What's New

This version of the Microsoft ODBC Driver for Oracle includes several improvements to performance and stability. You should find the driver has greater functionality and speed due to the following additions:

- Improved Extended Fetch with support for Row-wise binding. See Level 2 Functions. For more details, see the *Microsoft ODBC 3.0 Programmer's References and SDK Guide*.
- Integration with Microsoft® Transaction Server for distributed transactions. See Using Microsoft Transaction Server
- Support for Oracle Packaged Procedure syntax in catalog functions. See Returning Array Parameters from Stored Procedures
- Implementation of SQLDescribeParam to provide more accuracy in parameter descriptions for SQL statements. See Level 2 Functions
- Ability to return arrays from Stored Procedures. See Returning Array Parameters from Stored Procedures
- Support for Bookmarks. See Level 2 Functions and Statement Options Table
- Extended Help file

This version of the driver has greater stability through testing against more environments like Microsoft Transaction Server and Internet Information Server.

Overview

The Microsoft® ODBC Driver for Oracle allows you to connect your ODBC-compliant application to an Oracle database. This version adds additional performance and control features, including access to PL/SQL packages, XA/DTC integration, and Oracle access from within Internet Information Server (IIS).The ODBC driver conforms to the Open Database Connectivity (ODBC) specification described in the *ODBC Programmer's Reference (Version 2.0)* for your platform.

This Help file describes how to set up, configure, and use the ODBC driver, and includes the following sections:

System Requirements

To use the Microsoft ODBC Driver for Oracle, you must have, Windows 95 or Windows NT and Oracle Client Software, version 7.3 or higher installed on your Windows system. The Microsoft ODBC Driver for Oracle supports only SQL*Net 2.3 or later. For more information on Oracle products, refer to your Oracle documentation set.

Adding and Modifying Data Sources Using Setup

A data source identifies a path to data that may include a network library, server, database, and other attributes—in this case, the data source is the path to an Oracle database. In order to connect to a data source, the Driver Manager checks the Window registry for specific connection information.

The registry entry created by the ODBC Data source Administrator is used by the ODBC Driver Manager and ODBC drivers. This entry contains information about each data source and its associated driver. Before you can connect to a data source, its connection information must be added to the registry.

To add and configure data sources, access the ODBC Administrator through the 32bit ODBC Control Panel in Windows. The ODBC Administrator then updates your data source connection information. As you add data sources, the ODBC Administrator updates the registry information for you.

Adding a Data Source for Windows

1. To start the ODBC Administrator, double-click the ODBC icon in the Windows **Control Panel**.
2. When you see the **ODBC Data Source Administrator** dialog box, click the **Add** button. The **Create New Data Source** dialog box appears.
3. Select the **ODBC driver**, and then click **Finish**. The **Microsoft ODBC for Oracle Setup** dialog box appears.
4. In the **Data Source Name** box, enter the name of the data source you want to access. It can be any name that you choose.
5. In the **Description** box, enter the description for the driver. This is an optional field that describes the database driver that the data source connects to. It can be any name that you choose.
6. In the **User Name** box, enter your database user name. The user name is your database user id.
7. In the **Connect String** box, enter the connect string for the Oracle Server engine. The connect string identifies the Oracle Server engine that you want to access.
8. Click **OK** to add this data source.

   Note The **Data Sources** dialog box appears, and the ODBC Administrator updates the registry information. The User Name and connect string that you enter become the default data source connection values for this data source. That is, when you connect to the data source using either a dialog box or connection string, these values become the default entries for the data source connection.
9. Click **Add** to add another data source or click **Close** to exit.

Modifying a data source for Windows

1. Invoke the ODBC Administrator. The **Data Sources** dialog box appears.
2. In the **Data Sources** dialog box, select the Oracle data source you want to modify and then click **Setup**. The **Microsoft ODBC for Oracle Setup** dialog box appears.
3. Modify the applicable data source fields, and then click **OK**.

When you have finished modifying the information in this dialog box, the ODBC Administrator updates the registry information.

