	150
SQL_KEYSET_CURSOR_ATTRIBUTES2	151
SQL_KEYWORDS	89
SQL_LIKE_ESCAPE_CLAUSE	113
SQL_LOCK_TYPES	78
SQL_MAX_ASYNC_CONCURRENT_STMTS	10022
SQL_MAX_BINARY_LITERAL_LEN	112
SQL_MAX_CHAR_LITERAL_LEN	108
SQL_MAX_OWNER_NAME_LEN	32
SQL_MAX_PROCEDURE_NAME_LEN	33
SQL_MAX_QUALIFIER_NAME_LEN	34
SQL_MAX_ROW_SIZE_INCLUDES_LONG	103
SQL_MULT_RESULT_SETS	36
SQL_MULTIPLE_ACTIVE_TXN	37
SQL_NEED_LONG_DATA_LEN	111
SQL_NON_NULLABLE_COLUMNS	75
SQL_NUMERIC_FUNCTIONS	49
SQL_ODBC_API_CONFORMANCE	9

SQL_ODBC_INTERFACE_CONFORMANCE	152
SQL_ODBC_SAG_CLI_CONFORMANCE	12
SQL_ODBC_SQL_CONFORMANCE	15
SQL_ODBC_SQL_OPT_IEF	73
SQL_ODBC_VER	10
SQL_OJ_CAPABILITIES	65003
SQL_OUTER_JOINS	38
SQL_OWNER_TERM	39
SQL_OWNER_USAGE	91
SQL_PARAM_ARRAY_ROW_COUNTS	153
SQL_PARAM_ARRAY_SELECTS	154
SQL_POS_OPERATIONS	79
SQL_POSITIONED_STATEMENTS	80
SQL_PROCEDURE_TERM	40
SQL_PROCEDURES	21
SQL_QUALIFIER_LOCATION	114
SQL_QUALIFIER_NAME_SEPARATOR	41
SQL_QUALIFIER_TERM	42
SQL_QUALIFIER_USAGE	92
SQL_QUOTED_IDENTIFIER_CASE	93
SQL_ROW_UPDATES	11
SQL_SCHEMA_TERM	39
SQL_SCHEMA_USAGE	91
SQL_SCROLL_OPTIONS	44
SQL_SQL_CONFORMANCE	118
SQL_SQL92_DATETIME_FUNCTIONS	155
SQL_SQL92_FOREIGN_KEY_DELETE_RULE	156
SQL_SQL92_FOREIGN_KEY_UPDATE_RULE	157
SQL_SQL92_GRANT	158
SQL_SQL92_NUMERIC_VALUE_FUNCTIONS	159
SQL_SQL92_PREDICATES	160
SQL_SQL92_RELATIONAL_JOIN_OPERATORS	161
SQL_SQL92_REVOKE	162
SQL_SQL92_ROW_VALUE_CONSTRUCTOR	163
SQL_SQL92_STRING_FUNCTIONS	164
SQL_SQL92_VALUE_EXPRESSIONS	165
SQL_STANDARD_CLI_CONFORMANCE	166
SQL_STATIC_CURSOR_ATTRIBUTES1	167
SQL_STATIC_CURSOR_ATTRIBUTES2	168
SQL_STATIC_SENSITIVITY	83
SQL_STRING_FUNCTIONS	50
SQL_SUBQUERIES	95
SQL_SYSTEM_FUNCTIONS	51
SQL_TABLE_TERM	45
SQL_TIMEDATE_ADD_INTERVALS	109
SQL_TIMEDATE_DIFF_INTERVALS	110
SQL_TIMEDATE_FUNCTIONS	52
SQL_UNION	96
SQL_UNION_STATEMENT	96

The *infoValuePtr* argument retrieves the information regarding the driver and data source defined by *infoType*. The value returned depends on the type of information passed to *infoType*.

For more information, please see the SQLGetInfo function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711681(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method connects you to a data source and then retrieves information about the driver to find out the level of asynchronous support in the driver:

```
$result:=ODBC_SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION;3)
$result:=ODBC_SQLAllocConnect($connectionID)
$result:=ODBC_SQLConnect($connectionID;"oracle4d";"Admin";"admin1")
$result:=ODBC_SQLGetInfo($connectionID;10021;->asyncType) `SQL_ASYNC_MODE
```

ODBC_Macros

Macros, Introduction

The command in this chapter enables you to manage macros (*ODBC_LenDataAtExec*).

ODBC_LenDataAtExec

ODBC_LenDataAtExec (lengthData) -> Function result

Parameter	Type		Description
lengthData	Longint	→	Value to convert
Function result	Longint	←	Converted value

Description

The *ODBC_LenDataAtExec* command is used to pass the the parameter at execution time to *ODBC_SQLPutData*.

Example

See the example for the *ODBC_SQLPutData* command.

ODBC_Prepare requests

- Preparing SQL Requests
- ODBC_SQLBindParameter
- ODBC_SQLGetCursorName
- ODBC_SQLPrepare
- ODBC_SQLSetCursorName

Preparing SQL Requests

The commands in this chapter enable you to prepare SQL requests, by allowing you to do the following:

- Bind a parameter in an SQL Statement (*ODBC_SQLBindParameter*)
- Retrieve the cursor name associated with a statement (*ODBC_SQLGetCursorName*)
- Prepare an SQL string for execution (*ODBC_SQLPrepare*)
- Set the cursor name in a specific statement (*ODBC_SQLSetCursorName*)

ODBC_SQLBindParameter

ODBC_SQLBindParameter (stmtID ; paramNumber ; IOType ; paramType ; columnSize ; decimalDigits ; paramValPtr ; strLenOrIndPtr) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
paramNumber	Longint	➡	Parameter number, ordered sequentially in increasing parameter order, starting at 1
IOType	Longint	➡	Type of the parameter Input or Output
paramType	Longint	➡	SQL data type of the parameter
columnSize	Longint	➡	The size of the column or expression of the corresponding parameter defined
decimalDigits	Longint	➡	The decimal digits of the column or expression of the corresponding parameter defined
paramValPtr	Pointer	⬅	A pointer to a 4D variable or field
strLenOrIndPtr	Pointer	⬅	A pointer to the parameter's length if it's a Text value
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLBindParameter

Description

The *ODBC_SQLBindParameter* command binds a parameter and its value in an SQL Statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

paramNumber is the parameter number in the statement defined by *ODBC_SQLPrepare*, ordered sequentially in increasing parameter order, starting at 1.

IOType defines whether this bind parameter is for input, output, or for input and output and can have one of the following values:

Constant	Value
SQL_PARAM_INPUT	1
SQL_PARAM_INPUT_OUTPUT	2
SQL_PARAM_OUTPUT	4

paramType is the SQL data type of the parameter and can be equal to one of the following values:

Constant	SQL Data Type
SQL_CHAR	CHAR
SQL_VARCHAR	VARCHAR
SQL_LONGVARCHAR	LONG VARCHAR
SQL_DECIMAL	DECIMAL
SQL_NUMERIC	NUMERIC
SQL_SMALLINT	SMALLINT
SQL_INTEGER	INTEGER
SQL_REAL	REAL
SQL_FLOAT	FLOAT
SQL_DOUBLE	DOUBLE PRECISION
SQL_BIT	BIT
SQL_TINYINT	TINYINT
SQL_BIGINT	BIGINT
SQL_BINARY	BINARY
SQL_VARBINARY	VARBINARY
SQL_LONGVARBINARY	LONG VARBINARY
SQL_TYPE_DATE	Date
SQL_TYPE_TIME	Time
SQL_TYPE_TIMESTAMP	TIMESTAMP
SQL_INTERVAL_MONTH	INTERVAL MONTH
SQL_INTERVAL_YEAR	INTERVAL YEAR
SQL_INTERVAL_YEAR_TO_MONTH	INTERVAL YEAR TO MONTH
SQL_INTERVAL_DAY	INTERVAL DAY
SQL_INTERVAL_HOUR	INTERVAL HOUR
SQL_INTERVAL_MINUTE	INTERVAL MINUTE
SQL_INTERVAL_SECOND	INTERVAL SECOND
SQL_INTERVAL_DAY_TO_HOUR	INTERVAL DAY TO HOUR
SQL_INTERVAL_DAY_TO_MINUTE	INTERVAL DAY TO MINUTE
SQL_INTERVAL_DAY_TO_SECOND	INTERVAL DAY TO SECOND
SQL_INTERVAL_HOUR_TO_MINUTE	INTERVAL HOUR TO MINUTE
SQL_INTERVAL_HOUR_TO_SECOND	INTERVAL HOUR TO SECOND
SQL_INTERVAL_MINUTE_TO_SECOND	INTERVAL MINUTE TO SECOND

columnSize defines the size of the column or expression of the parameter defined.

decimalDigits defines the decimal digits of the column or expression of the parameter defined.

paramValPtr is a pointer to a 4D variable or field.

strLenOrIndPtr is a pointer to a variable that defines the parameter's length if *paramType* is of type Text, Picture, or BLOB. Use the *ODBC_LenDataAtExec* command to convert the actual length so that it can be processed by the MS ODBC API.

For more information, please see the SQLBindParameter function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710963(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLExecute* command.

ODBC_SQLGetCursorName

ODBC_SQLGetCursorName (stmtID ; cursorName) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
cursorName	String	⬅	Cursor name
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLGetCursorName

Description

The *ODBC_SQLGetCursorName* command returns the cursor name associated with a statement. Cursor names are used only in positioned update and delete statements.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

cursorName is the name of the cursor that was previously set by *ODBC_SQLSetCursorName*.

For more information, please see the SQLGetCursorName function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716209(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method sets a cursor name and then retrieves it. After some processing, the cursor is used to update data in the table:

```
$result:=ODBC_SQLSetCursorName($statementID;"C1")
$result:=ODBC_SQLExecDirect($statementID;"SELECT ID, Name FROM Employee")
$result:=ODBC_SQLGetCursorName($statementID;vNewCursorName)
.
. `more processing...
.
$result:=ODBC_SQLExecDirect($newStmt1;"UPDATE Employee SET Name='Test' WHERE 'CURRENT OF C1'")
```

ODBC_SQLPrepare

ODBC_SQLPrepare (stmtID ; statementText) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
statementText	Text	⇒	SQL text string
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLPrepare

Description

The *ODBC_SQLPrepare* command prepares a SQL string for execution passed in *statementText*.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

statementText is the SQL text string to be executed later with *ODBC_SQLExecute*.

For more information, please see the SQLPrepare function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710926(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLFetch* command.

ODBC_SQLSetCursorName

ODBC_SQLSetCursorName (stmtID ; cursorName) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
cursorName	String	➡	Name of the cursor to set
Function result	Longint	⮌	Returns the result of the MS ODBC API function SQLSetCursorName

Description

The *ODBC_SQLSetCursorName* command associates a cursor name with an active statement. If this command is not called, the driver generates cursor names needed for SQL statement processing.

For efficient processing, the cursor name should not include any leading or trailing spaces, and if the cursor name includes a delimited identifier, the delimiter should be positioned as the first character in the cursor name. Cursor names are used only in positioned update and delete statements and should not exceed 18 characters in length.