Configuring the ODBC Driver for Oracle

You can control performance of the ODBC driver by knowing the data environment and correctly setting the parameters of the data source connection through the ODBC Data Sources Administrator dialog box or through connect string parameters. The dialog box provides the following controls for connecting to a data source using the dialog box or using connect strings:

- **User DSN tab** Displays a list of the Data Source Names that are local to the computer.
- **System DSN tab** Displays a dialog box that allows you to add or remove a system data source. System data sources are accessible to all users on the local machine.
- **File DSN tab** Displays a dialog box that allows you to add or remove a file data source from the local machine. File data sources can be shared by all users who have the same driver installed.
- **ODBC Drivers tab** Displays a list of the installed ODBC drivers.
- **Tracing tab** Displays a dialog box that enables you to specify how the ODBC Driver Manager traces calls to ODBC functions. You can configure tracing separately for each installed ODBC application.
- **About tab** Displays a dialog box that lists the installed ODBC component files.

After you add a data source you can use the ODBC Data Sources Administrator dialog box to configure the access to your data source. Select a data source, then click on of the following tabs to edit or review the information.

Connecting to a Data Source

An ODBC application can pass connection information in a number of ways. For example, the application might have the driver always prompt the user for connection information. Or the application might expect a connection string that specifies the data source connection. How you connect to a data source depends on the connection method that your ODBC applications uses.

One common way of connecting to a data source is through the Data Source dialog box. If your ODBC application is set up to use a dialog box, that dialog box is displayed and prompts you for the appropriate data source connection information.

Another way is through use of the connection string.

Connecting to a data source using a dialog box

1. When you see the **Data Source** dialog box, select an Oracle data source and then click **OK**. The **Connect** dialog boxes appears.
2. Fill in the appropriate information for the **Connect** dialog box, and then click **OK**.

Once the connection information is verified, your application can access the information that the data source contains using the ODBC driver.

Connection String Attributes

Some applications may require a connection string that specifies data source connection information instead of using a dialog box to obtain this information. The connection string is made up of a number of attributes that specify how a driver connects to a data source. An attribute identifies a specific piece of information that the driver needs to know before it can make the appropriate data source connection. Each driver may have a different set of attributes, but the connection string format is always the same. A connection string has the following format:

"DSN=data-source-name[;SERVER=*value*] [;PWD=*value*] [;UID=*value*] [;<Attribute>=<value>]"

Note The version of the Microsoft ODBC Driver for Oracle supports the slightly different version 1 connection string format.

You must specify the data-source-name if you do not specify the UID, PWD, SERVER (or CONNECTSTRING) and DRIVER attributes. However, all other attributes are optional. If you do not specify an attribute, that attribute defaults to the one that you specified in the DSN tab of the ODBC Data Source Administrator. The attribute value might be case-sensitive.

The attributes for the connection string are as follows:

Attribute	Description	Default value
DSN	The data source name. This name is listed in the ODBC Drivers tab of the ODBC Data Sources Administrator.	""
PWD	The password for the Oracle server that you want to access.	""
SERVER	The connect string for the Oracle Server that you want to access.	""
UID	The Oracle Server user name. Depending on your system, this attribute may not be optional—that is, certain databases and tables may require this attribute for security purposes. Use "/" to use Oracle's operating system authentication.	""
BUFFERSIZE	The optimal buffer size used when fetching columns. The driver optimizes fetching so that one fetch from the Oracle Server returns enough rows to fill a buffer of this size. Larger values tend to increase performance, if you're fetching a lot of data.	65535
SYNONYMCOLUMNS	When this value is true (1), an SQLColumn( ) API call returns column information Otherwise SQLColumn( ) returns only columns for tables and views. The ODBC Driver provides faster access when this value is not set.	1
REMARKS	When this value is true (1) the driver returns Remarks columns for the	0

SQLColumns result set. The ODBC Driver provides faster access when this value is not set.

For example, a connection string that connects to the Employees data source using the mickey.world Oracle server as the Oracle User cindy, would be:

"DSN=Employees;UID=cindy;PWD=secret;SERVER=mickey.world"

Another example, a connection string that connects to the Payroll data source using operating system authentication and the moola Oracle server would be:

"DSN=Payroll;UID=/;PWD=;SERVER=moola"

ODBC Conformance Levels

ODBC defines two types of conformance standards for drivers—the API conformance standard and the SQL grammar conformance standard. API conformance refers to the functions that a driver supports. SQL conformance refers to the SQL grammar that the driver supports. Each conformance standard is made up of levels.

API Conformance Level

The ODBC driver supports the Core and Level 1 API Functions. The driver also supports the following Level 2 functions:

- SQLBrowseConnect( )
- SQLDataSources( )
- SQLDescribeParam( )
- SQLExtendedFetch( )
- SQLForeignKeys( )
- SQLMoreResults( )
- SQLNativeSql( )
- SQLNumParams( )
- SQLPrimaryKeys( )
- SQLProcedureColumns( )
- SQLProcedures( )
- SQLSetPos( )
- SQLSetScrollOptions( )

Supported Options

The driver supports the following options for the SQLGetConnectOption( ) and SQLSetConnectOption( ) Level 1 functions:

- SQL_ACCESS_MODE (SQLGetConnectOption( ) only)
- SQL_AUTOCOMMIT
- SQL_ODBC_CURSORS
- SQL_OPT_TRACEFILE
- SQL_OPT_TRACE
- SQL_TRANSLATE_DLL
- SQL_TRANSLATE_OPTION
- SQL_TXN_ISOLATION

The driver supports the following options for the SQLGetStmtOption( ) and SQLSetStmtOption( ) Level 1 functions:

- SQL_BIND_TYPE
- SQL_CONCURRENCY
- SQL_CURSOR_TYPE
- SQL_KEYSET_SIZE
- SQL_MAX_ROW
- SQL_ROWSET_SIZE

SQL Conformance Levels

The ODBC driver supports the Minimum SQL grammar and Core SQL grammar and also supports the following ODBC extensions to SQL:

- Date, time, and timestamp data
- Left and right Outer joins
- Numeric functions:

abs	log	round	tan
ceiling	log10	second	truncate
cos	mod	sign	
exp	pi	sin	
floor	power	sqrt	

- Date functions:

curdate	dayofweek	monthname	second
curtime	dayofyear	minute	week
dayname	hour	now	year
dayofmonth	month	quarter	

- String functions:

ascii	left	right	ucase
char	length	rtrim	
concat	ltrim	soundex	
lcase	replace	substring	

- Type-conversion function:
  convert
- The following system functions:

ifnull	user

Mapping Data Types

The Oracle Server supports a set of data types. The ODBC driver maps these data types to their appropriate ODBC SQL data types. The following table lists the Oracle 7.3 Server data type and its corresponding ODBC SQL data type.

Oracle Server Data Type	ODBC SQL Data Type
CHAR	SQL_CHAR
DATE	SQL_TIMESTAMP
FLOAT	SQL_DOUBLE
INTEGER	SQL_DECIMAL
LONG	SQL_LONGVARCHAR
LONG RAW	SQL_LONGVARBINARY
NUMBER	SQL_DECIMAL
RAW	SQL_VARBINARY
VARCHAR2	SQL_VARCHAR

Note ODBC SQL data types do not support the Oracle MLSLABEL data type.

Using Microsoft Transaction Server

You can enable an Oracle database to work with transactional Microsoft Transaction Server (MTS) components on Windows NT and Windows 95. To enable an Oracle database to work with MTS components that support transactions, systems administrators should create a view named V$XATRANS$. To create this script, you must run an Oracle-supplied script. For more information, see the *Microsoft Transaction Server Help* or your Oracle documentation.

Using Microsoft Internet Information Server

If you encounter difficulty connecting from within an Internet Information Server (IIS) script (particularly an ORA-12641 error), add the following line to the SQLNET.ORA file.

```
SQLNET.AUTHENTICATION_SERVICES = (none)
```

This will disable the Secure Network Services so that Anonymous authentication can work.