If the SQL statement is a SELECT statement and if you set a cursor name with a statement, then the driver uses the specified cursor. Otherwise, the driver generates a cursor name.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

cursorName is the name of the cursor that was previously set by *ODBC_SQLSetCursorName*.

For more information, please see the SQLSetCursorName function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711707(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLGetCursorName* command.

ODBC_Results

- Retrieving Results and Information, Introduction
- ODBC_SQLBindCol
- ODBC_SQLBulkOperations
- ODBC_SQLColAttribute
- ODBC_SQLDescribeCol
- ODBC_SQLFetch
- ODBC_SQLFetchScroll
- ODBC_SQLGetData
- ODBC_SQLGetDescField
- ODBC_SQLGetDescRec
- ODBC_SQLGetDiagField
- ODBC_SQLGetDiagRec
- ODBC_SQLMoreResults
- ODBC_SQLNumResultCols
- ODBC_SQLRowCount
- ODBC_SQLSetDescField
- ODBC_SQLSetDescRec
- ODBC_SQLSetPos

Retrieving Results and Information, Introduction

The commands in this chapter enable you to retrieve results and information about the results, by allowing you to do the following:

- Bind application data buffers to columns in the result set (*ODBC_SQLBindCol*)
- Perform bulk insertions and bulk bookmark operations, including update, delete, and fetch by bookmark (*ODBC_SQLBulkOperations*)
- Get the descriptor information for a column in a result set (*ODBC_SQLColAttribute*)
- Retrieve the result descriptor, such as column name, type, column size, decimal digits, and nullability, for one column in the result set (*ODBC_SQLDescribeCol*)
- Fetch the next rowset of data from the result set and returns data for all bound columns (*ODBC_SQLFetch*)
- Fetch the specified rowset of data from the result set and returns data for all bound columns (*ODBC_SQLFetchScroll*)
- Get data for a single column in the result set (*ODBC_SQLGetData*)
- Get the current setting or value of a single field of a descriptor record (*ODBC_SQLGetDescField*)
- Get the current settings or values of multiple fields of a descriptor record (*ODBC_SQLGetDescRec*)
- Get the current value of a field of a record of the diagnostic data structure (associated with a specified handle) that contains error, warning, and status information (*ODBC_SQLGetDiagField*)
- Get the current values of multiple fields of a diagnostic record that contains error, warning, and status information (*ODBC_SQLGetDiagRec*)
- Determine whether more results are available on a statement containing SELECT, UPDATE, INSERT or DELETE statements and, if so, initializes processing for those results (**ODBC_SQLGetMoreResults**)
- Find out the number of columns in a result set (*ODBC_SQLNumResultCols*)
- Get the number of rows affected by an UPDATE, INSERT, or DELETE statement (*ODBC_SQLRowCount*)
- Set the current setting or value of a single field of a descriptor record (*ODBC_SQLSetDescField*)
- Set the current settings or values of multiple fields of a descriptor record (*ODBC_SQLSetDescRec*)
- Set the cursor position in a rowset and allows an application to refresh data in the rowset or to update or delete data in the result set (*ODBC_SQLSetPos*)

ODBC_SQLBindCol

ODBC_SQLBindCol (stmtID ; colNb ; targetValPtr ; strLenOrInd) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
colNb	Longint	➡	Number of the result set column to bind
targetValPtr	Pointer	⬅	Pointer to the target to bind the column
strLenOrInd	Pointer	⬅	Pointer to the length/indicator buffer to bind to the column
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLBindCol

Description

The *ODBC_SQLBindCol* command binds application data buffers to columns in the result set.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

colNB is the number of the result set column to bind. Columns are numberes in increasing column order starting at 0, where the column 0 is the bookmark column.

targetValuePtr is a pointer to the variable, 4D field or array to bind to the column.

strLenOrInd is an optional parameter that is a pointer to the length of the parameter, which is a Longint, if *paramType* is of type Text, Picture, or BLOB. Use the *ODBC_LenDataAtExec* command to convert the actual length so that it can be processed by the MS ODBC API.

For more information, please see the SQLBindCol function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711010(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the examples for the *ODBC_SQLFetch* and *ODBC_SQLBulkOperations*.

ODBC_SQLBulkOperations

ODBC_SQLBulkOperations (stmtID ; operation) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
operation	Longint	➡	Operation to perform
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLBulkOperations

Description

The *ODBC_SQLBulkOperations* command performs bulk insertions and bulk bookmark operations, including update, delete, and fetch by bookmark.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

The operation to perform are the following:

Constant	Value
SQL_ADD	4
SQL_UPDATE_BY_BOOKMARK	5
SQL_DELETE_BY_BOOKMARK	6
SQL_FETCH_BY_BOOKMARK	7

For more information, please see the SQLBulkOperations function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712471(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method adds three rows from data in two arrays (arID and arEmployeeName) into the Employee table:

```
vattrVal:=SQL_CONCUR_ROWVER
$result:=ODBC_SQLSetStmtAttr($statementID;SQL_ATTR_CONCURRENCY;->vattrVal)
vattrVal:=SQL_CURSOR_KEYSET_DRIVEN
$result:=ODBC_SQLSetStmtAttr($statementID;SQL_ATTR_CURSOR_TYPE;->vattrVal)
vattrVal:=3   ` Size of the arrays that contain our values below
$result:=ODBC_SQLSetStmtAttr($statementID;SQL_ATTR_ROW_ARRAY_SIZE;->vattrVal)
$result:=ODBC_SQLSetStmtAttr($statementID;SQL_ATTR_ROW_STATUS_PTR;->arStatus;vIndic)
vattrVal:=SQL_UB_VARIABLE
$result:=ODBC_SQLSetStmtAttr($statementID;SQL_USE_BOOKMARKS;->vattrVal)
 `Use variable length bookmark
$result:=ODBC_SQLPrepare($statementID;"SELECT * FROM Employee")   `Define which table
$result:=ODBC_SQLExecute($statementID)
$result:=ODBC_SQLBindCol($statementID;1;->arID)   `Bind the columns to arrays
$result:=ODBC_SQLBindCol($statementID;2;->arEmployeeName)

arID{1}:=1006
arID{2}:=1007
arID{3}:=1008

arEmployeeName{1}:="John Smith"
arEmployeeName{2}:="Betty Jones"
arEmployeeName{3}:="Sally Peters"
```

```
$result:=ODBC_SQLBulkOperations($statementID;4)   `SQL_ADD

$result:=ODBC_SQLRowCount($statementID;vRowCount)
```

ODBC_SQLColAttribute

ODBC_SQLColAttribute (stmtID ; colNb ; fieldIdentifier ; characterAttrPtr) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
colNb	Longint	⇒	Number of the record from which the field value is to be retrieved
fieldIdentifier	Longint	⇒	Field identifier in row colNb that is to be returned
characterAttrPtr	Pointer	⇐	Value in the fieldID field of the colNb row if the field is a character string. Otherwise, the field is unused
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLColAttribute

Description

The *ODBC_SQLColAttribute* command returns descriptor information for a column in a result set.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

colNb is the number of the column from which the field value is to be retrieved.

fieldIdentifier defines which field in row *colNb* to be returned, which can be one of the following values:

Constant	Description
SQL_DESC_AUTO_UNIQUE_VALUE	SQL_TRUE if the column is an autoincrementing column. SQL_FALSE if the column is not an auto-incrementing column or is not numeric.
SQL_DESC_BASE_COLUMN_NAME	The base column name for the result set column.
SQL_DESC_BASE_TABLE_NAME	Name of the base table that contains the column.
SQL_DESC_CASE_SENSITIVE	SQL_TRUE if the column is treated as case-sensitive for collations and comparisons. SQL_FALSE if the column is not treated as case-sensitive for collations and comparisons or is noncharacter
SQL_DESC_CATALOG_NAME	Catalog of the table that contains the column.
SQL_DESC_CONCISE_TYPE	Concise data type.
SQL_DESC_COUNT	Number of columns available in the result set.
SQL_DESC_DISPLAY_SIZE	Maximum number of characters required to display data from the column.
SQL_DESC_DISPLAY_SIZE	Maximum number of characters required to display data from the column.
SQL_DESC_FIXED_PREC_SCALE	SQL_TRUE if the column has a fixed precision and nonzero scale that are data source–specific. SQL_FALSE if the column does not have a fixed precision and nonzero scale that are data source-specific.
SQL_DESC_LABEL	Column label or title.
SQL_DESC_LENGTH	A numeric value that is either the maximum or actual character length of a character string or binary data type.
SQL_DESC_LITERAL_PREFIX	Character or characters that the driver recognizes as a prefix for a literal of this data type.
SQL_DESC_LITERAL_SUFFIX	Character or characters that the driver recognizes as a suffix for a literal of this data type.
SQL_DESC_LOCAL_TYPE_NAME	Any localized (native language) name for the data type that may be different from the regular name of the data type.
SQL_DESC_NAME	Column alias, if it applies.
SQL_DESC_NULLABLE	SQL_ NULLABLE if the column can have NULL values; SQL_NO_NULLS if the column does not have NULL values; or SQL_NULLABLE_UNKNOWN if it is not known whether the column accepts NULL values.
SQL_DESC_NUM_PREX_RADIX	Returns 2 if the field is anapproximate numeric data type. If the data type is an exact numeric type, it returns 10 because the SQL_DESC_PRECISION field contains the number of decimal digits. This field is set to 0 for all non-numeric data types.
SQL_DESC_OCTET_LENGTH	Length, in bytes, of a character string or binary data type.
SQL_DESC_PRECISION	A numeric value that for a numeric data type denotes the applicable precision.
SQL_DESC_SCALE	A numeric value that is the applicable scale for a numeric data type.
SQL_DESC_SCHEMA_NAME	The schema of the table that contains the column.
SQL_DESC_SEARCHABLE	SQL_PRED_NONE if the column cannot be used in a WHERE clause. SQL_PRED_CHAR if the column can be used in a WHERE clause but only with the LIKE predicate. SQL_PRED_BASIC if the column can be used in a WHERE clause with all the comparison operators except LIKE. SQL_PRED_SEARCHABLE if the column can be used in a WHERE clause with any comparison operator.
SQL_DESC_TABLE_NAME	Name of the table that contains the column.
SQL_DESC_TYPE	Numeric value that specifies the SQL data type.
SQL_DESC_TYPE_NAME	Data source–dependent data type name, for example, "CHAR".
SQL_DESC_UNNAMED	If it contains a column alias or a column name, SQL_NAMED is returned. If there is no column name or column alias, SQL_UNNAMED is returned.
SQL_DESC_UNSIGNED	SQL_TRUE if the column is unsigned (or not numeric). SQL_FALSE if the column is signed.
SQL_DESC_UPDATABLE	Updatability of the column

characterAttrPtr is a pointer to the value returned based on *fieldIdentifier*.