Using Operating System Authentication

Oracle operating system authentication relies on the underlying operating system to control access to database accounts. Users need not enter a password when using this type of login.

To take advantage of this feature, specify "/" as the userid and do not specify a password when connecting using any of the connection APIs, SQLBrowseConnect, SQLConnect and SQLDriverConnect.

Limitations of Using Keyset-Driven Cursors

You must be able to retrieve a single ROWID column for the table queried. A keyset-driven cursor cannot be used on joins, queries, or statements containing DISTINCT, GROUP BY, UNION, INTERSECT or MINUS clauses.

Returning Array Parameters from Stored Procedures

In Oracle 7.3 there is no way to access a PL/SQL Record Type except from a PL/SQL program. If a packaged procedure/function has a formal argument defined as a PL/SQL Record Type, it is not possible to bind that formal argument as a parameter. Use the PL/SQL TABLE type in the Microsoft Oracle ODBC driver to invoke array parameters from procedures containing the correct escape sequences.

To invoke the procedure use the following syntax

```
{call  <package-name>.<proc-or-func>;
(..., {resultset <max-records-requested> ,<formal-array-param_1>,;
 <formal-array-param_2>,...,<formal-array-param_n> }, ... ) }
```

Note

- The `<max-records-requested>` parameter must be >= number of rows present in the result set. Otherwise Oracle returns an error that is passed to you by the driver.
- You cannot use PL/SQL records as array parameters. Each array parameter can represent only one column of a database table.

The following example defines a package containing two procedures that return different result sets, then provides two ways to return result sets from the package:

Package definition:

```
CREATE OR REPLACE PACKAGE SimplePackage AS

TYPE t_id is TABLE of  NUMBER(5)
    INDEX BY BINARY_INTEGER;

TYPE t_Course is TABLE of VARCHAR2(10)
    INDEX BY BINARY_INTEGER;

TYPE t_Dept is TABLE of VARCHAR2(5)
    INDEX BY BINARY_INTEGER;

PROCEDURE proc1
    (    o_id     OUT   t_id,
         ao_course   OUT   t_Course,
      ao_dept          OUT   t_Dept
    );

TYPE t_pk1Type1 IS TABLE OF VARCHAR2(100) INDEX BY BINARY_INTEGER;
TYPE t_pk1Type2 IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
PROCEDURE proc2
   (
   i_Arg1         IN    NUMBER,
   ao_Arg2        OUT   t_pk1Type1,
   ao_Arg3        OUT   t_pk1Type2
   );

END SimplePackage;

CREATE OR REPLACE PACKAGE BODY SimplePackage AS
    PROCEDURE  proc1 ( o_id OUT t_id,
    ao_course OUT t_Course, ao_dept OUT t_Dept   ) AS
    BEGIN
```

```
     o_id(1):= 200;
          ao_course(1) :=  'M101';
          ao_dept(1) :=  'EEE' ;

          o_id(2) := 201;
          ao_course(2) :=  'PHY320';
          ao_dept(2) :=  'ECE' ;

     END proc1;
PROCEDURE proc2
   (
   i_Arg1          IN    NUMBER,
   ao_Arg2         OUT   t_pk1Type1,
   ao_Arg3         OUT   t_pk1Type2
   )
AS
   i  NUMBER;
BEGIN
   FOR i IN 1 .. i_Arg1 LOOP
      ao_Arg2(i) := 'Row Number ' || to_char(i);
   END LOOP;
   FOR i IN 1 .. i_Arg1 LOOP
      ao_Arg3(i) := i;
   END LOOP;
END proc2;
END SimplePackage;
```

To invoke the procedure, PROC1

1. Return all the columns in a single result set:

   ```
   { call SimplePackage.Proc1( {resultset  3, o_id , ao_course, ao_dept  } )
   }
   ```

2. Return each column as a single result set:

   ```
   {call SimplePackage.Proc1( {resultset 3, o_id},  {resultset 3,
   ao_course}, {resultset 3, ao_dept} ) }
   ```

   This returns three result sets, one for each column.