For more information, please see the SQLColAttribute function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713558(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method returns the name of the second column in the Employee table:

```
$result:=ODBC_SQLPrepare($statementID;"SELECT * FROM Employee")
$result:=ODBC_SQLColAttribute($statementID;2;SQL_DESC_LABEL;->vColumnName)
```

ODBC_SQLDescribeCol

ODBC_SQLDescribeCol (stmtID ; colNb ; colName ; dataType ; colSize ; decimalDigits ; nullable) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
colNb	Longint	⇒	Column number of result data
colName	String	⇐	Column name
dataType	Longint	⇐	Data type of the column
colSize	Longint	⇐	Size of the column
decimalDigits	Longint	⇐	Number of decimal digits of the column
nullable	Longint	⇐	Indicates if the column allows NULL values
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLDescribeCol

Description

The *ODBC_SQLDescribeCol* command returns the result descriptor, such as column name, type, column size, decimal digits, and nullability, for one column in the result set.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

colNb is the column number containing the result data.

colName is the name of the column.

dataType can be one of the following values:

Constant	SQL Data Type
SQL_CHAR	CHAR
SQL_VARCHAR	VARCHAR
SQL_LONGVARCHAR	LONG VARCHAR
SQL_DECIMAL	DECIMAL
SQL_NUMERIC	NUMERIC
SQL_SMALLINT	SMALLINT
SQL_INTEGER	INTEGER
SQL_REAL	REAL
SQL_FLOAT	FLOAT
SQL_DOUBLE	DOUBLE PRECISION
SQL_BIT	BIT
SQL_TINYINT	TINYINT
SQL_BIGINT	BIGINT
SQL_BINARY	BINARY
SQL_VARBINARY	VARBINARY
SQL_LONGVARBINARY	LONG VARBINARY
SQL_TYPE_DATE	Date
SQL_TYPE_TIME	Time
SQL_TYPE_TIMESTAMP	TIMESTAMP
SQL_INTERVAL_MONTH	INTERVAL MONTH
SQL_INTERVAL_YEAR	INTERVAL YEAR
SQL_INTERVAL_YEAR_TO_MONTH	INTERVAL YEAR TO MONTH
SQL_INTERVAL_DAY	INTERVAL DAY
SQL_INTERVAL_HOUR	INTERVAL HOUR
SQL_INTERVAL_MINUTE	INTERVAL MINUTE
SQL_INTERVAL_SECOND	INTERVAL SECOND
SQL_INTERVAL_DAY_TO_HOUR	INTERVAL DAY TO HOUR
SQL_INTERVAL_DAY_TO_MINUTE	INTERVAL DAY TO MINUTE
SQL_INTERVAL_DAY_TO_SECOND	INTERVAL DAY TO SECOND
SQL_INTERVAL_HOUR_TO_MINUTE	INTERVAL HOUR TO MINUTE
SQL_INTERVAL_HOUR_TO_SECOND	INTERVAL HOUR TO SECOND
SQL_INTERVAL_MINUTE_TO_SECOND	INTERVAL MINUTE TO SECOND

colSize is the size of the column.

decimalDigits is the number of decimal digits of the column.

nullable indicates if the column allows NULL values and can have one of the following values:

Constant	Description
SQL_NO_NULLS	Does not allow NULL values
SQL_NULLABLE	Allows NULL values
SQL_NULLABLE_UNKNOWN	Driver cannot determine if the parameter allows NULL values

For more information, please see the SQLDescribeCol function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716289(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method returns the name of the third column in the Employee table:

```
$result:=ODBC_SQLPrepare($statementID;"SELECT * FROM Employee")
$result:=ODBC_SQLDescribeCol($statementID;3;vColumnName;vDataType;vcolSize;vDecimalDigits;vNullable)
```

ODBC_SQLFetch

ODBC_SQLFetch (stmtID) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
Function result	Longint	⮌	Returns the result of the MS ODBC API function SQLFetch

Description

The *ODBC_SQLFetch* command fetches the next rowset of data from the result set and returns data for all bound columns.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

For more information, please see the SQLFetch function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712424(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method selects all the records from the Employee table, ccreates a bind with our data source's Employee table and its four fields specified by the *ODBC_SQLPrepare* and *ODBC_SQLBindCol* commands, and then creates one record for each row of data in the 4D table [Employees]:

```
$result:=ODBC_SQLPrepare($statementID;"SELECT * FROM Employee")
$result:=ODBC_SQLExecute($statementID)

$result:=ODBC_SQLBindCol($statementID;1;->[Employees]Emp_ID)
$result:=ODBC_SQLBindCol($statementID;2;->[Employees]Emp_Fullname)
$result:=ODBC_SQLBindCol($statementID;3;->[Employees]Emp_HireDate)
$result:=ODBC_SQLBindCol($statementID;4;->[Employees]Emp_CurrentEmployee)

While($result#SQL_NO_DATA)
   CREATE RECORD([Employees])
   $result:=ODBC_SQLFetch($statementID)
   If($result#SQL_NO_DATA)
      SAVE RECORD([Employees])
   End if
End while
UNLOAD RECORD([Employees])

$result:=ODBC_SQLFreeStmt($statementID;SQL_UNBIND)
```

ODBC_SQLFetchScroll

ODBC_SQLFetchScroll (stmtID ; fetchOrientation ; fetchOffset) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
fetchOrientation	Longint	➡	Type of fetch
fetchOffset	Longint	➡	Number of the row to fetch
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLFetchScroll

Description

The *ODBC_SQLFetchScroll* command fetches the specified rowset of data from the result set and returns data for all bound columns. Rowsets can be specified at an absolute or relative position or by bookmark.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

fetchOrientation is the type of fetch and can be one of the following types:

Constant	Description
SQL_FETCH_NEXT	Return the next rowset
SQL_FETCH_PRIOR	Return the prior rowset
SQL_FETCH_FIRST	Return the first rowset
SQL_FETCH_LAST	Return the last rowset
SQL_FETCH_ABSOLUTE	Return the rowset starting at row *fetchOffset*.
SQL_FETCH_RELATIVE	Return the rowset *fetchOffset* from the start of the current rowset
SQL_FETCH_BOOKMARK	Return the rowset *fetchOffset* rows from the bookmark

fetchOffset is the offset to be used when the contant SQL_FETCH_ABSOLUTE or SQL_FETCH_RELATIVE is passed to the *fetchOrientation* argument

For more information, please see the SQLFetchScroll function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714682(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLGetData

ODBC_SQLGetData (stmtID ; colNb ; targetValPtr ; strLenOrInd) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
colNb	Longint	⇒	Number of the column for which to return data
targetValPtr	Pointer	⇐	Pointer to a variable in which to return the data
strLenOrInd	Longint	⇐	Length or indicator value
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLGetData

Description

The *ODBC_SQLGetData* command retrieves data for a single column defined by *colNb* in the result set. It can be called multiple times to retrieve variable-length data in parts.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

colNb is the number of the column to receive the data.

targetValPtr is a pointer to the variable in which to return the data.

strLenOrInd is the length or indicator value of the value returned in *targetValPtr.*

For more information, please see the SQLGetData function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms715441(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLGetDescField

ODBC_SQLGetDescField (connectionID ; recNumber ; fieldIdentifier ; valuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
recNumber	Longint	⇒	Descriptor record number
fieldIdentifier	Longint	⇒	Field of the descriptor whose value is to be returned
valuePtr	Pointer	⇐	Pointer to a variable to receive the descriptor information
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLGetDescField

Description

The *ODBC_SQLGetDescField* command returns the current setting or value of a single field of a descriptor record.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

Descriptor records are numbered from 0, with record number 0 being the bookmark record. If the *fieldIdentifier* argument indicates a header field, *recNumber* is ignored. If *recNumber* is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to *ODBC_SQLGetDescField* will return the default values of the fields.

recNumber is the descriptor record number.

fieldIdentifier is the field of the descriptor whose value is to be returned.

valuePtr is a pointer to the variable in which to receive the descriptor information.

For more information, please see the SQLGetDescField function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716370(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, SQL_NO_DATA, or SQL_INVALID_HANDLE.

SQL_NO_DATA is returned if *recNumber* is greater than the current number of descriptor records.

ODBC_SQLGetDescRec

ODBC_SQLGetDescRec (stmtID ; recNumber ; name ; type ; subType ; length ; precision ; scale ; nullable) -> Function result

Parameter	Type		Description
stmtID	Longint	→	Statement ID
recNumber	Integer	→	Descriptor record number
name	String	←	SQL_DESC_NAME field for the descriptor record
type	Longint	←	SQL_DESC_TYPE field for the descriptor record
subType	Longint	←	SQL_DESC_DATETIME_INTERVAL_CODE field for the descriptor record
length	Longint	←	SQL_DESC_OCTET_LENGTH field for the descriptor record
precision	Longint	←	SQL_DESC_PRECISION field for the descriptor record
scale	Longint	←	SQL_DESC_SCALE field for the descriptor record
nullable	Longint	←	SQL_DESC_NULLABLE field for the descriptor record
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLGetDescRec

Description

The *ODBC_SQLGetDescRec* command returns the current settings or values of multiple fields of a descriptor record. The fields returned describe the name, data type, and storage of column or parameter data.

StmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

Descriptor records are numbered from 0, with record number 0 being the bookmark record. If the *fieldIdentifier* argument indicates a header field, *recNumber* is ignored. If RecNumber is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to *ODBC_SQLGetDescField* will return the default values of the fields.

recNumber is the descriptor record number.

name is the SQL_DESC_NAME field for the descriptor record.

type is the SQL_DESC_TYPE field for the descriptor record.

subType is the SQL_DESC_DATETIME_INTERVAL_CODE field for the descriptor record (for records whose *type* is SQL_DATETIME or SQL_INTERVAL).

length is the SQL_DESC_OCTET_LENGTH field for the descriptor record.

precision is the SQL_DESC_PRECISION field for the descriptor record.

scale is the SQL_DESC_SCALE field for the descriptor record.

nullable is the SQL_DESC_NULLABLE field for the descriptor record.

For more information, please see the SQLGetDescRec function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710921(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLGetDiagField

ODBC_SQLGetDiagField (handleType ; handleID ; recNb ; diagID ; diagInfoPtr ; stringLengthPtr) -> Function result

Parameter	Type		Description
handleType	Longint	➡	Type of ID to pass to handleID
handleID	Longint	➡	Handle ID for the diagnostic data structure
recNb	Longint	➡	Indicates the status record from which the application seeks information
diagID	Longint	➡	Indicates the field of the diagnostic whose value is to be returned
diagInfoPtr	Pointer	⬅	Pointer to a variable in which to return the diagnostic information. The data type depends on the value of diagID
stringLengthPtr	Pointer	⬅	Total length of the string returned in diagInfoPtr
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLGetDiagField

Description

The *ODBC_SQLGetDiagField* command returns the current value of a field of a record of the diagnostic data structure (associated with a specified *handleID*) that contains error, warning, and status information.

handleType defines the type of ID to pass to *handleID*, which can be one of the following constants:

Constant	Description
SQL_HANDLE_ENV	Environment ID
SQL_HANDLE_DBC	Connection ID
SQL_HANDLE_STMT	Statement ID
SQL_HANDLE_DESC	Descriptor ID

handleID is a handle ID for the diagnostic data structure, of the type indicated by *handleType*. If *handleType* is SQL_HANDLE_ENV, this parameter is taken into account and the constant SQL_DEFAULT_ID can be then used.

handleID is the *connectionID* if *handleType* is equal to SQL_HANDLE_DBC. *connectionID* is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

If *handleType* is SQL_HANDLE_STMT, *handleID* is the *stmtID*, which is a valid statement ID returned by *ODBC_SQLAllocStmt*.

recNb indicates the status record from which the application seeks information.

diagID indicates the field of the diagnostic whose value is to be returned. It can be one of the following values:

Constant	Description
SQL_DIAG_CLASS_ORIGIN	A string that indicates the document that defines the class portion of the SQLSTATE value in this record
SQL_DIAG_COLUMN_NUMBER	The column number in the result set or theparameter number in the set of parameters
SQL_DIAG_CONNECTION_NAME	A string that indicates the name of the connection that the diagnostic record relates to.
SQL_DIAG_CURSOR_ROW_COUNT	The count of rows in the cursor.
SQL_DIAG_DYNAMIC_FUNCTION	This is a string that describes the SQL statement that the underlying function executed
SQL_DIAG_DYNAMIC_FUNCTION_CODE	This is a numeric code that describes the SQL statement that was executed by the underlying function.
SQL_DIAG_MESSAGE_TEXT	An informational message on the error or warning.
SQL_DIAG_NATIVE	A driver/data source–specific native error code.
SQL_DIAG_NUMBER	Number of status records that are available
SQL_DIAG_RETURNCODE	Return code returned by the function
SQL_DIAG_ROW_COUNT	Number of rows affected by an insert, delete, or update performed by *ODBC_SQLExecute*, **ODBC_SQLExecDirect**, *ODBC_SQLBulkOperations*, or *ODBC_SQLSetPos*
SQL_DIAG_SERVER_NAME	A string that indicates the server name that the diagnostic record relates to.
SQL_DIAG_SQLSTATE	A five-character SQLSTATE diagnostic code
SQL_DIAG_SUBCLASS_ORIGIN	A string with the same format and valid values as SQL_DIAG_CLASS_ORIGIN, that identifies the defining portion of the subclass portion of the SQLSTATE code

diagInfoPtr is a pointer to the variable in which the diagnostic information will be returned. Its type is dependent on the *diagID*.

stringLengthPtr is a pointer to a variable in which to return the length of the string/text returned in *diagInfoPtr*.

For more information, please see the SQLGetDiagField function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710181(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, SQL_INVALID_HANDLE, or SQL_NO_DATA.

ODBC_SQLGetDiagRec

ODBC_SQLGetDiagRec (handleType ; handleID ; recNb ; sqlState ; nativeError ; messageText ; textLength) -> Function result

Parameter	Type		Description
handleType	Longint	⇒	Type of ID to pass to handleID
handleID	Longint	⇒	Handle ID for the diagnostic data structure
recNb	Longint	⇒	Indicates the status record from which information is sought. Status records are numbered from 1
sqlState	String	⇐	Five-character SQLSTATE code pertaining to the diagnostic record recNb
nativeError	Longint	⇐	Native error code, specific to the data source
messageText	Text	⇐	Diagnostic message text string
textLength	Longint	⇐	Length of the string returned in messageText
Function result	Longint	⇒	Returns the result of the MS ODBC API function SQLGetDiagRec

Description

The *ODBC_SQLGetDiagRec* command returns the current values of multiple fields of a diagnostic record that contains error, warning, and status information. Call this command any time one of the other 4D ODBC PRO commands does not return SQL_SUCCESS.

handleType defines the type of ID to pass to *handleID*, which can be one of the following:

Constant	Description
SQL_HANDLE_ENV	Environment ID
SQL_HANDLE_DBC	Connection ID
SQL_HANDLE_STMT	Statement ID
SQL_HANDLE_DESC	Descriptor ID

handleID is a handle ID for the diagnostic data structure, of the type indicated by *handleType*. If *handleType* is SQL_HANDLE_ENV, this parameter is taken into account and the constant SQL_DEFAULT_ID can be then used.

handleID is the *connectionID* if *handleType* is equal to SQL_HANDLE_DBC. *connectionID* is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

Otherwise, *handleID* is the *stmtID*, which is a valid statement ID returned by *ODBC_SQLAllocStmt*.

recNb indicates the status record from which information is sought. Status records are numbered from 1.

sqlState is the five-character SQLSTATE code pertaining to *recNb*. The first two characters indicate the class; the next three indicate the subclass. This value comes from the SQL_DIAG_SQLSTATE diagnostic field.

nativeError is the native error code specific to the data source, from the SQL_DIAG_NATIVE diagnostic field.

messageText is the diagnostic message text string, which comes from the SQL_DIAG_MESSAGE_TEXT field.

For more information, please see the SQLGetDiagRec function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms716256(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method is called after the result from calling *ODBC_SQLExecute* is not equal to SQL_SUCCESS. The *debugMessage* variable will contain the error message:

```
If($result#SQL_SUCCESS)
   $resultDiag:=ODBC_SQLGetDiagRec(SQL_HANDLE_STMT;$statementID;1;SQLState;nativeError;
   debugMessage;vTextLen)
End if
```

ODBC_SQLMoreResults

ODBC_SQLMoreResults (stmtID) -> Function result

Parameter	Type		Description
stmtID	Longint	→	Statement ID
Function result	Longint	←	Returns the result of the MS ODBC API function SQLMoreResults

Description

The *ODBC_SQLMoreResults* command determines whether more results are available on a statement containing SELECT, UPDATE, INSERT or DELETE statements and, if so, initializes processing for those results.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

For more information, please see the SQLMoreResults function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714673(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_NO_DATA, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLNumResultCols

ODBC_SQLNumResultCols (stmtID ; columnCount) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
columnCount	Longint	⇐	Number of columns in the result set
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLNumResultCols

Description

The *ODBC_SQLNumResultCols* command returns the number of columns in a result set.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

columnCount is the number of columns in the result set. It does not include a bound bookmark column.

For more information, please see the SQLNumResultCols function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms715393(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLRowCount

ODBC_SQLRowCount (stmtID ; rowCount) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
rowCount	Longint	⇐	Number of rows affected by the request
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLRowCount

Description

The *ODBC_SQLRowCount* command returns the number of rows affected by an UPDATE, INSERT, or DELETE statement; an SQL_ADD, SQL_UPDATE_BY_BOOKMARK, or SQL_DELETE_BY_BOOKMARK operation in *ODBC_SQLBulkOperations*; or an SQL_UPDATE or SQL_DELETE operation in *ODBC_SQLSetPos*.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

rowCount is the number of rows affected by the result set.

For more information, please see the SQLRowCount function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms711835(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLBulkOperations* command.

ODBC_SQLSetDescField

ODBC_SQLSetDescField (connectionID ; recNumber ; fieldIdentifier ; valuePtr) -> Function result

Parameter	Type		Description
connectionID	Longint	⇒	Connection ID
recNumber	Longint	⇒	Descriptor record number
fieldIdentifier	Longint	⇒	Field of the descriptor whose value is to be set
valuePtr	Pointer	⇒	Pointer to the value to set fieldIdentifier
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetDescField

Description

The *ODBC_SQLSetDescField* command sets the value of a single field of a descriptor record.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

Descriptor records are numbered from 1, with *recNumber* equal to 0 being the bookmark record. The *recNumber* argument must be less than or equal to the value of SQL_DESC_COUNT. If *recNumber* is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to *ODBC_SQLSetDescField* will return the default values of the fields.

recNumber is the descriptor record number.

fieldIdentifier is the field of the descriptor whose value is to be set.

valuePtr is a pointer to the variable to set *fieldIdentifier*.

For more information, please see the SQLSetDescField function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713560(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLSetDescRec

ODBC_SQLSetDescRec (stmtID ; recNumber ; type ; subType ; length ; precision ; scale ; dataPtr ; stringLengthPtr ; indicatorPtr) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
recNumber	Integer	⇒	Descriptor record number
type	Integer	⇒	SQL_DESC_TYPE field for the descriptor record
subType	Integer	⇒	SQL_DESC_DATETIME_INTERVAL_CODE field for the descriptor record
length	Integer	⇒	SQL_DESC_OCTET_LENGTH field for the descriptor record
precision	Integer	⇒	SQL_DESC_PRECISION field for the descriptor record
scale	Integer	⇒	SQL_DESC_SCALE field for the descriptor record
dataPtr	Pointer	⇒	SQL_DESC_DATA_PTR field for the descriptor record
stringLengthPtr	Pointer	⇒	SQL_DESC_OCTET_LENGTH_PTR field for the descriptor record
indicatorPtr	Pointer	⇒	SQL_DESC_INDICATOR_PTR field for the descriptor record
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetDescRec

Description

The *ODBC_SQLSetDescRec* command sets multiple descriptor fields that affect the data type and buffer bound to a column or parameter data.

StmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

Descriptor records are numbered from 1, with *recNumber* equal to 0 being the bookmark record. The *recNumber* argument must be less than or equal to the value of SQL_DESC_COUNT. If *recNumber* is less than or equal to SQL_DESC_COUNT but the row does not contain data for a column or parameter, a call to *ODBC_SQLSetDescRec* will return the default values of the fields.

recNumber is the descriptor record number.

type is the SQL_DESC_TYPE field for the descriptor record.

subType is the SQL_DESC_DATETIME_INTERVAL_CODE field for the descriptor record (for records whose type is SQL_DATETIME or SQL_INTERVAL).

length is the SQL_DESC_OCTET_LENGTH field for the descriptor record.

precision is the SQL_DESC_PRECISION field for the descriptor record.

scale is the SQL_DESC_SCALE field for the descriptor record.

dataPtr is the SQL_DESC_DATA_PTR field for the descriptor record.

stringLengthPtr is the SQL_DESC_OCTET_LENGTH_PTR field for the descriptor record.

indicatorPtr is the SQL_DESC_OCTET_INDICATOR_PTR field for the descriptor record.