To invoke procedure, PROC2

1. Return all the columns in a single result set:

   ```
   {call SimplePackage.Proc2( 5 , {resultset  5, ao_Arg2, ao_Arg3} ) }
   ```

2. Return each column as a single result set:

   ```
   {call SimplePackage.Proc2( 5 , {resultset 5, ao_Arg2}, {resultset 5,
   ao_Arg3} ) }
   ```

Ensure that your applications fetch all the result sets using SQLMoreResults API. For more information, refer to the Microsoft ODBC SDK documentation.

Note In the Microsoft ODBC for Oracle driver version 2.0, you cannot use Oracle functions that return PL/SQL arrays to return result sets.

Notes on API Functions

The Microsoft ODBC Driver for Oracle supports the Core, Level 1 and Level 2 API functions. These functions are listed in ODBC Conformance Levels.

- Core level interface(CLI) conformance provides features defined in the ISO CLI specification and the mandatory features defined in the X/Open CLI specification.
- Level 1 conformance provides Core level interface functionality as well as additional features such as transactions.
- Level 2 conformance provides Level 1 functionality as well as additional features like bookmarks, dynamic parameters, and asynchronous execution of ODBC functions.

Thread-Safety Notes on API Functions

The Microsoft ODBC driver for Oracle is thread-safe, however Oracle does not allow active multiple concurrent statements on a single connection. The driver enforces this restriction for you. In other words, in multi-threaded applications, though any thread may call into the Oracle ODBC driver at any time, the driver blocks any other thread from the driver on the same connection until the original thread leaves the driver.

The driver does not block if there are two statements on two different connections. However, if there is a single connection with two statements, there is potential for blocking.

Core Level API Functions

Functions at this level comprise the minimum level of interface conformance for ODBC drivers.

API Function	Notes
SQLAllocConnect	Allocates memory for a connection handle, *hdbc*, within the environment identified by *henv*. The Driver Manager processes this call and calls the driver's SQLAllocConnect whenever **SQLConnect**, SQLBrowseConnect, or **SQLDriverConnect** is called.
SQLAllocEnv	Displays a dialog box specifying the requirement for Oracle Client software, then returns SQL_NULL_HANDLE. If the Oracle Client software is not installed, this function Allocates memory for an environment handle, *henv*, and initializes the ODBC call level interface for use by an application.
SQLAllocStmt	Allocates memory for a statement handle and associates the statement handle with the connection specified by *hdbc*. The Driver Manager passes this call to the driver, which allocates the memory for the *hstmt* structure.
SQLBindCol	Assigns storage space for a result column and specifies the type of the result.
SQLCancel	Cancels the processing on a statement handle, *hstmt*. In some cases, Oracle does not allow you to cancel a running statement. This means that a running statement will continue until Oracle completes the process, at which time the results from the statements are cancelled by the ODBC driver.
SQLColAttributes	Returns descriptor information for a column in a result set. Descriptor information is returned as a character string, a 32-bit descriptor-dependent value, or an integer value.
SQLConnect	Connects to a data source. To use Oracle Operating System Authentication, specify "/" as the szUID and "" as the szAuthStr parameters.
SQLDescribeCol	Returns the name, type, precision, scale and nullability of the given result column.
SQLDisconnect	Closes a connection. If connection pooling is enabled for a shared environment, and an application calls SQLDisconnect on a connection in that environment, the connection is returned to the connection pool, and is still available to other components using the same shared environment.
SQLError	Returns error or status information about the last error. The driver maintains a stack or list of errors that can be returned for the *hstmt*, *hdbc*, and *henv* arguments, depending on how the call to **SQLError** is made. The error queue is flushed after each statement. Usually retrieves an Oracle error message, and is otherwise empty.
SQLExecDirect	Executes a new, preparable SQL statement. The driver uses the current values of the parameter marker variables if any parameters exist in the statement. If your table, view or

	field names contain spaces, enclose the names in back quote marks. For example, if your database contains a table named `My Table` and the field `My Field`, enclose each element of the identifier as shown below:
	`SELECT “My Table”. “Field1”,;`
	`“My Table”.“Field2” FROM “My Table”`

SQLExecute	Executes a prepared SQL statement (a statement already prepared by **SQLPrepare**). The driver uses the current values of the parameter marker variables if any parameters exist in the statement.
SQLFetch	Retrieves one row from a result set into the locations specified by the previous calls to **SQLBindCol**. Prepares the driver for a call to **SQLGetData** for the unbound columns.
SQLFreeConnect	Releases a connection handle and frees all memory allocated for the handle.
SQLFreeEnv	Closes the ODBC Driver and releases all memory associated with the driver.
SQLFreeStmt	Stops processing associated with a specific *hstmt*, closes any open cursors associated with the *hstmt*, discards pending results, and, optionally, frees all resources associated with the statement handle.
SQLGetCursorName	Returns the name of the cursor associated with the given *hstmt*.
SQLNumResultCols	Returns the number of columns in a result set cursor.
SQLPrepare	Prepares an SQL statement by planning how to optimize and execute the statement. The SQL statement is compiled for execution by **SQLExecDirect**. If your table, view or field names contain spaces, enclose the names in back quote (“) marks. For example, if your database contains a table named `My Table` and the field `My Field`, enclose each element of the identifier as shown below: `SELECT * FROM “My Table”.“My Field”` For information on using result sets containing arrays as formal parameters, see. Returning Array Parameters from Stored Procedures.
SQLRowCount	Oracle does not provide a way to determine the number of rows in a result set until after you fetch the last row, so it returns -1.
SQLSetCursorName	Associates a cursor name with an active statement handle, *hstmt*.
SQLSetParam	Replaced by **SQLBindParameter** in ODBC 2.x
SQLTransact	Requests a commit or rollback operation for all active operations on all statement handles (*hstmt*s) associated with a connection, or all connections associated with the environment handle, *henv*. If a commit fails when in manual mode, the transaction remains active; you can choose to rollback the transaction or retry the commit operation. If a commit operation fails when in automatic transaction mode,

the transaction is rolled back automatically; the transaction cannot be inactive.

Level 1 API Functions

Functions at this level provide Core interface conformance, plus additional functionality such as transaction support.

API Function	Notes
SQLColumns	Creates a result set for a table which is the column list for the specified table or tables. When you request columns for a PUBLIC synonym, you must have set the SYNONYMCOLUMNS connection attribute and specify an empty string as the szTableOwner argument. When returning columns for PUBLIC synonyms, the driver sets the TABLE NAME column to an empty string. The result set contains an additional column, ORDINAL POSITION, at the end of each row. This value is the ordinal position of the column in the table.
SQLDriverConnect	Connects to an existing data source. For details, see Connection String Attributes.
SQLGetConnectOption	Returns the current setting of a connection option. This function is partially supported: the driver supports all values for the *fOption* argument, but does not support some of *vParam* values for the *fOption* argument **SQL_TXN_ISOLATION**. For more information, see the Connect Options Table.
SQLGetData	Retrieves the value of a single field in the current record of the given result set.
SQLGetFunctions	Returns TRUE for all supported functions. Implemented by the Driver Manager.
SQLGetInfo	Returns information about the ODBC driver and data source associated with a connection handle, *hdbc*.
SQLGetStmtOption	Returns the current setting of a statement option. For more information, see the Statement Options Table.
SQLGetTypeInfo	Returns information about the data types supported by a data source. The driver returns the information in a SQL result set.
SQLParamData	Used in conjunction with **SQLPutData** to specify parameter data at statement execution time.
SQLPutData	Allows an application to send data for a parameter or column to the driver at statement execution time.
SQLSetConnectOption	Provides access to options that govern aspects of the connection. This function is partially supported: the driver supports all values for the *fOption* argument, but does not support some of *vParam* values for the *fOption* argument **SQL_TXN_ISOLATION**. For more information, see the Connect Options Table.
SQLSetStmtOption	Sets options related to a statement handle, *hstmt*. For more information, see the Statement Option Table.
SQLSpecialColumns	Retrieves the optimal set of columns that uniquely identifies a row in the table.
SQLStatistics	Retrieves a list of statistics about a single table and the