For more information, please see the SQLSetDescRec function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714675(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLSetPos

ODBC_SQLSetPos (stmtID ; rowNb ; operation ; lockType) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
rowNb	Longint	⇒	Position of the row in the rowset on which to perform the operation specified with the operation argument
operation	Longint	⇒	Operation to perform
lockType	Longint	⇒	Specifies how to lock the row after performing the operation
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLSetPos

Description

The *ODBC_SQLSetPos* command sets the cursor position in a rowset and allows an application to refresh data in the rowset or to update or delete data in the result set.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

rowNb is the position of the row in the rowset on which to perform the operation specified with the *operation* argument. If 0, then the operation applies to every row in the rowset.

operation is the operation to perform and can be one of the following constants:

Constant	Value	Description
SQL_POSITION	0	Driver positions the cursor on the row specified by *rowNb*
SQL_REFRESH	1	Driver positions the cursor on the row specified by *rowNb* and refreshes data in the rowset for that row
SQL_UPDATE	2	Driver positions the cursor on the row specified by *rowNb* and updates the underlying row of data with the values in the row set
SQL_DELETE	3	Driver positions the cursor on the row specified by *rowNb* and deletes the underlying row of data

lockType specifies how to lock the row after performing the operation:

Constant	Value	Description
SQL_LOCK_NO_CHANGE	0	Driver or data source ensures that the row is in the same locked or unlocked state as it was before **ODBC_SQLSetPos** was called
SQL_LOCK_EXCLUSIVE	1	Driver or data source locks the row exclusively
SQL_LOCK_UNLOCK	2	Driver or data source unlocks the row

For more information, please see the SQLSetPos function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713507(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_Submit requests

- Submitting Requests
- ODBC_SQLDescribeParam
- ODBC_SQLExecDirect
- ODBC_SQLExecute
- ODBC_SQLNativeSql
- ODBC_SQLNumParams
- ODBC_SQLParamData
- ODBC_SQLPutData

Submitting Requests

The commands in this chapter enable you to submit SQL requests, by allowing you to do the following:

- Retrieve the description of a parameter associated with a prepared SQL statement (*ODBC_SQLDescribeParam*)
- Execute a statement, using the current values of the parameter marker variables if any parameters exist in the statement that you pass directly to it (*ODBC_SQLExecDirect*)
- Execute a prepared statement, using the current values of the parameter marker variables if any parameter markers exist in the statement (*ODBC_SQLExecute*)
- Return the SQL string as modified by the driver but do not execute the SQL statement (*ODBC_SQLNativeSQL*)
- Obtain the number of parameters in an SQL statement (*ODBC_SQLNumParams*)
- Supply parameter data at statement execution time before calling *ODBC_SQLPutData* (*ODBC_SQLParamData*)
- Send data for a parameter or column to the driver (*ODBC_SQLPutData*)

ODBC_SQLDescribeParam

ODBC_SQLDescribeParam (stmtID ; paramNb ; dataType ; paramSize ; decimalDigits ; nullable) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
paramNb	Longint	⇒	Parameter marker number ordered sequentially in increasing parameter order, starting at 1
dataType	Longint	⇐	SQL data type of the parameter
paramSize	Longint	⇐	Size of the column or expression of the corresponding parameter marker as defined by the data source
decimalDigits	Longint	⇐	Number of decimal digits of the column or expression of the corresponding parameter as defined by the data source
nullable	Longint	⇐	Indicates whether the parameter allows NULL values
Function result	Longint	⇒	Returns the result of the MS ODBC API function SQLDescribeParam

Description

The *ODBC_SQLDescribeParam* command returns the description of a parameter associated with a prepared SQL statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

paramNb is the parameter marker number ordered sequentially in increasing parameter order, starting at 1.

dataType is the SQL data type of the parameter. See the *ODBC_SQLGetTypeInfo* command for possible data types.

paramSize is the size of the column or expression.

decimalDigits is the number of decimal digits of the column or expression.

The *nullable* parameter indicates whether the parameter allows NULL values and can be equal to one of the following values:

Constant	Description
SQL_NO_NULLS	Does not allow NULL values
SQL_NULLABLE	Allows NULL values
SQL_NULLABLE_UNKNOWN	Driver cannot determine if the parameter allows NULL values

For more information, please see the SQLDescribeParam function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms710188(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLExecDirect

ODBC_SQLExecDirect (stmtID ; stmtText) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
stmtText	Text	⇒	SQL statement to be executed
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLExecDirect

Description

The *ODBC_SQLExecDirect* command executes a preparable statement, using the current values of the parameter marker variables if any parameters exist in the statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

stmtText is an SQL statement to be executed.

For more information, please see the SQLExecDirect function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713611(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_ERROR, SQL_NO_DATA, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLGetCursorName* command.

ODBC_SQLExecute

ODBC_SQLExecute (stmtID) -> Function result

Parameter	Type		Description
stmtID	Longint	➡	Statement ID
Function result	Longint	⮌	Returns the result of the MS ODBC API function SQLExecute

Description

The *ODBC_SQLExecute* command executes a prepared statement, using the current values of the parameter marker variables if any parameter markers exist in the statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

For more information, please see the SQLExecute function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713584(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_ERROR, SQL_NO_DATA, or SQL_INVALID_HANDLE.

If *ODBC_SQLExecute* returns SQL_NEED_DATA, you can use the *ODBC_SQLParamData* and *ODBC_SQLPutData* commands.

Example

The following method creates a bind with our data source's Employee table and its four fields specified by the *ODBC_SQLPrepare* command and then inserts the data defined in the *ODBC_SQLBindParameter* command once the statement is executed:

```
$result:=ODBC_SQLPrepare($statementID;"INSERT INTO Employee (ID, Name, Hire_Date,
Current_Employee)VALUES(?,?,?,?)")

vEmployeeHireDate:=Current date
vEmployeeID:=6
vEmployeeFullname:="Betty Jones"
vEmployeeCurrent:=True

$result:=ODBC_SQLBindParameter($statementID;1;1;SQL_SMALLINT;0;0;->vEmployeeID)
$result:=ODBC_SQLBindParameter($statementID;2;1;SQL_CHAR;10;0;->vEmployeeFullname)
$result:=ODBC_SQLBindParameter($statementID;3;1;SQL_TYPE_DATE;0;0;->vEmployeeHireDate)
$result:=ODBC_SQLBindParameter($statementID;4;1;SQL_BIT;0;0;->vEmployeeCurrent)

$result:=ODBC_SQLNumParams($statementID;$numparams)
$result:=ODBC_SQLExecute($statementID)
```

ODBC_SQLNativeSql

ODBC_SQLNativeSql (connectionID ; inStatementText ; outStatementText) -> Function result

Parameter	Type		Description
connectionID	Longint	➡	Connection ID
inStatementText	Text	➡	SQL text string to be translated
outStatementText	Text	⬅	Translated SQL string
Function result	Longint	↩	Returns the result of the MS ODBC API function SQLNativeSql

Description

The *ODBC_SQLNativeSql* command returns the SQL string as modified by the driver but does not execute the SQL statement.

connectionID is a valid connection ID returned by *ODBC_SQLAllocConnect* and a connection must be established using the *ODBC_SQLConnect* command.

inStatementText is the SQL text string to be translated.

outStatementText is the translated SQL string.

For more information, please see the SQLNativeSql function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms714575(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE.

ODBC_SQLNumParams

ODBC_SQLNumParams (stmtID ; parameterCount) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
parameterCount	Longint	⇐	Number of parameters in the statement
Function result	Longint	⇔	Returns the result of the MS ODBC API function SQLNumParams

Description

The *ODBC_SQLNumParams* command returns the number of parameters in an SQL statement.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

parameterCount is the number of parameters in the statement specified by *stmtID*.

For more information, please see the SQLNumParams function in the MS ODBC API at http://msdn.microsoft.com/en-us/library/ms715409(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLExecute* command.

ODBC_SQLParamData

ODBC_SQLParamData (stmtID ; valuePtr) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
valuePtr	Pointer	⇐	Pointer to the parameter data or column data
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLParamData

Description

The *ODBC_SQLParamData* command is used in conjunction with *ODBC_SQLPutData* to supply parameter data at statement execution time.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

valuePtr is equal to *paramValPtr* passed to *ODBC_SQLBindParameter* (for parameter data) or *targetValuePtr* passed to *ODBC_SQLBindCol* (for column data).

For more information, please see the SQLParamData function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms712366(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

See the example for the *ODBC_SQLPutData* command.

ODBC_SQLPutData

ODBC_SQLPutData (stmtID ; valuePtr ; strLenOrInd) -> Function result

Parameter	Type		Description
stmtID	Longint	⇒	Statement ID
valuePtr	Pointer	⇒	Pointer to the actual data for the parameter or column
strLenOrInd	Longint	⇒	Amount of data to send
Function result	Longint	⇐	Returns the result of the MS ODBC API function SQLPutData

Description

The *ODBC_SQLPutData* command sends data for a parameter or column to the driver at statement execution time.

stmtID is a valid statement ID returned by *ODBC_SQLAllocStmt*.

valuePtr is a pointer to the data for the parameter or column.

strLenOrInd is an optional parameter that defines the amount of data to send if *paramType* is of type Text, Picture, or BLOB. Use the *ODBC_LenDataAtExec* command to convert the actual length so that it can be processed by the MS ODBC API.

For more information, please see the SQLPutData function in the MS ODBC API Reference at http://msdn.microsoft.com/en-us/library/ms713824(VS.85).aspx.

Function Results

SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_ERROR, or SQL_INVALID_HANDLE.

Example

The following method creates a bind with our data source's Employee table and inserts data into its four fields. If *ODBC_SQLPrepare* command returns SQL_NEED_DATA, we find out which parameter need data by calling *ODBC_SQLParamData* and insert a value using *ODBC_SQLPutData*:

```
vIndic:=ODBC_LenDataAtExec(5)
$result:=ODBC_SQLPrepare($stmtID;"INSERT INTO Employee (ID, Name, Hire_Date, Current_Employee)
VALUES(?,?,?,?)")

vEmployeeHireDate:=Current date
vEmployeeID:=6
vEmployeeFullname:=""
vEmployeeCurrent:=True

$result:=ODBC_SQLBindParameter($statementID;1;1;SQL_SMALLINT;0;0;->vEmployeeID)
$result:=ODBC_SQLBindParameter($statementID;2;1;SQL_CHAR;10;0;->vEmployeeFullname;->vIndic)
$result:=ODBC_SQLBindParameter($statementID;3;1;SQL_TYPE_DATE;0;0;->vEmployeeHireDate)
$result:=ODBC_SQLBindParameter($statementID;4;1;SQL_BIT;0;0;->vEmployeeCurrent)

$result:=ODBC_SQLExecute($stmtID)
While($result=SQL_NEED_DATA)
   $result:=ODBC_SQLParamData($stmtID;vWhichField)   `Returns a pointer to the
expected parameter
   vWhichField->:="More data needed"
   $result:=ODBC_SQLPutData($stmtID;vWhichField)
End while
```

Appendixes

Appendix A, 4D ODBC PRO Error Codes

This section describes all the error codes returned by 4D ODBC PRO:

Constant	Value
SQL_ERROR	-1
SQL_INVALID_HANDLE	-2
SQL_NEED_DATA	99
SQL_NO_DATA	100
SQL_STILL_EXECUTING	2
SQL_SUCCESS	0
SQL_SUCCESS_WITH_INFO	1

Appendix B, 4D ODBC PRO Constants

This section details all of the SQL constants that can be used in 4D ODBC PRO.

Important Note: In the current release of the 4D ODBC PRO plug-in, not all of the following SQL constants are available as such, but you can still use the associated value. In the list below, the constants that have not been created are followed by a * sign.