	indexes, or tag names, associated with the table. The driver returns the information as a result set.
SQLTables	Returns the list of table names specified by the parameter in the **SQLTables** statement. If no parameter is specified, returns the table names stored in the current data source. The driver returns the information as a result set. Enumeration type calls will not receive a result set entry for remote views or local parameterized views. However, a call to **SQLTables** with a unique table name specifier will find a match for such a view if present with that name; this allows the API to used to check for name conflicts prior to creation of a new table. PUBLIC synonyms are returned with a TABLE_OWNER value of “”. VIEWS owned by SYS or SYSTEM are identified as SYSTEM VIEW.

Level 2 API Functions

Functions at this level provide Level 1 interface conformance, plus additional functionality such as support for bookmarks.

API Function	Notes
SQLBindParameter	Associates a buffer with a parameter marker in a SQL statement.
SQLBrowseConnect	Returns successive levels of attributes and attribute values.
SQLDataSources	Lists data source names. Implemented by the Driver Manager
SQLDescribeParam	Returns the description of a parameter marker associated with a prepared SQL statement. Returns a best guess of what the parameter is, based upon parsing the statement. If the parameter type cannot be determined, SQL_VARCHAR returns with length 2000.
SQLDrivers	Implemented by the Driver Manager.
SQLExtendedFetch	Similar to **SQLFetch**, but returns multiple rows using an array for each column. The result set is forward-scrollable and can be made backward-scrollable if the cursor is defined to be static, not forward-only. For forward-only cursors with default column binding, column data from data sets larger than the BUFFERSIZE connection attribute is fetched directly into data buffers. Does not support variable-length bookmarks and does not support fetching a rowset at an offset (other than 0) from a bookmark
SQLForeignKeys	Returns a list of foreign keys in a single table or a list of foreign keys in other tables that refer to a single table.
SQLMoreResults	Determines whether more results are pending on a statement handle, *hstmt*, containing SELECT, UPDATE, INSERT, or DELETE statements and, if so, initializes processing for those results. Oracle supports multiple result sets only from stored procedures, when using {resultset… } escape sequences.
SQLNativeSql	For information on usage, see Returning Array Parameters from Stored Procedures.
SQLNumParams	Returns the number of parameters in a SQL statement. The number of parameters should equal the number of question marks in the SQL statement passed to **SQLPrepare**.
SQLPrimaryKeys	Returns the column names that comprise the primary key for a table.
SQLProcedureColumns	Returns a list of input and output parameters, the return value, and the columns in the result set of a single procedure. For packaged procedures, the PROCEDURE NAME column is in *packagename.procedurename* format.
SQLProcedures	Returns a list of procedures in the data source. For packaged procedures, the PROCEDURE NAME column is in *packagename.procedurename* format. Since Oracle does not provide a way to distinguish packaged procedures from packaged functions, the driver

	returns SQL_PT_UNKNOWN for the PROCEDURE_TYPE column.
SQLSetPos	Sets the cursor position in a rowset. You can use **SQLSetPos** with **SQLGetData** to retrieve rows from unbound columns after positioning the cursor to a specific row in the rowset. Rows added to the result set using fOption SQL_ADD are added after the last row in the result set.
SQLSetScrollOptions	Sets options that control the behavior of cursors associated with a statement handle, *hstmt*. For details, see the supported Cursor Type and Concurrency Combinations Table.

Connect Options Table

These options allow customization of the database connection within an application.