Constant	Value
SQL_ACCESS_MODE	101
SQL_ACCESSIBLE_PROCEDURES	20
SQL_ACCESSIBLE_TABLES	19
SQL_ACTIVE_CONNECTIONS*	0
SQL_ACTIVE_ENVIRONMENTS	116
SQL_ACTIVE_STATEMENTS*	1
SQL_AGGREGATE_FUNCTIONS	169
SQL_ALL_TYPES	0
SQL_ALTER_DOMAIN	117
SQL_ALTER_TABLE	86
SQL_AM_CONNECTION	1
SQL_AM_NONE	0
SQL_AM_STATEMENT	2
SQL_API_SQLALLOCCONNECT	1
SQL_API_SQLALLOCENV	2
SQL_API_SQLALLOCHANDLE	1001
SQL_API_SQLALLOCSTMT	3
SQL_API_SQLBINDCOL	4
SQL_API_SQLBINDPARAM	1002
SQL_API_SQLBINDPARAMETER	72
SQL_API_SQLBROWSECONNECT	55
SQL_API_SQLCANCEL	5
SQL_API_SQLCLOSECURSOR	1003
SQL_API_SQLCOLATTRIBUTES	6
SQL_API_SQLCOLUMNPRIVILEGES	56
SQL_API_SQLCOLUMNS	40
SQL_API_SQLCONNECT	7
SQL_API_SQLCOPYDESC	1004
SQL_API_SQLDATASOURCES	57
SQL_API_SQLDESCRIBECOL	8
SQL_API_SQLDESCRIBEPARAM	58
SQL_API_SQLDISCONNECT	9
SQL_API_SQLDRIVERCONNECT	41
SQL_API_SQLDRIVERS	71
SQL_API_SQLENDTRAN	1005
SQL_API_SQLERROR	10
SQL_API_SQLEXECDIRECT	11
SQL_API_SQLEXECUTE	12
SQL_API_SQLEXTENDEDFETCH	59
SQL_API_SQLFETCH	13
SQL_API_SQLFETCHSCROLL	1021
SQL_API_SQLFOREIGNKEYS	60
SQL_API_SQLFREECONNECT	14
SQL_API_SQLFREEENV	15
SQL_API_SQLFREEHANDLE	1006
SQL_API_SQLFREESTMT	16
SQL_API_SQLGETCONNECTATTR	1007
SQL_API_SQLGETCONNECTOPTION	42

SQL_API_SQLGETCURSORNAME	17
SQL_API_SQLGETDATA	43
SQL_API_SQLGETDESCFIELD	1008
SQL_API_SQLGETDESCREC	1009
SQL_API_SQLGETDIAGFIELD	1010
SQL_API_SQLGETDIAGREC	1011
SQL_API_SQLGETENVATTR	1012
SQL_API_SQLGETFUNCTIONS	44
SQL_API_SQLGETINFO	45
SQL_API_SQLGETSTMTATTR	1014
SQL_API_SQLGETSTMTOPTION	46
SQL_API_SQLGETTYPEINFO	47
SQL_API_SQLMORERESULTS	61
SQL_API_SQLNATIVESQL	62
SQL_API_SQLNUMPARAMS	63
SQL_API_SQLNUMRESULTCOLS	18
SQL_API_SQLPARAMDATA	48
SQL_API_SQLPARAMOPTIONS	64
SQL_API_SQLPREPARE	19
SQL_API_SQLPRIMARYKEYS	65
SQL_API_SQLPROCEDURECOLUMNS	66
SQL_API_SQLPROCEDURES	67
SQL_API_SQLPUTDATA	49
SQL_API_SQLROWCOUNT	20
SQL_API_SQLSETCONNECTATTR	1016
SQL_API_SQLSETCONNECTOPTION	50
SQL_API_SQLSETCURSORNAME	21
SQL_API_SQLSETDESCFIELD	1017
SQL_API_SQLSETDESCREC	1018
SQL_API_SQLSETENVATTR	1019
SQL_API_SQLSETPARAM	22
SQL_API_SQLSETPOS	68
SQL_API_SQLSETSCROLLOPTIONS	69
SQL_API_SQLSETSTMTATTR	1020
SQL_API_SQLSETSTMTOPTION	51
SQL_API_SQLSPECIALCOLUMNS	52
SQL_API_SQLSTATISTICS	53
SQL_API_SQLTABLEPRIVILEGES	70
SQL_API_SQLTABLES	54
SQL_API_SQLTRANSACT	23
SQL_ARD_TYPE	-99
SQL_ASYNC_ENABLE	4
SQL_ASYNC_ENABLE_DEFAULT	0
SQL_ASYNC_ENABLE_OFF	0
SQL_ASYNC_ENABLE_ON	1
SQL_ASYNC_MODE	10021
SQL_AT_ADD_COLUMN	1
SQL_AT_ADD_CONSTRAINT	8
SQL_AT_DROP_COLUMN	2

SQL_ATTR_ACCESS_MODE	101
SQL_ATTR_APP_PARAM_DESC	10011
SQL_ATTR_APP_ROW_DESC	10010
SQL_ATTR_ASYNC_ENABLE	4
SQL_ATTR_AUTO_IPD	10001
SQL_ATTR_AUTOCOMMIT	102
SQL_ATTR_CONCURRENCY	7
SQL_ATTR_CONNECTION_DEAD	1209
SQL_ATTR_CONNECTION_POOLING	201
SQL_ATTR_CONNECTION_TIMEOUT	113
SQL_ATTR_CP_MATCH	202
SQL_ATTR_CURRENT_CATALOG	109
SQL_ATTR_CURSOR_SCROLLABLE	-1
SQL_ATTR_CURSOR_SENSITIVITY	-2
SQL_ATTR_CURSOR_TYPE	6
SQL_ATTR_DISCONNECT_BEHAVIOR	114
SQL_ATTR_ENABLE_AUTO_IPD	15
SQL_ATTR_ENLIST_IN_DTC	1207
SQL_ATTR_ENLIST_IN_XA	1208
SQL_ATTR_FETCH_BOOKMARK_PTR	16
SQL_ATTR_IMP_PARAM_DESC	10013
SQL_ATTR_IMP_ROW_DESC	10012
SQL_ATTR_KEYSET_SIZE	8
SQL_ATTR_LOGIN_TIMEOUT	103
SQL_ATTR_MAX_LENGTH	3
SQL_ATTR_MAX_ROWS	1
SQL_ATTR_METADATA_ID	10014
SQL_ATTR_NOSCAN	2
SQL_ATTR_ODBC_CURSORS	110
SQL_ATTR_ODBC_VERSION	200
SQL_ATTR_OUTPUT_NTS	10001
SQL_ATTR_PACKET_SIZE	112
SQL_ATTR_PARAM_BIND_OFFSET_PTR	17
SQL_ATTR_PARAM_BIND_TYPE	18
SQL_ATTR_PARAM_OPERATION_PTR	19
SQL_ATTR_PARAM_STATUS_PTR	20
SQL_ATTR_PARAMS_PROCESSED_PTR	21
SQL_ATTR_PARAMSET_SIZE	22
SQL_ATTR_QUERY_TIMEOUT	0
SQL_ATTR_QUIET_MODE	111
SQL_ATTR_RETRIEVE_DATA	11
SQL_ATTR_ROW_ARRAY_SIZE	27
SQL_ATTR_ROW_BIND_OFFSET_PTR	23
SQL_ATTR_ROW_BIND_TYPE	5
SQL_ATTR_ROW_NUMBER	14
SQL_ATTR_ROW_OPERATION_PTR	24
SQL_ATTR_ROW_STATUS_PTR	25
SQL_ATTR_ROWS_FETCHED_PTR	26
SQL_ATTR_SIMULATE_CURSOR	10

SQL_ATTR_TRACE	104
SQL_ATTR_TRACEFILE	105
SQL_ATTR_TRANSLATE_LIB	106
SQL_ATTR_TRANSLATE_OPTION	107
SQL_ATTR_TXN_ISOLATION	108
SQL_ATTR_USE_BOOKMARKS	12
SQL_AUTOCOMMIT	102
SQL_AUTOCOMMIT_DEFAULT	1
SQL_AUTOCOMMIT_OFF	0
SQL_AUTOCOMMIT_ON	1
SQL_BATCH_ROW_COUNT	120
SQL_BATCH_SUPPORT	121
SQL_BEST_ROWID	1
SQL_BIGINT	-5
SQL_BINARY	-2
SQL_BIND_BY_COLUMN	0
SQL_BIND_TYPE	5
SQL_BIND_TYPE_DEFAULT	0
SQL_BIT	-7
SQL_BOOKMARK_PERSISTENCE	82
SQL_C_CHAR	1
SQL_C_DEFAULT	99
SQL_C_DOUBLE	8
SQL_C_FLOAT	7
SQL_C_LONG	4
SQL_C_NUMERIC	2
SQL_C_SHORT	5
SQL_CATALOG_LOCATION	114
SQL_CATALOG_NAME	10003
SQL_CATALOG_NAME_SEPARATOR	41
SQL_CATALOG_TERM	42
SQL_CATALOG_USAGE	92
SQL_CB_CLOSE	1
SQL_CB_DELETE	0
SQL_CB_PRESERVE	2
SQL_CD_FALSE	0
SQL_CD_TRUE	1
SQL_CHAR	1
SQL_CLOSE	0
SQL_CODE_DATE	1
SQL_CODE_DAY	3
SQL_CODE_DAY_TO_HOUR	8
SQL_CODE_DAY_TO_MINUTE	9
SQL_CODE_DAY_TO_SECOND	10
SQL_CODE_HOUR	4
SQL_CODE_HOUR_TO_MINUTE	11
SQL_CODE_HOUR_TO_SECOND	12
SQL_CODE_MINUTE	5
SQL_CODE_MINUTE_TO_SECOND	13

SQL_CODE_MONTH	2
SQL_CODE_SECOND	6
SQL_CODE_TIME	2
SQL_CODE_TIMESTAMP	3
SQL_CODE_YEAR	1
SQL_CODE_YEAR_TO_MONTH	7
SQL_COL_PRED_BASIC	2
SQL_COL_PRED_CHAR	1
SQL_COLLATION_SEQ	10004
SQL_COLUMN_ALIAS*	87
SQL_COMMIT	0
SQL_CONCAT_NULL_BEHAVIOR	22
SQL_CONCUR_DEFAULT	1
SQL_CONCUR_LOCK	2
SQL_CONCUR_READ_ONLY	1
SQL_CONCUR_ROWVER	3
SQL_CONCUR_VALUES	4
SQL_CONCURRENCY	7
SQL_CONVERT_BIGINT	53
SQL_CONVERT_BINARY	54
SQL_CONVERT_BIT	55
SQL_CONVERT_CHAR	56
SQL_CONVERT_DATE	57
SQL_CONVERT_DECIMAL	58
SQL_CONVERT_DOUBLE	59
SQL_CONVERT_FLOAT	60
SQL_CONVERT_FUNCTIONS	48
SQL_CONVERT_GUID*	173
SQL_CONVERT_INTEGER	61
SQL_CONVERT_INTERVAL_DAY_TIME	123
SQL_CONVERT_INTERVAL_YEAR_MONTH	124
SQL_CONVERT_LONGVARBINARY	71
SQL_CONVERT_LONGVARCHAR	62
SQL_CONVERT_NUMERIC	63
SQL_CONVERT_REAL	64
SQL_CONVERT_SMALLINT	65
SQL_CONVERT_TIME	66
SQL_CONVERT_TIMESTAMP	67
SQL_CONVERT_TINYINT	68
SQL_CONVERT_VARBINARY	69
SQL_CONVERT_VARCHAR	70
SQL_CONVERT_WCHAR*	122
SQL_CONVERT_WLONGVARCHAR*	125
SQL_CONVERT_WVARCHAR*	126
SQL_CORRELATION_NAME	74
SQL_CP_DEFAULT	0
SQL_CP_MATCH_DEFAULT	0
SQL_CP_OFF	0
SQL_CP_ONE_PER_DRIVER	1

SQL_CP_ONE_PER_HENV	2
SQL_CP_RELAXED_MATCH	1
SQL_CP_STRICT_MATCH	0
SQL_CREATE_ASSERTION	127
SQL_CREATE_CHARACTER_SET	128
SQL_CREATE_COLLATION	129
SQL_CREATE_DOMAIN	130
SQL_CREATE_SCHEMA	131
SQL_CREATE_TABLE	132
SQL_CREATE_TRANSLATION	133
SQL_CREATE_VIEW*	134
SQL_CUR_DEFAULT	2
SQL_CUR_USE_DRIVER	2
SQL_CUR_USE_IF_NEEDED	0
SQL_CUR_USE_ODBC	1
SQL_CURRENT_QUALIFIER	109
SQL_CURSOR_COMMIT_BEHAVIOR	23
SQL_CURSOR_DYNAMIC	2
SQL_CURSOR_FORWARD_ONLY	0
SQL_CURSOR_KEYSET_DRIVEN	1
SQL_CURSOR_ROLLBACK_BEHAVIOR	24
SQL_CURSOR_SENSITIVITY	10001
SQL_CURSOR_STATIC	3
SQL_CURSOR_TYPE	6
SQL_CURSOR_TYPE_DEFAULT	0
SQL_DATA_AT_EXEC	-2
SQL_DATA_SOURCE_NAME	2
SQL_DATA_SOURCE_READ_ONLY	25
SQL_DATE	9
SQL_DATE_LEN	10
SQL_DATETIME	9
SQL_DATETIME_LITERALS*	119
SQL_DB_DEFAULT	0
SQL_DB_DISCONNECT	1
SQL_DB_RETURN_TO_POOL	0
SQL_DBMS_NAME	17
SQL_DBMS_VER	18
SQL_DDL_INDEX	170
SQL_DECIMAL	3
SQL_DEFAULT	99
SQL_DEFAULT_TXN_ISOLATION	26
SQL_DESC_ALLOC_AUTO	1
SQL_DESC_ALLOC_TYPE	1099
SQL_DESC_ALLOC_USER	2
SQL_DESC_ARRAY_SIZE	20
SQL_DESC_ARRAY_STATUS_PTR	21
SQL_DESC_AUTO_UNIQUE_VALUE	11
SQL_DESC_BASE_COLUMN_NAME	22
SQL_DESC_BASE_TABLE_NAME	23

SQL_DESC_BIND_OFFSET_PTR	24
SQL_DESC_BIND_TYPE	25
SQL_DESC_CASE_SENSITIVE	12
SQL_DESC_CATALOG_NAME	17
SQL_DESC_CONCISE_TYPE	2
SQL_DESC_COUNT	1001
SQL_DESC_DATA_PTR	1010
SQL_DESC_DATETIME_INTERVAL_COD	1007
SQL_DESC_DATETIME_INTERVAL_PRE	26
SQL_DESC_DISPLAY_SIZE	6
SQL_DESC_FIXED_PREC_SCALE	9
SQL_DESC_INDICATOR_PTR	1009
SQL_DESC_LABEL	18
SQL_DESC_LENGTH	1003
SQL_DESC_LITERAL_PREFIX	27
SQL_DESC_LITERAL_SUFFIX	28
SQL_DESC_LOCAL_TYPE_NAME	29
SQL_DESC_MAXIMUM_SCALE	30
SQL_DESC_MINIMUM_SCALE	31
SQL_DESC_NAME	1011
SQL_DESC_NULLABLE	1008
SQL_DESC_NUM_PREC_RADIX	32
SQL_DESC_OCTET_LENGTH	1013
SQL_DESC_OCTET_LENGTH_PTR	1004
SQL_DESC_PARAMETER_TYPE	33
SQL_DESC_PRECISION	1005
SQL_DESC_ROWS_PROCESSED_PTR	34
SQL_DESC_SCALE	1006
SQL_DESC_SCHEMA_NAME	16
SQL_DESC_SEARCHABLE	13
SQL_DESC_TABLE_NAME	15
SQL_DESC_TYPE	1002
SQL_DESC_TYPE_NAME	14
SQL_DESC_UNNAMED	1012
SQL_DESC_UNSIGNED	8
SQL_DESC_UPDATABLE	10
SQL_DESCRIBE_PARAMETER	10002
SQL_DIAG_ALTER_DOMAIN	3
SQL_DIAG_ALTER_TABLE	4
SQL_DIAG_CALL	7
SQL_DIAG_CLASS_ORIGIN	8
SQL_DIAG_COLUMN_NUMBER	-1247
SQL_DIAG_CONNECTION_NAME	10
SQL_DIAG_CREATE_ASSERTION	6
SQL_DIAG_CREATE_CHARACTER_SET	8
SQL_DIAG_CREATE_COLLATION	10
SQL_DIAG_CREATE_DOMAIN	23
SQL_DIAG_CREATE_INDEX	-1
SQL_DIAG_CREATE_SCHEMA	64

SQL_DIAG_CREATE_TABLE	77
SQL_DIAG_CREATE_TRANSLATION	79
SQL_DIAG_CREATE_VIEW	84
SQL_DIAG_CURSOR_ROW_COUNT	-1249
SQL_DIAG_DELETE_WHERE	19
SQL_DIAG_DROP_ASSERTION 2	24
SQL_DIAG_DROP_CHARACTER_SET 2	25
SQL_DIAG_DROP_COLLATION 2	26
SQL_DIAG_DROP_DOMAIN 2	27
SQL_DIAG_DROP_INDEX	-2
SQL_DIAG_DROP_SCHEMA	31
SQL_DIAG_DROP_TABLE	32
SQL_DIAG_DROP_TRANSLATION	33
SQL_DIAG_DROP_VIEW	36
SQL_DIAG_DYNAMIC_DELETE_CURSOR	38
SQL_DIAG_DYNAMIC_FUNCTION	7
SQL_DIAG_DYNAMIC_FUNCTION_CODE	12
SQL_DIAG_DYNAMIC_UPDATE_CURSOR	81
SQL_DIAG_GRANT	48
SQL_DIAG_INSERT	50
SQL_DIAG_MESSAGE_TEXT	6
SQL_DIAG_NATIVE	5
SQL_DIAG_NUMBER	2
SQL_DIAG_RETURNCODE	1
SQL_DIAG_REVOKE	59
SQL_DIAG_ROW_COUNT	3
SQL_DIAG_ROW_NUMBER	-1248
SQL_DIAG_SELECT_CURSOR	85
SQL_DIAG_SERVER_NAME	11
SQL_DIAG_SQLSTATE	4
SQL_DIAG_SUBCLASS_ORIGIN	9
SQL_DIAG_UNKNOWN_STATEMENT	0
SQL_DIAG_UPDATE_WHERE	82
SQL_DM_VER*	171
SQL_DOUBLE	8
SQL_DRIVER_HDBC	3
SQL_DRIVER_HDESC	135
SQL_DRIVER_HENV	4
SQL_DRIVER_HLIB	76
SQL_DRIVER_HSTMT	5
SQL_DRIVER_NAME	6
SQL_DRIVER_ODBC_VER	77
SQL_DRIVER_VER	7
SQL_DROP	1
SQL_DROP_ASSERTION	136
SQL_DROP_CHARACTER_SET	137
SQL_DROP_COLLATION	138
SQL_DROP_DOMAIN	139
SQL_DROP_SCHEMA	140

SQL_DROP_TABLE	141
SQL_DROP_TRANSLATION	142
SQL_DROP_VIEW	143
SQL_DTC_DONE	0
SQL_DYNAMIC_CURSOR_ATTRIBUTES1	144
SQL_DYNAMIC_CURSOR_ATTRIBUTES2	145
SQL_ERROR	-1
SQL_EXPRESSIONS_IN_ORDERBY	27
SQL_FALSE	0
SQL_FD_FETCH_ABSOLUTE	16
SQL_FD_FETCH_FIRST	2
SQL_FD_FETCH_LAST	4
SQL_FD_FETCH_NEXT	1
SQL_FD_FETCH_PRIOR	8
SQL_FD_FETCH_RELATIVE	32
SQL_FETCH_ABSOLUTE	5
SQL_FETCH_DIRECTION	8
SQL_FETCH_FIRST	2
SQL_FETCH_LAST	3
SQL_FETCH_NEXT	1
SQL_FETCH_PRIOR	4
SQL_FETCH_RELATIVE	6
SQL_FILE_USAGE	84
SQL_FLOAT	6
SQL_FORWARD_ONLY_CURSOR_ATTRS1	146
SQL_FORWARD_ONLY_CURSOR_ATTRS2	147
SQL_GD_ANY_COLUMN	1
SQL_GD_ANY_ORDER	2
SQL_GET_BOOKMARK	13
SQL_GETDATA_EXTENSIONS	81
SQL_GUID	-11
SQL_GROUP_BY	88
SQL_HANDLE_DBC	2
SQL_HANDLE_DESC	4
SQL_HANDLE_ENV	1
SQL_HANDLE_SENV	5
SQL_HANDLE_STMT	3
SQL_IC_LOWER	2
SQL_IC_MIXED	4
SQL_IC_SENSITIVE	3
SQL_IC_UPPER	1
SQL_IDENTIFIER_CASE	28
SQL_IDENTIFIER_QUOTE_CHAR	29
SQL_INDEX_ALL	1
SQL_INDEX_CLUSTERED	1
SQL_INDEX_HASHED	2
SQL_INDEX_KEYWORDS	148
SQL_INDEX_OTHER	3
SQL_INDEX_UNIQUE	0