Connect Option	Notes
SQL_AUTOCOMMIT	If you choose SQL_AUTOCOMMIT_OFF, your application must explicitly commit or roll back transactions with **SQLTransact**.
SQL_ODBC_CURSORS	This connection attribute is implemented in the Driver Manager.
SQL_OPT_TRACE	This connection attribute is implemented in the Driver Manager.
SQL_OPT_TRACEFILE	This connection attribute is implemented in the Driver Manager.
SQL_TRANSLATE_DLL	Returns error: "Driver not capable".
SQL_TRANSLATE_OPTION	A 32-bit value passed to the translation DLL.
SQL_TXN_ISOLATION	The driver allows only: SQL_TXN_READ_COMMITTED The following *vParam*s are not supported: SQL_TXN_READ_UNCOMMITTED SQL_TXN_REAPEATABLE_READ SQL_TXN_SERIALIZABLE
SQL_ATTR_ENLIST_IN_DTC	This ODBC 3.0 connection attribute allows you to use the Oracle ODBC driver in Distributed Transactions coordinated by Microsoft Transaction Server; provide the Interface pointer, pITransaction, to the transaction as the vParam argument.
SQL_ATTR_CONNECTION_DEAD	This read-only ODBC 3.5 connection attribute allows you to determine whether the connection to the Oracle server has failed. Get only, cannot Set.

Statement Options Table

These options allow customization of a specific execution statement within an application.

Statement Options	Notes
SQL_BIND_TYPE	Cannot exceed 2,147,483,647 bytes or available memory.
SQL_CONCURRENCY	For allowed values, see the Supported Cursor Type and Concurrency Combinations Table
SQL_CURSOR_TYPE	The driver does not allow SQL_CURSOR_DYNAMIC. See **SQLSetScrollOptions** for more information. For allowed values, see the Supported Cursor Type and Concurrency Combinations Table
SQL_GET_BOOKMARK	Returns a 32-bit integer value that is the bookmark for the current record number. Get only, cannot Set.
SQL_KEYSET_SIZE	Can only be set to 0.
SQL_MAX_ROWS	The maximum number of rows to return from a result set.
SQL_ROW_NUMBER	Returns a 32-bit integer specifying the position of the current row within the result set. Get only, cannot Set.
SQL_ROWSET_SIZE	Cannot exceed 4,294,967,296 rows, however you must have enough virtual memory in your computer to handle whatever you request.
SQL_USE_BOOKMARKS	Supports setting SQL_USE_BOOKMARKS to SQL_UB_ON, and exposes fixed-length bookmarks.

Cursor Type and Concurrency Combinations Table

Cursor types control the functionality of the cursor provided to the user. Concurrency options control the updatability and locking behavior of a result set.

Cursor Type	Concurrency (allowed values)
SQL_CURSOR_FORWARD_ONLY	SQL_CONCUR_READ_ONLY
SQL_CURSOR_STATIC	SQL_CONCUR_READ_ONLY
SQL_CURSOR_KEYSET_DRIVEN*	SQL_CONCUR_READ_ONLY SQL_CONCUR_LOCK ** SQL_CONCUR_VALUES

Notes

* See Limitations of Using Keyset-Driven Cursors.

** SQL_CONCUR_LOCK is only supported when the **SQL_AUTOCOMMIT** connection option is set to SQL_AUTOCOMMIT_OFF.

Error Messages

When an error occurs, the Microsoft ODBC Driver for Oracle returns the SQLSTATE (an ODBC error code), and an error message. The driver derives this information both from errors detected by the driver and errors returned by the Oracle Server.

Messages returned by Oracle ODBC Driver

If there is an Oracle Error message available, it will be returned preceded by the [Microsoft], [ODBC Driver for Oracle], and [Oracle] tags, otherwise the message is returned without the [Oracle] tag as in the following examples:

Oracle error message:

```
[Microsoft][ODBC driver for Oracle][Oracle]ORA-nnnnn message-text
```

Oracle ODBC Driver error message

```
[Microsoft][ODBC driver for Oracle]
```