SQL_INFO_SCHEMA_VIEWS	149
SQL_INSENSITIVE	1
SQL_INSERT_STATEMENT	172
SQL_INTEGER	4
SQL_INTEGRITY	73
SQL_INTERVAL	10
SQL_INTERVAL_DAY	103
SQL_INTERVAL_DAY_TO_HOUR	108
SQL_INTERVAL_DAY_TO_MINUTE	109
SQL_INTERVAL_DAY_TO_SECOND	110
SQL_INTERVAL_HOUR	104
SQL_INTERVAL_HOUR_TO_MINUTE	111
SQL_INTERVAL_HOUR_TO_SECOND	112
SQL_INTERVAL_MINUTE	105
SQL_INTERVAL_MINUTE_TO_SECOND	113
SQL_INTERVAL_MONTH	102
SQL_INTERVAL_SECOND	106
SQL_INTERVAL_YEAR	101
SQL_INTERVAL_YEAR_TO_MONTH	107
SQL_INVALID_HANDLE	-2
SQL_IS_INTEGER	-6
SQL_IS_POINTER	-4
SQL_IS_SMALLINT	-8
SQL_IS_UINTEGER	-5
SQL_IS_USMALLINT	-7
SQL_KEYSET_CURSOR_ATTRIBUTES1	150
SQL_KEYSET_CURSOR_ATTRIBUTES2	151
SQL_KEYSET_SIZE	8
SQL_KEYSET_SIZE_DEFAULT	0
SQL_KEYWORDS*	89
SQL_LIKE_ESCAPE_CLAUSE	113
SQL_LOCK_TYPES*	78
SQL_LOGIN_TIMEOUT	103
SQL_LOGIN_TIMEOUT_DEFAULT	15
SQL_LONGVARBINARY	-4
SQL_LONGVARCHAR	-1
SQL_MAX_ASYNC_CONCURRENT_STMTS	10022
SQL_MAX_BINARY_LITERAL_LEN	112
SQL_MAX_CATALOG_NAME_LEN	34
SQL_MAX_CHAR_LITERAL_LEN	108
SQL_MAX_COLUMN_NAME_LEN	30
SQL_MAX_COLUMNS_IN_GROUP_BY	97
SQL_MAX_COLUMNS_IN_INDEX	98
SQL_MAX_COLUMNS_IN_ORDER_BY	99
SQL_MAX_COLUMNS_IN_SELECT	100
SQL_MAX_COLUMNS_IN_TABLE	101
SQL_MAX_CONCURRENT_ACTIVITIES	1
SQL_MAX_CURSOR_NAME_LEN	31
SQL_MAX_DRIVER_CONNECTIONS	0

SQL_MAX_IDENTIFIER_LEN	10005
SQL_MAX_INDEX_SIZE	102
SQL_MAX_LENGTH	3
SQL_MAX_LENGTH_DEFAULT	0
SQL_MAX_MESSAGE_LENGTH	512
SQL_MAX_OWNER_NAME_LEN*	32
SQL_MAX_PROCEDURE_NAME_LEN	33
SQL_MAX_QUALIFIER_NAME_LEN*	34
SQL_MAX_ROW_SIZE	104
SQL_MAX_ROW_SIZE_INCLUDES_LONG	103
SQL_MAX_ROWS	1
SQL_MAX_ROWS_DEFAULT	0
SQL_MAX_SCHEMA_NAME_LEN	32
SQL_MAX_STATEMENT_LEN	105
SQL_MAX_TABLE_NAME_LEN	35
SQL_MAX_TABLES_IN_SELECT	106
SQL_MAX_USER_NAME_LEN	107
SQL_MODE_DEFAULT	0
SQL_MODE_READ_ONLY	1
SQL_MODE_READ_WRITE	0
SQL_MULTIPLE_ACTIVE_TXN	37
SQL_MULT_RESULT_SETS	36
SQL_NAMED	0
SQL_NC_HIGH	0
SQL_NC_LOW	1
SQL_NEED_DATA	99
SQL_NEED_LONG_DATA_LEN	111
SQL_NO_DATA	100
SQL_NO_DATA_FOUND	100
SQL_NO_NULLS	0
SQL_NON_NULLABLE_COLUMNS	75
SQL_NONSCROLLABLE	0
SQL_NOSCAN	2
SQL_NOSCAN_DEFAULT	0
SQL_NOSCAN_OFF	0
SQL_NOSCAN_ON	1
SQL_NTS	-3
SQL_NTSL	-3
SQL_NULL_COLLATION	85
SQL_NULL_DATA	-1
SQL_NULL_HANDLE	0
SQL_NULL_HDBC	0
SQL_NULL_HDESC	0
SQL_NULL_HENV	0
SQL_NULL_HSTMT	0
SQL_NULLABLE	1
SQL_NULLABLE_UNKNOWN	2
SQL_NUMERIC	2
SQL_NUMERIC_FUNCTIONS	49

SQL_ODBC_API_CONFORMANCE*	9
SQL_ODBC_CURSORS	110
SQL_ODBC_INTERFACE_CONFORMANCE	152
SQL_ODBC_SAG_CLI_CONFORMANCE*	12
SQL_ODBC_SQL_CONFORMANCE*	15
SQL_ODBC_SQL_OPT_IEF*	73
SQL_ODBC_VER	10
SQL_OJ_ALL_COMPARISON_OPS	64
SQL_OJ_CAPABILITIES	115
SQL_OJ_CAPABILITIES	65003
SQL_OJ_FULL	4
SQL_OJ_INNER	32
SQL_OJ_LEFT	1
SQL_OJ_NESTED	8
SQL_OJ_NOT_ORDERED	16
SQL_OJ_RIGHT	2
SQL_OPT_TRACE	104
SQL_OPT_TRACEFILE	105
SQL_ORDER_BY_COLUMNS_IN_SELECT	90
SQL_OUTER_JOINS	38
SQL_OV_ODBC2	22
SQL_OV_ODBC3	33
SQL_OWNER_TERM*	39
SQL_OWNER_USAGE*	91
SQL_PACKET_SIZE	112
SQL_PARAM_ARRAY_ROW_COUNTS	153
SQL_PARAM_ARRAY_SELECTS	154
SQL_PARAM_BIND_BY_COLUMN	0
SQL_PARAM_BIND_TYPE_DEFAULT	0
SQL_PARAM_INPUT*	1
SQL_PARAM_INPUT_OUTPUT*	2
SQL_PARAM_OUTPUT*	3
SQL_PC_NON_PSEUDO	1
SQL_PC_PSEUDO	2
SQL_PC_UNKNOWN	0
SQL_POS_OPERATIONS*	79
SQL_POSITIONED_STATEMENTS*	80
SQL_PRED_BASIC	2
SQL_PRED_CHAR	1
SQL_PRED_NONE	0
SQL_PROCEDURES	21
SQL_PROCEDURE_TERM	40
SQL_PT_FUNCTION	2
SQL_PT_PROCEDURE	1
SQL_PT_UNKNOWN	0
SQL_QUALIFIER_LOCATION*	114
SQL_QUALIFIER_NAME_SEPARATOR*	41
SQL_QUALIFIER_TERM*	42
SQL_QUALIFIER_USAGE*	92

SQL_QUERY_TIMEOUT	0
SQL_QUERY_TIMEOUT_DEFAULT	0
SQL_QUIET_MODE	111
SQL_QUOTED_IDENTIFIER_CASE	93
SQL_RD_DEFAULT	1
SQL_RD_OFF	0
SQL_RD_ON	1
SQL_REAL	7
SQL_RESET_PARAMS	3
SQL_RETRIEVE_DATA	11
SQL_ROLLBACK	1
SQL_ROW_IDENTIFIER	1
SQL_ROW_NUMBER	14
SQL_ROW_UPDATES	11
SQL_ROWSET_SIZE	9
SQL_ROWSET_SIZE_DEFAULT	1
SQL_ROWVER	2
SQL_SC_NON_UNIQUE	0
SQL_SC_TRY_UNIQUE	1
SQL_SC_UNIQUE	2
SQL_SCCO_LOCK	2
SQL_SCCO_OPT_ROWVER	4
SQL_SCCO_OPT_VALUES	8
SQL_SCCO_READ_ONLY	1
SQL_SCHEMA_TERM	39
SQL_SCHEMA_USAGE	91
SQL_SCOPE_CURROW	0
SQL_SCOPE_SESSION	2
SQL_SCOPE_TRANSACTION	1
SQL_SCROLL_CONCURRENCY	43
SQL_SCROLL_OPTIONS	44
SQL_SCROLLABLE	1
SQL_SEARCH_PATTERN_ESCAPE	14
SQL_SENSITIVE	2
SQL_SERVER_NAME	13
SQL_SIGNED_OFFSET	-20
SQL_SIMULATE_CURSOR	10
SQL_SMALLINT	5
SQL_SPECIAL_CHARACTERS	94
SQL_SQL_CONFORMANCE	118
SQL_SQL92_DATETIME_FUNCTIONS*	155
SQL_SQL92_FOREIGN_KEY_DELETE_RULE*	156
SQL_SQL92_FOREIGN_KEY_UPDATE_RULE*	157
SQL_SQL92_GRANT*	158
SQL_SQL92_NUMERIC_VALUE_FUNCTIONS*	159
SQL_SQL92_PREDICATES*	160
SQL_SQL92_RELATIONAL_JOIN_OPERATORS*	161
SQL_SQL92_REVOKE*	162
SQL_SQL92_ROW_VALUE_CONSTRUCTOR*	163

SQL_SQL92_STRING_FUNCTIONS*	164
SQL_SQL92_VALUE_EXPRESSIONS*	165
SQL_STANDARD_CLI_CONFORMANCE*	166
SQL_STATIC_CURSOR_ATTRIBUTES1	167
SQL_STATIC_CURSOR_ATTRIBUTES2	168
SQL_STATIC_SENSITIVITY*	83
SQL_STILL_EXECUTING	2
SQL_STRING_FUNCTIONS	50
SQL_SUBQUERIES*	95
SQL_SUCCESS	0
SQL_SUCCESS_WITH_INFO	1
SQL_SYSTEM_FUNCTIONS	51
SQL_TABLE_TERM	45
SQL_TC_ALL	2
SQL_TC_DDL_COMMIT	3
SQL_TC_DDL_IGNORE	4
SQL_TC_DML	1
SQL_TC_NONE	0
SQL_TIME	10
SQL_TIME_LEN	8
SQL_TIMEDATE_ADD_INTERVALS	109
SQL_TIMEDATE_DIFF_INTERVALS	110
SQL_TIMEDATE_FUNCTIONS	52
SQL_TIMESTAMP	11
SQL_TIMESTAMP_LEN	19
SQL_TINYINT	-6
SQL_TRANSLATE_DLL	106
SQL_TRANSLATE_OPTION	107
SQL_TRUE	1
SQL_TXN_CAPABLE	46
SQL_TXN_ISOLATION	108
SQL_TXN_ISOLATION_OPTION	72
SQL_TXN_READ_COMMITTED	2
SQL_TXN_READ_UNCOMMITTED	1
SQL_TXN_REPEATABLE_READ	4
SQL_TXN_SERIALIZABLE	8
SQL_TYPE_DATE	91
SQL_TYPE_TIME	92
SQL_TYPE_TIMESTAMP	93
SQL_UB_DEFAULT	0
SQL_UB_FIXED	1
SQL_UB_OFF	0
SQL_UB_ON	1
SQL_UB_VARIABLE	2
SQL_UNBIND	2
SQL_UNICODE	-95
SQL_UNICODE_CHAR	-95
SQL_UNICODE_LONGVARCHAR	-97
SQL_UNICODE_VARCHAR	-96

SQL_UNION	96
SQL_UNKNOWN_TYPE 0	0
SQL_UNNAMED	1
SQL_UNSIGNED_OFFSET	-22
SQL_UNSPECIFIED	0
SQL_USE_BOOKMARKS	12
SQL_USER_NAME	47
SQL_VARBINARY	-3
SQL_VARCHAR	12
SQL_WCHAR	-8
SQL_WLONGVARCHAR	-10
SQL_WVARCHAR	-9
SQL_XOPEN_CLI_YEAR	10000